Review, Comparison On Different Path Planning Sample Based Algorithm

By:- *Nayan Madhav Sarode*

Submitting To:- *QSTP Robotic Automation 2020*

Rapidly Exploring Random Tree:-

The concept of RRT is actually quite straight forward. Points are randomly generated and connected to the closest available node. Each time a vertex is created, a check must be made that the vertex lies outside of an obstacle. Furthermore, chaining the vertex to its closest neighbour must also avoid obstacles. The algorithm ends when a node is generated within the goal region, or a limit is hit. RRT is a developed algorithm on which fast continuous domain path planners can be based. This search algorithm finds path in high dimension cluttered environments. The RRT is a probabilistic sampling based approach, and the generated random points should lead towards the goal. Samples are generated such that the 20% to 25% of the generated random samples should be goal itself. This will lead the tree towards the goal. RRT may get stuck in much cluttered environments. Thus an iteration limit is given so that it does not go in an infinite loop. Once the iteration limit is reached it comes out of the loop and the path is shown irrespective of whether the goal point is reached or not. Also this method is not useful in case of moving targets, because the position of the object is changing and it may intersect with the path.

```
RRT Pseudo Code
Qgoal //region that identifies success
Counter = 0 //keeps track of iterations
lim = n //number of iterations algorithm should run for
G(V,E) //Graph containing edges and vertices, initialized as
empty
While counter < lim:
    Xnew  = RandomPosition()
    if IsInObstacle(Xnew) == True:
        continue
    Xnearest = Nearest(G(V,E),Xnew) //find nearest vertex
    Link = Chain(Xnew,Xnearest)
    G.append(Link)
    if Xnew in Qgoal:
        Return G
Return G
```

Rapidly Exploring Random Tree*:-

RRT* is a modified version of RRT. When the number of nodes approaches infinity, the RRT* algorithm will deliver the shortest possible path to the goal. While realistically unfeasible, this statement suggests that the algorithm does work to develop a shortest path. The basic principle of RRT* is the same as RRT, but two key additions to the algorithm result in significantly different results.

First, RRT* records the distance each vertex has travelled relative to its parent vertex. This is referred to as the cost () of the vertex. After the closest node is found in the graph, neighbourhoods of vertices in a fixed radius from the new node are examined. If a node with a cheaper cost() than the proximal node is found, the cheaper node replaces the proximal node. The second difference RRT* adds is the renovating of the tree. After a vertex has been connected to the cheapest neighbour, the neighbours are again examined. Neighbours are checked if being renovated to the newly added vertex will make their cost decrease. If the cost does indeed decrease, the neighbour is rewired to the newly added vertex. This feature makes the path smoother.

```
RRT* Pseudo Code
Rad = r
G(V,E) //Graph containing edges and vertices
For itr in range(0…n)
    Xnew = RandomPosition()
    If Obstacle(Xnew) == True, try again
    Xnearest = Nearest(G(V,E),Xnew)
    Cost(Xnew) = Distance(Xnew,Xnearest)
    Xbest,Xneighbors = findNeighbors(G(V,E),Xnew,Rad)
    Link = Chain(Xnew,Xbest)
    For x' in Xneighbors
        If Cost(Xnew) + Distance(Xnew,x') < Cost(x')
            Cost(x') = Cost(Xnew)+Distance(Xnew,x')
            Parent(x') = Xnew
            G += {Xnew,x'}
    G += Link
Return G
```

Rapidly exploring random trees (RRT) and probabilistic roadmaps (PRM) are sampling-based techniques being extensively used for robot path planning.

Rapidly Exploring Random Graph:-

It is similar to RRT the random points are plotted on graph by connecting to nearer nodes. The graph is used to represent the random point which uses shorter distance between the neighbouring nodes. But RRG is not mostly preferred than RRT as in gives more straight line path with less point over the obstacles.

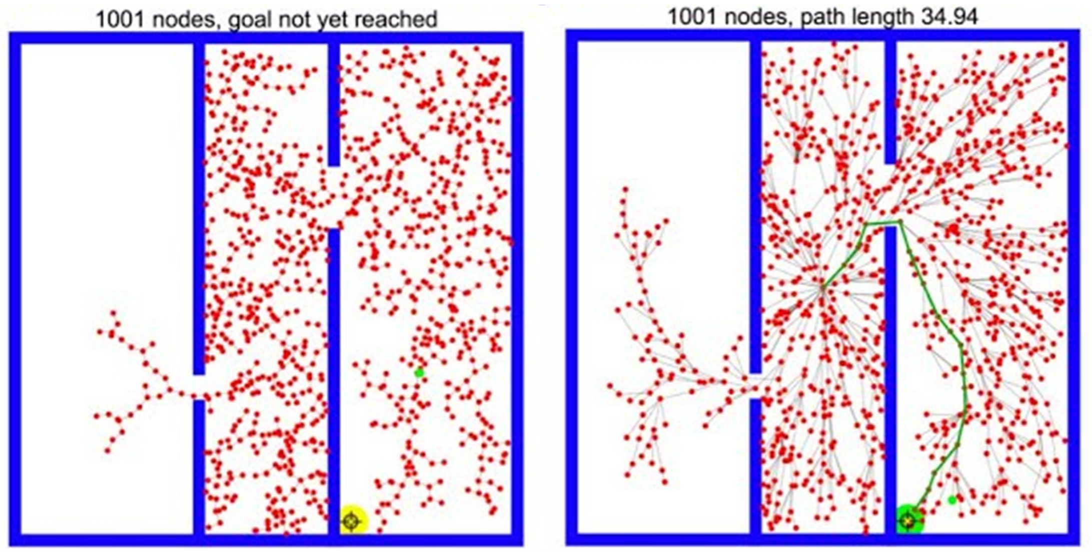
Probabilistic Road Map:-

PRM, similarly to RRT, generates a limited number of random points within a given area. There are numerous methods of generating random points. While decent results are produced, the biases of the random generator are fairly apparent in the resulting PRM graphs. After a new node is generated, the PRM algorithm clusters nodes into connected components. The connected component the node belongs to needs to be stored. To perform clustering, all nodes within a fixed radius of the randomly generated node are collected. These neighbouring nodes are ordered by increasing distance (or the desired metric). Looping through the ordered neighbouring nodes, check if the randomly generated node is not in the same cluster as the examined node. The new node is added to the cluster if the condition is satisfied. A node can, and should, belong to multiple clusters.

PRM Pseudo Code

```
G(V,E) = Null //Initialize a graph as empty
limit = n //number of nodes to make graph out of
Rad = r //radius of neighborhoods
For itr in 0...limit:
    Xnew = RandomPosition()
    Xnearest = Near(G(V,E),Xnew,Rad) //find all nodes within a
Rad
    Xnearest = sort(Xnearest) //sort by increasing distance
    For node in Xnearest:
        if not ConnectedComp(Xnew,node) and not
Obstacle(Xnew,node):
            G(V,E) += {Xnew,node} //add edge and node to graph
            Xnew.comp += node.comp//add Xnew to connected
component
Return G(V,E)
```

The dimension of the radius is given by the user and its value defines the structure of the roadmap. Some of the formed clusters will from circular blobs that have this radius. The chosen radius will also impact speed and performance of the generated roadmap. A larger radius will involve parsing through more neighbours and determining their cluster relationship. Another parameter the user determines is the number of nodes. The PRM algorithm ends once the number of desired nodes is generated. Having to generate too many nodes can prolong the time it takes to develop the roadmap. More time will be dedicated to examine regions of neighbours. Clusters, especially in high density obstacle regions, can become disconnected from the remainder of the road map. Paths generated on a sparsely formed road map can be more irregular as well. Fewer nodes decrease possible routes when using a shortest path planning algorithm.


1001 nodes, goal not yet reached
1001 nodes, path length 34.94

Conclusion:-

It can be concluded that for robotic path planning algorithm Rapidly Exploring Random Tree (RRT) and Probabilistic Road Method (PRM) is mostly preferred algorithm as they give maximum straight line path by connecting to nearest nodes and less error of random point being plotted on obstacles. For RRT points are being examined to see that if nodes are away from obstacles. Also more the number of points/nodes more are accuracy in path forming. RRT* is modified form of RRT with less number of nodes.