This article can be cited as R. Gayathri and V. Uma, Performance Analysis of Robotic Path Planning Algorithms in a Deterministic Environment, International Journal of Imaging and Robotics, vol. 19, no. 4, pp. 83-108, 2019.
Copyright©2019 by CESER Publications

Performance Analysis of Robotic Path Planning Algorithms in a Deterministic Environment

R. Gayathri[1] **and V. Uma**[2]

Department of Computer Science
Pondicherry University
Puducherry, India
[1]gayathrir339@gmail.com [2]umabskr@gmail.com

ABSTRACT

Motion planning is one of the fundamental challenging problems in the field of robotics. The objective is to find the shortest path traversed by the robot encountering any type of obstacles in the space. A number of intelligent path planning algorithms have been designed for solving motion planning problems. This paper presents the performance analyses of various robotic path planning algorithms for avoiding obstacles in a deterministic environment both quantitatively and qualitatively. This paper also discusses the benefits and drawbacks of each method and shows the simulation results of each algorithm on the explored space. These algorithms are tested under various situations with different degrees of complexities. The results analysis shows that with respect to time and path length, Rapidly Exploring Random Tree (RRT) algorithm perform better in finding the optimal path and reaching the specified target point in a given planning horizon.

Keywords: Path Planning Algorithms, Robotics, Deterministic environment, Planning, Plan representation, Path optimization.

Computing Classification System (CCS): I Computing methodologies, I.2 Artificial Intelligence, I.2.9 Robotics $\rightarrow$ planning.

1 Introduction

Path planning can be defined as finding a path from the initial position to a goal position passing through all the obstacles in the real workspace. The basic motion planning problem is referred to as the "piano movers problem" which is used for searching a collision-free path from the initial state to the goal state (LaValle, 2006). One of the basic capabilities of robotic path planning is autonomous routing that has several applications in the field of self-driving car, vacuum cleaning robot, etc. Path planning research works have been going on in various areas such as robotic automation, space exploration, hydrokinetic turbine, automated harvesters, desalination plants, industrial robotics, military robots, medical robots, mobile robots, computer graphics, video games, active noise control (Ginter and Pieper, 2011),(Haidegger, Kovács, Precup, Benyó, Benyó and Preitl, 2012),(Yacoub, Bambang, Harsoyo and Sarwono, 2014),(Vrkalovic, Lunca and Borlea, 2018),(Aravind, Raja and Pérez Ruiz, 2017),(Chen, Luo, Mei, Yu and Su, 2016) etc.

Robot path planning is concerned with generating the feasible path and avoiding possible collisions with known and unknown obstacles in the state. Path planning defines the trajectories in various state environments such as deterministic, nondeterministic, static, dynamic, discrete, continuous (Mac, Copot, Tran and De Keyser, 2016) etc. Path planning algorithms can be specifically classified as deterministic and nondeterministic state based algorithms considering the knowledge of the robot about its environment (Kavraki, Svestka, Latombe and Overmars, 1996). In the deterministic state, the robot has the complete knowledge about its environment, obstacles, position of start and goal state for execution of a task. The robots in the deterministic environment take the advantage of knowing the environment before execution of a task. This complete knowledge of information can be specified as informative path planning. The robot can avoid collisions with the known obstacles in the state and estimate the feasible path by using the path planning algorithms (Yang, Qi, Song, Xiao, Han and Xia, 2016). The more challenging environment is nondeterministic environment, where the environment is full of ambiguous scenarios such as unstructured background, uncertain factors and unsafe motion faced by the robot.

To generate a suitable path, various robot path planning algorithms are used. Recently, several real world motion planning problems were solved by different approaches namely Sampling based algorithms, Node based algorithms, Bio-inspired algorithms, Mathematical model based algorithms and Multifusion algorithms. A short description of these algorithms is given in Section 2. These algorithms are used for addressing the robot path planning problems in a 3D environment (Yang et al., 2016).

In our previous works (Gayathri and Uma, 2019),(Gayathri and Uma, 2018), we have analyzed the knowledge representation and reasoning techniques for robotic path planning. Knowledge representation and reasoning (KR& R) based planning techniques can represent the structural domain knowledge that aids Description Logic (DL), temporal, ontological and spatial reasoning in a modeled environment. These can be used to successfully accomplish the tasks with lesser computational time.

As an extension of our previous works, this paper mainly focuses on algorithms that can facilitate safe and effortless movement of the robot after tackling the obstacles in the deterministic environment. This paper mainly concentrates on three categories of path planning algorithms namely sampling based algorithms, node based algorithms and bio-inspired algorithms. The performance analyses of these path planning algorithms are discussed in this paper. Hence, these algorithms are tested under various challenging situations and the result analyses demonstrate that these algorithms find the optimal path with different computation time and path lengths. In future, we have plans to integrate our previous works with this work and find the optimal path using the algorithm which performs better with respect to the chosen environment.

The rest of the paper is organized as follows. In section 2, comprehensive discussion of path planning algorithms is presented. In section 3, explains the architecture of robotic path planner. Section 4 provides a comparison between various path planning algorithms and section

5 discusses the benefits and limitations of each algorithm by analyzing the performance of different path planning algorithms. Finally, the paper is concluded in Section 6.

2 Path Planning Algorithms

Various path planning algorithms have been proposed to overcome the path planning problems under different environments (Yang et al., 2016),(Galceran and Carreras, 2013). This section first lists the classification of three intelligent path planning algorithms for planning safe and effortless motions of a robot in a modeled environment. The comprehensive overview of three different path planning algorithms is also presented in this section. Representation of planning domain knowledge is necessary for an agent to act in the environment according to the action sequence. The agent perceives the environment and undergoes a sequence of actions that transform initial state to the goal state (Meyer and Filliat, 2003). Planning domain is represented by the states, actions and goal.

2.1 Classification of Path Planning Algorithms

Although, five approaches of 3D path planning algorithms are available for motion planning problems, in this paper, we discuss the three main classification of 3D algorithms namely sampling based algorithms, node based algorithms and bio-inspired algorithms. This is because multifusion algorithms perform planning by combining the above said algorithms and the performance of mathematical model based algorithms is lower with respect to the time complexity when compared with above said algorithms (Yang et al., 2016). These categories are distinguished from each other by their unique characteristics. For instance, sampling based planner is based on Monte Carlo Sampling approach in taking actions and node based algorithms are based on grid based techniques in achieving the goal. These approaches are used for addressing the path planning problems in deterministic (informative) environment. The taxonomy of path planning algorithms is illustrated in Figure 1.

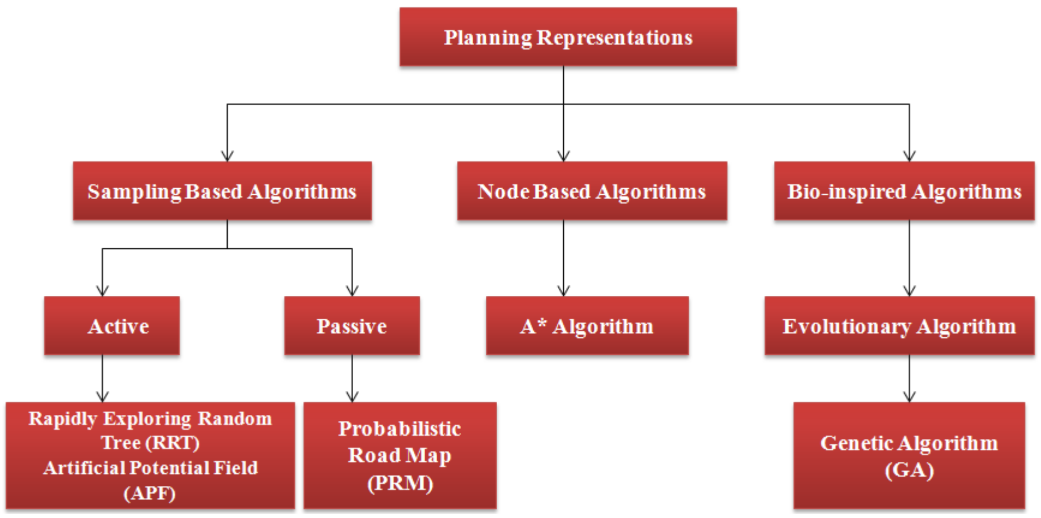


Figure 1: Classification of Path Planing Algorithms

2.2 Sampling Based Algorithms

The principle use of this method is to generate a random sample in the environment as set of nodes, cells or in other forms that search the specific goal point to achieve a feasible path. This algorithm requires the prior knowledge of the environment for the agent to perform a task in the state (Yang et al., 2016). The elements of sampling based algorithms can be separated into two types namely active and passive elements. These elements are based on the roadmap methods. Using active elements, the agent can achieve the best feasible path towards the goal by its own processing procedure. Using passive elements, the agent generates a roadnet map on the workspace that is based on the position of start and goal nodes. As a consequence, combination of these two approaches can select the best feasible path using this road map.

2.2.1 Rapidly Exploring Random Tree (RRT)

RRT is a space-filling tree structure and this algorithm is designed efficiently. The intended RRT algorithm efficiently searches the non-convex high dimensional spaces by using the random sampling approach. The structure of RRT enables a greater exploration from multiple points in the graph and the explorations are combined to produce a well-connected tree structure. RRT intuitively considers Monte-Carlo Sampling approach in generating a random point into the Voronoi regions. These regions are explored by the RRT algorithm in an attempt to generate an optimal path with the minimum path distance.

RRT is extensively used for solving the path planning problems that involve collision, differential constraints (in terms of bi-tree, non-holonomic or kinodynamic constraints). RRT explorer is used to find the sub-optimal path in terms of global or local optimality (Kala, 2013). To avoid obstacles, global optimality technique considers a different strategy (like move from the left, right or in-between). Local optimality technique is used for maintaining the distances from the obstacles. These two techniques are applied in RRT algorithm based on the surroundings so that path is found through the interpolated points in the graph.

Single-RRT The position of start and the goal points are set in the collision-free space of the configuration. Single RRT algorithm generates a tree from the starting point towards the goal point by using random sampling approach. This approach is used to generate random points which are considered as nodes in the graph. Then the algorithm intuitively terminates to generate a tree at certain point by using the stopping configuration method (Knispel and Matousek, 2013). This method runs till the expansion of tree reaches the specified goal point. The pseudocode of RRT is illustrated in Algorithms 1.

Algorithm 1. RRT algorithm

```
Input: Configuration space (C-space)
Output: Path generated from initial node to goal node
Begin
Set the initial node ($V_i \leftarrow q_{start} \in C_{free}$) and goal node ($V_i \leftarrow q_{goal} \in C_{free}$) in configuration space
(C-space).
```

```
    while (! Pathfound) do
        $q_{rand}$ = random_state($V_i \leq P_g$) [$0 \leq P_g \geq 1$]
        $q_{near}$ = min(distancecost($q_{rand}$ , $q_{goal}$ ))
        $q_{new}$ = $q_{near}$ + stepsize
        if checkpath($q_{new}$ , $q_{near}$ ) then
            $V_i \leftarrow q_{new}$ + add _ vertex ($V_i \leftarrow V_1, V_2, V_3, V_4 \dots V_n$)
            $E_i \leftarrow q_{new}$ + add _ edges ($E_i \leftarrow E_1, E_2, E_3, E_4 \dots E_n$)
        end if
        if distancecost $\leq$ Threshold then
            Pathfound = True
        else
            Return
        end if
    end while
end
```

Bi-directional RRT Bi-directional search algorithm is called as the balanced search algorithm that generates a tree simultaneously from the initial positions and from the opposite directions (goal positions). The first tree initiates and grows from the source towards the goal. The second tree initiates and grows from the goal towards source. Then both the trees are stopped at certain point where it meets each other. At this point, the tree finds the best solutions by balancing searches (Knispel and Matousek, 2013). RRT Bi-directional tree search has more power to connect a tree and finds the solution faster than the single tree search RRT algorithm. To connect the two end nodes, path distance is calculated using

$$d(p) = \sum_{i=1}^{N-1} \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

Probabilistic Road Map (PRM) PRM is the probabilistically complete algorithm that finds the solution with a high probability and generates a number of samples in the graph (Kannan, Gupta, Tiwari, Prasad, Khatri and Kala, 2016). PRM is based on the roadmap methods. The roadmap method generates a roadnet from the start state to goal state in the space.

An analysis of PRM operating in configuration space (C-space) be an open subset $[0, 1]^k$ and the distance (d) be the Euclidean metric on C^k space. The local planner for the PRM connects points x,y $\in$ C. The measure p denotes the volume of a region of space that represents p($[0,1]^k$) = 1. If A $\subset$ C is a subset and x is a random point chosen by sampling function (Al-Hmouz, Gulrez and Al-Jumaily, 2004) is

$$Pr(x \in A) = \frac{p(A)}{p(C_{free})} \tag{2}$$

The probabilistic planner generates vast number of nodes and multiple edges in the space. The Probabilistic planner operates in two phases (Mac et al., 2016). First phase is the offline roadmap (learning or construction) phase and the second phase is the online planning (query) phase. In Construction phase, a simple graph is build in the map. A simple graph of map is probabilistic because it selects random points in the collision-free map. The roadmap vertices are connected with all other random point vertices by using the local planning algorithm. A common local planning algorithm is used to connect the sample vertices in a straight line and add edges if the sample points are in the collision-free graph.

In query phase, the roadmap graph is used for planning the path of the robot. During this phase, it finds the optimal path from the start node to goal node by using any of the graph search algorithm (depth first search, breadth first search, heuristic search and replanning methods). Hence, the resulting path of this algorithm is too complicated and it contains a lot of possible redundant moves. The problem is solved by using the RRT algorithm that generates a tree based on the position of start and goal states in the space (Knispel and Matousek, 2013).

Artificial Potential Field (APF) Artificial Potential Field (APF) is widely used in robot path planning problems. APF is based on the reactive planning technique that reacts immediately and takes an instant action that guides the robot to move away from the obstacles given the direction to reach the goal. This technique operates based on two forces (Ahmed, Abdalla and Abed, 2015)

1. Repulsive Force
2. Attractive Force

Repulsive force technique is applied to the mobile robot such that it takes immediate action and thus moves away from the obstacles. All the obstacles repel away from the robot and the force is inversely proportional to the distance from the obstacle. Target point is attracted by the attractive force. The combination of these two forces is called as the total potential force. The total potential force is used for guiding the robot to reach the target point that can be represented as

$$U_t(x) = U_a(x) + U_r(x) \tag{3}$$

In forumula 3, U$_t$ (x) denotes the total potential force at state x, U$_a$(x) is attractive potential force at state x, U$_r$ (x) represents the repulsive potential force at state x.

Additionally, APF uses a scalar function which is called as a potential function that contains two values (Pimenta, Fonseca, Pereira, Mesquita, Silva, Caminhas and Campos, 2005). The first is the minimum value and the second is the maximum value. Minimum value is marked as the target point and the maximum value is marked as obstacles.

The most common problem in the artificial potential field is local minima problems. The local minima problem occurs in the following three main conditions and the sum of all forces is zero (Ahmed et al., 2015).

- The position of the robot, target point and the obstacles are present in the same line where the obstacles are placed in the middle of the robot and the target point.

- The goal point appears within the obstacles region so that the robot can move away from the obstacles by using the repulsive force technique. This problem is commonly known as “Goals non-reachable with obstacles nearby” (Hamani and Hassam, 2012).
- Robot faces a complex environment that contains a non-convex obstacle so that the robot cannot avoid the obstacle and reach the target point under artificial potential field.

These types of problems can be solved by using the chaos optimization method (Chen, Wu and Lu, 2015).

2.2.2 Node based algorithms

The principle method of this algorithm is exploring the decomposed graph to search the optimal path in the graph. The decomposition graph based on grid methods can be divided into three categories namely approximate cell decomposition, exact cell decomposition and adaptive cell decomposition. Node based algorithms explore a set of nodes in the map to find the optimal path according to certain decomposition.

A* algorithm A* is a grid-based search algorithm that controls the robot to obtain the shortest path between source and destination on the explored space. It works similar to the Dijkstra’s algorithm, but A* uses the heuristics estimate function for finding the shortest path. This algorithm is used to compute the shortest path that involves checking many adjacent matrix nodes in the graph. The nodes are explored towards the regions that converge to the goal indicated by the heuristics function (Kala, Shukla, Tiwari, Rungta and Janghel, 2009).

The total travel cost from the start node to goal node via the shortest path is calculated by the heuristic function.

g(n) = Cost from start node to the goal node

h(n) = Cost from the current node to the goal node

The total travel cost approximation is done using

$$f(n) = g(n) + h(n) \tag{4}$$

The total travel cost can be approximated by the heuristic function and the least travel cost node can be considered as the most promising node in the graph.

2.2.3 Bio-inspired algorithms

This kind of method deals with the NP-hard problems to generate a near optimal path based on stochastic approaches. Evolutionary algorithm, a category of bio-inspired algorithm stems from analyzing the behavior of certain species. Evolutionary algorithm have many categories but this paper mainly analyzes Genetic algorithm (GA). GA is the most famous population-based optimization method.

Genetic Algorithm (GA) Genetic Algorithm is a search-based optimization technique that evolves towards better solutions. GA is used to solve the optimization problem that is modeled by selection, genetic and evaluation techniques (Hosseinzadeh and Izadkhah, 2010). GA maintains a population of candidate solutions in optimization problem. Each candidate solution is coded as a binary string called as chromosome. A set of chromosomes form a population that is evaluated and ranked by their fitness function. Here, the total number of populations is considered as the number of paths in the workspace. Fitness function plays a vital role in GA because it provides the convergence and stability (Qu, Xing and Alexander, 2013).

The fitness function for a path (p) is defined as

$$F(p) = f(p) + C \tag{5}$$

$$f(p) = \frac{1}{w_d d(p) + w_s s(p)} + w_h h(p) \tag{6}$$

In (6), d(p) is a path distance, s(p) is path safety, h(p) is the smoothness of path and C is a constant. The parameters w$_d$, w$_s$, w$_h$ are the weights of the path length, safety and smoothness for a feasible path p.

In GA, three main operators namely reproduction or selection, crossover and mutation are used to generate a solution to optimization problem. The number of paths can be estimated during the reproduction cycle. This computation is based on the path's fitness value that depends on the suitable path according to the problem. Path estimation is determined by the path length, processing time and the number of steps in each path of the population. The length of the path can be optimized using fitness value.

During the reproduction operation, crossover is applied to the parent chromosomes with a probability which determines the frequency of crossover. The principle use of this operator is to interchange two parent chromosomes in generating better solution. Then, the operator can search the best solution by using the two existing solutions.

A mutation operator changes the order of gene's value (location or direction) within the chromosomes. This operator is used to check every gene's value and decides whether it should be mutated or not. Then, it finds the best solution based on the decision made. Finally, every gene's value is re-evaluated by their fitness function and the value is stored immediately after making changes in the specified location.

3 Representation and Architecture of Robotic Path Planner

Path planning enables point-to-point motion of a robot in order to reach the designated target location. The robot finds the path avoiding the obstacles in configuration. A systematic representation of robotic path planning architecture comprising of configuration space, path planner and controller is shown in Figure 2.

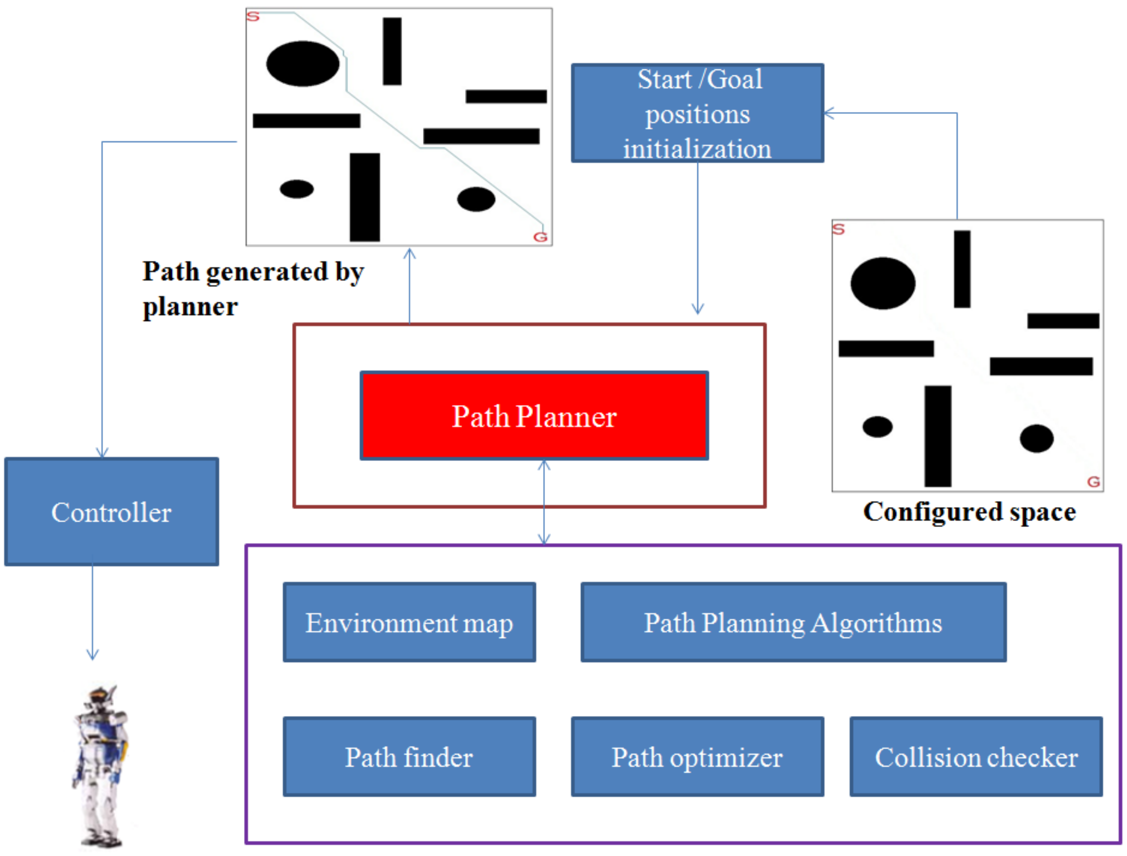


Figure 2: Architecture of robotic path planner

3.1 Configuration Space

With respect to the robotic architecture, configured space given as input of path planner. The configured space has the following premises. They are

- Environment of a robot's motion is a 2-D workspace, where several obstacles are represented by their absolute coordinates, and the height of obstacles are not considered.
- The information of the static obstacles and positions are well known.
- No dynamic obstacles are introduced in the environment.
- Robots moves with a fixed speed.

3.2 Path Planner

A planner constructs a plan that can be executed by the agent. A planner is used to plan a sequence ordered of actions in solving the path planning problems such as a control system. A planner interacts with basic functions such as environment map, collision checker, path optimizer, path smoother, path finder and path planning algorithms. These modules are explained below.

3.2.1 Environment map

An environment map is generated by mapping the configuration space coordinates into workspace coordinates.

3.2.2 Path planning algorithms

There are various path planning algorithms proposed by various researchers. The explanation of these algorithms have already been provided in Section 2.

3.2.3 Collision checker

Collision checker using its sensors identify the obstacles in the space and the information is fed to the path planning algorithm. This will enable a robot to handle several obstacles in the space and navigate in the free space.

3.2.4 Path finder

Path finder is responsible for finding alternate traversable paths from the start node to the end node. This includes the steps taken in a particular direction using the entire sequence of actions and rewards.

3.2.5 Path optimizer

A path optimizer searches optimal path on random sampled nodes by connecting configured pairs.

3.3 Controller

The controller provides the directions such as left, right, forward or reverse to an actuator based on the path generated by the path planner. In our work, we have not integrated the path planner with the controller and hence we have not configured the controller with any of the parameters.
So, in our work we have applied the path planner to find the optimal path with minimum time and minimum distance given a start location, goal location and set of obstacles distributed in the workspace. The next section explains the experimental setup and the analyses that has been performed in an elaborative manner.

4 Comparative Analysis of Path Planning Algorithms

This section discusses the experiment that has been carried out to analyze the various path planning algorithms.

4.1 Experimental Setup

In order to execute the path planning algorithms implemented in MATLAB 2016a version, 64-bit machine with i5 processor having 4GB RAM capacity is used. The implemented RRT algorithm can be downloaded from
`https://drive.google.com/file/d/1un3V98vmfUXF6_pPURuJxrrxrrdjQiTW`

4.2 Evaluation Metrics

In this paper, the path length and the processing time are considered as metrics for evaluating the agent performance. In evaluating the various path planning algorithms, parameters are considered and the details are given in the following section.

4.3 Explanation of the Parameters for RRT

The parameters that is required for the execution of algorithms to model static environment is explained in this section. The initial graph is given as an input image. In the image, obstacles are represented as black region and the path is represented as white region. The image map has 500 $\times$ 500 resolution. Parameters considered for the implementation of RRT are as follows.

The step size of the robot is 20ft, maximum failed attempt value is 10,000 (iterations) and the distance threshold value which is always proportional to the step size is set as 20 m. The algorithms were tested under various positions of start and goal coordinate points and the results are shown in Table 1.

Increase in the step size value leads to lesser computation time. The objective function of RRT algorithm is to find the shortest path with minimum execution time. RRT algorithm runs from the starting point until the expansion of tree reaches the specified goal point. Otherwise, the

Table 1: Quantitative analysis of RRT Algorithm

Iterations	Start position of robot (Coordinate positions of source)	Target position of robot (Coordinate positions of goal	Processing Time (Sec)	Path Length (m)
1	Source = [10, 10]	Goal = [490, 490]	1.256922e+02	8.207567e+02
2	Source = [5, 5]	Goal = [400, 400]	4.007778e+01	7.184744e+02
3	Source = [5, 5]	Goal = [100, 100]	1.819326e+01	2.810579e+02
4	Source = [5, 5]	Goal = [400, 350]	3.226578e+01	6.909495e+02
5	Source = [10, 5]	Goal = [300, 250]	3.796443e+01	7.315111e+02

algorithm will display an “no path found” error message when it reaches the maximum failure value.

4.3.1 Simulation results for single RRT

The performance of RRT algorithm is being evaluated considering the path length of the generated shortest path and the execution time of the algorithm. The results obtained for different positions of start and goal co-ordinate values are shown in Table 1. Table 2 shows the simulation results for RRT in which a tree is generated based on the position of start and goal locations and the shortest path is found from start to goal position avoiding all the obstacles in the testbed image.

4.4 Explanation of the Parameters for Bi-RRT

Bi-directional RRT algorithm works similar to Single RRT algorithm, but the main difference is that it explores the tree from the two faces of the nodes. The first face of node generates tree from the source point towards the goal and the second face of node generates tree from the goal point towards the source. The results obtained are shown in Table 3.

4.4.1 Simulation results for Bidirectional RRT

The performance of Bi-RRT algorithm is being evaluated considering the path length of the generated shortest path and the execution time of the algorithm. The results obtained for different positions of start and goal co-ordinates values are shown in Table 3. Table 4 shows the simulation results for Bi-RRT in which a tree is generated simultaneously from the start position and goal position. The tree generated from source to goal in blue in color and the tree generated from goal to source is red in color. The algorithm completes when two trees merges and this indicates that the best path is found in the testbed image.

4.5 Explanation of the Parameters for PRM

The input parameter of PRM is the number of vertices (K) and the experiment is performed considering the value of (k) as 50 and 100. Although increase in the number of vertices will

Table 2: Simulation Results for RRT

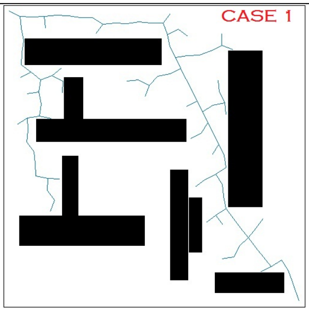
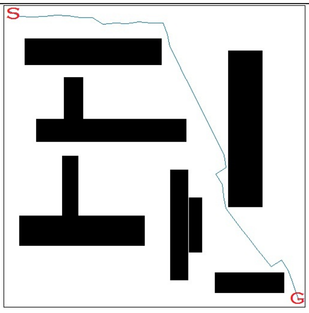
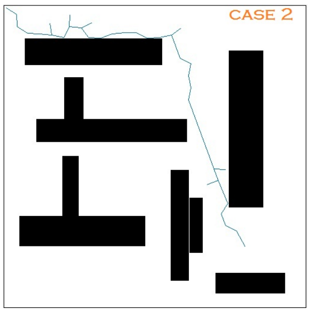
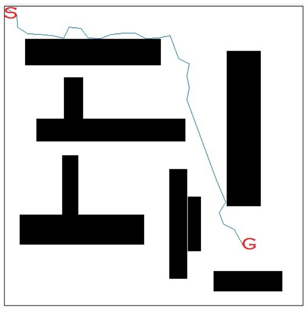
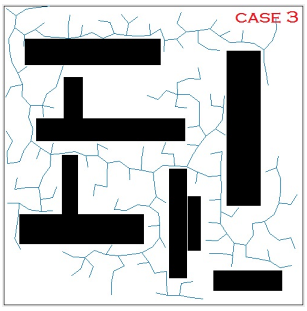
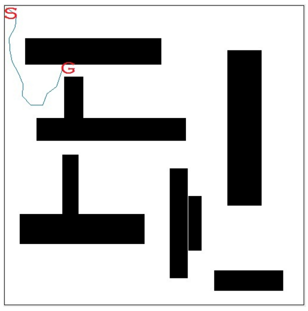
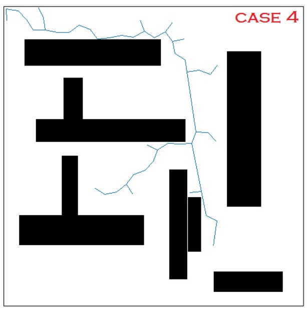
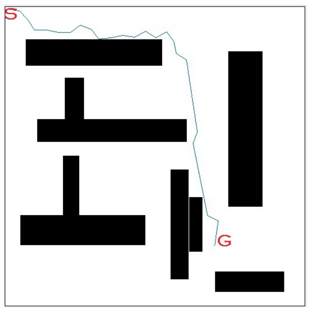
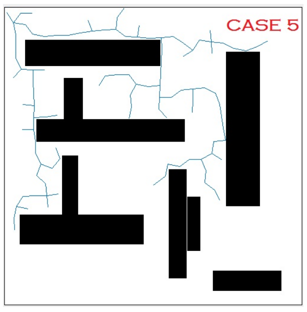
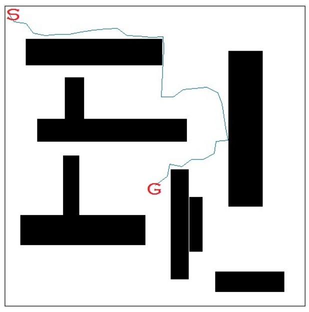
Testbed image for Simulation of RRT	Simulation showing path generated using RRT	Path Length (m)	Processing Time (sec)
		8.20	1.25
		7.18	4
		2.81	1.81
		6.90	3.22
		7.31	3.79

Table 3: Quantitative analysis of Bi-RRT Algorithm

Iterations	Start position of robot (Coordinate positions of source)	Target position of robot (Coordinate positions of goal	Processing Time (Sec)	Path Length (m)
1	Source = [10, 10]	Goal = [490, 490]	5.323438e+01	7.996150e+02
2	Source = [5, 5]	Goal = [400, 400]	2.995807e+01	7.550670e+02
3	Source = [5, 5]	Goal = [100, 100]	3.098862e+01	5.390868e+02
4	Source = [10, 10]	Goal = [100, 100]	2.073294e+01	1.770750e+02

Table 4: Simulation Results for Bi-RRT

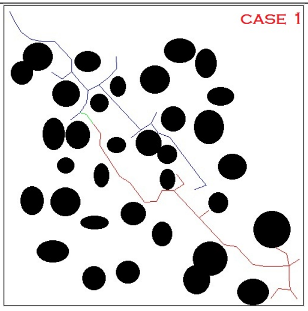
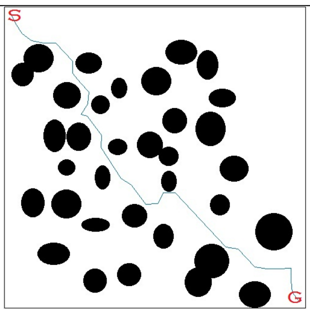
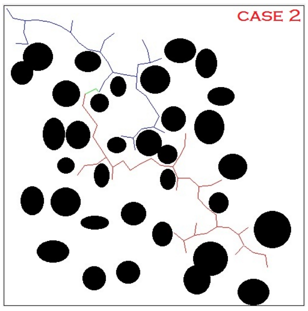
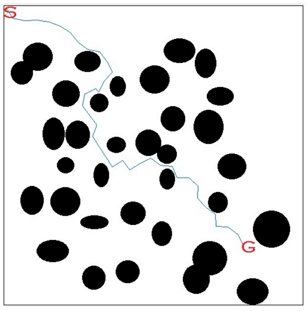
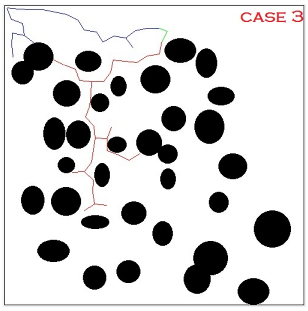
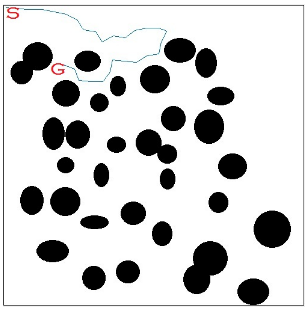
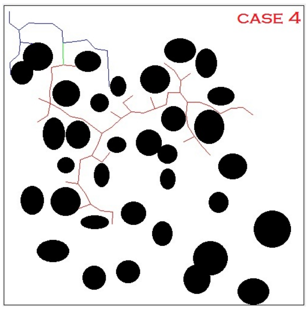
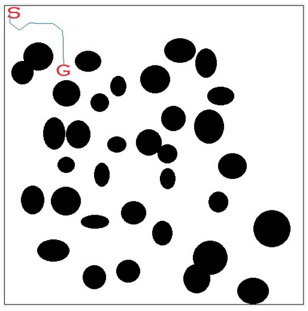
Testbed image for Simulation of Bi-RRT	Simulation showing path generated using Bi-RRT	Path Length (m)	Processing Time (sec)
		7.99	5.32
		7.55	2.99
		5.39	3.09
		1.77	2.07

Table 5: Quantitative analysis of PRM Algorithm

Iterations	Start position of robot (Coordinate positions of source)	Target position of robot (Coordinate positions of goal	Processing Time (Sec)	Path Length (m)
1 (k=50)	Source = [10, 10]	Goal = [490, 490]	1.940287e+01	8.700981e+02
2 (k=50)	Source = [10, 10]	Goal = [300, 300]	1.674716e+01	5.542155e+02
3 (k=50)	Source = [5, 5]	Goal = [500, 400]	1.571003e+01	8.751453e+02
4 (k=50)	Source = [10, 5]	Goal = [500, 400]	9.181762e+00	7.577930e+02
5 (k=100)	Source = [10, 10]	Goal = [490, 490]	1.850295e+01	8.385454e+02

lead to higher computation time, shortest path can be found. The algorithm was tested under various position of start and goal coordinate points and results are shown in Table 5.

4.5.1 Simulation results for PRM

The performance of PRM algorithm is being evaluated considering the path length of the generated shortest path and the execution time of the algorithm. The results obtained for different positions of start and goal co-ordinates values are shown in Table 5. Table 6 shows the simulation results for PRM. Large number of random nodes are generated when k value is more. It is found that when large numbers of random nodes are present, optimal path is achieved with a high probability.

4.6 Explanation of the Parameters for A*

The input parameter of A* is the connection matrix and the experiment is performed considering the dimension of a matrix as [3 $\times$ 3] and [5 $\times$ 5]. Increase in the matrix dimension will always give better result than the lower dimension matrix. In a coded matrix, '1' is marked as possible moves and '0' as impossible moves. The default position of the robot is marked as '2' in the matrix. The algorithm was tested using various matrices and the different coordinate positions of source and goal points. The results are shown in Table 7.

4.6.1 Simulation results for A*

The performance of A* algorithm is being evaluated considering the path length of the generated shortest path and the execution time of the algorithm. The results obtained for different positions of start and goal co-ordinates values are shown in Table 7. Table 8 shows the simulation results for A*. It is found that the robot successfully navigates through all the obstacles and finds the optimal path when a matrix with higher dimension is used.

4.7 Explanation of the Parameters for Artificial Potential Field (APF)

The input parameter of APF algorithm is attractive and repulsive potential functions. The scaling factor for attractive potential is 300000 (ratio of 1: 300000) and the repulsive potential is

Table 6: Simulation Results for PRM

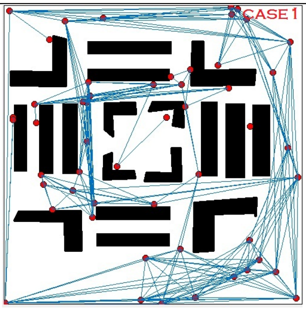
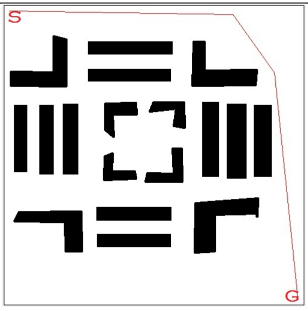
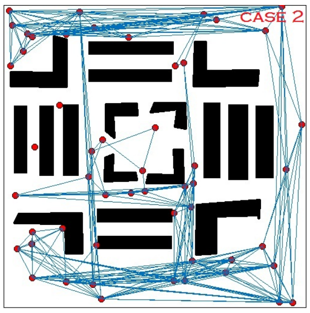
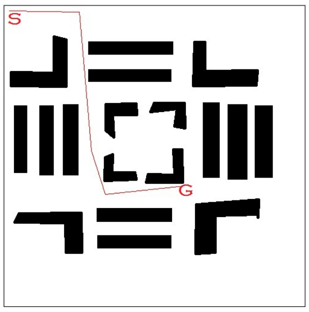
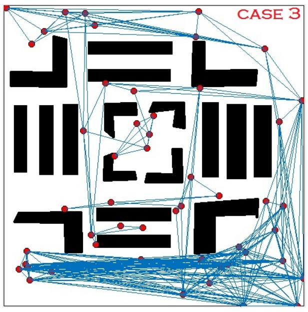
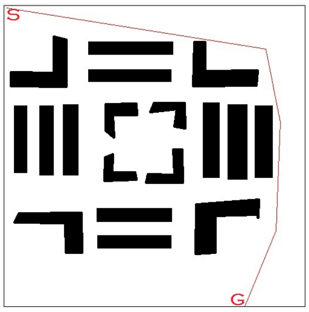
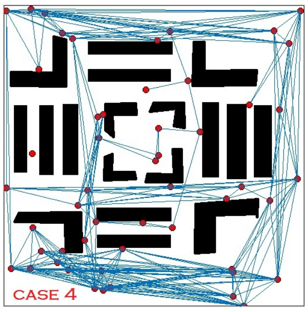
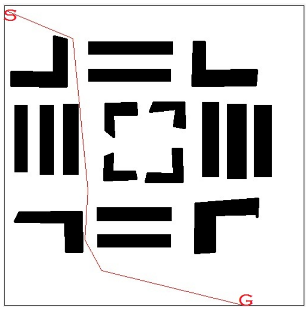
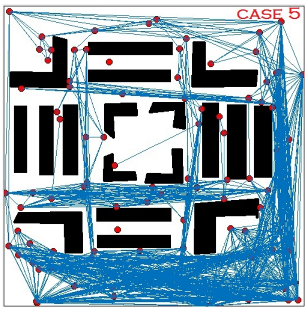
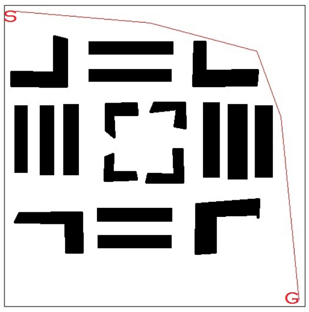
Testbed image for Simulation of PRM	Simulation showing path generated using PRM	Path Length (m)	Processing Time (sec)
		8.70	1.94
		5.54	1.67
		8.75	1.57
		7.57	9.18
		8.38	1.85

Table 7: Quantitative analysis of A* Algorithm

Iterations	Start position of robot (Coordinate positions of source)	Target position of robot (Coordinate positions of goal	Processing Time (Sec)	Path Length (m)
1 [3 $\times$ 3]	Source = [10, 10]	Goal = [490, 490]	1.160474e+02	7.432590e+02
2 [3 $\times$ 3]	Source = [5, 5]	Goal = [190, 190]	7.602119e+01	3.026346e+02
3 [3 $\times$ 3]	Source = [10, 10]	Goal = [190, 290]	4.268528e+01	3.662742e+02
4 [3 $\times$ 3]	Source = [5, 5]	Goal = [370, 370]	7.710836e+01	5.630509e+02
5 [5 $\times$ 5]	Source = [5, 5]	Goal = [370, 370]	3.462974e+01	5.345476e+02

300000 (ratio of 1:300000). A scale of 1:300000 means that 1 cm on the map represents an actual distance of 300000 cm (or 3000 m or 3 km). Initial robot angular direction is mentioned as π /8, so that the robot moves away from the obstacles at specified angles namely forward, left, right, left diagonal and right diagonal. Maximum robot speed is 10 m/sec. Distance threshold value is set as 30 m so that the robot can select the nodes within this threshold value. The algorithm was tested under various positions of start and goal coordinate points and the results obtained are shown in Table 9.

4.7.1 Simulation results for Artificial Potential Field (APF)

The performance of Potential Field algorithm is being evaluated considering the path length of the generated shortest path and the execution time of the algorithm. The results obtained for different positions of start and goal co-ordinates values are shown in Table 9. Table 10 shows the simulation results for Potential Field in which the robot reaches the goal avoiding all the obstacles in the testbed image when repulsive and attractive forces are applied.

4.8 Explanation of the Parameters for Genetic Algorithm

The objective function of GA algorithm is the length of the path. Number of populations, number of generations and fixed number of points are considered as the input parameter of GA. The optimization variable is fixed number of points in the map that is equal to the maximum number of robotic turns in the map. All the points are connected by a straight line and each point in the path is a mark representing point of turn.

Increase in the number of points in the map results in larger computation time, useless turns and high path length. Specifying less number of points also make a robot ineffective. So, specifying exact number of points makes a robot more effective in the testbed. The algorithm was tested under various locations of the start and goal coordinate points and results obtained are shown in Table 11.

4.8.1 Simulation results for Genetic Algorithm

The performance of GA algorithm is being evaluated considering the path length of the generated shortest path and the execution time of the algorithm. The results obtained for different

Table 8: Simulation Results for A*

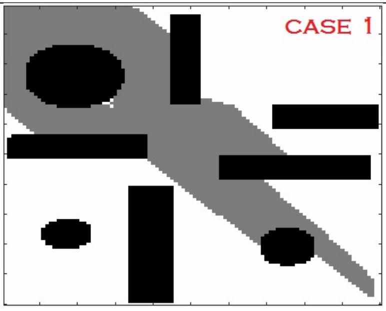
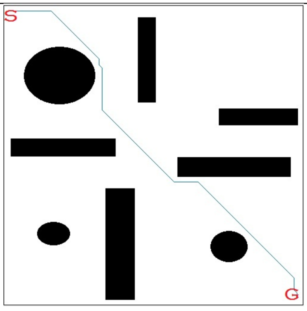
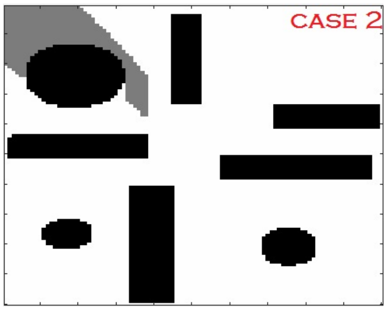
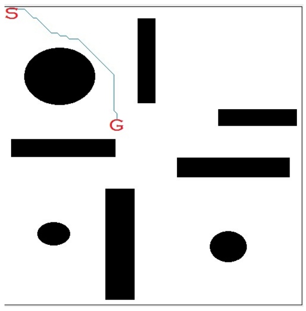
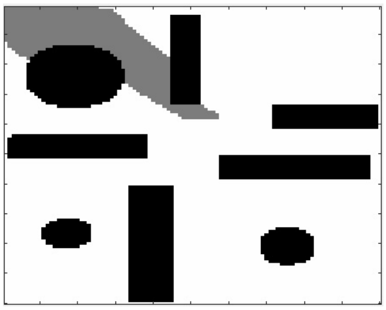
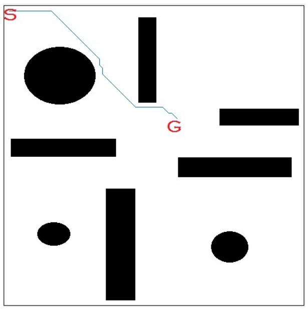
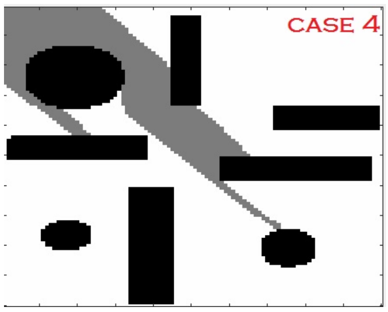
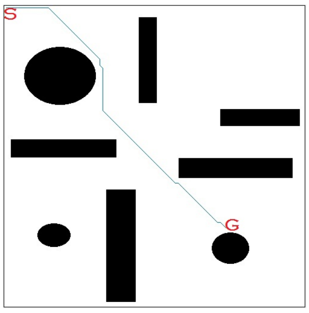
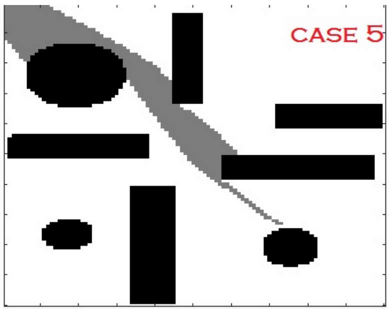
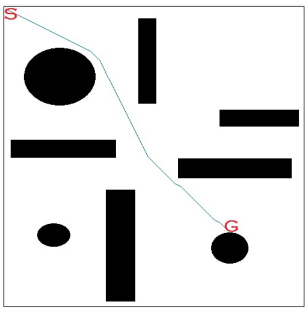
Testbed image for Simulation of A*	Simulation showing path generated using A*	Path Length (m)	Processing Time (sec)
		7.43	1.16
		3.02	7.6
		3.66	4.26
		5.63	7.7
		5.34	3.46

Table 9: Quantitative analysis of Artificial Potential Field Algorithm

Iterations	Start position of robot (Coordinate positions of source)	Target position of robot (Coordinate positions of goal	Processing Time (Sec)	Path Length (m)
1	Source = [50, 50]	Goal = [250, 250]	3.515630e+01	5.428395e+02
2	Source = [10, 10]	Goal = [200, 200]	2.382031e+00	3.685265e+02

Table 10: Simulation Results for APF

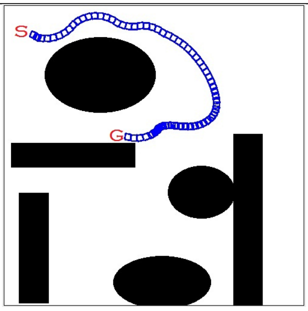
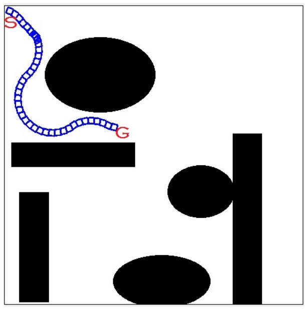
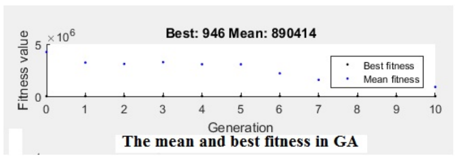
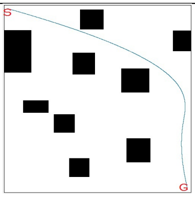
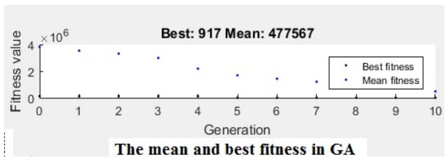
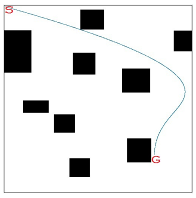
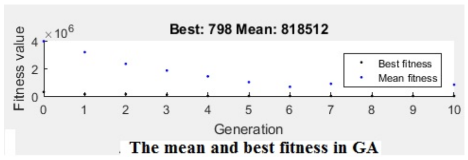
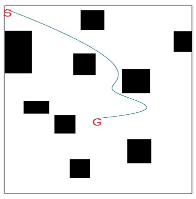
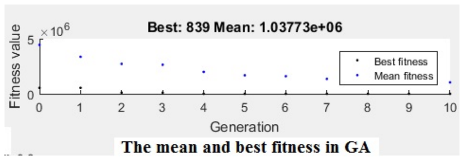
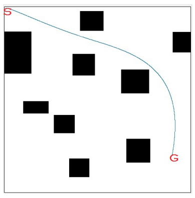
Testbed image for Simulation of APF	Path Length (m)	Processing Time (sec)
	5.42	3.5
	3.68	2.38

Table 11: Quantitative analysis of GA Algorithm

Iterations	Start position of robot (Coordinate positions of source)	Target position of robot (Coordinate positions of goal	Processing Time (Sec)	Path Length (m)
1	Source = [10, 10]	Goal = [490, 490]	4.318317e+01	946
2	Source = [5, 5]	Goal = [400, 400]	4.165122e+01	917
3	Source = [10, 5]	Goal = [300, 250]	4.208790e+01	798
4	Source = [5, 5]	Goal = [400, 450]	4.118612e+01	839

Table 12: Simulation Results for GA

Testbed image for Simulation of GA	Simulation showing path generated using GA	Path Length (m)	Processing Time (sec)
		946	4.31
		917	4.16
		798	4.2
		839	4.11

positions of start and goal co-ordinates values are shown in Table 11. Table 12 shows the simulation results for GA in which number of points is plotted in the map based on the maximum number of turns a robot takes to reach the goal.

5 Discussion on the Experimental Results

We implemented various path planning algorithms and obtained the results. Each algorithm has its unique characteristics. A comparative analyses of all algorithms is explained in this section.

- RRT gives better performance in open-space and it connects to the target point in fewer iterations.
- Single RRT always performs better than Bi-directional RRT because it spends less time in constructing a tree and computing the path. With respect to execution time both perform similarly.
- The resulting path of PRM is too complicated because it generates lots of possible redundant moves based on the number of vertices.

- PRM will not be able to follow the non-holonomic constraints as RRT algorithm. But the execution time is less even for problems with high dimensionality.
- A* algorithm returns a solution in higher computation time when the distance is more between source and goal.
- RRT might be more efficient than PRM for holonomic path planning. This is because RRT does not require any connections made between pairs of state but PRM requires thousands of connections.
- RRT can be directly applied to nonholonomic and kinodynamic constraints.
- A* algorithm finds optimal path faster than GA but it has higher computation time.
- APF algorithms find a solution faster than A* but it has higher path length.
- Single-RRT is the best path planner and achieves the best performance considering the performance metrics.

5.1 Benefits and Limitations of Robotic Path Planning Algorithms

Based on the analyses done Table 13 summarizes the findings by providing the details of environmental applicability, approach, parameters that are necessary to find a shortest path and the features of various path planning algorithms.

5.2 Graphical Analyses

This section discusses the comparative analyses of various path planning algorithms with respect to path distance and execution time by considering different start and goal positions. Five categories of start and goal coordinate locations are considered. The (x,y) coordinate positions of nodes 1) A and B are (10, 10) and (400, 400) respectively 2) C and D are (10, 10) and (350, 490) respectively 3) E and F are (40, 50) and (395, 495) respectively 4) G and H are (10, 10) and (295, 495) respectively and 5) I to J are (10, 10) and (425, 415) respectively.

5.2.1 Analyses of path distance

The experimental results show that PRM and A* are provides optimal path when compared to the other four path planning algorithms namely Genetic Algorithm, Potential Field, single RRT and Bi-RRT. Considering the path between A and B nodes, the performance of A* with respect to path length is better compared to PRM when a matrix with low dimension is used. Considering the path between I and J nodes, the performance of A* is lower in terms of path length in comparison with PRM when a matrix with high dimension is used. Considering the remaining nodes (C-D, E-F and G-H), both perform similar with respect to path length.

Table 13: Comparative analyses of various path planning algorithms

Path planning Algorithm	Environment	Approaches	Parameters	Experimental Findings
Single RRT	Deterministic	Sampling Approach	Step size Path length Processing time	Increase in step size value leads to lesser computation time.
Bidirectional RRT	Deterministic	Sampling Approach	Step size Path length Processing time	Spends less time in constructing a tree but has higher path length.
Probabilistic Road Map (PRM)	Deterministic & NonDeterministic	Roadmap Methods	Number of Vertices Path length Processing time	Provides better path but generates a lot of redundant moves.
Artificial Potential Field (APF)	Deterministic & NonDeterministic	Reactive Planning Technique	Scaling factor for APF and RPF, directions, speed, Path length, processing time	Finds a faster solution but supports only limited directions.
A* Algorithm	Deterministic	Heuristics Function	Matrix, Path length, processing time	Finds the shortest path but has higher computation time.
Genetic Algorithm	Deterministic & NonDeterministic	Evolutionary Approach	Number of populations, Number of generations, Fitness function, path length, processing time	Provides the longest path with high memory requirements and takes lot of time to generate the path.

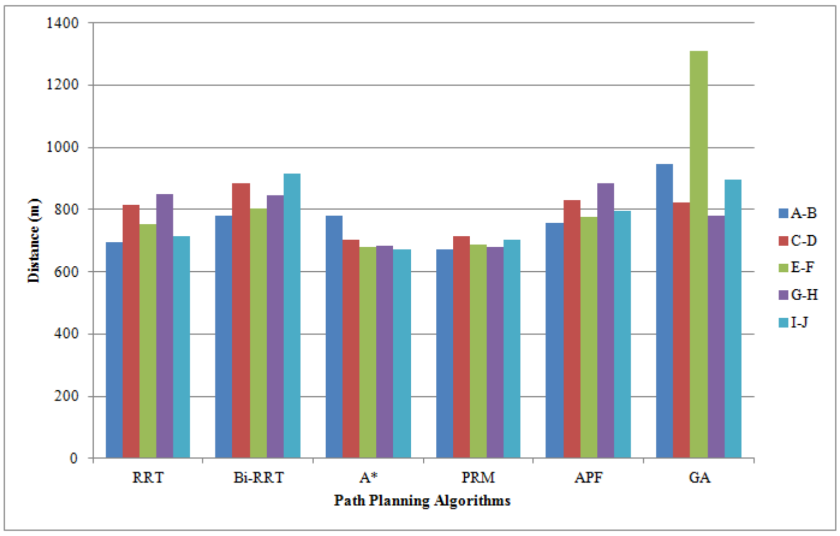


Figure 3: Path distance analyses of various path planning algorithms considering different start and goal positions

5.2.2 Analyses of execution time

The execution time of each robot path planning algorithm is being evaluated considering the time taken to reach the specified goal considering different locations and is shown in Figure 3. It is found that single-RRT and APF performs better. A* algorithm takes more time in finding a solution when compared with other four algorithms for large size inputs.

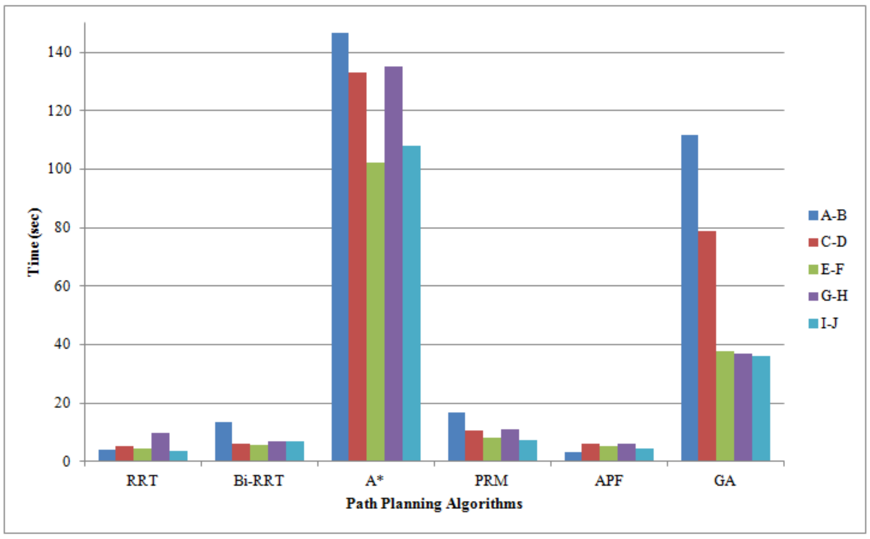


Figure 4: Execution time analyses of various path planning algorithms considering different start and goal positions

6 Conclusions and Future Work

In this paper, robot path planning algorithms are tested using various positions of start and goal coordinate locations of a modeled environment and the performance analyses of intelligent path planning algorithms is presented. Three intelligent path planning algorithms are evaluated in terms of the execution time and path length. From our experiments, it is found that PRM and A* performs better with respect to path length. Single RRT and APF performs better in terms of execution time. Considering both the performance metrics, Single RRT is found to be optimal in a deterministic environment.

In future, we will integrate our previous works on knowledge representation & reasoning with RRT algorithm to generate collision-free path in a non-deterministic environment. We would also try to integrate deep learning approaches to predict the imminent trajectories based on the action of an agent in previous time instances. Single RRT is not reliable in real-time (non-deterministic) robot path planning applications. Therefore, RRT algorithm can be combined with Long Short Term Memory (LSTM) model to perform autonomous robot navigation. The ability of LSTM model to learn and reproduce long sequences will help in predicting the next direction of movement in future instances.

References

Ahmed, A. A., Abdalla, T. Y. and Abed, A. A. 2015. Path planning of mobile robot using fuzzy-potential field method., *Iraqi Journal for Electrical & Electronic Engineering* **11**(1): 32–41.

Al-Hmouz, R., Gulrez, T. and Al-Jumaily, A. 2004. Probabilistic road maps with obstacle avoidance in cluttered dynamic environment, *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, IEEE, pp. 241–245.

Aravind, K. R., Raja, P. and Pérez Ruiz, M. 2017. Task-based agricultural mobile robots in arable farming: A review, *Spanish Journal of Agricultural Research* **2017**(15 (1)): 1–16.

Chen, W., Wu, X. and Lu, Y. 2015. An improved path planning method based on artificial potential field for a mobile robot, *Cybernetics and Information Technologies* **15**(2): 181–191.

Chen, Y.-b., Luo, G.-c., Mei, Y.-s., Yu, J.-q. and Su, X.-l. 2016. UAV path planning using artificial potential field method updated by optimal control theory, *International Journal of Systems Science* **47**(6): 1407–1420.

Galceran, E. and Carreras, M. 2013. A survey on coverage path planning for robotics, *Robotics and Autonomous systems* **61**(12): 1258–1276.

Gayathri, R. and Uma, V. 2018. Ontology based knowledge representation technique, domain modeling languages and planners for robotic path planning: A survey, *ICT Express* **4**(2): 69–74.

Gayathri, R. and Uma, V. 2019. A review of description logic-based techniques for robot task planning, *Integrated Intelligent Computing, Communication and Security*, Springer, pp. 101–107.

Ginter, V. J. and Pieper, J. K. 2011. Robust gain scheduled control of a hydrokinetic turbine, *IEEE Transactions on Control Systems Technology* **19**(4): 805–817.

Haidegger, T., Kovács, L., Precup, R.-E., Benyó, B., Benyó, Z. and Preitl, S. 2012. Simulation and control for telerobots in space medicine, *Acta Astronautica* **81**(1): 390–402.

Hamani, M. and Hassam, A. 2012. Mobile robot navigation in unknown environment using improved APF method, *The 13th international Arab Conference on information Technology ACIT'2012*, pp. 10–13.

Hosseinzadeh, A. and Izadkhah, H. 2010. Evolutionary approach for mobile robot path planning in complex environment, *International Journal of Computer Science Issues* **7**(4): 1–9.

Kala, R. 2013. Rapidly exploring random graphs: motion planning of multiple mobile robots, *Advanced Robotics* **27**(14): 1113–1122.

Kala, R., Shukla, A., Tiwari, R., Rungta, S. and Janghel, R. 2009. Mobile robot navigation control in moving obstacle environment using genetic algorithm, artificial neural networks and A* algorithm, *Computer Science and Information Engineering, 2009 WRI World Congress on*, Vol. 4, IEEE, pp. 705–713.

Kannan, A., Gupta, P., Tiwari, R., Prasad, S., Khatri, A. and Kala, R. 2016. Robot motion planning using adaptive hybrid sampling in probabilistic roadmaps, *Electronics* **5**(2): 16.

Kavraki, L. E., Svestka, P., Latombe, J.-C. and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE transactions on Robotics and Automation* **12**(4): 566–580.

Knispel, L. and Matousek, R. 2013. A performance comparison of rapidly-exploring random tree and Dijkstra's algorithm for holonomic robot path planning, *Institute of Automation and Computer Science, Faculty of Mechanical Engineerig, Brno University of Technology* pp. 154–162.

LaValle, S. 2006. Planning Algorithms, Cambridge University Press, Cambridge, UK.

Mac, T. T., Copot, C., Tran, D. T. and De Keyser, R. 2016. Heuristic approaches in robot path planning: A survey, *Robotics and Autonomous Systems* **86**: 13–28.

Meyer, J.-A. and Filliat, D. 2003. Map-based navigation in mobile robots:: Ii. a review of map-learning and path-planning strategies, *Cognitive Systems Research* **4**(4): 283–317.

Pimenta, L. C., Fonseca, A. R., Pereira, G. A., Mesquita, R. C., Silva, E. J., Caminhas, W. M. and Campos, M. F. M. 2005. On computing complex navigation functions, *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, IEEE, pp. 3452–3457.

Qu, H., Xing, K. and Alexander, T. 2013. An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots, *Neurocomputing* **120**: 509–517.

Vrkalovic, S., Lunca, E.-C. and Borlea, I.-D. 2018. Model-free sliding mode and fuzzy controllers for reverse osmosis desalination plants, *International Journal of Artificial Intelligence* **16**: 208–222.

Yacoub, R. R., Bambang, R. T., Harsoyo, A. and Sarwono, J. 2014. DSP implementation of combined FIR-functional link neural network for active noise control, *International Journal of Artificial Intelligence*^{TM} **12**(1): 36–47.

Yang, L., Qi, J., Song, D., Xiao, J., Han, J. and Xia, Y. 2016. Survey of robot 3D path planning algorithms, *Journal of Control Science and Engineering* **2016**: 5.