ROS-based Path Planning for Turtlebot Robot using Informed Rapidly Exploring Random Trees (Informed RRT*)

Arpit Aggarwal
University of Maryland
College Park, U.S.A.
arpitagg@umd.edu

Shantam Bajpai
University of Maryland
College Park, U.S.A.
sbajpai@terpmail.umd.edu

***Abstract*—Informed RRT* motion planning algorithm outperforms RRT* motion planning algorithm in rate of convergence, final solution cost, and ability to find difficult passages while demonstrating less dependence on the state dimension and range of the motion planning problem. Therefore, Informed RRT* algorithm is a powerful yet simple algorithm used for navigation. Informed RRT* motion planning algorithm will be used on ROS Turtlebot 2 to navigate in a configuration space consisting of static obstacles. The path generated by the Informed RRT* motion planning algorithm will be compared with the path generated by the RRT* motion planning algorithm on the basis of the optimal time and optimal path from the start node to the goal node.**

***Index Terms*—Informed RRT*, RRT*, Path Planning, Motion Planning, Robot Operating System (ROS), Sampling-based Motion Planning, Turtlebot 2**

I. Introduction

Intelligent robot systems have a robot framework that perceives data about its condition through sensors, decides the state and produces the action to move from the beginning point to the objective point so as to achieve a given task [1]. The need of utilizing an efficient motion planning algorithm on the robot is to guarantee that the robot moves from the beginning point to the objective point in an ideal manner and in ideal time. The motion planning problem is commonly solved by discretizing the state space with a grid for action-based planning or through random sampling for sample-based planning[2][3]. The issue with action-based planning is that as the configuration space expands, the time required to find an optimal path from the beginning point to the objective point increments exponentially.

Sampling-based motion planning algorithms, for example RRT and RRT* have been broadly utilized for the motion planning of robots lately[2][3]. Sampling-based motion planning algorithms give fast answers for high dimensional issues utilizing randomized exploring in search space [4][5]. Rapidly-exploring Random Trees (RRT) is a sampling-based planning that gives the path rapidly yet neglects to give an optimal path. The variants of the RRT motion planning algorithms like RRT* motion planning algorithm guarantees to be asymptotically ideal, with the likelihood of finding an optimal path as the number of iterations reach infinity. [7]. Although RRT*[8] extend the RRTs to finding the optimal solution, however, they find the optimal path from the beginning point to every other point in the configuration space. This tends to be inefficient in scenarios where a single query nature exists. On the other hand, Informed RRT* motion planning algorithm, a variant of RRT* algorithm, finds an optimal path from the beginning point to the objective point and retains the same probabilistic guarantees on completeness and optimality as RRT* while improving the convergence rate and solution quality[6].

In this paper, the Informed RRT* motion planning algorithm will be used on ROS Turtlebot 2 to navigate in a configuration space consisting of static obstacles. Turtlebot 2 belongs to the class of differential drive mobile robots whose movement is based on the relative rate of rotation of its two independently driven wheels placed on either side of the robot[9]. The path generated by the Informed RRT* motion planning algorithm will be compared with the path generated by the RRT* motion planning algorithm on the basis of the optimal time and optimal path from the beginning point to the objective point. This will help us verify that Informed RRT* algorithm outperforms RRT* algorithm in rate of convergence, final solution cost, and ability to find difficult passages while demonstrating less dependence on the state dimension and range of the motion planning problem.

II. Related Work

In [6], Gammell et al. demonstrated the simulation results of testing the Informed RRT* algorithm and the RRT* algorithm in an unknown environment. Informed RRT* algorithm, a variant of RRT* algorithm, finds an optimal path from the start node to the goal node and retains the same probabilistic guarantees on completeness and optimality as RRT* while improving the convergence rate and solution quality. In [10], Kuwata et al. demonstrated their variant of the RRT algorithm, that is a real-time motion planning algorithm. In [9], Zhang et al. presented their variant of the

RRT algorithm where RRT is combined with simultaneous localization and mapping (SLAM). In [1], Chang-an et al. presented the simulation results of testing their variant of the RRT algorithm and the traditional RRT algorithm in an unknown environment. Their variant of the RRT motion planning algorithm performed better than the traditional RRT motion planning algorithm. In [4], Seif et al. demonstrated a new path planning algorithm that is applicable to dynamic environments. In [3], Karaman et al. presented the overview of various sample-based algorithms used in motion planning.

III. Method

A rapidly exploring random tree (RRT) is a sampling-based motion planning algorithm designed to efficiently search non-convex, high dimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from the samples drawn randomly from the search space and is inherently biased to grow towards large unsearched areas of the problem. RRT* motion planning algorithm is an optimized version of RRT motion planning algorithm. When the number of nodes approaches infinity, the RRT* motion planning algorithm will deliver the shortest possible path to the objective point. It extends the RRTs to finding an optimal solution from the beginning point to every other point in the configuration space. The RRT* motion planning algorithm can be divided into 7 steps. Steps 1-5 explain the pseudo code given in Algorithm 1, whereas step 6 and 7 explain Algorithms 2 and 3, respectively.

RRT* Motion Planning Algorithm Steps:

A. Initialization:

The first step before implementing the RRT* motion planning algorithm is deciding the root node. In this case, the root node will be the start node. Next, the step size is decided. The step size is the maximum distance from Q_{near} to Q_{new}. The step size was taken as 6 cm. A larger step size is preferred in configuration spaces where there are sparse obstacles and a smaller step size is preferred where there are many obstacles.

B. Generating random node:

After the initialization step, the next step is generating the random node. The random point is biased towards exploring, as the larger unexplored area has higher chances of getting a random point than the smaller explored area.

C. Finding Nearest Neighbour Node:

After generating the random node, the next step is finding the nearest neighbour node. The nearest neighbour node is found by iterating through the nodes in the graph and finding the node which has the least euclidean distance with the random node.

Algorithm 1: $T = (V, E) \leftarrow$RRT*(q_{init})

```
T ←InitializeTree()
T ←InsertNode(∅,q_init,T)
for k ← 1 to N do
    q_rand ←RandomSample(k)
    q_nearest ←NearestNeighbor(q_rand,Q_near,T)
    q_min ←ChooseParent(q_rand,Q_near,q_nearest, Δq)
    T ←InsertNode(q_min, q_rand, T)
    T ←Rewire(T, Q_near, q_min, q_rand)
end
```

Algorithm 2: $q_{min} \leftarrow$ChooseParent$(q_{rand}, Q_{near}, q_{nearest},$

```
q_min ← q_nearest
c_min ←Cost(q_nearest) + c(q_rand)
for q_near ∈ Q_near do
    q_path ←Steer(q_near, q_rand, Δq)
    if ObstacleFree(q_path) then
        c_new ←Cost(q_near) + c(q_path)
        if c_new < c_min then
            c_min ← c_new
            q_min ← q_near
        end
    end
end
return q_min
```

Algorithm 3: $T \leftarrow$Rewire$(T, Q_{near}, q_{min}, q_{rand})$

```
for q_near ∈ Q_near do
    q_path ←Steer(q_rand, q_near)
    if ObstacleFree(q_path) and Cost(q_rand) + c(q_path) <
     Cost(q_near) then
        T ←ReConnect(q_rand, q_near, T)
    end
end
return T
```

Fig. 1. RRT* Algorithm, taken from [11]

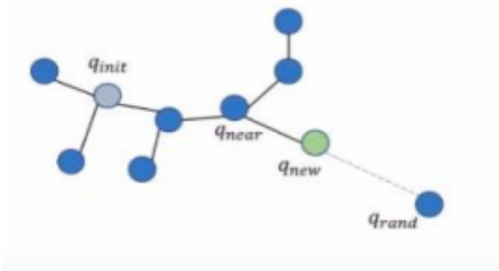

Fig. 2. Generating Random Node

D. Finding a new node:

After finding the nearest neighbour node, the next step is finding the new node. The new node is found by moving a distance of step size from the nearest neighbour node towards the random node. Assuming the nearest neighbour node to be (x_0, y_0), the new node (x, y) is found by the following equations:

$$x = x_0 \pm l\sqrt{\frac{1}{1+m^2}} \tag{1}$$

$$y = y_0 \pm ml\sqrt{\frac{1}{1+m^2}} \tag{2}$$

E. Finding the best parent:

Using a multiple of the step factor, a neighbourhood region around the new node is found. All the nodes that are the part of the graph and lie in the neighbourhood region are added to the list. The node with the least euclidean distance with the new node is considered to be the updated nearest neighbour node for the new node found in step 4.

F. Rewiring the graph:

The next step is rewiring the graph. After finding the updated nearest neighbour node for the new node, the list of neighbours are checked for an optimal path from the new node. For the nodes having an optimal path from the new node, the graph is updated.

G. Distance from the goal:

The euclidean distance between the new node and goal node is checked. If the distance is within the goal threshold, the algorithm is terminated and then the path is traced from the new node to the start node. Otherwise, the process is repeated from step 2.

Informed RRT* motion planning algorithm is a variant of RRT* motion planning algorithm. It finds an optimal path from the start node to the goal node and retains the same probabilistic guarantees on completeness and optimality as RRT* while improving the convergence rate and solution quality.
Informed RRT* Motion Planning Algorithm Steps:

1) Like RRT* algorithm, Informed RRT* algorithm searches for the optimal path to a planning problem by incrementally building a tree in state space, T = (V, E), consisting of a set of vertices, $V \subseteq X_{free}$, and edges, $E \subseteq X^2_{free}$. New vertices are added by growing the graph in free space towards randomly selected states. The graph is rewired with each new vertex such that the cost of the nearby vertices are minimized.

2) The algorithm differs from RRT* motion planning algorithm in that once a solution is found, it focuses the search on the part of the planning problem that can improve the solution. It does this through direct sampling of the ellipsoidal heuristic. In the following steps, the added sub-functions are described in detail.

3) SampleUnitBall Sub-Function: This sub-function returns a uniform sample from the volume of a n-ball of unit radius centred at the origin.

4) RotationToWorldFrame Sub-Function: Given two points as the focal points of a hyperellipsoid, this sub-function returns the rotation matrix from the hyperellipsoid aligned frame to the world frame.

5) InGoalRegion Sub-Function: Given a pose, this sub-function returns True if and only if the state is in the goal region, otherwise it returns False.

6) Sample Sub Function: Given two poses, x_{from}, x_{to} X_{free} and a maximum heuristic value, c_{max}, the function Sample (x_{from}, x_{to}, c_{max}) returns independent and identically distributed samples from the state space, x_{new} X, such that the cost of an optimal path between x_{from} and x_{to} that is constrained to go through x_{new} is less than c_{max}.

IV. Implementation

The Informed RRT* motion planning algorithm was used on ROS Turtlebot 2 to navigate in a configuration space consisting of static obstacles. Turtlebot 2 was treated as a rigid robot with the following properties:

1) The clearance, i.e the distance of the rigid robot from the obstacle was defined as 25 cm.

2) The radius of the Turtlebot 2 robot was taken as 20 cm.

3) The distance between the wheels of the Turtlebot 2 robot was taken as 34 cm.

4) The wheel radius of the Turtlebot 2 robot was taken as 3.8 cm.

Before implementing the motion planning algorithms, the configuration space for the rigid robot had to be defined. The configuration space is shown in Figure 5. Next step was implementing the Informed RRT* and RRT* motion planning algorithms in Python. The algorithms were compared on the basis of path generated and the time taken from the start node to the goal node in the given configuration space. Two test cases were taken which are given below and the results obtained are shown in Figures 7-14.

1) Test case 1: Start Node - (400, 300), Goal Node - (0, 400)

2) Test case 2: Start Node - (400, 300), Goal Node - (-400, -300)

Algorithm 1: Informed RRT*$(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}})$

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

Fig. 3. Informed RRT* Algorithm, taken from [6]

Algorithm 2: $\texttt{Sample}(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}}, c_{\max})$

if $c_{\max} < \infty$ **then**
  $c_{\min} \leftarrow ||\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}}||_2$;
  $\mathbf{x}_{\text{centre}} \leftarrow (\mathbf{x}_{\text{start}} + \mathbf{x}_{\text{goal}})/2$;
  $\mathbf{C} \leftarrow \texttt{RotationToWorldFrame}(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}})$;
  $r_1 \leftarrow c_{\max}/2$;
  $\{r_i\}_{i=2,\dots,n} \leftarrow \left(\sqrt{c_{\max}^2 - c_{\min}^2}\right)/2$;
  $\mathbf{L} \leftarrow \text{diag}\{r_1, r_2, \dots, r_n\}$;
  $\mathbf{x}_{\text{ball}} \leftarrow \texttt{SampleUnitNBall}$;
  $\mathbf{x}_{\text{rand}} \leftarrow (\mathbf{C}\mathbf{L}\mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{centre}}) \cap X$;
else
  $\mathbf{x}_{\text{rand}} \sim \mathcal{U}(X)$;
return $\mathbf{x}_{\text{rand}}$;

Fig. 4. Sub-Functions of the Informed RRT* Algorithm, taken from [6]

After testing the correctness of the algorithms in Python, the algorithms were deployed on Turtlebot 2 using Robot Operating System (ROS). The path generated by the algorithms was published on the ROS Topic, namely "/move_base_simple/goal", to move the Turtlebot 2 from a given point to another point.

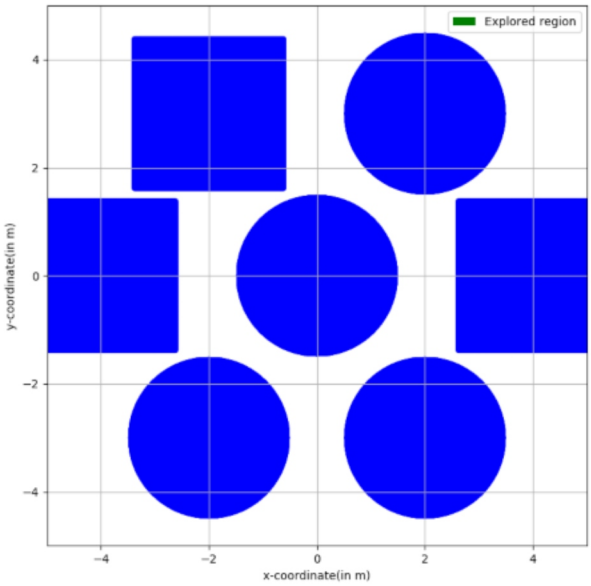

Fig. 5. Configuration Space for the rigid robot

Fig. 6. Turtlebot 2

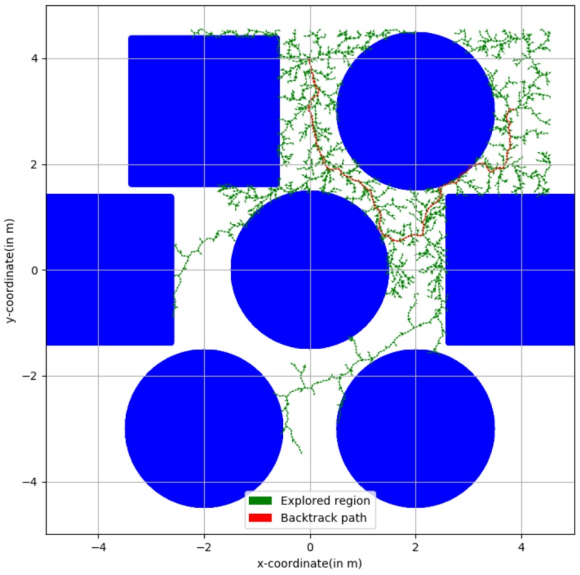

Fig. 7. Informed RRT* Algorithm for Test case 1, Iterations: 10000, Path Length = 6.2 metres, Time taken: 26 seconds

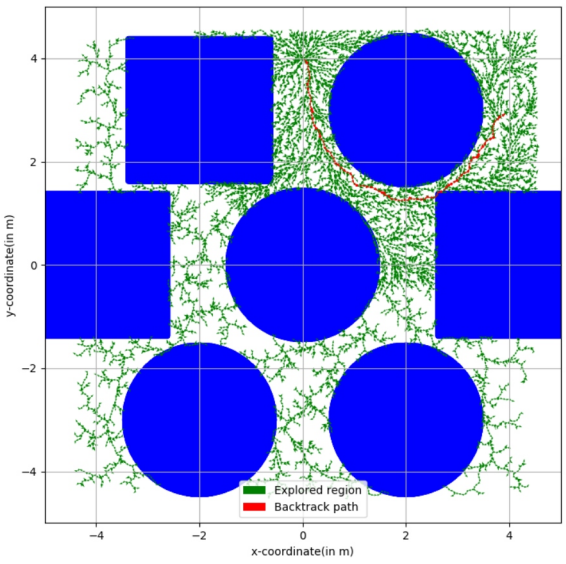

Fig. 8. Informed RRT* Algorithm for Test case 1, Iterations: 25000, Path Length = 5.8 metres, Time taken: 121 seconds

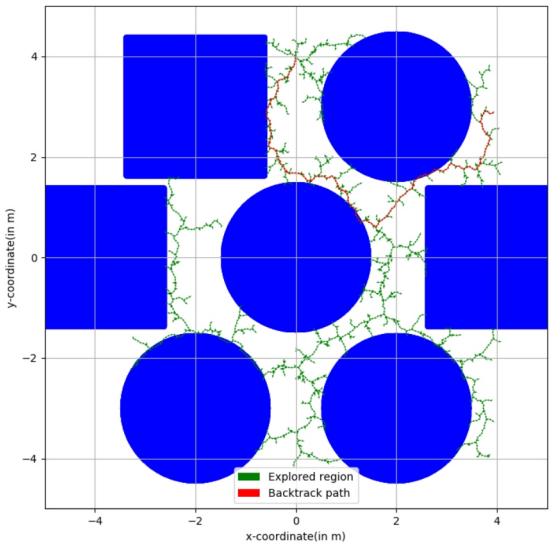

Fig. 9. RRT* Algorithm for Test case 1, Iterations: 10000, Path Length = 7.1 metres, Time taken: 16 seconds

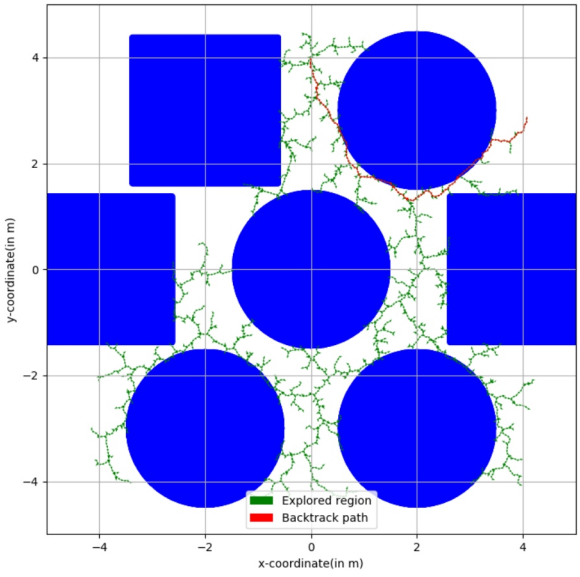

Fig. 10. RRT* Algorithm for Test case 1, Iterations: 25000, Path Length = 6.5 metres, Time taken: 23 seconds

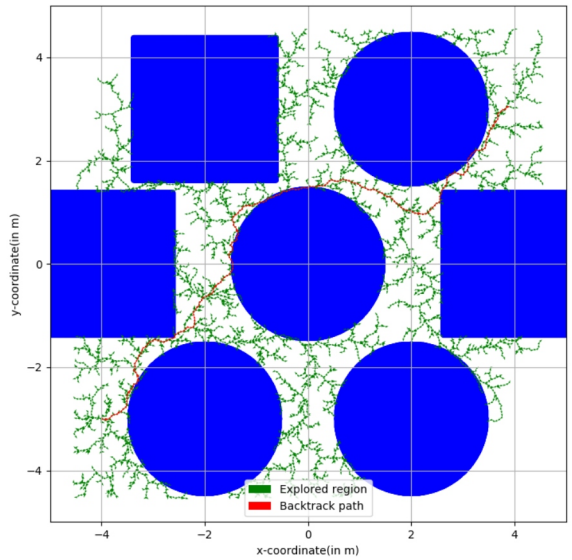

Fig. 11. Informed RRT* Algorithm for Test case 2, Iterations: 10000, Path Length = 15.8 metres, Time taken: 29 seconds

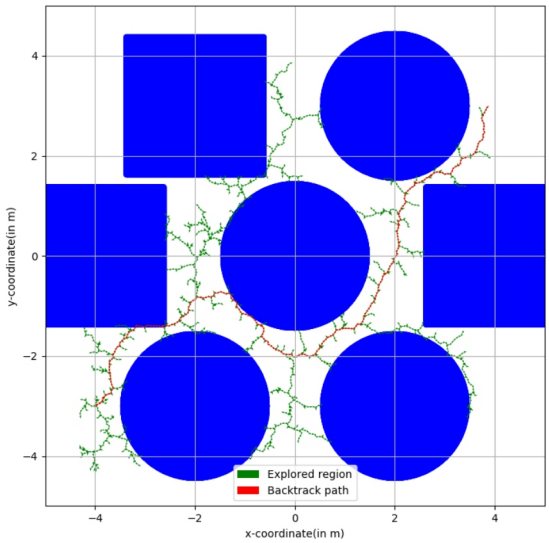

Fig. 13. RRT* Algorithm for Test case 2, Iterations: 10000, Path Length = 16.6 metres, Time taken: 19 seconds

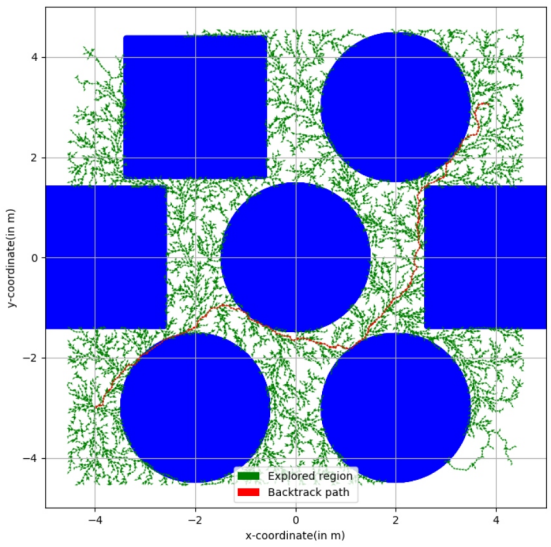

Fig. 12. Informed RRT* Algorithm for Test case 2, Iterations: 25000, Path Length = 15.5 metres, Time taken: 176 seconds

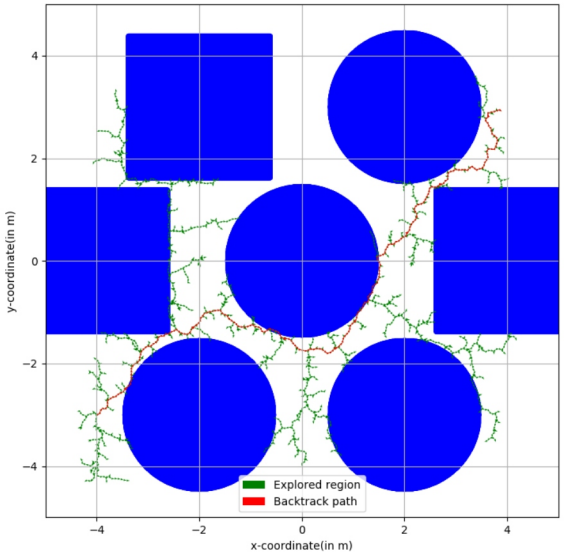

Fig. 14. RRT* Algorithm for Test case 2, Iterations: 25000, Path Length = 16.4 metres, Time taken: 25 seconds

V. Results

The Informed RRT* and RRT* motion planning algorithms were used on ROS Turtlebot 2 to navigate in a configuration space consisting of static obstacles. The configuration space was loaded using Gazebo and is shown in Figure 15. As shown in Figures 7-14, the algorithms were compared on two test cases on the basis of optimal path generated and time taken to move from the start node to the goal node in the configuration space.

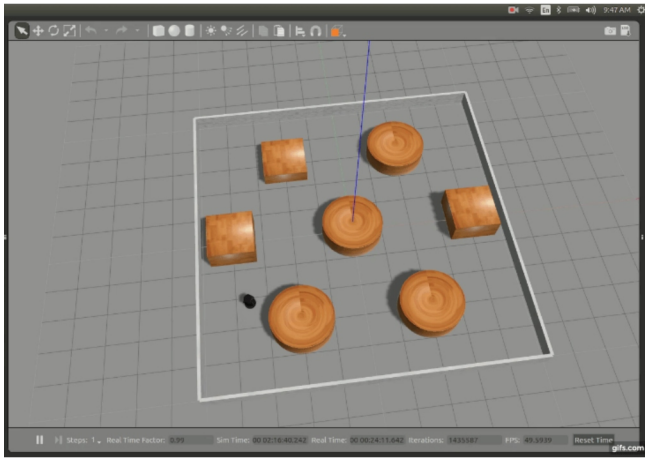

Fig. 15. Configuration space in Gazebo simulation

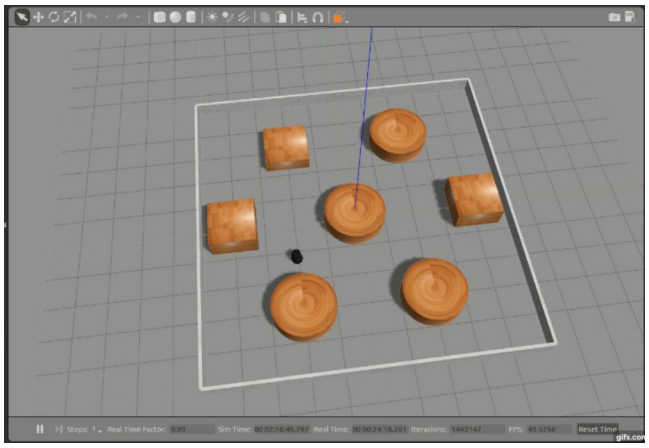

Fig. 16. Informed RRT* Algorithm on ROS Turtlebot 2

VI. Conclusion

In this paper, we examined the Informed RRT* and RRT* sampling-based motion planning algorithms for rigid robots operating in the static configuration environment. The algorithms were first implemented in Python and then they were deployed on Turtlebot 2 in the static configuration environment. From the simulations, we can see that as the number of iterations increase, the Informed RRT* sampling-based algorithm gives an optimal path between the beginning and objective point. On comparing the algorithm with the RRT* sampling-based algorithm on the basis of optimal path and time taken to move from the beginning point and objective point, we can infer that Informed RRT* sampling-based algorithm outperformed the RRT* sampling-based algorithm in rate of convergence, final solution cost, and ability to find difficult paths while demonstrating less dependence on the state dimension and range of the motion planning problem.

References

[1] L. Chang-an, C. Jin-gang, L. Guo-dong, and L. Chunyang, "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.

[2] S. M. Lavalle, Planning Algorithms: Cambridge University Press, 2006.

[3] S. Karaman, and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning", The International Journal of Robotics Research, vol. 30, pp. 846-894, 2011.

[4] 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.

[5] M. Elbanhawi, and M. Simic, "Sampling-Based Robot Motion Planning: A Review survey", IEEE Access, vol. 2, pp. 56-77, 2014.

[6] Gammell, J., Srinivasa, S., Barfoot, T. (2014). Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. IEEE/RSJ International Conference on Intelligent Robots and Systems.

[7] Noreen, I., Khan, A., Habib, Z. (2016). A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms. IJCSNS International Journal of Computer Science and Network Security, 16(10), 20-27.

[8] Karaman, S., Walter, M., Perez, A., Frazzoli, E., Teller, S. (2011). Anytime motion planning using the RRT*. ICRA, 1478–1483.

[9] H. Pan and J. Zhang, "Extending RRT for Robot Motion Planning with SLAM," Trans Tech Publications, Switzerland. Applied Mechanics and Materials vol. 151 (2012) pp. 493-497.

[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] Dynamic path planning and replanning with RRT*, Devin Connel, Hung Manh La.