

OBSTACLE AVOIDANCE

Obstacle avoidance, or collision avoidance, is the problem of ensuring that the manipulator(s) don't collide with objects or each other.

There can be both hardware and software solutions: hardware solutions are restrictive, and software solutions are still an active area of robotics research. We'll only deal with software solutions.

Some robot tasks allow the environment to be ordered in such a way as to make collisions impossible e.g. pick and place tasks. All non-manipulated objects are placed outside the workspace, all robot paths are planned in advance, and all manipulated objects are positioned very precisely. Such a system is incapable of responding to foreign objects or errors.

The collision avoidance problem may be divided into two major subtasks:

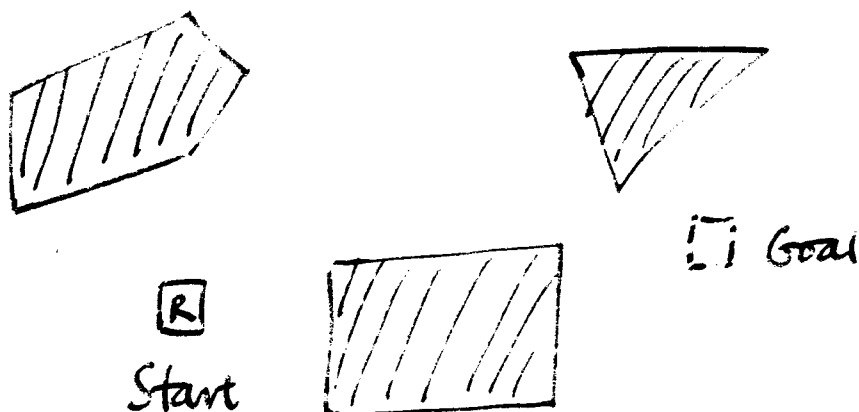
- path planning
- trajectory control.

We'll consider path planning first. The problem is to find an algorithm that will

- find safe paths through a field of obstacles
- guarantee that these paths are short with respect to some prespecified metric, eg., shortest distance, shortest time, least power etc.

We'll start with the following assumptions:

- the field of obstacles is known in advance
- the robot has no additional information pertaining to foreign obstacles or errors in positioning
- both the obstacles and the robot will be modelled as polyhedra, and the robot is assumed to be mobile



eg. robot moving through a room with furniture.

Even under these assumptions it can be shown that the problem is NP hard and so further constraints are usually imposed to find efficient algorithms in specific cases.

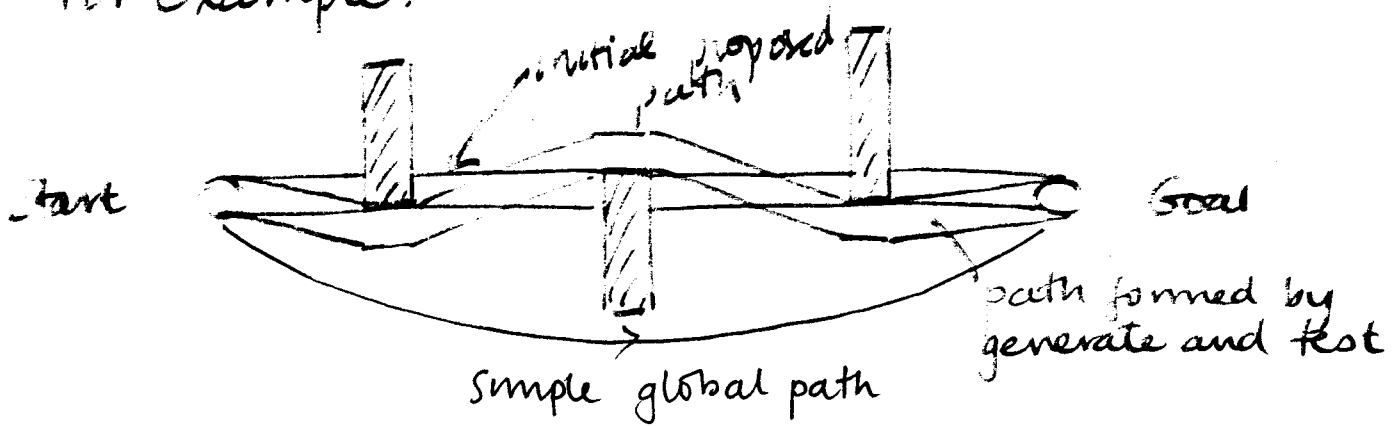
The first approach was a generate and test idea, known as the swept volume method. The algorithms consisted of three steps:

1. calculate the volume swept out by the moving object along a proposed path
2. determine the intersection between the swept volume and the obstacles
3. if $\neq \emptyset$, propose a new path and goto 1.

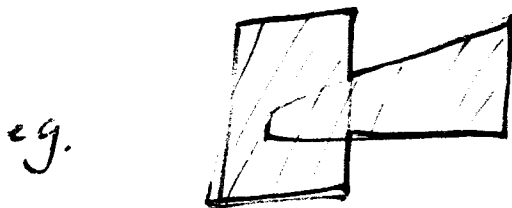
This approach has a number of problems. Firstly, step 2) is computationally expensive and very difficult for general obstacle shapes. Even when obstacles and robot are modelled by simple shapes, the total geometry of the scene can be complicated.

Secondly, each proposed path provides only local information about potential collisions hence only suggesting local path changes. A more global view is often necessary.

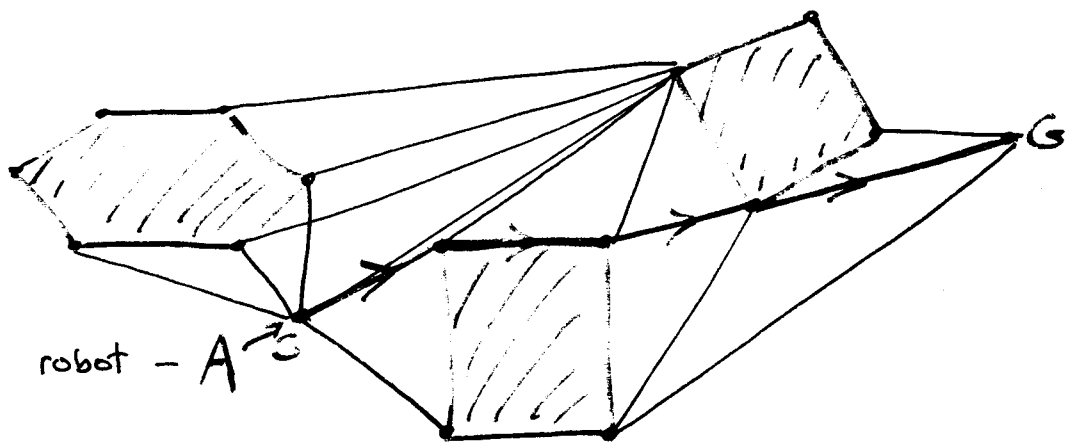
For example:



An alternative approach is to pose the problem as an optimisation problem by explicitly stating the constraints on the position of the robot caused by the obstacles. If the robot and obstacles are modelled as convex polyhedra these constraints can be stated in terms of the vertices of the objects (in 2D). Non convex objects are modelled by overlapping convex polyhedra:



The geometric problem is then converted to a graph theoretic problem and the solution is to find the shortest path through the graph.



We construct an undirected graph $VG(N, L)$, called the visibility graph, with nodes N and links L as follows:

Let $V =$ set of all vertices of the obstacles.

Then $N = V \cup \{S, G\}$

$L =$ set of straight lines joining nodes which don't overlap obstacles

That is, $(n_i, n_j) \in L$ if there is a line between nodes n_i and n_j not intersecting obstacles.

Here we find the shortest path using the Euclidean metric on the links. The simplicity of the scheme comes from modelling A as a point.

The simplest generalization of this scheme is to assume A is rotationally symmetric, modelled as a disc of radius r . Now the geometry of A can affect the obstacles and hence the path.