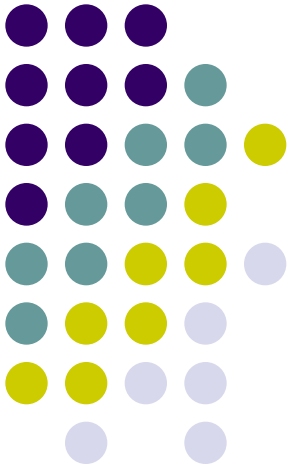
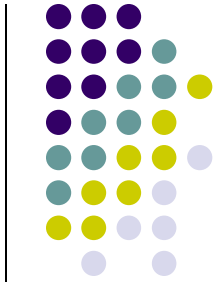
Motion Planning by Configuration Space and Visibility Graphs

Lecture 9

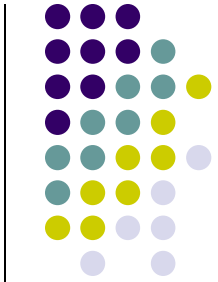


The Robot Motion Planning Problem



Most General Form: Given an a-priori known workspace with stationary objects of arbitrary shape whose representations are known and moving objects whose future trajectories are known find a collision free path that is optimal in some metric for a robot with arbitrary degrees of freedom between any two given points if one exists. (Generally unsolvable if optimality is required!)

Solvable form with optimality: Stationary objects of regular shapes (convex/concave polygonal), moving objects whose trajectories are linear and velocities uniform and a robot with limited degrees of freedom (2 or 3 in number)



COMPLEXITY

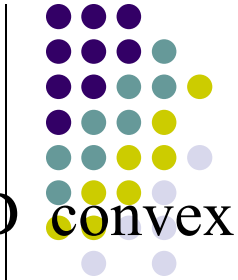
• Complexity of motion planning problem in presence of moving objects is inherently harder than the stationary case

• For stationary objects and a robot with two degrees of freedom the motion planning problem can be solved in polynomial time generally

• The problem of finding a path for a point robot in a plane with ***bounded*** velocity in presence of moving objects that are convex polygons moving with linear velocities ***without*** rotation the problem is classified as **NP Hard**

• For stationary objects and robots with arbitrary degrees of freedom the notion of probabilistic completeness is used for ascertaining complexity

Configuration Space Approach



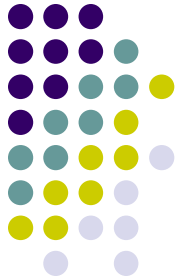
• This method of 2-D motion planning assumes a set of 2-D convex polygonal obstacles and a 2-D convex polygonal mobile robot.

• The general idea is grow the obstacles by the size of the mobile robot, thereby reducing the analysis of the robot's motion from a moving area to a single moving point.

• The point will always be a safe distance away form each obstacle due the growing step of each obstacle. Once we shrink the robot to a point, we can then find a safe path for the robot using a graph search technique.

• Reference: *An Algorithm for Planning Collision Free Paths Among Polyhedral Obstacles* by T. Lozano-Perez and M. Wesley. *(Communications of the ACM 22: 560-570)*

Configuration Space Approach



- Please note the following:
  - The planning is done for a point on the robot. The CS approach makes sure that if a path is collision free for this point it is collision free for the whole robot
  - The number of degrees of freedom of the robot is the dimension of the CS
  - The configuration obstacle (usually denoted as CB) is constructed by tracing the locus of the reference point of the robot for those configurations where the robot just graces the obstacle (Boundary between collision and no collision) or the limiting case

The Overall Algorithm with Visibility Graph

• Grow each obstacle in the scene by the size of the mobile robot. This is done by finding a set of vertices that determine the grown obstacle.

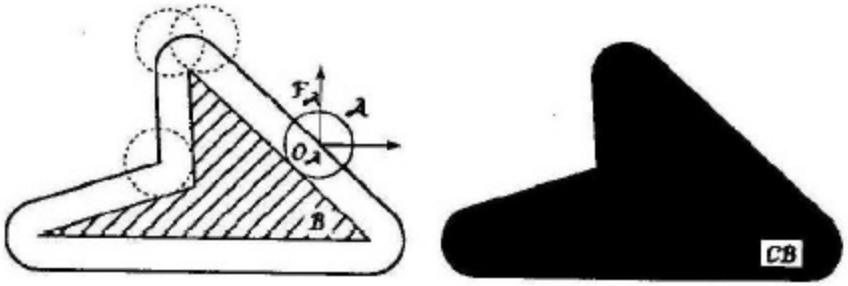
•We can now create a visibility graph. A visibility graph is an undirected graph G = (V,E) where the V is the set of vertices of the grown obstacles plus the start and goal points, and E is a set of edges consisting of all polygonal obstacle boundary edges, or an edge between any 2 vertices in V that lies entirely in free space except for its endpoints. Intuitively, if you place yourself at a vertex, you create an edge to any other vertex you can see (i.e. is visible).

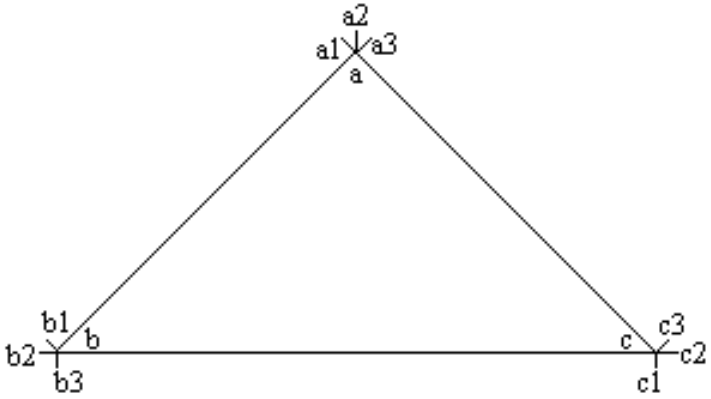
•

• The shortest path in distance can be found by searching the graph G using the shortest path search (Dijkstra's Algorithm) or other heuristic search method

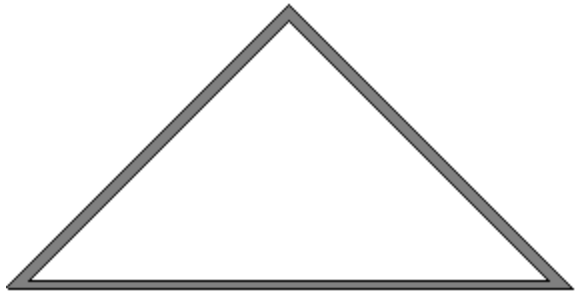
Computing the Configuration Space

- For the simple case of a circular robot





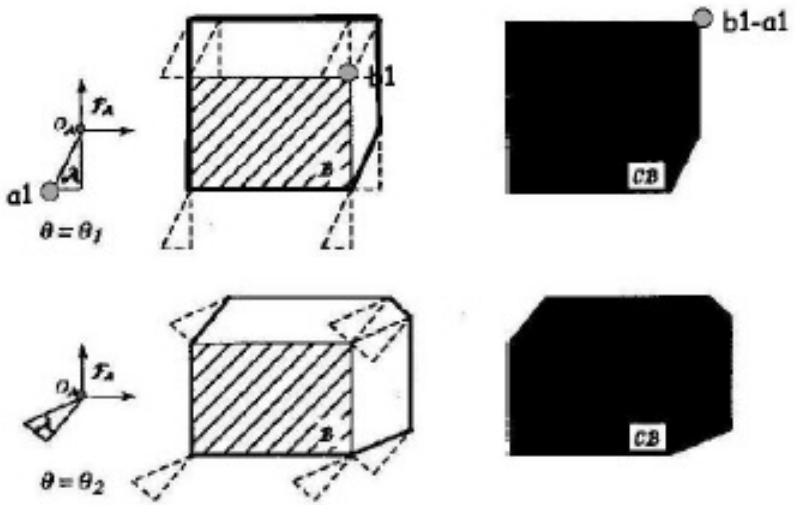
Points for the convex hull

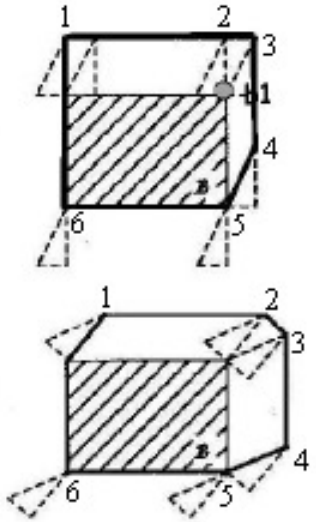


Simplification that you would use for your algorithms

Computing the Configuration Space (contd.)

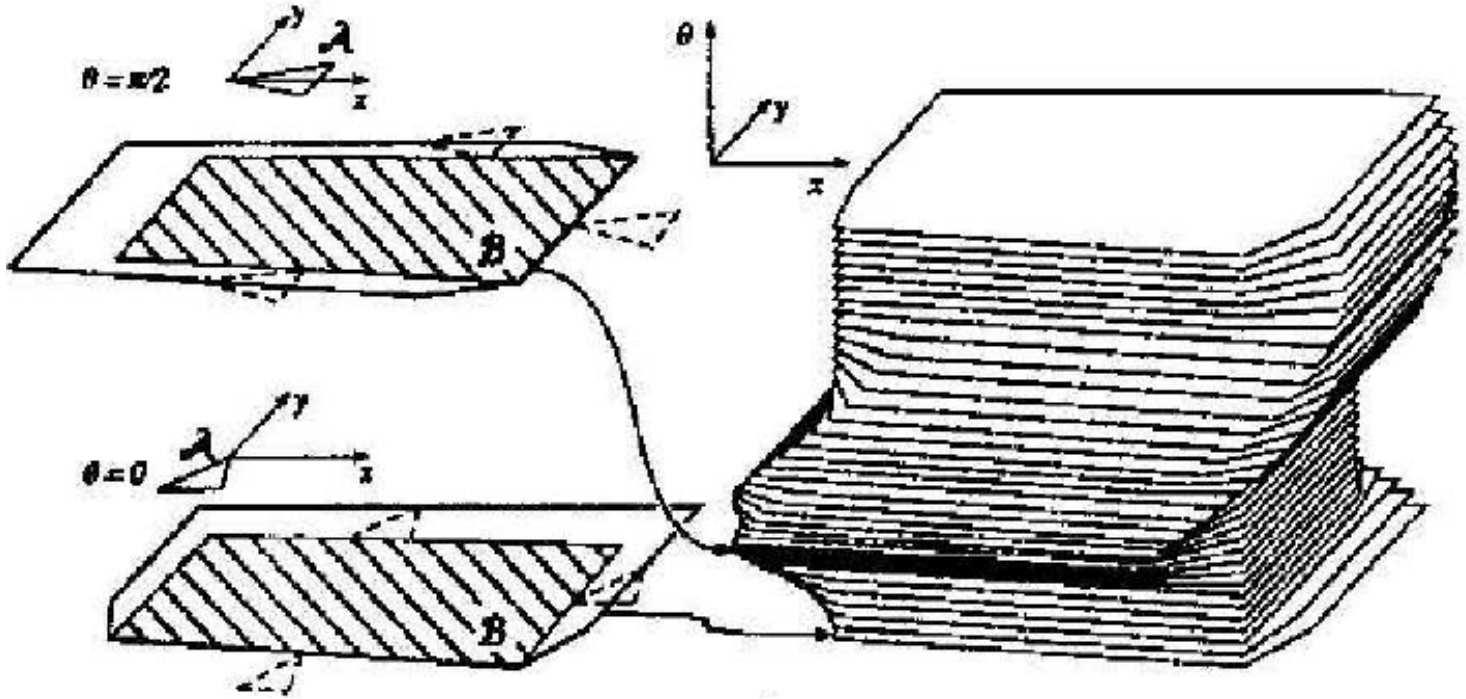
- For a convex polygonal robot



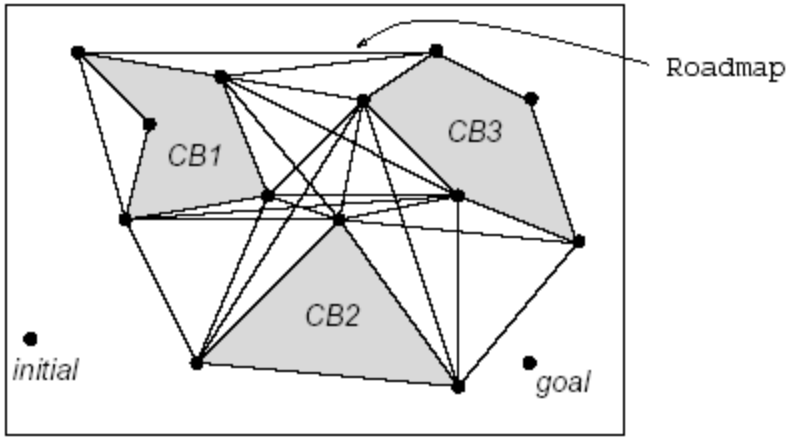


Enumerating the vertices

Problems with CS approach

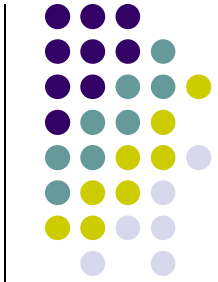


Constructing the Visibility Graph



The visibility graph is an undirected graph G = (V,E) where the V is the set of vertices of the grown obstacles plus the start and goal points, and E is a set of edges consisting of all polygonal obstacle boundary edges, or an edge between any 2 vertices in V that lies entirely in free space except for its endpoints. Intuitively, if you place yourself at a vertex, you create an edge to any other vertex you can see (i.e. is visible).

An Algorithm for Visibility Graph



Brute Force Method:

A simple algorithm to compute G is the following. Assume all N vertices of the G are connected. This forms N(N-1)/2 edges. Now check each edge to see if it intersects (excepting its endpoints) any other edge in the graph. If so reject this edge. The remaining edges are the edges of the visibility graph. This algorithm is brute force and slow but simple to compute. Faster algorithms are known

Fact:

If the initial and final locations of the robot are vertices on the G, then shortest path is contained on the visibility graph. In other words if a computation between S and T involves the visibility graph then it needs to be the shortest path as long as S and T are connected to the vertices of G by straight line segments