3.4.4 (Path planning

1. Introduction to the path planning module

Path planning is one of the essential functions of autonomous mobile robots, which given the ability of a robot to find the shortest path between two points. As for the ICRA, the priority is to find out the path between robot and destination then get there as soon as possible, so we decided to use some of the classical path-planning algorithms this time.

1. Introduction to algorithms

We have two types of the path planning algorithms are required on the robot path planning: global and local and two alternative algorithms that we are still not determined which one we are going to use for the global planning:

One of them is called A\* algorithm; A\* is an informed search algorithm that widely used in pathfinding and graph traversal, it finding the path by maintaining a tree of paths originating at the start node and extending those paths one edge at the time until its termination criterion is satisfied. The algorithm determines which of its paths to extend based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal; then it selects the path that minimises.

Equation:

f(n)=g(n)+h(n)

where n is the next node on the path, g(n) is the cost of the path from the start

node to n and h(n) is a heuristic function that estimates the cost of the cheapest path from n to the goal.

Pros of the algorithm: The obstacles and illegal terrains are highlights to reduce the unnecessary exploration, hence, increase the processing speed.

The other one is called the RRT algorithm, RRT(rapidly-exploring random tree) is an algorithm designed to search high-dimensional spaces by randomly building a space-filling tree efficiently. An RRT randomly grows tree roots at the starting configuration; if the connection is feasible, this also results of the new state to the tree. The first path who reached the goal be selected.

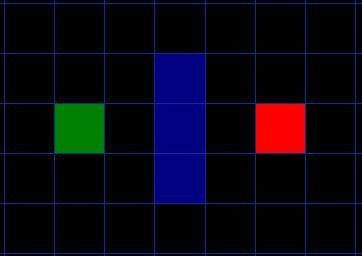
Pros of the algorithm: There is no need to adjust specific parameters for RRT, the exploring speed of RRT is faster than the A\* when facing an unknown environment.

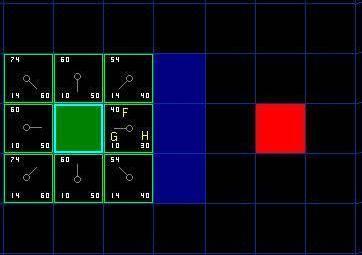
Cons of the algorithm: RRT is a purely random search algorithm that is not sensitive to the environment, so when there are a large number of obstacles or narrow channel constraints exist in the environment, the algorithm could converge slowly, which means the performance drops dramatically.

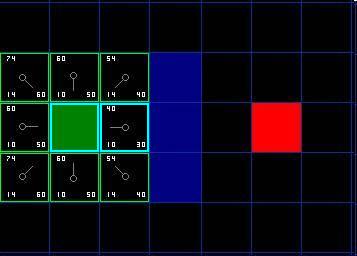
We were using the TEB local planner for local planning, TEB local planner able to optimize the robot trajectory online and produce alternative trajectories in distinctive topologies, moreover, it supports forward and backwards driving of robot, consider the mercurial environment of the battlefield(stage) the TEB would be the best choice for local planning.

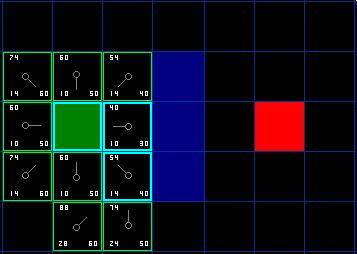
1. Planning processes

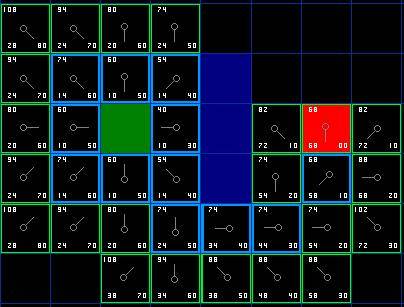
A\* algorithm:











Assume that point A(green) wants to get to point B(red) and there is a wall(blue) separates the two points.

Step1: algorithm add point A into an ‘open list'.

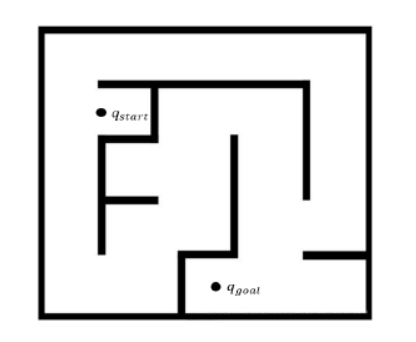
Step2: it starts to look at all the reachable squares adjacent to the point inside the open list, ignoring squares with illegal terrain like walls. Add them to the open list too. For each of these squares, save point A as ‘parent square' which be used to trace our path later.

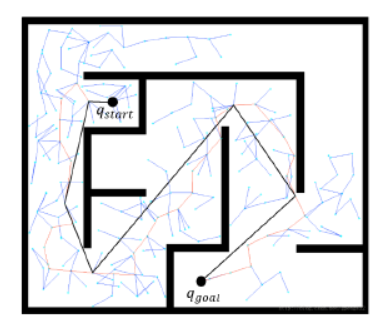
Step3: algorithm move the starting square(point A) from the open list to ‘close list' after reachable squares were added into open list and start to determine which square to use when figuring out the path using the equation: f(n)=g(n)+h(n)

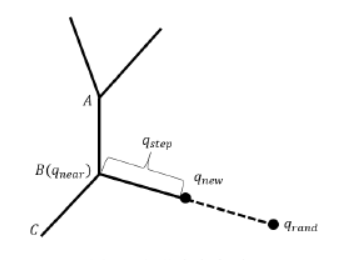
Where G is the movement cost to move from the given location to the goal square on the grid, following the path generated to get there. H is the estimated movement cost to move from the given square on the gird to the final destination.

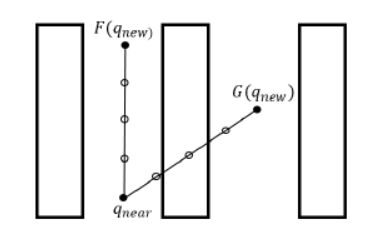
Choice the square with the smallest f index and repeat step 1,2,3 until the final goal is inside an open list.

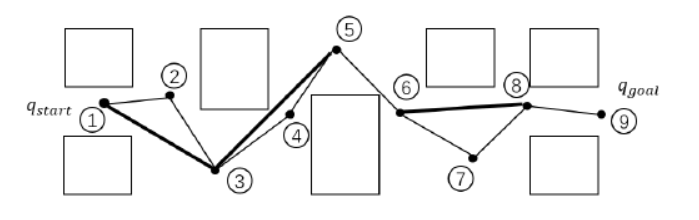
RRT algorithm:











Assume we that q start wants to get the q goad, as we can see the path has created by RRT algorithm. 1)The algorithm randomly creates a point record as q rand; then it finds out the nearest node of the tree to the q rand and records it as q near. Connect q near and q new with a line, let a point on this line be q new, the distance between q near and q new record as q step. 2)Check whether there is obstacle exist between q near and q new. If there is no obstacle exists, add q new and the line as a node and branch separately into the tree, otherwise, skip the loop. 3)After it finally reaches the q goal, the algorithm starts the traversal for optimisation, generating the shortest path.