# Path planning for autonomous robot using Informed-RRT*: Extension of RRT* by using rapid convergence rate

Ishan Patel[1] Nakul Patel[2]

*Abstract*— **For the motion planning of any autonomous robots, many sampling based algorithms are available of which, RRT is one of the quickest and most efficient obstacle free path-finding algorithm[1]. However, since RRT cannot guarantee optimal path although being probabilistic complete, an extension of RRT, i.e RRT* algorithm is introduced. This method claims to achieve convergence towards the optimal solution thus assuring asymptotic optimality. But, it takes infinite time to do so and hence has slow convergence rate. Hence, a novel extension to RRT* is introduced in this paper, named Informed-RRT*. The goal of the proposed algorithm is to increase the rate of convergence, in order to reach to the optimum/near optimum solution at a faster rate, thus reducing the execution time. Furthermore, two techniques – Path Optimization and Intelligent Sampling, have been implemented for the proposed informed-RRT* algorithm. Having introduced these versions of RRT algorithm ,we will compare the performance results, along with statistical comparison of these algorithms when employed on a particular autonomous robot in various obstacle cluttered environments.**

**Keywords:** RRT, RRT*, Informed-RRT*, Path Optimization, Intelligent Sampling, Map Exploration, Rewiring, Ellipsoidal Subsampling.

## I. INTRODUCTION

The industry of robotics and autonomous robots is rapidly growing at the pace, like no other industry. That being said, the foremost task at the hand for any application of the autonomous robot, is to perform the path-planning, followed by motion planning. In the path-planning problem, obstacle avoidance is of utmost importance, for which the choice of algorithm depends on various parameters. Many path planning algorithms, including geometric, grid-based, potential field, neural networks, genetic and sampling based algorithms, have been proposed in recent years for various kind of static and dynamic environments. There are many algorithms that have been developed, RRT, A*, Dijkstra, PRM to name few[1]. Each of these algorithms has its own advantages and disadvantages in finding the most efficient path planning solution in terms of space and time complexity and path optimization[1]. Among these, we would want to select the algorithm that is faster and efficient for obstacle free path planning. RRT* is one such algorithm, which ensures probabilistic completeness, though it doesn't guarantee the most optimal path[1].

Sampling based algorithm are widely used path planning algorithms because of the following reasons. As compared to other probabilistically complete algorithms, they are computationally less complex, and have the ability to find solutions without using explicit information about the obstacles in the configuration space of the robot. Furthermore, they depend on a collision checking module and build a roadmap of feasible trajectories made by connecting together a set of points sampled from the obstacle-free space[12]. However, since it was obvious that the generated trajectories are not optimal, a new concept of asymptotic stability, which corresponds to convergence to optimal solution with increased number of samples. This concept has been utilized in the RRT*.

RRT* claims to achieve convergence towards the optimal solution, and ensures asymptotic optimality along with probabilistic completeness, but this takes an infinite time to do so and has slow convergence rate[1]. Thus, in order to overcome the limitations of RRT*, the new algorithm called Informed-RRT* has been introduced. Its major advantage over other algorithms is that it finds an initial path very quickly and then later keeps on optimizing it as the number of samples increases[1].

## II. METHOD

### A. *RRT (Feasible Path Solution:)*

RRT is the sampling-based algorithm that employs randomization to explore large state spaces efficiently as it avoids state explosion[3]. Due to the incremental nature of RRT, it can maintain kinematic constraints, and can be used to develop paths based on the kinodynamic and non-holonomic constraints.

The main advantage of RRT algorithm is that in their building process they are intrinsically biased towards regions with a low density of configurations[13].

RRT constructs a tree using random sampling in search space[6]. When a point in the space is randomly sampled, it is checked if that point collides with an obstacle in the space[6]. If the sampled point has no collisions, it is then checked if the straight line path between the sampled point and the nearest existing point in the tree has any collisions. If this straight line path has no collisions, the sampled point is added to the tree with the nearest point as its parent node. If there is a collision, this point is thrown out. Examples of these three outcomes can be seen in Figure-1[11].

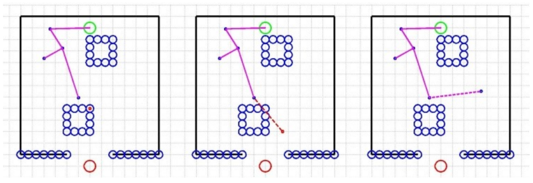


Fig. 1

Each time after a node is added to the tree and than node is less than some threshold distance from the goal position, it is checked if the goal can be reach in a straight line path from the added node. If the goal position is reachable, the goal position is added to the tree with the recently added node as its parent. At this point, the path planning is complete. If the goal position is still unreachable, additional points are sampled. An example of a completed path is shown in Figure-2 [11].

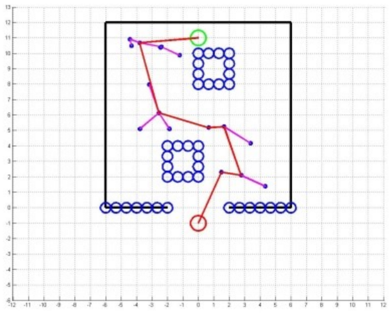


Fig. 2

The algorithm for RRT is shown in Table-1:

However, since RRT gives only feasible solution, and not optimal solution, we need to develop modified algorithm, that can yield optimal path from start point to goal point in finite time and iterations of random sampling. Hence, a variant of RRT, called RRT* is introduced and explained in the subsequent subsection.

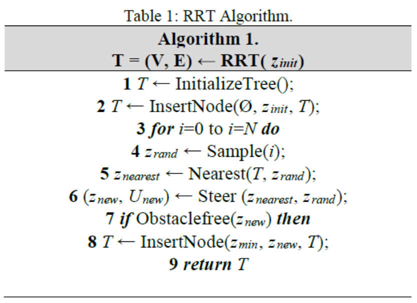


Table 1: RRT Algorithm.

```
Algorithm 1.
T = (V, E) ← RRT( $z_{init}$)
1 $T \leftarrow$ InitializeTree();
2 $T \leftarrow$ InsertNode(Ø, $z_{init}$, $T$);
3 for $i$=0 to $i$=$N$ do
4 $z_{rand} \leftarrow$ Sample($i$);
5 $z_{nearest} \leftarrow$ Nearest($T$, $z_{rand}$);
6 ($z_{new}$, $U_{new}$) $\leftarrow$ Steer ($z_{nearest}$, $z_{rand}$);
7 if Obstaclefree($z_{new}$) then
8 $T \leftarrow$ InsertNode($z_{min}$, $z_{new}$, $T$);
9 return $T$
```

Fig. 3

## B. *RRT* (Optimal Path Solution:)*

RRT* works similar to RRT and inherits all the properties of RRT. However, it introduces two promising features called near neighbor search and rewiring tree operations . Near neighbor operations finds the best parent node for the new node before its insertion in tree[6]. This process is performed within the area of a ball of radius defined by

$$k = \gamma \left( \frac{\log(n)}{n} \right)^{\frac{1}{d}}$$

where d is the search space dimension and  is the planning constant based on environment. Rewiring operation rebuilds the tree within this radius of area k to maintain the tree with minimal cost between tree connections as shown in Figure 5 . As the number of iterations increase, RRT* improves its path cost gradually due to its asymptotic quality, whereas RRT does not improves its jaggy and suboptimal path[6].

If X is the configuration space and $X_{obs}$ is the obstacle region, then $X_{free} = X/X_{obstacle}$ will be the obstacle-free region, and $X_{goal}$ is the goal region. As seen in the algorithm, which is labeled as Table 2, the following functionalities have been incorporated[1].

**Sample:** This function randomly samples a state $Z_{rand} \in X_{free}$ from the obstacle-free configuration space[1].

**Distance:** This function will return the cost(in terms of Euclidean distance) of the path between two states assuming the region between them is obstacle free[1].

**Nearest Neighbor:** The function Nearest(T,$z_{rand}$) will return the nearest node from T=(V,E) to $Z_{rand}$ in terms of cost which is determined by the distance function[1].

**Steer:** This function takes parameters $Z_{nearest}, z_{rand}$ and solves for a control input u[0,T] that drives the system from $x(0) = Z_{rand}$ to $x(T) = Z_{nearest}$ along the path x:[0,T] $\mapsto Xandgivesz_{new}$ at the distance of $\delta q$(incremental distance) from the $z_{nearest}$ towards $z_{rand}$[1].

**Check Collision:** The function Obstaclefree(x) will determine whether the path x[0,T] lies in the obstacle-free region $X_{free}$ for all t=0 to t=T [1].

**Nearby Vertices:** The function Near(T,$z_{rand}$, n) returns the nearby neighbor nodes that lie in a sphere of volume given by the above equation of k, with $z_{rand}$ as the centre[1].

**Insert node:** This function InsertNode ($z_{parent}$,$z_{new}$, T) adds a node $z_{new}$ to V in tree T = (V,E) and connects it to an existing node $z_{parent}$ as its parent, and adds the edge to E. Also, a cost is assigned to $z_{new}$ which equals the sum total of the cost of its parent and the Euclidean cost returned by distance function between $Z_{new}$ and its parent $z_{parent}$ [1].

**Rewiring:** The function Rewire (T,$z_{near}$,$z_{min}$,$z_{new}$) will check if the cost to the nodes in neighbor nodes is less through $z_{new}$ as compared to their older costs. If it is less for a particular node, its parent is changed to $Z_{new}$ [1].

The algorithm for RRT* is shown in Table-2:

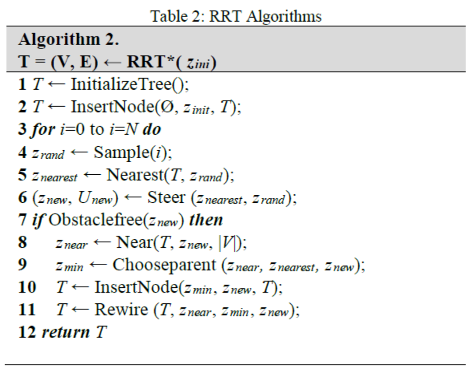


Table 2: RRT Algorithms

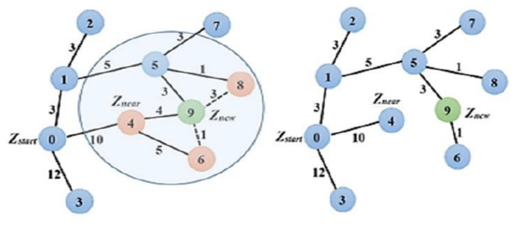
```
Algorithm 2.
T = (V, E) ← RRT*( $z_{ini}$)
1 $T$ ← InitializeTree();
2 $T$ ← InsertNode(Ø, $z_{init}$, $T$);
3 for $i$=0 to $i$=$N$ do
4   $z_{rand}$ ← Sample($i$);
5   $z_{nearest}$ ← Nearest($T$, $z_{rand}$);
6   ($z_{new}$, $U_{new}$) ← Steer ($z_{nearest}$, $z_{rand}$);
7   if Obstaclefree($z_{new}$) then
8       $z_{near}$ ← Near($T$, $z_{new}$, $|V|$);
9       $z_{min}$ ← Chooseparent ($z_{near}$, $z_{nearest}$, $z_{new}$);
10      $T$ ← InsertNode($z_{min}$, $z_{new}$, $T$);
11      $T$ ← Rewire ($T$, $z_{near}$, $z_{min}$, $z_{new}$);
12 return $T$
```

Fig. 4

The following figures are the detailed interpretation to extra features (rewiring and new best parent).

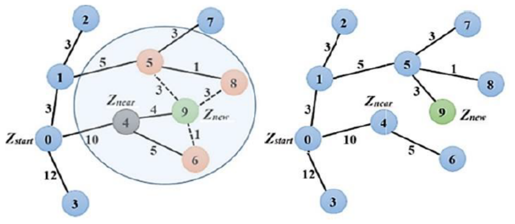
## C. *Informed RRT*(Convergence to Optimal Solution:)*

Informed-RRT* is the most recent modification of RRT*, and has two additional features: Path Optimization and Intelligent Sampling[1]. Since RRT* is asymptotically optimal everywhere that



(c) Check cost for rewiring (d) Rewired tree

Fig. 5: [6]



(a) Find near neighbors (b) Select best parent

Fig. 6: [6]

would be unnecessary for single-query planning. Hence, we should limit our node search to the subspace that would have a better solution.

Informed RRT* behaves as RRT* until a first solution is found, and afterwards all possible improvements are contained in an ellipse defined by the path length, the start and the goal[8]. Sampling the ellipsoidal subset will improve the convergence rate of solution.

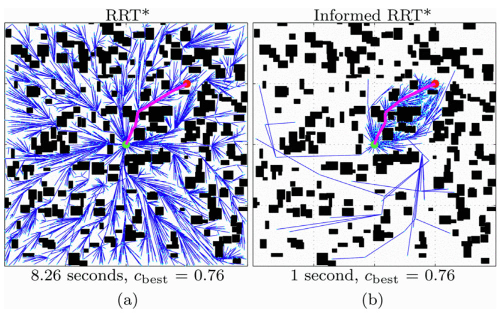


Fig. 7

By directly sampling the ellipse, we focus the search to only the nodes/states that have the possibility of improving the final solution. This will increase the solution improvement rate, and causes the ellipse to quickly shrink and further focusing the search for improvements[8].

The algorithm for Informed-RRT* is shown in Figure-9:

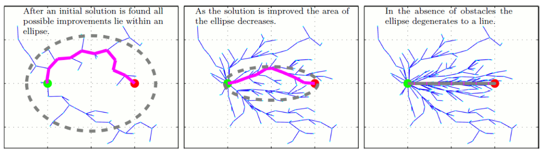


Fig. 8

```
Algorithm 1: Informed RRT*($x_{start}, x_{goal}$)
$V \leftarrow \{x_{start}\}$;
$E \leftarrow \emptyset$;
$X_{soln} \leftarrow \emptyset$;
$\mathcal{T} = (V, E)$;
for iteration $= 1 \ldots N$ do
    $c_{best} \leftarrow \min_{x_{soln} \in X_{soln}} \{\texttt{Cost}(x_{soln})\}$;
    $x_{rand} \leftarrow \texttt{Sample}(x_{start}, x_{goal}, c_{best})$;
    $x_{nearest} \leftarrow \texttt{Nearest}(\mathcal{T}, x_{rand})$;
    $x_{new} \leftarrow \texttt{Steer}(x_{nearest}, x_{rand})$;
    if CollisionFree($x_{nearest}, x_{new}$) then
        $V \leftarrow \cup \{x_{new}\}$;
        $X_{near} \leftarrow \texttt{Near}(\mathcal{T}, x_{new}, r_{RRT*})$;
        $x_{min} \leftarrow x_{nearest}$;
        $c_{min} \leftarrow \texttt{Cost}(x_{min}) + c \cdot \texttt{Line}(x_{nearest}, x_{new})$;
        for $\forall x_{near} \in X_{near}$ do
            $c_{new} \leftarrow \texttt{Cost}(x_{near}) + c \cdot \texttt{Line}(x_{near}, x_{new})$;
            if $c_{new} < c_{min}$ then
                if CollisionFree($x_{near}, x_{new}$) then
                    $x_{min} \leftarrow x_{near}$;
                    $c_{min} \leftarrow c_{new}$;
        $E \leftarrow E \cup \{(x_{min}, x_{new})\}$;
        for $\forall x_{near} \in X_{near}$ do
            $c_{near} \leftarrow \texttt{Cost}(x_{near})$;
            $c_{new} \leftarrow \texttt{Cost}(x_{new}) + c \cdot \texttt{Line}(x_{new}, x_{near})$;
            if $c_{new} < c_{near}$ then
                if CollisionFree($x_{new}, x_{near}$) then
                    $x_{parent} \leftarrow \texttt{Parent}(x_{near})$;
                    $E \leftarrow E \setminus \{(x_{parent}, x_{near})\}$;
                    $E \leftarrow E \cup \{(x_{new}, x_{near})\}$;
        if InGoalRegion($x_{new}$) then
            $X_{soln} \leftarrow X_{soln} \cup \{x_{new}\}$;
return $\mathcal{T}$;
```

Fig. 9

### D. *Problem Definition:*

The problem definition for Informed-RRT* is as follows: Let $X \subseteq R^n$ be the state space and $X_{obs} \subseteq X$ be the obstacle space, then $X_{free} = X/X_{obs}$ is the resulting set of permissible states. Let $x_{start}$ and $x_{goal}$ be the initial and goal states respectively[8].

Now, if f(x) is the cost of an optimal path from $x_{start}$ to $x_{goal}$. Then the subset of states that can improve the current solution can be expressed as[8]:

$$X_f = \{\mathrm{x} \in X | f(\mathrm{x}) < c_{\text{best}}\}$$

Also, since f(.) is generally unknown, a heuristic function $\widehat{f}(\cdot)$ may be used as an estimate. This ellipsoidal heuristic is referred to as admissible if it never overestimates the true cost of the path[8].

Now, since the ellipsoidal informed subset as shown in figure 10 is used as the subspace domain, the heuristic-based sample rejection will attempt to

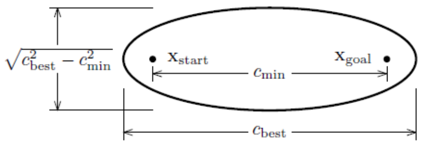


Fig. 10

increase the real-time rate of sampling $X_f$ by using rejection sampling on X to sample $X_{\overline{f}}$.[8]

Given a positive cost function, the cost of the optimal path from $x_{start}$ to $x_{goal}$ f(x), is equal to the cost of the optimal path from $x_{start}$ and $x$, g(x), plus the cost of optimal path from $x$ to $x_{goal}$, h(x). For the problems seeking to minimize the path length, the Euclidean distance is an admissible heuristic. This heuristic that may improve the current solution, $X_{\widehat{f}} \supseteq X_f$, can be expressed as [8]: $\mathrm{X}_{\widehat{f}} = \{\mathrm{x} \in X \quad \|\mathrm{x}_{\text{ start}} - \mathrm{x}\|_2 + \|\mathrm{x} - \mathrm{x}_{\text{ goal}}\|_2 \leq c_{\text{ best}}\}$

The equation is the general equation of the n-dimensional hyperspheroid, and the focal points are the start and goal points respectively, the traverse diameter being $c_{best}$ and the conjugate diameters are $\sqrt{c_{\text{best}}^2 - c_{\text{min}}^2}$.

**Path Optimization:**
This is the main feature of informed-RRT*, which is essential for path optimization. As discussed earlier, informed-RRT* behaves as RRT* till the first intial path has been found, after that the nodes in the path x: $[x_{start}, x_{goal}] \mapsto X$ thsat are visible to each other are directly connected. However, in this method, an iterative process starts to check for direct connections with successive parent's of each node until the obstacles are found[1]. Hence, by the end of this process, there will be no more connected nodes. Thus, we can say that the path is optimized based on the triangle inequality.

This leads to reduction in the number of nodes on the path,as compared to that in RRT*, and these nodes are termed as Beacons($x_{beacons}$), that will inturn form the basis for intelligent sampling.

The interpolation method used to do the visibility check between two nodes works by constructing every point on the line and ensures that the newly added points lie in the free configuration space[1].

**Intelligent Sampling:** As discussed earlier in path optimization, the newly considered nodes after the path has been optimized with the interpolation method, will form the basis for this important process of intelligent sampling. Also, since the interpolation method used for collision free-checking is independent of the shape of the obstacles, it is less expensive.

The approach here is to generate the nodes as close as possible to the obstacle points and achieve optimality by visibility graph technique. As the algorithm iterates, an optimized path is calculated whenever a new RRT* path has been found[1]. If the cost of this path is smaller than the previous optimized path, then new beacons are generated, closer to the vertices, and this process continues till the predefined iterations are complete [1],[6].

Thus, due to this enhanced capability of path optimization and intelligent sampling, informed-RRT* is used as the sampling based algorithm, and can be employed on any autonomous robot.

## III. Results:

In this section, we analyze three algorithms RRT, RRT* and Informed-RRT*. We have used RRL(Robotics Realization Laboratory) Map as our obstacle space. We have performed two experiments. In the first experiment we implemented RRT, RRT* and Informed-RRT* on the same obstacle space for the same start and goal node in python. In the second experiment, we have carried out comparison between RRT* and Informed-RRT* using simulation results at varying number of iterations. Lastly, we have implemented map exploration algorithm using Informed-RRT* in ROS(Robot Operating System).

### *A. Comparison between RRT, RRT* and Informed-RRT**

In this section, we analyze the result obtained by implementing RRT,RRT* and Informed-RRT* on RRL-map obstacle space. The obstacle space has been created using half plane and semi-algebraic equations. The start node is [400,400] and the goal node is [550,600] for all of the above implementations. The number of iterations have been kept constant i.e. n = 500. The final path obtained using RRT, RRT* and Informed-RRT* are as follows:

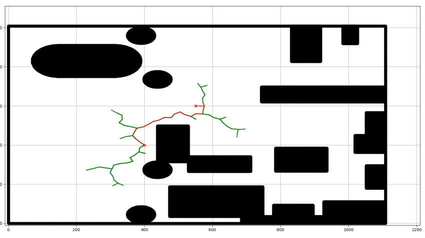


Fig. 11: RRT Simulation

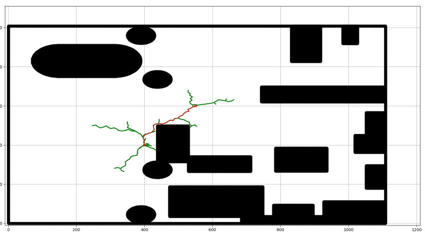


Fig. 12: RRT* Simulation

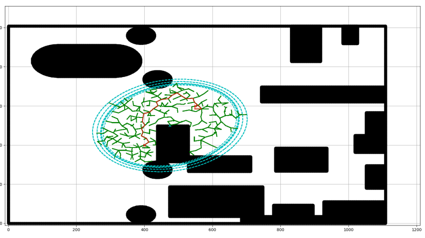


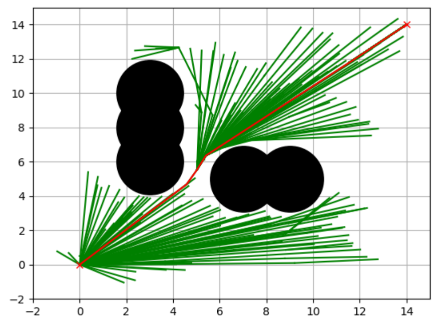
Fig. 13: Informed-RRT* Simulation

From the Figure-12, we can say that RRT algorithm gives a feasible path as compared to the optimal path given by RRT* and Informed-RRT*. As RRT does not perform path optimization, it is faster than the other two algorithms. As the RRT* algorithm uses incremental rewiring of the graph, we obtain a optimal path solution. But due to its slower rate of convergence and asymptotic optimality, it takes high amount of time to compute an optimal path. The Informed-RRT* overcomes this drawback by directly sampling ellipsoidal heuristic. Thus, it takes less time to converge to an optimal path. The ellipsoid shrinks as a path with lesser cost is found as shown in Figure-14.

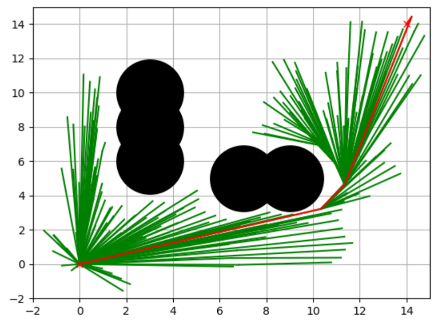
### *B. Performance Comparison of RRT* and Informed-RRT**

In this experiment, we compare the performance of RRT* and informed-RRT* for the following number of iterations: n = 300, n = 400 and n = 500. We have used a different obstacle space for this experiment.

For n = 300 iterations, RRT* finds a path with cost:14.92 and Informed-RRT* finds a path with cost:13.86. For n = 400 iterations, RRT* finds a path with cost:14.52 and Informed-RRT* finds a

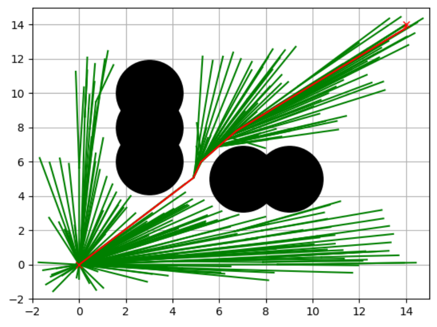


(a) RRT*

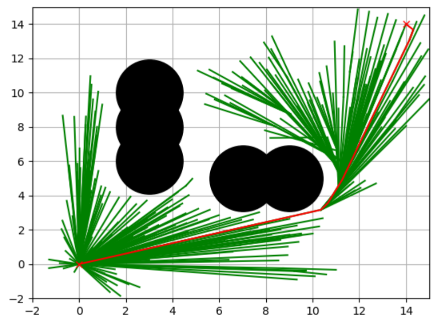


(b) Informed-RRT*

Fig. 14: Number of Iterations: n = 300



(a) RRT*



(b) Informed-RRT*

Fig. 15: Number of Iterations: n = 400

path with cost:12.95. For n = 500 iterations, RRT* finds a path with cost:13.91 and Informed-RRT* finds a path with cost:12.54.

From the above results, we can say that Informed-RRT* has a much higher rate of convergence as compared to RRT*. It also finds the optimal path quicker than RRT*.

## C. *Map Exploration using Informed-RRT**

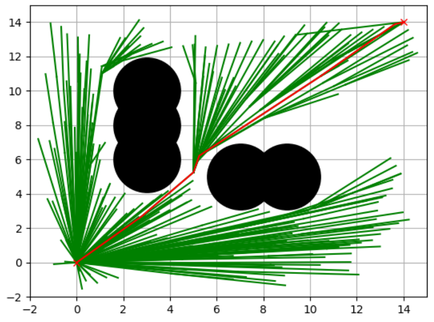
In this experiment, we simulated an autonomous exploration of map using Informed-RRT* in ROS(Figure 17). We used the following packages of ROS: Gmapping, Navigation Stack and Opencv. We also used Opencv frontier detection to extract frontier points. Frontiers are boundaries that separates known space from unknown space. We have used Informed-RRT* for map exploration because it is biased towards unexplored regions and it can also be used for higher dimensional spaces. [14]

## D. *Complexity Analysis*

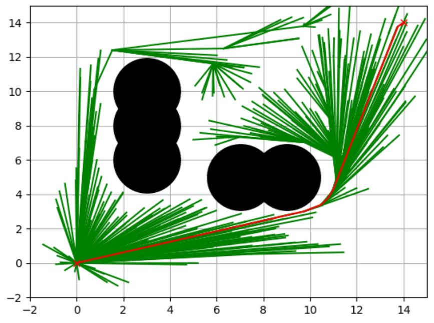
The space and time complexity of RRT* is given by O(n) and O(n*log(n)). According to mathematical analysis, Informed-RRT* has same space and time complexities as RRT, but the value of n is significantly reduced in order to achieve same optimality[1].

## E. *Time Complexity*

Time complexity of an algorithm is defined as the amount of time that is required by the algorithm to execute a problem of size n. Due to the additional steps of path optimization, intelligent sampling and the collision checking method of the interpolation, Informed-RRT* has complexities that are insignificant enough to have no effect on time-complexities. RRT* and Informed-RRT*, shows that for the same number of iterations the time for RRT*-Smart is significantly less as compared to RRT*[1].



(a) RRT*



(b) Informed-RRT*

Fig. 16: Number of Iterations: n = 500

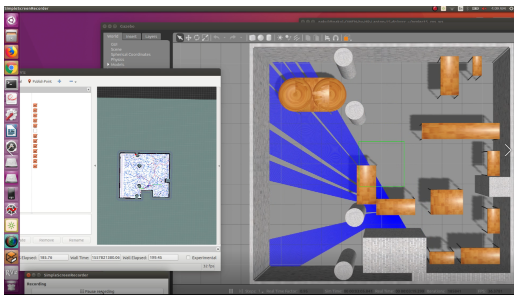


Fig. 17: RRT Simulation

## IV. Conclusions and Future Work:

After analyzing the methods of this algorithm, it is clear that the RRT algorithm yields feasible solution, while RRT* and informed-RRT* yield the optimal solutions.

RRT and its variants employs randomization to explore large state spaces efficiently as it avoids the state explosion that discretizes faces, making RRT a lot faster algorithm compared to other motion planning algorithms[3]. It can also be used to develop paths based on kinodynamic and non-holonomic constraints. Further, to produce optimal path, RRT* has been augmented with two additional steps: ‘finding best parent’ and ‘rewiring’.

Furthermore, the rate of convergence of informed-RRT* is higher as compared to RRT* since it uses the elliptical based heuristic subsampling. This can be validated by the simulation results discussed above. Hence, we can conclude that informed-RRT* is superior as compared to RRT* in terms of time and cost.

Owing to these nice results, the informed-RRT* algorithm can be extended to higher dimensional spaces. Further, the algorithm can be made more computationally efficient for its application on the autonoumous robots.

## References

[1] J. Nasir et al., ”RRT*-SMART: A Rapid Convergence Implementation of RRT*”, International Journal of Advanced Robotic Systems, vol. 10, pp. 1-12, 2013.

[2] M. Kanehara, S. Kagami, J.J. Kuffner, S. Thompson, H. Mizoguhi (2007) ”Path shortening and smoothing of grid-based path planning with consideration of obstacles”, IEEE International Conference on Systems, Man and Cybernetics, (ISIC) , pp. 991-996.

[3] Canberk Gurel, Rajendra Sathyam, Akash Guha “ROS-based Path Planning for Turtlebot Robot using Rapidly Exploring Random Trees (RRT*)”.

[4] L. Chang-an, C. Jin-gang, L. Guo-dong, and L. Chun-yang, “Mobile Robot Path Planning Based on an Improved Rapidly-exploring Random Tree in Unknown Environment,” IEEE International Conference on Automation and Logistics, pp. 2375-2379, September 2008.

[5] Sertac Karaman, Emilio Frazzoli, “Asymptotic Optimality in Sampling-based Motion Planning”. The International Journal of Robotics Research, vol. 30, pp. 846-894, 2011.

[6] I. Noreem, A. Khan, Z. Habib, “A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms,” IJCSNS International Journal of Computer Science and Network Security, vol. 16, no. 10, 2016.

[7] R. Seif and M. Oskoei, “Mobile Robot Path Planning by RRT* in Dynamic Environments,” I.J. Intelligent Systems and Applications, pp. 24-30, May 2015.

[8] J. D. Gammell, S. S. Srinivasa and T. D. Barfoot, ”Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic,” 2014 IEEE/RSJ International Conference on

Intelligent Robots and Systems, Chicago, IL, 2014, pp. 2997-3004.

[9] S. Karaman and E. Frazzoli (2011) “Sampling-based Algorithms for Optimal Motion Planning”, International Journal of Robotics Research, vol. 30, no. 7, pp. 846-894.

[10] Y. Kuwata, G. Fiore, and E. Frazzoli, “Real-time Motion Planning with Applications to Autonomous Urban Driving,” IEEE Transactions on Control Systems Technology, vol. 17, no. 5, September 2009.

[11] http://coecsl.ece.illinois.edu/ge423/spring13/RickRekoskeAvoid/rrt.html

[12] Wilbert G. Aguilar and Stephanie G. Morales, ”3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms”.

[13] V. Alcazar, M. Veloso and D. Borrajo, ”Adapting a Rapidly-exploring Random Tree for Automated Planning”.Published in SOCS 2011

[14] H. Umari and S. Mukhopadhyay, ”Autonomous robotic exploration based on multiple rapidly-exploring randomized trees”.Published in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)