ENPM661

Planning of Autonomous Systems

Final Project

Triangular-Geometrized RRT* (TG-RRT*)

ADITYA VARADARAJ

(UID: 117054859)

SAURABH PALANDE

(UID: 118133959)

(SECTION: 0101)

M.Eng. Robotics (PMRO),

University of Maryland - College Park, College Park, MD - 20742, USA.

Date of Submission: 8th May, 2022

I. Introduction

In the effort of finding efficient solutions for practical motion planning problems, extensive research has been devoted to exploring incremental sampling based algorithms, like Rapidly-exploring Random Trees (RRT) and the Probabilistic Road Map (PRM). These probabilisticaly complete algorithms generate a solution, provided that one exists, irrespective of an obstacle's geometry in the configuration space. In these algorithms, the probability of finding a path solution in a given environment is one, if number of iterations utilized are allowed to approach infinity. Rapidly exploring random trees quickly find their initial path in a given problem, and due to their effectiveness, have been improved in number of ways.

RRT* is an improved version of the **RRT** algorithm which rewires the graph and keeps track of the cost. While RRT concentrates on simply finding an initial obstacle-free path, RRT* guarantees eventual convergence to an optimum, collision-free path for any given geometrical environment. On the other hand, the main limitations of RRT* include its slow processing rate and high memory utilization due to the large number of iterations required to achieve optimal path solution.

The method proposed in the primary reference paper [1] (**TG-RRT***) uses geometrical centres of triangle of start, goal and random sample to reduce the dispersion of random samples by bringing them towards the triangle formed by start, goal and random nodes.

Aims:

- To improve performance and reeduce computational time of RRT* using Triangular Geometry
- Use Geometrical Centres like Centroid, Incenter, Orthocenter and Circumcenter.

II. Background and Related Work

A. RRT and RRT*

The **RRT* [2]** algorithm uses **Rewiring** concept based on cost of each vertex in the tree generated by **RRT**. The cost of the best path in the **RRT** converges almost surely to a **non-optimal value**. **RRT*** mainitains the asymptotic optimality of the **RRG [?]** algorithm while preserving the **tree structure** of **RRT**.

The algorithm of **RRT*** is shown in algorithm 1 and 2. The way **RRT*** works is that it samples random points x from the free space and then finds the nearest point $x_{nearest}$ in graph by **Nearest** function, it finds the new point x_{new} by steering from $x_{nearest}$ towards x using the actionset and **Steer** function. Then if edge $(x_{nearest}, x_{new})$ is Obstacle Free, then it adds the vertex x_{new} to the vertices of the graph. Next, it uses **Near** function to find the set of all vertices X_{near} in a ball of radius $r_n = min\left\{\gamma\left(\frac{\log n}{n}\right)^{1/d}, \eta\right\}$, where, γ is a constant, η is the maximum allowable distance between 2 vertices which are hyperparameters. $\Re^d$ is the space which the graph belongs to according to vertex (E.g.: if vertex is (x, y, θ), $d = 3$). Then it finds the vertex x_{min} in X_{near} that gives the minimum total cost (cost-to-come to x_{near} + cost of line from x_{near} to x_{new}). Then, it adds the edge (x_{min}, x_{new}) to the edge set of the graph. Then, rewiring is performed if the sum of the cost to x_{new} and cost from x_{new} to x_{near} is less than current cost to x_{near}.

Algorithm 1 General RRT*/RRT/TG-RRT* Algorithm Body

$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$; $i \leftarrow 0$
while $i < N$ **do**
  $G \leftarrow (V, E)$;
  $x_{rand} \leftarrow$ SAMPLE(i); $i \leftarrow i + 1$;
  $(V, E) \leftarrow$ EXTEND(G, x_{rand});

Algorithm 2 EXTEND$_{RRT*}$ (G, x)

$V' \leftarrow V; E' \leftarrow E;$
$x_{nearest} \leftarrow$ NEAREST$(G, x);$
$x_{new} \leftarrow$ STEER$(x_{nearest}, x);$
if OBSTACLEFREE$(x_{nearest}, x_{new})$ **then**
  $V' \leftarrow V' \cup \{x_{new}\};$
  $x_{min} \leftarrow x_{nearest};$
  $X_{near} \leftarrow$ NEAR$(G, x_{new}, |V|);$
  for all $x_{near} \in X_{near}$ **do**
    if OBSACLEFREE(x_{new}, x_{near}) **then**
      $c' \leftarrow Cost(x_{near}) + c(Line(x_{near}, x_{new}));$
      if $c' < Cost(x_{new})$ **then**
        $x_{min} \leftarrow x_{near};$
  $E' \leftarrow E' \cup \{(x_{min}, x_{new})\};$
  for all $x_{near} \in X_{near} \setminus \{x_{min}\}$ **do**
    if OBSTACLEFREE(x_{new}, x_{near}) **and** $Cost(x_{near}) > Cost(x_{new}) + c(Line(x_{new}, x_{near}))$ **then**
      $x_{parent} \leftarrow$ PARENT$(x_{near});$
      $E' \leftarrow E' \setminus \{(x_{parent}, x_{near})\};$
      $E' \leftarrow E' \cup \{(x_{new}, x_{near})\};$
return $G' = (V', E')$

B. Triangular Geometry

There are four common triangular centres, namely Circumcentre, Incentre, Orthocentre and Centroid. For the purpose of this paper, all four methods were applied to RRT* and performance results in each case were separately investigated. In triangular geometrical methods, we use centres of the triangle between start, goal and random point. The results obtained indicated that out of the four methods used, only two types of centres, i.e, the incentre and centroid, were able to produce satisfactory results. The other two methods, when applied in a number of different environments, caused RRT* to take an infinite amount of time to locate the optimum path solution. This is due to the fact that Orthocentres and Circumcentres do not necessarily lie inside the triangle formed by the goal, root and random nodes. Therefore, there exists a possibility that these centres take the random sample away from the goal region instead of closer to it, resulting in increase in the execution time of the algorithm. The types of centres of triangle are as follows (shown in Fig.1):

- **Circumcentre:** The point where the perpendicular bisectors of all three sides of triangle intersect is called the Circumcentre of triangle.
- **Orthocentre:** This is a point where all three altitudes of a triangle meet. Altitude is the line passing through a vertex and perpendicular to the opposite side.
- **Centroid:** This is the point where medians of the triangle intersect. Medians are lines joining a vertex to the midpoint of the opposite side of the triangle. Centroid is also called as the centre of gravity.
- **Incentre:** A point inside the triangle where the angle bisectors intersect. This is essentially the centre of the incircle, the largest circle enclosed inside the triangle.

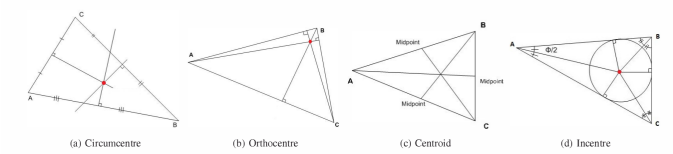

Fig. 1. Centres of the Triangle: (a) Circumcentre, (b) Orthocentre, (c) Centroid an (d) Incentre

III. METHODOLOGY: TRIANGULAR GEOMETRISED RRT* (TG-RRT*)

*A. TG-RRT**

Triangular Geometrised RRT* (TG-RRT*) algorithm further extends **RRT*** to include the geometrical

methods discussed above. In TG-RRT*, every time a random sample x is generated from the free configuration space, the geometrical methods discussed are used to find the geometrical centre of the triangle with start, goal and the random sample as vertices. Now, the geometrical centre is assigned to the new random sample variable x_{gc}. This is now treated as the random sample in Extend function of RRT*.

GeometricCentre: The function returns the geometric centre of the triangle and assigns it to new random sample x_{gc}. If $choice = 1$, normal RRT* is performed. If $choice = 2$, Incentre-based (IC-RRT*) is performed. If $choice = 3$, Centroid-based (C-RRT*) is performed.

Incentre-based RRT*: new random sample is the incentre of the traingle having start, goal and random sample as vertices. The formula for the same is given as follows:

$$\begin{aligned} c &= length(line(start, goal)) \\ a &= length(line(x, goal)),\ x\ is\ random\ sample \\ b &= length(line(start, x)) \\ start &\equiv\ point\ A(x_1, y_1),\ goal \equiv\ point\ B(x_2, y_2), \\ x &\equiv\ point\ C(x_3, y_3) \\ x_{incentre} &= \frac{ax_1 + bx_2 + cx_3}{a+b+c} \\ y_{incentre} &= \frac{ay_1 + by_2 + cy_3}{a+b+c} \end{aligned} \tag{1}$$

Centroid-based RRT*: new random sample is the centroid of the traingle having start, goal and random sample as vertices. The formula for the same is given as follows:

$$\begin{aligned} x_{centroid} &= \frac{x_{start} + x_{goal} + x}{3} \\ y_{centroid} &= \frac{y_{start} + y_{goal} + y}{3} \end{aligned} \tag{2}$$

We think that since centroid for any random sample in the workspace lies in the centre of the workspace, we probably need to change the sampling for centroid to sample x from around 3 times the dimensions of the workspace so that the new samples can cover the existing workspace.

The algorithm for **TG-RRT*** is given in Algorithm 3:

Algorithm 3 $\textsc{Extend}_{TG-RRT*}\ (G, x, choice)$

$V' \leftarrow V; E' \leftarrow E;$
$x_{gc} \leftarrow \textsc{GeometricCentre}_{choice}$
$x_{nearest} \leftarrow \textsc{Nearest}(G, x_{gc});$
$x_{new} \leftarrow \textsc{Steer}(x_{nearest}, x_{gc});$
if $\textsc{ObstacleFree}(x_{nearest}, x_{new})$ **then**
  $V' \leftarrow V' \cup \{x_{new}\};$
  $x_{min} \leftarrow x_{nearest};$
  $X_{near} \leftarrow \textsc{Near}(G, x_{new}, |V|);$
  for all $x_{near} \in X_{near}$ **do**
    if $\textsc{ObsacleFree}(x_{new}, x_{near})$ **then**
      $c' \leftarrow Cost(x_{near}) + c(Line(x_{near}, x_{new}));$
      if $c' < Cost(x_{new})$ **then**
        $x_{min} \leftarrow x_{near};$
  $E' \leftarrow E' \cup \{(x_{min}, x_{new})\};$
  for all $x_{near} \in X_{near} \setminus \{x_{min}\}$ **do**
    if $\textsc{ObstacleFree}(x_{new}, x_{near})$ **and** $Cost(x_{near}) > Cost(x_{new}) + c(Line(x_{new}, x_{near}))$ **then**
      $x_{parent} \leftarrow \textsc{Parent}(x_{near});$
      $E' \leftarrow E' \setminus \{(x_{parent}, x_{near})\};$
      $E' \leftarrow E' \cup \{(x_{new}, x_{near})\};$
return $G' = (V', E')$

B. Gazebo Map creation:

The dimensions of the obstacle space we created are shown in the Fig.2. Note that these dimensions multiplied by $\frac{3}{100}$ give the dimensions in Gazebo. The inflated obstacle space with $clearance = 5$ and radius of robot as $r = 10.5$ is shown in Fig.3.

We first opened a blank Gazebo environment, then we added walls and cylinders to the world and saved as .world file. Then, we edited the .world file to adjust

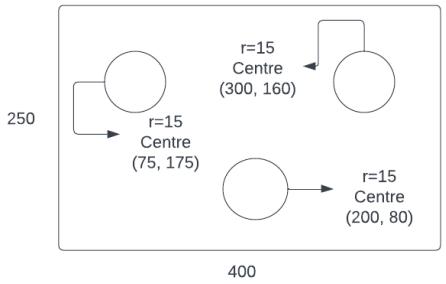

Fig. 2. Arena dimensions

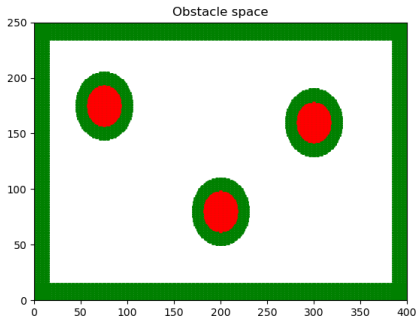

Fig. 3. Inflated Obstacle space dimensions

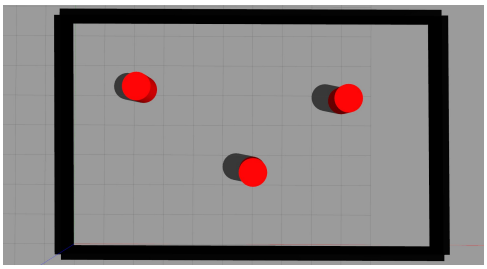

Fig. 4. World in Gazebo

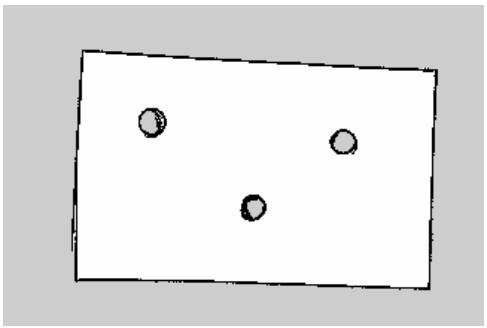

Fig. 5. SLAM Map in Rviz

position and scale of the walls and cylinders and added colors to them. We resaved the world file (Fig. 4). We spawned Turtlebot Burger in the world and used SLAM-gmapping and Teleop to create a map of the world (map.yaml and map.pgm files) (Fig. 5). The arena and its map are shown in Fig. 4 and 5 respectively. We kept the map.world in the worlds folder and the map.yaml and map.pgm in map folder of the package. We took the path from the path planning algorithm and published few of those points by skipping appropriate no. of points in between to MoveBase AMCL server to move the Turtlebot3 burger.

IV. Simulations and Discussions

This section presents simulation results of the TG-RRT* in clutterd environment. The new algorithm is run with both the Centroid and Incentre extensions separately and the iterations and cost required to reach optimum path solution in each environment are calculated. RRT* algorithm is also run in the same environment for comparison purposes. In all figures depicting simulation results, obstacles in the configuration space are seen as red objects while the pink network of lines represents a tree while the single black line indicates the path deduced by the algorithm after n iterations.

C-RRT* indicates RRT* extended by applying the Centroid geometrical method while IC-RRT* represents RRT* extended using the Incentre geometrical method. In all performance evaluations presented in this project, cost is the sum of the **Euclidean distances** of subsequent nodes connecting the starting and ending nodes of the path and is denoted by C while N is the number of iterations taken by the algorithm to find an optimal path. To compare the results of these 3 algorithms, three different pairs of start and goal locations are selected.

The simulation results for each start and goal location is shown in the following seb-sections

A. Start - (50, 50) and Goal - (375,225)

Figures show the performance of all three types of algorithms in finding an initial path from starting to goal node in a cluttered environment. It can be seen that out of the three algorithms, RRT* takes more number of iterations and also explores the entire space. Between IC-RRT* and RRT*, the IC-RRT* takes lesser number of iterations and also explores in the area between the start and goal location only. C-RRT* takes the least number of iterations but the total cost is compromised here. The comparison between cost and number of iterations is shown in Table 1.

TABLE I

Algorithm	Cost (C)	Number of iterations (N)
RRT*	92.68	883
TG-RRT*	71.19	733
C-RRT*	217.66	542

The simulation videos can be found here: RRT*, IC-RRT*, C-RRT*

The path for each algorithm is also simulated in Gazebo. The Gazebo simulation videos can be found here: RRT*, IC-RRT*, C-RRT*

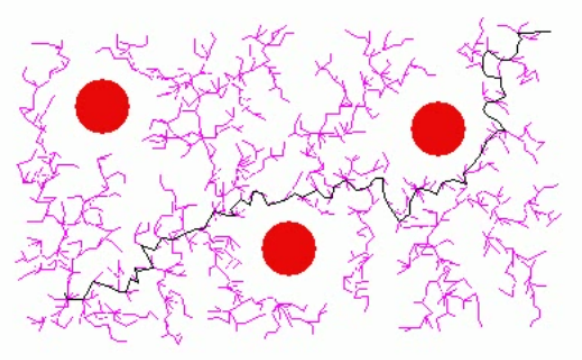

Fig. 6. RRT*

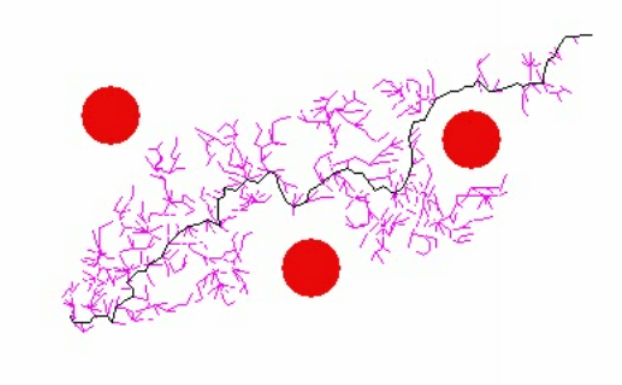

Fig. 7. IC-RRT*

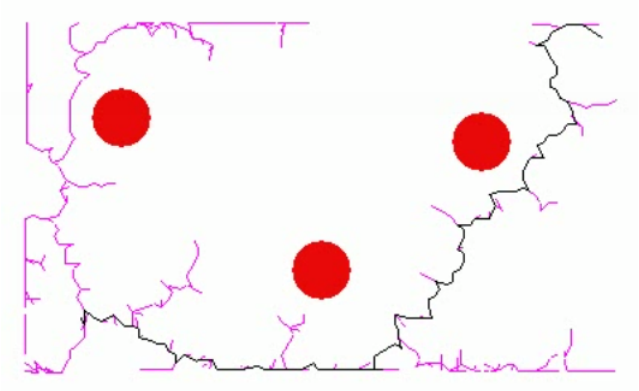

Fig. 8. C-RRT*

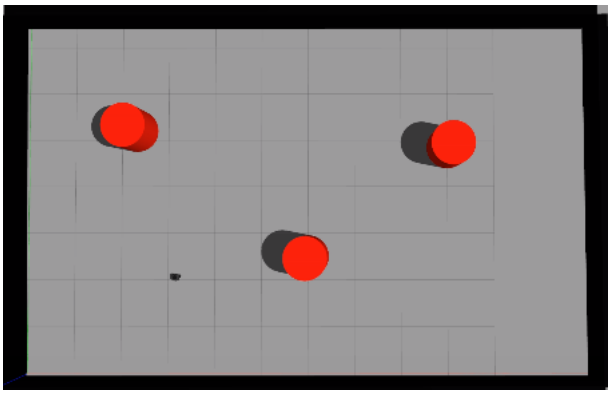

Fig. 9. Gazebo Simulation

B. Start - (150, 50) and Goal - (375,225)

Figures show the performance of all three types of algorithms in finding an initial path from starting to

goal node in a cluttered environment from (150, 50) to (375, 225). It can be clearly seen from the results that inspite of the start location selected approximately at the center of the space, the RRT* algorithm searches and explores the entire space whereas the IC-RRT* algorithm explores only in the area between the start and goal thus reducing the number of iterations required to find the path and hence reducing the time.The comparison between cost and number of iterations is shown in Table 2.

TABLE II

Algorithm	Cost (C)	Number of iterations (N)
RRT*	45.68	1566
TG-RRT*	39.07	796
C-RRT*	186.35	512

The simulation videos can be found here: RRT*, IC-RRT*, C-RRT*

The Gazebo simulation videos can be found here: RRT*, IC-RRT*, C-RRT*

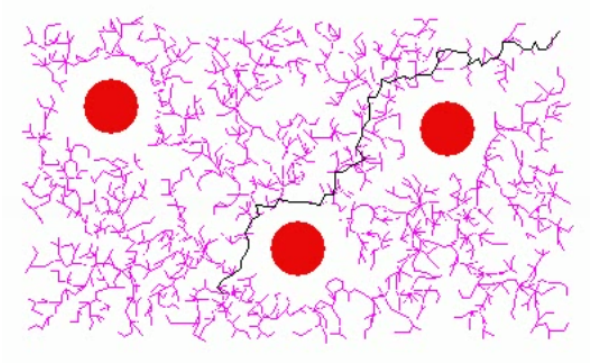

Fig. 10. RRT*

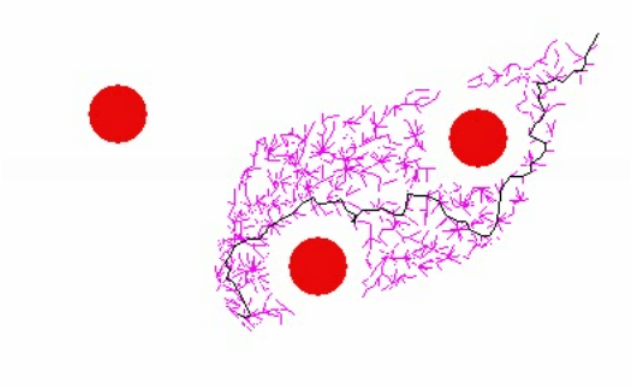

Fig. 11. IC-RRT*

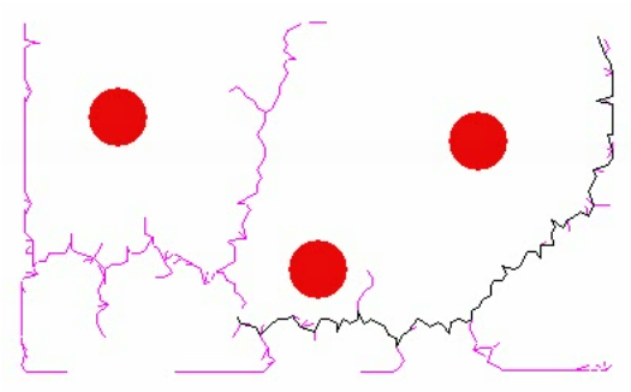

Fig. 12. C-RRT*

C. Start - (50, 225) and Goal - (250,50)

Figures show the performance of all three types of algorithms in finding an initial path from starting to goal node in a cluttered environment from (50, 225) to (250, 50).The comparison between cost and number of iterations is shown in Table 3.

TABLE III

Algorithm	Cost (C)	Number of iterations (N)
RRT*	89.52	303
TG-RRT*	45.85	176
C-RRT*	187.47	665

The simulation videos can be found here: RRT*, IC-RRT*, C-RRT*

The Gazebo simulation videos can be found here: RRT*, IC-RRT*, C-RRT*

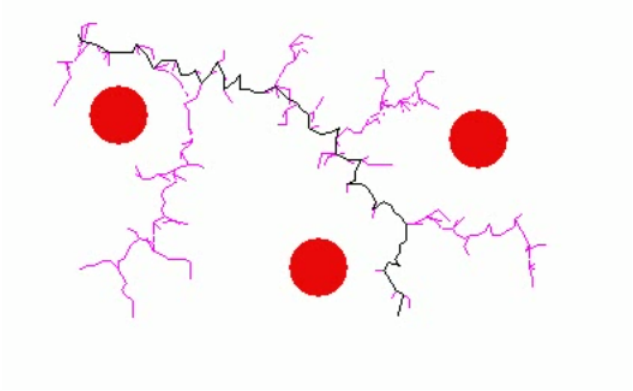

Fig. 13. RRT*

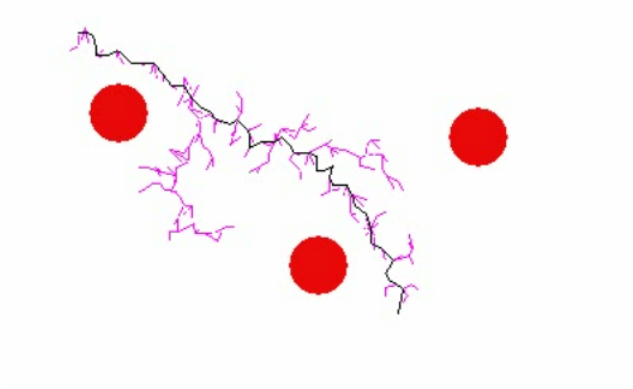

Fig. 14. IC-RRT*

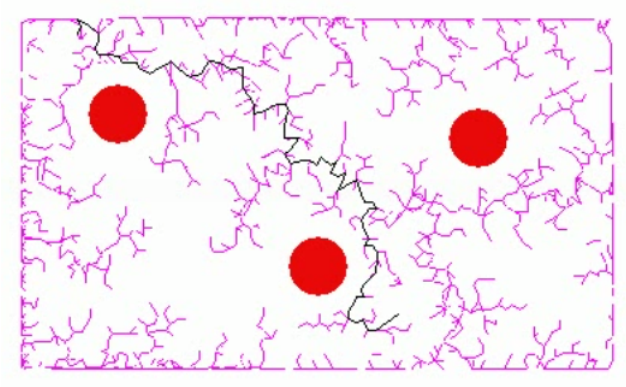

Fig. 15. C-RRT*

V. Conclusions

The three tables mentioned above shows the comparison between the three algorithms in terms of number of iterations required by each to converge to an initial path and the total cost.

Time complexity for all algorithms is the same i.e. O(nlogn) as no extra looping is involved. Differences in processing times of the algorithms are due to the extra calculations being performed for every sample to direct them closer to the goal region.

In first two cases, RRT* requires much greater number of iterations to reach the same initial solution that is reached by C-RRT* and IC-RRT* in significantly less number of iterations.

Comparing C-RRT* and IC-RRT*, it can be observed that C-RRT* gives a better performance in first two cases since it requires least number of iterations. In all cases, IC-RRT* takes lesser number of iterations than RRT* and returns a path with the least cost.

Since there exist situations(last case) in which C-RRT* gives a poor performance, it can be concluded that IC-RRT* is the most robust algorithm of the three, allowing efficient motion planning in almost every type of start and goal location.

Also while working on this project, we observed that

the sampling space changes according to the type of geometric center selected. But due to time constraint, the exact relation between the geometric center and the sampling space could not be found and thus this problem can be declared as the future scope of this project.

REFERENCES

[1] A. H. Qureshi et al., "Triangular geometry based optimal motion planning using RRT-motion planner," 2014 IEEE 13th International Workshop on Advanced Motion Control (AMC), 2014, pp. 380-385, doi: 10.1109/AMC.2014.6823312.

[2] Karaman S and Frazzoli E, "Incremental sampling-based algorithms for optimal motion planning", 2010b Robotics: Science and Systems (RSS), 2010, doi: 10.48550/arXiv.1005.0416.