Efficient Path Planning for Quadrotor/Mobile Robots using RRT with Safety Certificates

Lokesh Bansal
RBCCPS
Indian Institute of Science, Bangalore
lokeshbansal@iisc.ac.in

Suprakas Saren
Aerospace Department
Indian Institute of Science, Bangalore
suprakass@iisc.ac.in

Vishwas Banavara Chandramouly
RBCCPS
Indian Institute of Science, Bangalore
vishwasc@iisc.ac.in

***Abstract*—This paper uses an efficient collision checking procedure for path planning of Quadrotors/mobile robots to avoid static obstacles and reach the goal. The collision checking process for RRT, a sampling-based motion planning algorithm, is done by creating safety certificates around the sample nodes and then finding a viable path to the goal point through those safety certificates. At first, the problem is solved by considering a point mass robot. This problem is extended to the multi-agent system, where a collection of robots performs specific tasks (such as carrying a glass of fixed dimension from one place to another). Numerical simulations and demonstrations show that the method gives a feasible path by efficiently avoiding obstacles for both cases.**

***Index Terms*—RRT, safety certificates, path planning, Quadrotor**

I. Introduction

Path planning is searching for a possible collision-free path within a goal region from any start point. Several studies have been done to solve such problems and applied in various fields such as industry robot locomotion, aerospace, underwater exploration, and autonomous driving. There are several algorithms available to consider, but they should satisfy some essential criteria, such as

- The path planning algorithm should work on the actual robot in the practical workspace.
- The path planning algorithm should give a path close to optimal.
- The path planning algorithm must consider the obstacles present in the workspace.

One standard algorithm that satisfies all the above criteria is Rapidly-exploring Random Trees (RRT)[5]. It is an online, single-query based incremental sampling-based planning algorithm that generates a tree whose root is at the starting point. Random points are sampled within a known environment except for obstacle points. The tree moves towards the nearest neighbour point via a connection line and grows as more points are sampled. The tree will stop growing when it finds a viable path connecting the start to the goal point in the workspace. In sampling-based algorithms many incremental variations have also been proposed such as Rapidly exploring Random Graphs (RRG). This variation not only guarantees optimal solution but also improves the convergence rate for the optimal solution [1]. In the given work the collision checking is extended to the application of efficient collision checking for multi-robot assembly. A fast collision checking technique for moving polyhedral objects is proposed in the paper [6]. An approach for fast collision checking for the multi-robot teams are proposed in [2].

Nearest-neighbour searches within the graph constructed by the algorithms are the computational bottleneck for sampling-based motion planning algorithms. The same is true for RRT, as the computation complication occurs while finding the nearest neighbour to expand the tree. However, the high complexity of collision checking has prevented it from being realized in practice, as seen in the previous works. So, this paper uses a certificate-based approach to avoid the complexity of collision checking in sampling-based motion planning algorithms [3]. A certificate is defined around a collision-free point, containing the subset of free space around that point. New points sampled within the certificate can skip the normal collision checking because they are guaranteed to be collision-free.

Similarly, the subset of any trajectory passing through this certificate will also be collision-free. The procedure of the safety certificate algorithm is shown in Fig.1. The green region is defined as a safe region, and any sampled point from this region is implicitly declared a collision-free point. The black shapes are obstacle regions.

As the number of nodes increases, more and more configuration space is covered and declared safe regions by safety certificates, and for any nodes sampled in that region collision checking is not required. If all the space is covered then the number of collision checks required tends to zero as shown in Fig. 2.

Any kind of sampling-based algorithm such as RRT and PRM[4] can be modified by using safety certificates to find a viable path for a mobile robot by avoiding static obstacles.

The path obtained by modified RRT with safety certificates is used to solve two problems:

1) *Path planning for a single Quadrotor:* Using RRT with safety certificate to generate a trajectory for avoiding static obstacles and reaching a static target position. Here the safety certificates generated is by assuming that Quadrotor is a point object without considering its dimensions.
2) *Path planning for multiple robots carrying a fragile object:* Two robots with fixed formation are carrying

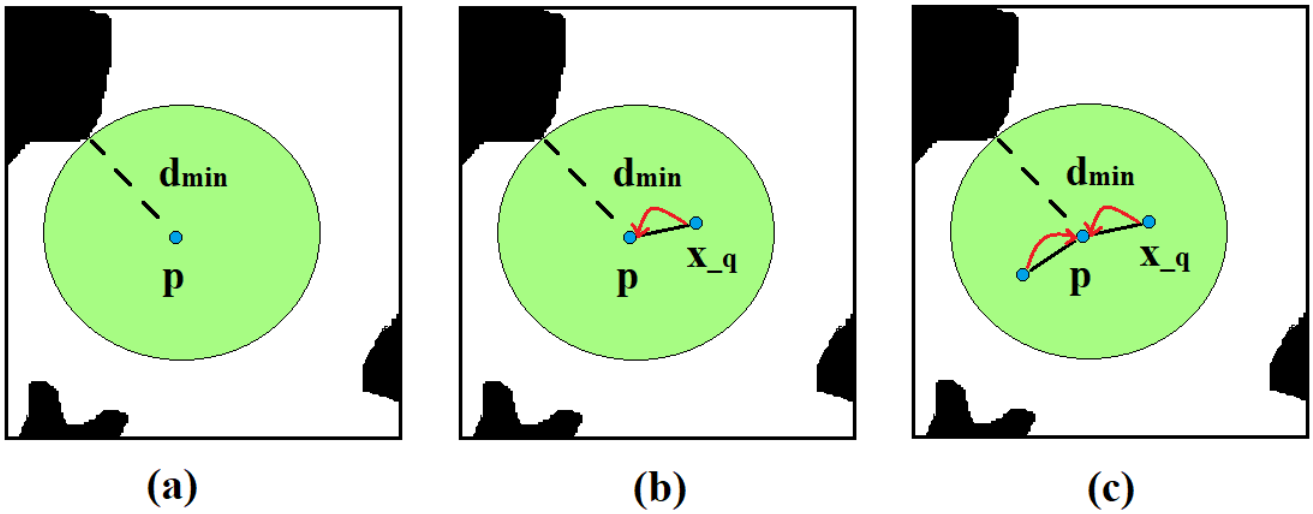


Fig. 1: (a) "Safety certificates" (green) defined by d_{min} (Radius denoted as dashed black lines) for point p. (b) Next sampled node, x_q, is within a certificate need not have collision checking. (c) Pointers (solid red arrows) are maintained to certifying nodes. used

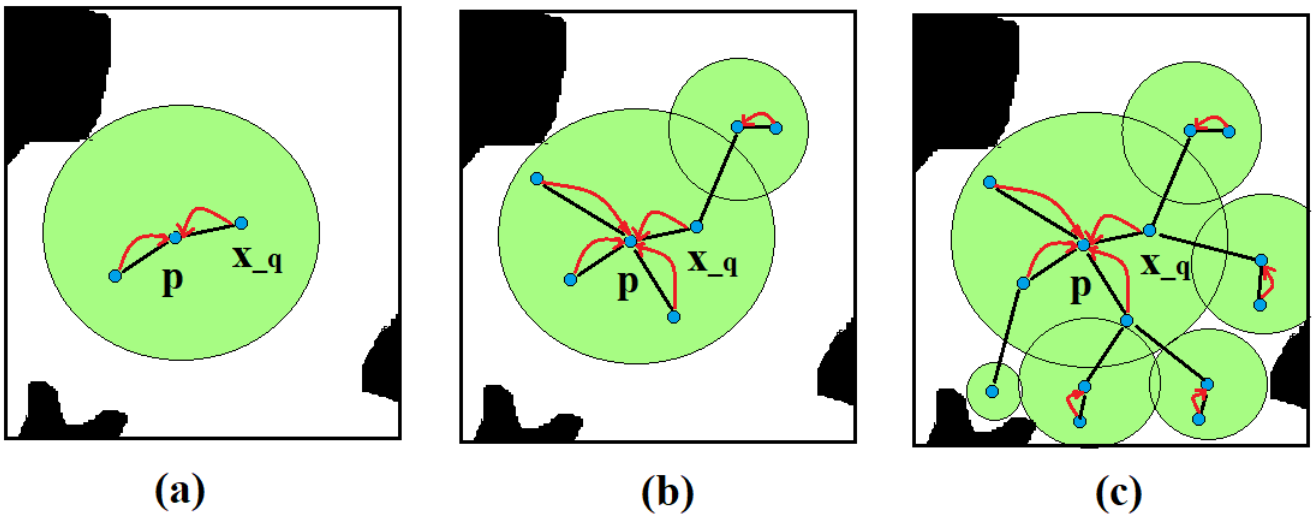


Fig. 2: Certificates (green discs) asymptotically cover the space as graph size increases toward infinity. The ratio of collision checks versus graph size (nodes) approaches zero in the limit as graph size approaches infinity.

a fragile object of known dimensions should reach the target position avoiding all the static obstacles. Here the safety certificates generated is by considering the dimensions of the robots and the fragile object.

The rest of this paper is organized as follows. Modified RRT with safety certificates is discussed in section II. Section III explains Path planning for Quadrotor using Modified RRT and Simulation in VREP. Section IV explains modification of safety certificates to consider the robot not as a point object but as a formation of robots with some dimensions. Section V presents the results. Conclusion and future work is presented in Section VI.

II. Modified RRT with safety certificates:

The configuration space, $X \subset R^D$, is the set of all configurations of the robotic system. X_{obs} , X_{start} , and X_{goal} are obstacle set, initial set, and goal set respectively. The free space is given by X_{free} = cl $(X \setminus X_{obs})$, where cl $(\cdot)$ denotes set closure.

ω is an infinite sample sequence such that x $\in$X for all x $\in \omega$. S in general, denotes a set of points.

V and E are vertex and edge sets, V $\subset X_{free}$ and E $\subset$ V $\times$ V.

The sets $S_{free} \in X_{free}$ and $S_{obs} \in X_{obs}$, are sets of points for which an explicit collision check has been made. Dist(x)

for x$\in$ X defines the radius of the ball that is used for the certificate.

In this paper, our problem is divided into two parts. In the first part, we have to define a path planning procedure for mobile robots (i.e., Quadrotor) by modifying RRT with safety certificates to avoid obstacles and reach a static goal. Then in the second part, we will have multiple robots that work collaboratively to move about the path by maintaining a safe distance between each other and reaching the goal.

A. Premitive Subroutines

1) *Sampling*. ω is an infinite sequence of samples from X . Sample$_i(\omega)$ returns the ith element of ω.
2) *Nearest neighbors*: The Subroutine Nearest (S, x) returns the member of the point set S that is closest to x, i.e. Nearest : (S, x) $\rightarrow$ s $\in$ S such that

$$Nearest(S, x) = \underset{s \in S}{\operatorname{argmin}} ||x - s|| \tag{1}$$

3) *Set Distance:* The Subroutine SetDistance(S, x) returns a non-trivial lower bound on the minimum distance from a point x to S, i.e. for some $\alpha \in$ (0, 1]. We assume that S $\subset$ X is a closed set and that the point x $\in$ X . Formally,

$$\alpha \min_{s \in S} ||s - x|| \leq SetDistance(S, x) \leq \min_{s \in S} ||s - x|| \tag{2}$$

4) *Furthest safe point along a trajectory:* The Subroutine FarSafe(x, y) returns the furthest point along the trajectory [x, y] that is certified by the same certificate as x (without leaving and then re-entering that certificate).
5) *Certifying node:* The Subroutine CertifierOf (x) returns the point y such that y certified x when the node located at x was added to RRT tree.
6) *Segment collision test:* The Subroutine CFreePath(x, y) returns True if the path segment between points x and y lies in X$_{free}$, otherwise it returns False.

B. Modified collision-checking procedures:

Collision-checking procedures are shown in Algorithms 1 and 2, respectively. Algorithm 1 is used for checking whether sampled node lies in safe region or obstacle region. Algorithm 2 is used for checking whether the edge connecting two nodes is collision free or not.

Algorithm 3 is RRT algorithm where collision checking procedure is modified using Algorithm 1 and 2.

1) Algorithm 1: Point collision test: If any new node x$_q$ is sampled, algorithm 1 will check whether it lies within any certificate or not, if it lies inside any safety certificate, algorithm returns true. If it does not lie in any safety certificate, explicit collision checking is done for that node and hence checks whether the sampled node is inside obstacle or not. If the node lies within obstacle space, X$_{obs}$ the algorithm return false value, if it does not lie within obstacle space X$_{obs}$, it is declared as collision free and a new safety certificate is created with the node itself as the center and its shortest distance to nearby obstacle is taken as the radius of the safety certificate and returns true.

Algorithm 1: $CFreePoint(x_q)$

```
x_free ← Nearest(S_free, x_q);
x_obs ← Nearest(S_obs, x_q);
if ||x_q − x_free|| ≤ Dist(x_free) then
    CertifierOf(x_q) ← x_free;
    return True
else
    if ||x_q − x_obs|| ≤ Dist(x_obs) then
        return False
    end
end
d_obs ← SetDistance(X_obs, x_q);
if d_obs > 0 then
    S_free ← S_free ∪ x_q;
    Dist(x_q) ← d_obs;
    CertifierOf(x_q) ← x_q;
    return True
else
    S_obs ← S_obs ∪ x_q;
    Dist(x_q) ← SetDistance(X_free, x_q);
    return False
end
```

Algorithm 2: $BatchCFreePath(S_{near}, x_q)$

```
H ← ∅
c_q ←CertifierOf(x_q);
if ||x_q − x_free|| ≤ Dist(x_free) then
    H ← H ∪ (x, x_q);
else
    c ←CertifierOf(x_q);
    return False
    if ||FarSafe(x_q, x) − c|| ≤ Dist(c) then
        H ← H ∪ (x, x_q);
    else
        if CFreePath(S_near,x_q) then
        H ← H ∪ (x, x_q);
    end
end
```

2) Algorithm 2: Path set collision test: If the sampled node x$_q$ is declared as collision free, from algorithm 1, the next task is to check whether the edge connecting the node and its nearest neighbor, y (which is already in safe region) is collision free or not. Algorithm 2 does this job. There are three cases. First case if both the sampled node x$_q$ and its nearest neighbor, y is within a safety certificate algorithm returns true. Second case is if sampled node x$_q$ and its nearest neighbor, y are in different but overlapping safety certificate. Overlapping is guaranteed such that if FarSafe(x$_q$,y) point is certified by y certifier. These two cases are explained by taking nodes as x$_1$ and x$_2$ instead of x$_q$ and y in Fig.3 (taken from [3]). Third case if both safety certificates are not overlapping. It means entire edge is not in safe region, so explicit collision check

for the edge is done. CFreePath(x_q,y) returns True if the path segment between points x and y lies is collision free. If there is obstacle in between it returns false.

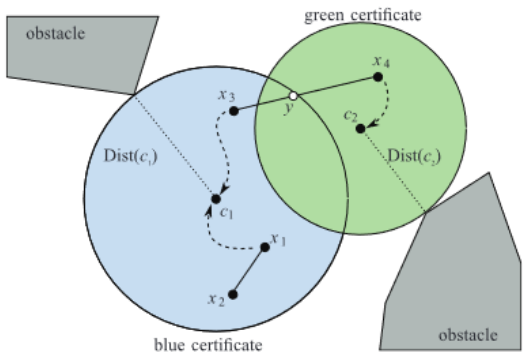


Fig. 3: First case: Path segment [x_1, x_2] is guaranteed to be collision-free because it exists completely within a single certificate. Second case: Path segment [x_2, x_4] is guaranteed to be collision-free because it exists within the union of both certificates

Algorithm 3: Modified RRT

V $\leftarrow x_{init}$; $E \leftarrow \emptyset$; $S_{free} \leftarrow \emptyset$; $S_{obs} \leftarrow \emptyset$
for *i = 1,....,n-1* **do**
  $x_{rand} \leftarrow Sample(\omega)$;
  if *CFreePoint*(x_{rand}) **then**
    $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
    $H \leftarrow BatchFreePath(x_{nearest}, x_{rand})$;
    if $H \neq \emptyset$ **then**
      $V \leftarrow V \cup x_{rand}$
      $E \leftarrow E \cup H$ **end**
    end
  end

If sampled node x_q is not lying inside obstacle (validated by algorithm 1) and the edge joining x_q and its nearest neigbor y, is also collision free (validated by algorithm 2), then x_q is added as a new Vertex to RRT tree, and the edge is also added. y is updated as x_q's parent node. Like this the RRT tree keeps expanding and it can be stopped if required number or nodes are sampled or if the goal is reached.

C. Implementation of Modified RRT in Matlab

We considered a 2-D configuration space (100 x 100 units) with obstacles as shown in Fig.4. We can observe in (a) that there is a safety certificate created in around start node. In (b) a random sample node is generated and is within safety certificate. In (c) a random node generated is outside safety certificate and it is not inside obstacle, so a new certificate is formed around it. In (d) the path generated from modified RRT with safety certificate can be seen.

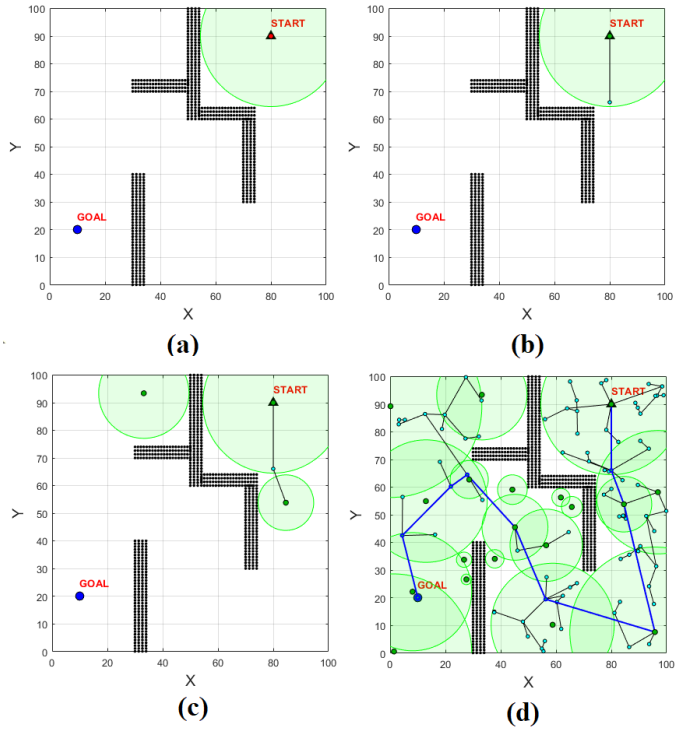


Fig. 4: Implementation of Modified RRT Algorithm in Matlab

III. Path Planning for Quadrotor using Modified RRT and Simulation in VREP

Our aim is to demonstrate the efficient path planning using RRT with safety certificates. For this demonstration path planning is carried out for a Quadrotor to move the Quadrotor from a start location to a static goal location in static obstacles environment. For this, we have Implemented a 6 DOF Quadrotor with PD Controller for trajectory tracking for Quadrotor. Simulation have been done in MATLAB and CoppeliaSim (VREP). VREP is a 3D robot simulation software with an integrated development environment that allows us to model, edit, program, and simulate any robot or robotic system. We used MATLAB to do the computational aspects for path planning and VREP is used as a visualization platform. In addition to this, we applied this algorithm to a six DOF Quadrotor whose modelling is done in MATLAB.

The program can be divided into two. MATLAB implementation, which is the brain of the model and V-REP for modelling and executing the algorithms. In V-REP, the environment for the simulation of the drone has been set up. There are obstacles of different sizes. The source robot (Quadrotor) is using the RRT with Safety certification to get the obstacle free trajectory to reach a static goal. MATLAB scripts is of 2 parts. In the first part, the RRT with Safety certification algorithm, is implemented, in which the environment skeleton is hard coded. The communication between MATLAB and V-REP is running continuously to get the visualization of the

drone traversing. The RRT with Safety certification function is accepting the start and final target coordinates. The algorithm ensures avoidance of any possibility of collision of the drone with the obstacles. In the second part, the control program executed the same RRT with Safety certification function and displayed the trajectory followed by the drone from the start of the program.

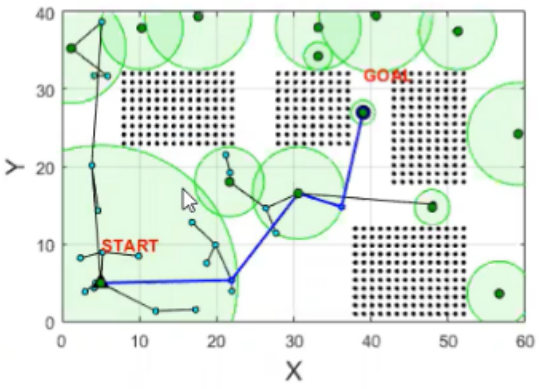


Fig. 5: Path found for Quadrotor

IV. Path planning for multiple robots carrying fragile object

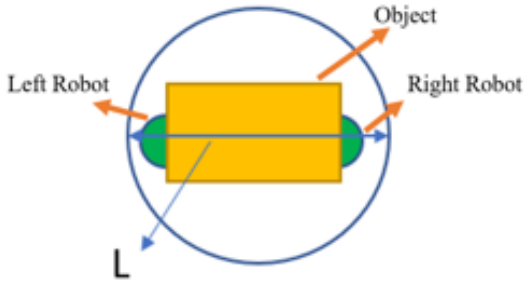


Fig. 6: Two robots carrying fragile object

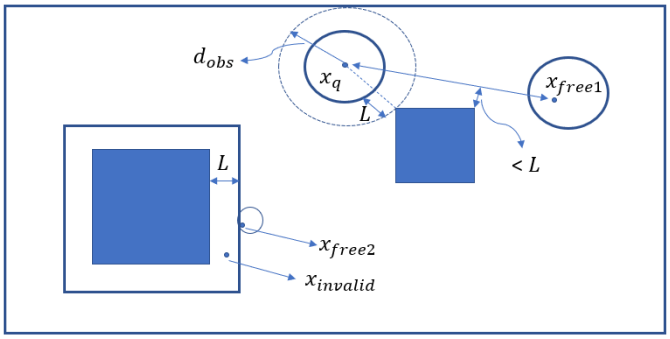


Fig. 7: Modified Safety certificates for multi-robot's assembly

A. Modification of safety certificate to accommodate multiple robot

Consider two robot carrying a fragile object as shown in Fig.6. To account for the robot's dimension, consider the robot's maximum dimension from the centroid of the robot as 'L' m. We consider a circle of diameter 'L' around the centroid of the robot, which is a certificate of the robot. Now, we can modify the above algorithm as follows The overall

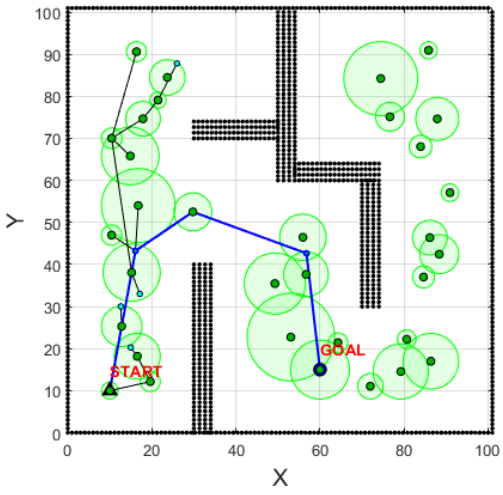
Algorithm 4: $ModifiedCFreePoint(x_q)$

$x_{free} \leftarrow Nearest(S_{free}, x_q)$;
$x_{obs} \leftarrow Nearest(S_{obs}, x_q)$;
if $\|x_q - x_{free}\| \leq Dist(x_{free})$ **then**
  $CertifierOf(x_q) \leftarrow x_{free}$;
  return True
else
  if $\|x_q - x_{obs}\| \leq Dist(x_{obs})$ **then**
    return False
  end
end
$d_{obs} \leftarrow SetDistance(X_{obs,x_q})$;
if $d_{obs} > L$ **then**
  $d_{obs} \leftarrow d_{obs} - L$;
  $S_{free} \leftarrow S_{free} \cup x_q$;
  $Dist(x_q) \leftarrow d_{obs}$;
  $CertifierOf(x_q) \leftarrow x_q$;
  return True
else
  if $d_{obs} < marginalvalue$ **then**
    $S_{obs} \leftarrow S_{obs} \cup x_q$;
    return False
  end
end
else $invalid \leftarrow invalid \cup x_q$;

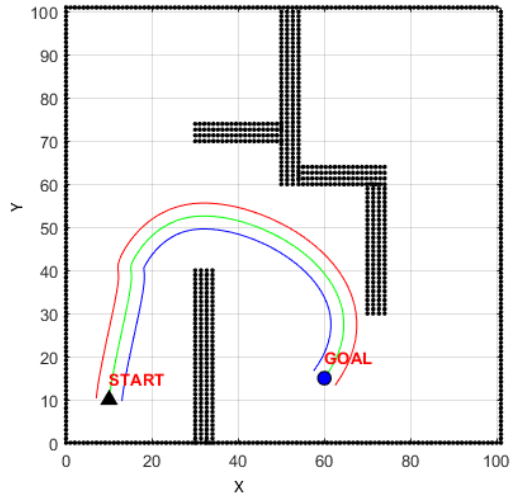
Algorithm 5: $ModifiedCFreePatht(x, x_q)$

$Min_{Distance} \leftarrow argmin\|(x, y) - x_{obs}\|$;
if $Min_{Distance} > L$ **then**
  $H \leftarrow H \cup (x, x_q)$;
  return True
else
  return False
end

procedure is explained in the above Fig.7 Basically, we take a query point x_q and will find the minimum distance from every obstacle. After finding the minimum distance, we compare this minimum distance with the multi-robot system's greatest dimension and check whether the distance is greater than the robot's max dimension. If it is smaller than the multi-robot system's greatest dimension, we will not consider the point for further iteration and put it into an invalid list. As iteration continues, more samples will be put into an invalid



(a) Path obtained from RRT



(b) Confidence Interval of smooth path

Fig. 8: Path found for multirobot system

list, creating an imaginary boundary of distance L from the obstacle. Next, if the distance is greater than the multi-robot system's greatest dimension, we will create a safety certificate of radius [d_{obs}-L], and its certifier will be x_q. As iteration goes on, more safety certificates will generate, and our next task will be to create a path through those collision-free points inside the safety certificates. When generating the edge connecting two free nodes such that the two nodes lie inside non-overlapping safety certificates then, we discretize the edge into discrete points, and for each point, we check the minimum distance from the obstacles. If the minimum distance found is smaller than the multi-robot system's greatest dimension, we will not consider the path between them and look for other paths.

V. Results

For the demonstration of the path planning by the improved RRT with safety certificates, path planning for Quadrotor is performed. We performed computation in MATLAB for node sampling and execution of improved RRT for trajectory generation. The generated trajectory is fed in VREP simulator where a map is created with static obstacles. The task is to move the Quadrotor from start location to goal location by avoiding obstacles. Efficient path planning and simulation has been executed successfully.

A. Robot as single mass system

In the Fig.8, node sampling is visualized from sampling initialization to connecting the nodes from start to goal. The safety certificates are represented by green circles. Any node sampled inside the green circles are implicitly declared to be collision free. The green dots are sampled inside the safety certificates and no explicit collision checking is required for them. Finally, collision free path represented by blue line is generated via collision free nodes as explained in the algorithm. Further this path is fed into VREP simulator. The VREP environment is shown in Fig.9a and Fig.9b. In Fig.9a the Quadrotor is starting from start node and in the Fig.9b the Quadrotor has reached to goal by following the trajectory generated by the improved RRT algorithm. In addition to VREP simulation we also visualized the Quadrotor motion in MATLAB as shown in Fig.9c by giving the trajectory to a trajectory tracking module for the Quadrotor. In this module 6 degree of freedom model is used with PD controller to track the trajectory generated by the improved RRT.

To demonstrate the reduction in execution time due to the efficient collision checking by the safety certificates, a comparison is executed between RRT with safety certificates (SC) and RRT without safety certificates (SC).

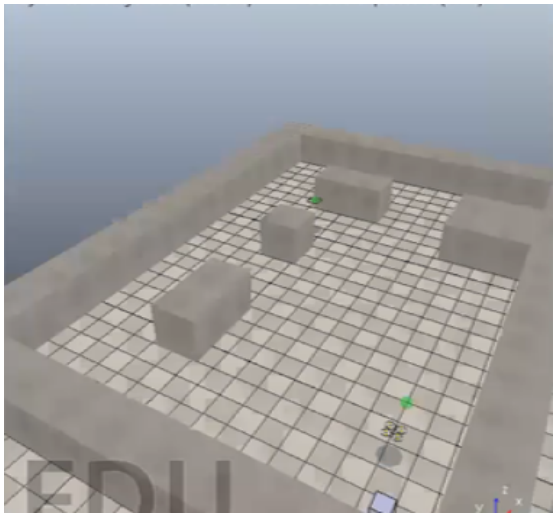
In the absence of goal node, both the algorithms are executed to sample 100, 500, 750 and 1000 nodes randomly. In the Fig.10 in the comparison curves it can be seen that the run time is significantly reduced because of the efficient collision checking by safety certificates. Run time is reduced by approximately 6 times for sampling 1000 nodes.

B. Multi-robot assembly

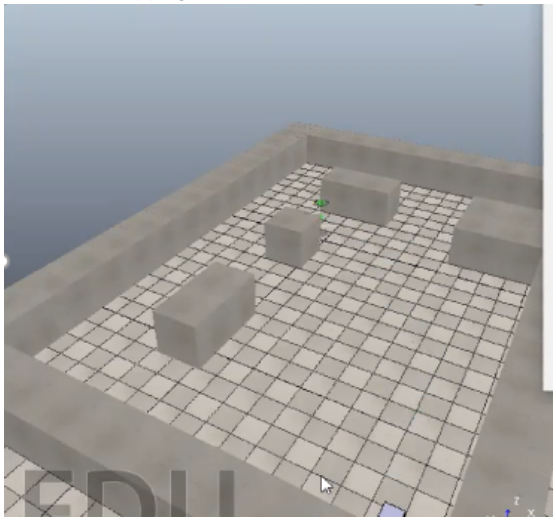
The application of the algorithm RRT with safety certificates is further extended to get a collision free path for a multi-robot assembly deployed to carry a fragile object in static environment. For this, two identical ground robots are considered to be deployed in a maze representing warehouse having static obstacles. The robots are assumed carrying a rectangular object of length 'l', breadth 'b' and diagonal 'L'. the two robots are geometrically similar having circular shape of radius r such that:

$$r + \frac{l}{2} \leq \frac{L}{2} \tag{3}$$

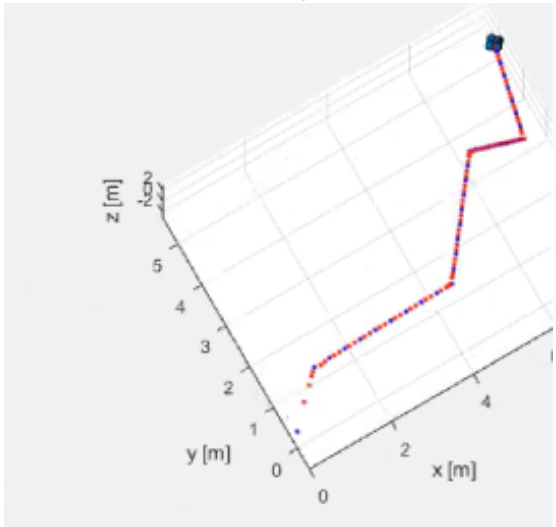
It can be seen in Fig.8a the safety certificates for this case are at least L unit away from the obstacles instead of zero unit as was the case for the previous problem and the edges



(a) Quadrotor at start location



(b) Quadrotor at goal location



(c) Matlab visualization with path from start to goal

Fig. 9: VREP simulation of the Quadrotor

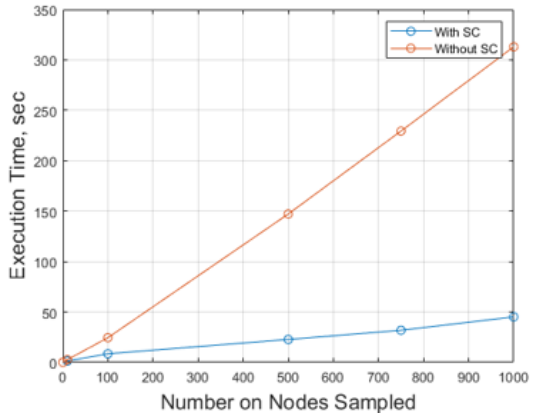


Fig. 10: Execution time comparison

connecting any two safe nodes are also at least L distance away from any obstacle. A safe path is successfully found for the multi-agent robot assembly. The generated trajectory represented by blue line would be the path for the centroid location of the multi agent robot assembly. The trajectory for the robots can be generated as per the formation of the robots in the assembly.

For the present implementation the formation is shown in Fig.8b Here two robots are carrying the object in side-by-side formation. The overall end-to-end length of the formation is less than or equal to L as per equation 3.

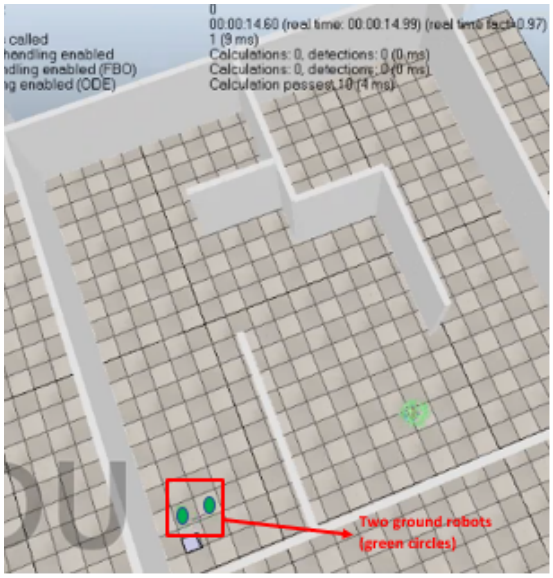
The generated path for the centroid of the assembly is used to generate two smooth and parallel paths one for each robot. The two generated paths are fed to VREP simulator as shown in the Fig.11. In the VREP simulator a map is created representing the grid of the warehouse where the two ground robots are deployed to carry the object. It is demonstrated successfully that the two robots are moving in a fixed formation and reaching to the static goal location by avoiding the static obstacles.

In the given demonstration the two robots are placed underneath the object at the parallel opposite edges of the object. Further, they can be placed at any location within the confidence interval shown in Fig.8b such that the end-to-end length of the assembly is less than or equal to L as per equation 3.

In the Fig.8a the confidence interval is shown. The green curve is the path for the centroid of the assembly, the red curve is the path for the left most location of the left robot and the blue curve is the path for the right most location on the robot. With in this red and left curves any path can be formed for the robots.

VI. Conclusion and Future Work

In this paper, we implemented an improved version of RRT algorithm with safety certificate to avoid static obstacles and reach goal. The simulation results and demonstration on the Quadrotor shows that the path obtained from the algorithm is collision-free and also the execution time is less compared



(a) Multi-robot assembly at start location

(b) Matlab visualization with path from start to goal

(c) Multi-robot assembly at goal location

Fig. 11: VREP simulation of the Multi-robot assembly

to original RRT. This is also extended to the problem where multi-robots are present to do some tasks collaboratively. It is seen that the improved algorithm can generate path efficiently by making a safe distance from the obstacles by considering the dimension of the multi-robot system. This verified that the algorithm can be used for any dimensions and orientation of the multi-robots for path planning.

In this paper the environment is supposed to be static that is the obstacles and goal are not moving. So, this work can be further extended for dynamic environment where we will have moving obstacles or goal and the task for multi-robot system to move from starting point to goal point.

ACKNOWLEDGMENT

The work presented in this report is the result of motivation and help given by our faculties, family, friends and relatives. We are highly thankful to Dr. Debasish Ghose, Professor, Aerospace Department, Indian Institute of Science for his guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project. We would like to express our gratitude towards Dr. Mukunda Bharatheesha, Senior Member Of Technical Staff, ARTPARK, Indian Institute of Science for his kind co-operation and providing inputs for bonus problem which gave a challenging outlook to this project. We would like to thank Mr. Amit Kumar and Mr. Suryadeep Research Scholars, Indian Institute of Science for continuously encouraging and supporting us throughout this journey.

REFERENCES

[1] Arslan, O. and Tsiotras, P. (2013). Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *2013 IEEE International Conference on Robotics and Automation*, pages 2421–2428. IEEE.

[2] Bialkowski, J., Otte, M., and Frazzoli, E. (2013). Fast collision checking: From single robots to multi-robot teams. *arXiv preprint arXiv:1305.2299*.

[3] Bialkowski, J., Otte, M., Karaman, S., and Frazzoli, E. (2016). Efficient collision checking in sampling-based motion planning via safety certificates. *The International Journal of Robotics Research*, 35(7):767–796.

[4] Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894.

[5] LaValle, S. M., Kuffner, J. J., Donald, B., et al. (2001). Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions*, 5:293–308.

[6] Redon, S., Kheddar, A., and Coquillart, S. (2002). Fast continuous collision detection between rigid bodies. In *Computer graphics forum*, volume 21, pages 279–287. Wiley Online Library.