[Motion Planning]
Computational Geometry (CSc 635)

Jagdish Bhatta

Central Department of Computer Science & Information Technology
Tribhuvan University

Motion Planning/ Path Planning

The field of robotics has inspired the exploration of a collection of problems in computational geometry, called motion planning problems. The motion planning problem has to be solved whenever any kind of robot wants to move in physical space. In general, the robot has to be able to plan its own motion, for this certain information about the environment in which it is moving is provided by the floor plan that consist of layout of obstacles using the information about the environment, the robot has to move to its goal position without colliding with any of the obstacles.

The motion planning algorithms can also be applied to wire routing on chips, to planning path in GIS (Geographic information systems), to virtual navigation in computers.

Problem Specification:

Assume a fixed environment of impenetrable obstacles, usually polygons in 2D (Polyhedra in 3D) within this environment, consider a robot R, a movable object with some geometric characteristics; it may be a point, a line segment, a convex polygon etc.

- The robot R is at some initial position S (start)
- The task is to plan motions that will move R to some specified final position t (terminus), such that throughout the motion, collision between the robot and all obstacles is avoided.

✡ Collision: collision occurs when a point of the robot coincides with an interior point of an obstacle. The sliding contact with the boundary of the obstacles does not constitute a collision.

✡ A free path: A collision avoiding path is called a free path.

Within these general classes of problems, we have three specific questions as;

- Decision question: Does there exists a free path for R from S to +.
- Path Construction: Find a free path for R from S to +
- Shortest Path: Find the shortest path for R from S to t.

Here, each question is asking more information that proceeding one

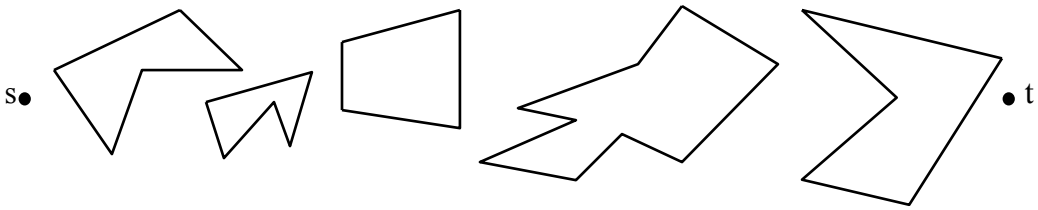
- A solution to question (3) solves (2), which in turn solves 1.

Path planning Problem (PPP):

Given a collection of disjoint polygons representing obstacles, a start point s and a terminus point t now the problem is;

Find a shortest collision free path from s to t.

An instance of such problem can be shown as;



Visibility Graph:

A visibility graph constructed between n-nodes is a graph, in which there is an edge between two nodes if and only if they are visible to each other. Two nodes are said to be visible to each other if the line segment connecting them does not intersect any obstacle.

Thus if,

$V = \{ v/v$ is either vertex of obstacles or point s or t$\}$

$E = \{(v_i, v_j)/v_i, v_j \in V$ and visible $(v_i, v_j) = true\}$

Then graph G (V, E) is a visibility graph induced by the obstacle set. The edges of obstacles are also included in the graph. Finding visibility graph is nearly identical to finding polygon diagonals, the only differences are inessential external diagonals. So, we have to consider the exterior visibility instead of interior.

A naïve approach is; for each vertex x & for each vertex y, check xy against every edge. This takes O (n) time. There is O (n^3) such edges. So, total complexity turns out to be O (n^3)

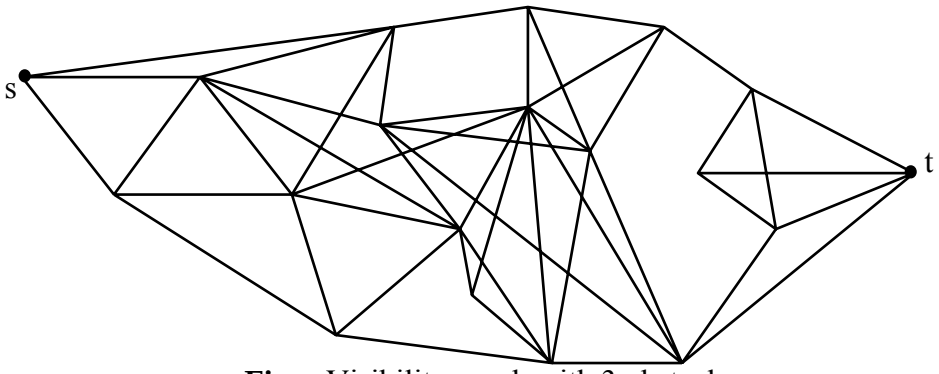


Fig: - Visibility graph with 3 obstacles

Thus, once we have the visibility graph, we can compute the shortest path for the robot from its initial position s, to the final position t, which is collision free by using the Dijkstra's shortest path algorithm.

Lemma: A shortest collision free path is contained in the visibility graph of the obstacle polygons (A shortest path is a sub path of the visibility graph of the vertices of the obstacle polygons)

Proof: -

Following points are sufficient to justify this lemma;

- As visibility graph consists of vertices of all obstacles & arcs between two vertices that are visible to each other. Thus, visibility is not considered if the line segments joining two vertices intersect the interior of polygonal obstacle. Thus the path will be free path.
- The shortest path consists of straight line segments. Clearly, the path along boundary of obstacle polygon can not be curved as polygon boundaries are straight line segments. There may be some curved subsection of the path that does not touch any polygon. Such curved path can be made shorter by straight lines (by geometry)
- The turning points of the path are polygon vertices i.e. the vertices of shortest path connecting s to t must be vertices of polygonal obstacles.
- If the shortest collision free path touches the obstacle then it must be such that the entry point & exit point of the shortest path with the obstacle must be vertices of obstacle and between these points the path is edge of the obstacle.

Thus, the visibility graph consist s, t & all the vertices of obstacles, from above three facts it shows that shortest collision free path is contained in the visibility graph of the obstacle polygons.

Algorithm for computing shortest collision free path:

To compute the shortest path from s to t, we have following steps;

- Construct visibility graph induced by polygonal obstacles

- Each edge of visibility graph is assigned weight equal to the Euclidean distance between the vertices of the edge.
- Now, apply the Dijkstra's shortest path algorithm to compute the shortest path in the visibility graph.

Analysis:

- Complexity visibility graph, it takes O (n^3) as O (n^2) edges are examined for visibility O (n) time.
- Assigning weight to each edge it requires O (n^2) as there are at most O (n^2) edges.
- Dijkstra's shortest path algorithm takes O (n^2) time.

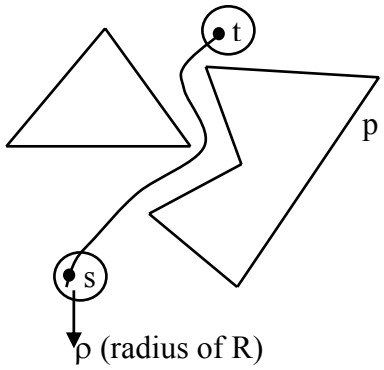
  Thus, total complexity for finding shortest collision free path is O (n^3).

  (But using sophisticated data-structures & proper arrangement of visibility graph, this algorithm can be reduced to O (n^2).)

Motion of Disk Robot:

If the robot is disc rather than point then the case is much different;

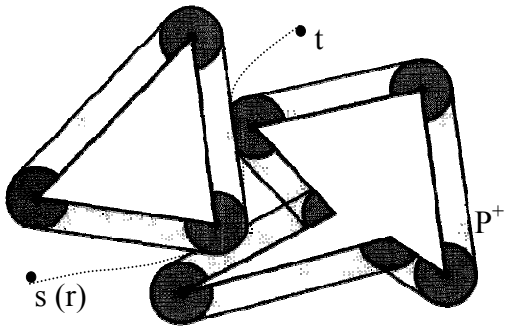
- Suppose a robot R is a disc centered at s, the start point. Now the goal is to move R so that it becomes centered at t, the terminus point. During its motion it should never penetrate any obstacle.



Here, in this figure, the path shown couldn't be a free path for the robot R because the robot is too wide to fit through the gap between the obstacles. To view whether the path is free or not, let us assume r be the reference point at the center of the disk R. clearly, r can't get closer to any polygon p than the radius of R, say ρ.

Thus, we can expand the obstacle p by radius "ρ" of R and shrink the disk robot to a point robot, thereby reducing the problem to moving a point among obstacles. Let p+ be the grown obstacle set.

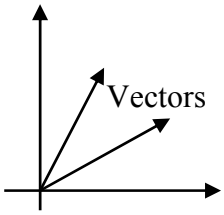
After growing the obstacles, if r stays outside of P^+, then R will not intersect p, so there is a collision free path for R. if r is inside P^+, R must intersect p.



Here, we don't have the free path as the grown obstacles are overlapping so moving point robot r will lie inside P^+

Minkowski Sum:

Let A & B be the two sets of points in the plane. If we establish the coordinate system then the points can be viewed as vectors in that coordinate system.



The Minkowski sum of the point sets A & B is defined as;

$A \oplus B = \{x+y/\ x \in A \ \& \ y \in B\}$

where, x+y is the vector sum of two points x & y.

This sum is also known as point wise sum of A & B.

Similarly the Minkowski sum of a point x and point set B can be defined as;

$x \oplus B = \{x+y/y \in B\}$.

This sum $x \oplus B$ is just a copy of B translated by the vector x for each point y of set B. (i.e. each y of B is moved by x).

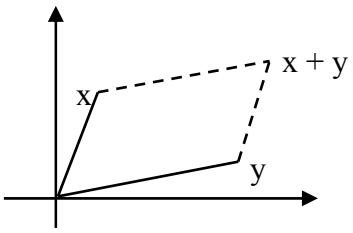


Fig: - Minkowski sum of two points x & y

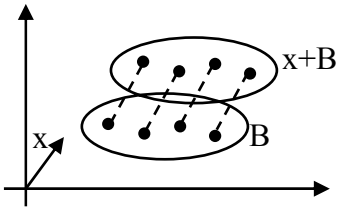


Fig: - Minkowski sum of a point x & a point set B

So, we can define Minkowski sum of two point sets A & B as the union of copies B translated by every x∈A

i.e. $A \oplus B = \bigcup_{x \in A} x \oplus B$

Conceptual Algorithm for moving a disk robot:

This conceptual algorithm is based on the Minkowski sum discussed in previous section.

Algoritm:

Let p_1, p_2, ----------, p_n are obstacles & let R is a disc robot.

Step 1: Grow every obstacle by the radius of the disc robot R, constructing the Minkowski sum with R.

i.e., $p_i^+ = p_i \oplus R$ for each pi in P.

Step 2: Form the union of grown obstacles,

i.e. $p^+ = \bigcup_i Pi^+$

Step 3: if the terminus point t (destination) is in different component of the plane than the start s, then there is no free path from s to t.

If s & t are in the same component of the plane, then there is a free path and shortest path can be found by modifying the visibility graph to include the appropriate arcs of circles. The time complexity of this algorithm is $O(n^2 \log n)$.

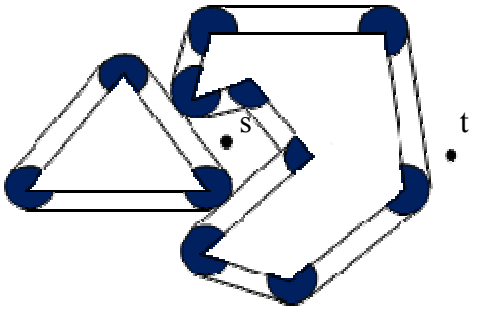


Fig: - s & t are in different component of plane
So no free path exists from s to t

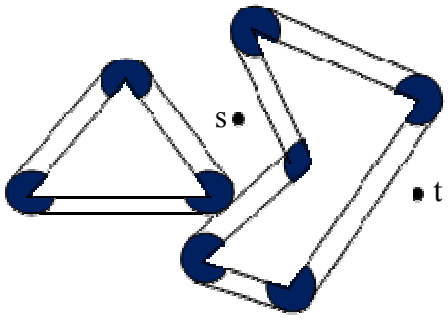
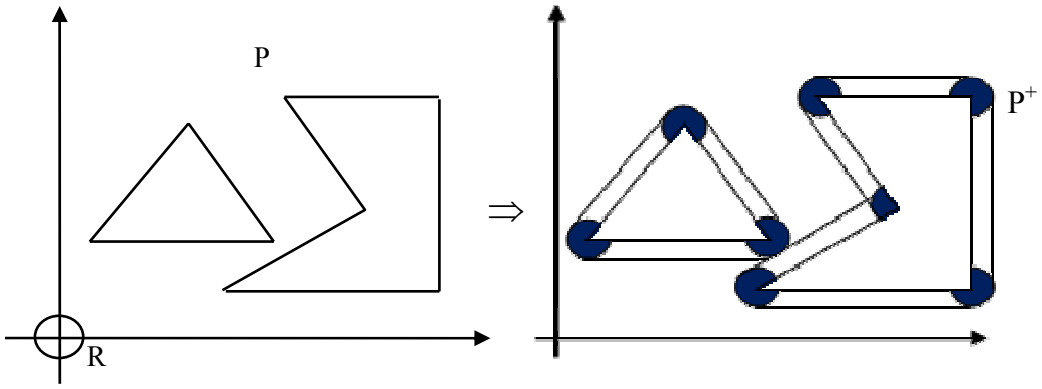


Fig: - s & t are in same component
So free path between s & t exists

Use of Minkowski Sum for growing the obstacles while moving a disk Robot:

By using the Minkowski sum, we can obtain the grown obstacles for moving a disc robot R.

Let A is a polygon set, (obstacles) p and B is a disc robot R centered at origin. Then the Minkowski sum p⊕R can be viewed as many copies of R translated by x for all x∈p.



Since, R is centered at the origin, x⊕R will be centered at x. so P⊕R amounts to placing a copy of R centered on each x∈P. here we take each x, for every vertices of polygons. So P contains all vertices of polygonal obstacles. Thus, with P⊕R, the robot R will be centered at each vertex of obstacles. Assume the tangents to each R at every vertex x∈P, we can obtain the obstacles growing by the radius of R as shown in figure above.

Hence, P⊕R is results the expanded obstacle region P^+.

Translating a convex polygon:

When a robot is convex polygon, then along with the Minkowski sum, we need some complicated transformations of robot. Consider, robot is a square, choose the reference point r to be the lower left corner of the square and let p be a polygonal obstacle.

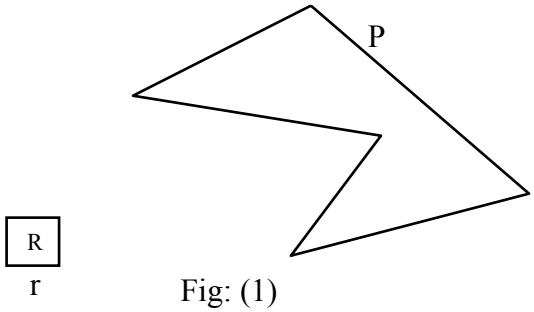


Fig: (1)

As R moves around the boundary of P, r traces out the boundary of P^+, the grown obstacle, i.e. expanded P, that r cannot penetrate. Here the situation is somewhat different from that with a disk, since we choose the reference point at a corner of R: P grows by different amount along each edge.

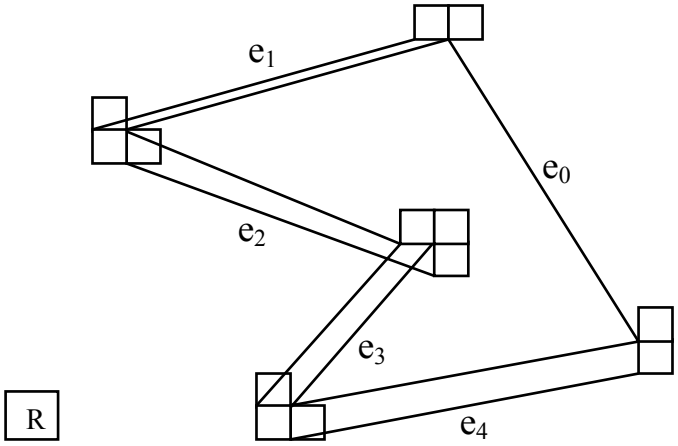


Fig: (2)

In the figure (2), growing of P along e_0 is same in P & P^+ i.e. P & P^+ match along e_0. The offset of P^+ from edges e1 & e4 of P is different; the width of R horizontally from e1, and the height of R vertically from e4. Also at the reflex vertex, r traces out the self crossing path.

Use of Minkowski Sum for Translating Convex Polygon:

Consider the square robot R with its reference point r at its lower left corner. Also consider the polygonal obstacle P. The Minkowski sum $P^+ = P \oplus R$ which works for disk motion does not work for square (by above discussion). The appropriate computation is rather to take Minkowski sum of P with a reflection of R through the reference point r. since r is the origin for the purpose of the Minkowski sum formulation, the reflection of R is simply –R, where every point of R is negated so each point $p \in R$ has an effect of holding r away from the boundary of P by –P.

Now, the Minkowski sum of P with the reflection of R is $P \oplus -R$ which produces P+, the grown obstacle as shown below in figure 3.

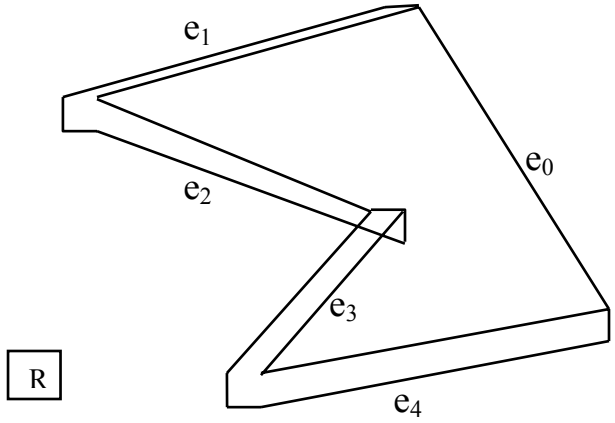


Fig: (3)

Thus, applying the p ⊕ -R, we can get the growing object as in the figure above, which is the exact requirement for growing so that reference point (r) does not penetrate the wall of obstacle.

Note: when R is disk, it is centrally symmetric about origin then R= -R. So this sum P^+ = P⊕ -R also works for motion of disk robot.

Algorithm:

Now, let us device an algorithm for planning a motion for a convex polygon. Let the obstacles be P_1, P_2, ------- P_m with total of n vertices. Now the algorithm is as;

(1) Grow all the obstacles using Minkowski sum as;

$P_i^+ = P_i \oplus -R$

$P^+ = \bigcup_i Pi$

(2) Find the component containing s & t. (start & terminus position)

(3) If s & t are in different component of the plane, path between s & t does not exist. Else find path between s & t in the component.

This algorithm also works for disk motion as R = -R. But for disc, the translation from R to –R is not required. So the algorithm is as;

- Grow all obstacles $P_i^+ = P_i \oplus R$ (R = -R) & shrink R into a point.
  Now, $P^+ = \bigcup_i Pi^+$
- Find the component containing start position s & terminus position t.
- If s & t are in different component of the plane, path between s & t does not exist. Otherwise find the path between s & t in that component modifying the visibility graph.