On-line path planning simulation for mobile robots in cluttered environments

Francisco Javier Garcia Rosas [garciar@rhrk.uni-kl.de], M. Sc. Yanhao He [yanhao@eit.uni-kl.de]

***Abstract*— A general problem of path planning for mobile robots can be decomposed into three main parts: Motion control in charge of controlling the movement of robot's actuators, local collision avoidance for keeping the robot in a safety zone during the trajectory, and a global search-based planning algorithm, which tries to find a suitable sequence of points to move the robot from an initial position to a goal point. This article describes a basic pipeline for addressing the overall on-line path planning problem in cluttered environments, present a software simulator developed by the authors using Matlab that allows a rapid exploration of path planning algorithms and perform some experiments in order to analyze and compare four different approaches for global search-based planning algorithm.**

I. INTRODUCTION

As many mobile robotic platforms become increasingly popular nowadays, their applications has been moved from static lab scenarios to highly dynamic environments with complex interactions between people and robots, making more difficult to find a suitable path (sequence of points from origin to goal destination) that might be able to deal with surrounding movable objects, this process is known as path planning and is widely use for cooperative robotics, in-home robots, rescue platforms among others.

Even considering that path planning is a widely researched topic, there is not yet an "universal algorithm" that solves the problem for any condition/configuration, and we can highlight some common challenges that are intrinsically involved on the process such as optimality of planned path, kinematic and dynamic constrains of the robot, limited or null information about the environment, computational load and efficiency, poor quality of sensing systems and many others, for that reason we will focus this article on analyzing different methods to solve the problem and make a comparison between them in order to understand the advantages/disadvantages of each method, making some assumptions in order to simplify the dynamic environment and point out clearly the key features of each approach.

Motion planning for mobile robots can be decomposed into three main steps:

- Motion control: Handle the movement of actuators in the robot, mainly based on control theory
- Local collision avoidance: Keep the robot in safe zones (no crash with dynamic objects) during the trajectory
- Global search-based planning algorithm: Find a suitable sequence of points to move the robot from an initial position to a goal point

This article keep focus on the last concept, we will discuss four different algorithm for global planning which can be divide into two main groups: deterministic (A* and Artificial potential field) and randomized (Rapidly exploring random trees and probabilistic road-map), we present simulations for this methods in different scenarios and provide a comparison of their main features and possible applications.

In order to simplify analysis, the following assumptions were considered:

- The map already exists and is given as input
- We have a robot arena with an overhead camera, which can capture the location of robot relative to the map (solve the localization problem)
- The path planning is performed in R^2 (2D configuration space).

In section II, we discuss about previously mentioned algorithms, highlighting properties and summarizing key features, then we present in section III simulations for different scenarios and finally we provide some conclusions about performance and possible applications of this path planning approaches on section IV.

II. GLOBAL PLANNER

In the following section, we present a brief description of each path planning algorithm, pointing out key features, advantages, disadvantages and some relevant remarks.

A. Algorithm A*

Developed by Peter Hart, Nils Nilsson and Bertram Raphael of Stanford Research Institute, it solves the path planning problem and finds a solution searching for the smallest cost to travel from origin to the goal. Formulated as a weighted graph, and creating a tree starting from a specific node and expanding until it reaches the goal point. A* algorithm can be summarized into the following steps:

- Specify admissible movements for robot
- Define a cost function

$$f(n) = g(n) + h(n)$$

  $g(n)$ Movement cost: Corresponds to the cost of changing the actual cell to someone placed into the neighbor.
  $h(n)$ Heuristic cost: Corresponds to the cost of moving from actual to goal position.
- Evaluate total cost $f(n)$ and move to the cell with lowest value
- Repeat the same approach until reach goal position
- When the goal is reached, compute the final path following the parents with lowest cost

As an example, figure 1 represent how the algorithm evolve from initial position (robot) until it reaches the goal

(red cross). Inside the box, numbers has the following meaning: bottom left: movement cost, bottom right: heuristic cost and upper left: total cost.

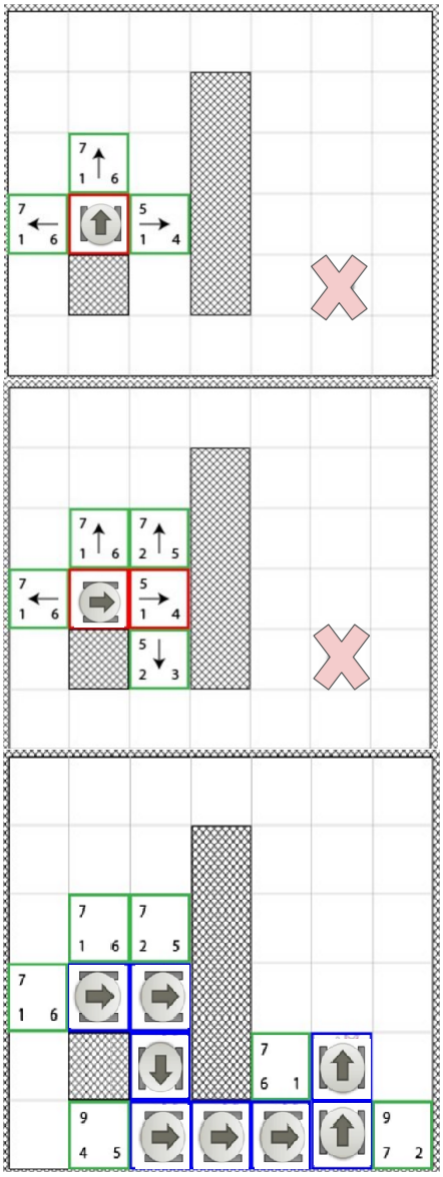


Fig. 1. A* algorithm

The figure 2 shows an example of path planning using A*, where black dots represent obstacles, white dots are free space, gray ones are the searched space and red ones correspond to the path with smallest value from origin to the goal position.

1) Advantages of A:*

- If the solution exists, A* will find an optimal one based on its cost function
- The algorithm is conceptually simple and easy for implementation

2) Disadvantages of A:* The main drawback of A* is its memory requirement, since the entire map must be stored, it can be problematic with large scenarios or high dimensional operational spaces.

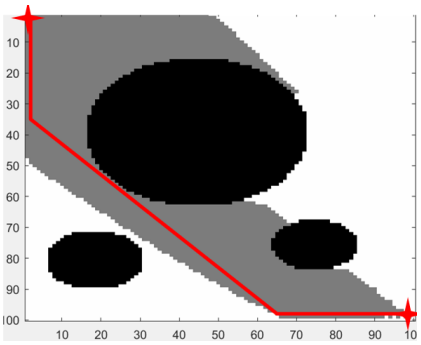


Fig. 2. Generated path of A*

B. Artificial potential field (APF)

Proposed by Oussama Khatib [1] in 1986, this approach is widely used in robotics for obstacle avoidance and path planning [2]. APF model the environment (map) as a fields that attract or repel the robot, all obstacles repel the robot with a magnitude inversely proportional to the distance and conversely, it is attracted the goal position. Total potential field can be defined as:

$$U_{total} = U_{attra} + U_{Rep}$$

where attractive field U_{attra} is defined as $\frac{1}{2}\sqrt{(X - X_g)^2 + (Y - Y_g)^2}$ and (X, Y, X_g, Y_g) represent current and goal position of the robot.

Similarly, the repulsive field U_{Rep} can be modeled as $\frac{1}{2}(Rd - Di)^2$ where Rd is the influence's distance of the object to robot and Rd is the length to nearest obstacle.

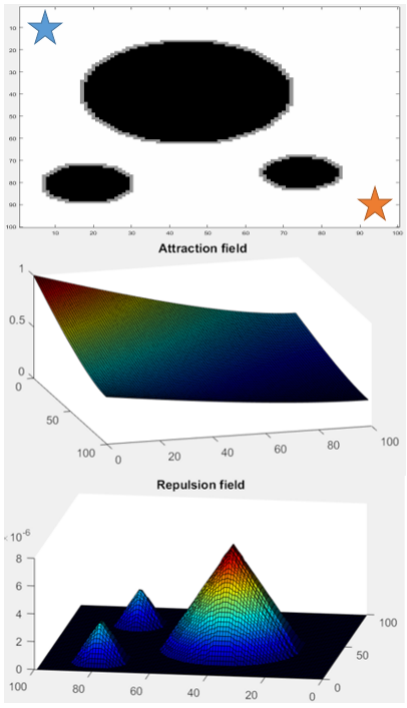


Fig. 3. APF: Map, attractive and repulsive field

As an example, figure 3 shows a demo map, with initial and final position marked with blue and red stars respectively, as well as resulting attractive and repulsive field generated by APF. The distance of obstacles at all angles from the robot is measured, the resulting force can be used also for controlling the speed of robot.

Figure 4 shows the corresponding generated path for previous example using APF.

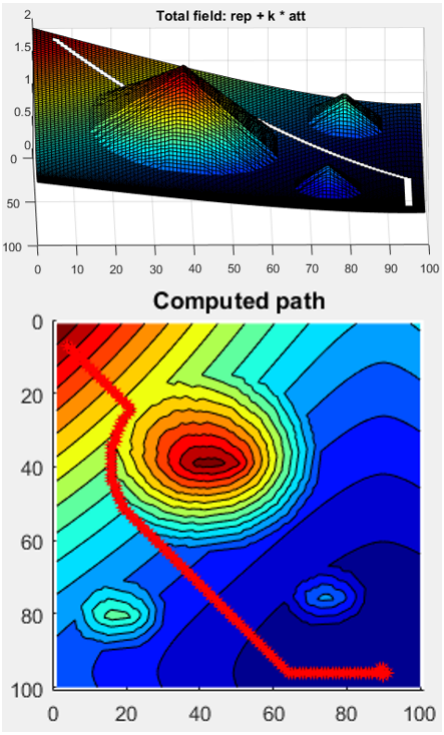


Fig. 4. Generated path of APF

The plane with negative slope represent U_{attra} and the cones are U_{Rep} generated from the obstacles. Although these methods are fast and efficient, they have the following drawbacks and limitations [3]:

- Trap situations due to local minima
- No passage between closely spaced obstacles
- Oscillations in the presence of objects and narrow passages

The local minimum problem for APF has been extensively researched and some approaches were proposed to avoid it, for instance, add a virtual obstacle to escape local minima [4], use simulated annealing [5] or implement an improved repulsive field model that changes weight of the obstacle potential field function adaptively to make the robot escape from the local minima [6].

APF method is mainly used for local path planning and obstacle avoidance because it is fast and computationally cheap.

C. Rapidly exploring random trees (RRT)

1) RRT: This algorithm was developed by Steven Lavalle in 1998 and can be summarized as a randomized algorithm that provides a way to search high-dimensional spaces efficiently, it consists of a tree data structure of samples in space and connections in a way that provides good coverage [7]. The tree-construction algorithm basically repeats a loop with the following operations:

- Pick a random sample in the search space
- Find the nearest neighbor of that sample
- Select an action from the neighbor that heads towards the random sample
- Create a new node based on the outcome of action applied to the neighbor
- Add new sample to the existing tree and connect it to its neighbor
- Stop the loop when goal position is reached by a node

In order to speed up the path planning, we can use a slightly different variant of this algorithm and include a second tree on the process; this approach is described in the next subsection.

2) Bidirectional RRT: It uses the same algorithm as RRT but it creates two different trees: one starting from the source and growing towards goal, the second one starting from the goal and growing towards the source, when both trees meet a solution is found.

Figure 5 presents an example of BRRT, on the left image we can observe two trees (blue and red) joined by a green line that shows the point in which both trees are intersected, and on the right image, the computation of final path between initial position (left upper point) and goal (bottom right point). It can be observed that the generated path is not optimal due to the randomized search.

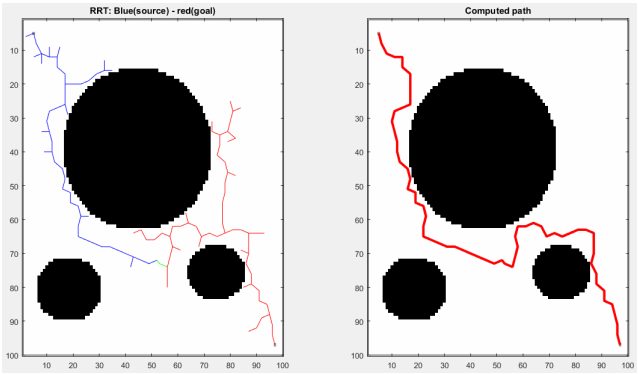


Fig. 5. Generated path of BRRT

One of the biggest advantages of this method is that the computation effort is reduced due to randomized sampling of the configuration space, leading to fast sub-optimal paths.

D. Probabilistic road-map PRM

In 1996, Kavraki and Svestka have developed the PRM algorithm that basically takes random samples from the configuration space of the robot and connect them to other nearby configurations, current and goal points are added and a graph-search based algorithm is applied to determine a path between the starting and goal configurations[8].

The PRM algorithm has two main stages:

- Build an off-line road-map: Draw a graph across the workspace, by randomly selecting points that are not inside of obstacles and connect all pairs of vertices using straight lines, checking that all vertices and edges are collision-free.
- Online planning: Since a graph (road-map) is already known, we can use a graph-search based algorithm for path planning, in this article we use A*, defining euclidean distance between connecting points as local cost and euclidean distance to the goal as heuristic cost.

PRM is probabilistic complete, meaning that when the time tends to infinity, it tends to find a solution. This algorithm build a graph over state space and is focused on generating a path, but computing exact solutions with complex geometries are probably computationally exponential. Sampling based planners can often create plans in high-dimensional spaces efficiently but for this algorithm, rather than compute the C-space explicitly, it sample it. [9]

Figure 6 shows an example of path planning using PRM.

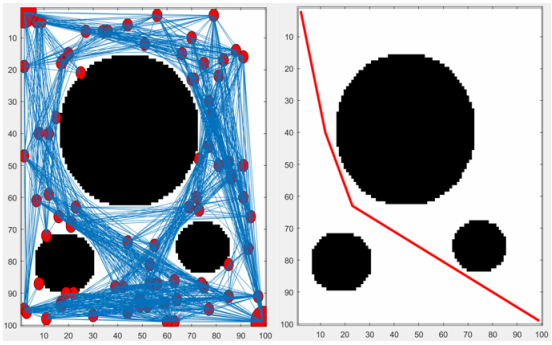


Fig. 6. Generated path of PRM

The more random samples are taken, the better output is given, but also a higher computation time is required.

III. SIMULATIONS

In order to analyze the algorithms exposed previously in this article, we have developed a simulator tool using matlab, that allows us to change easily some settings such as map, type of algorithm and type of simulation, the GUI is presented in the figure 7

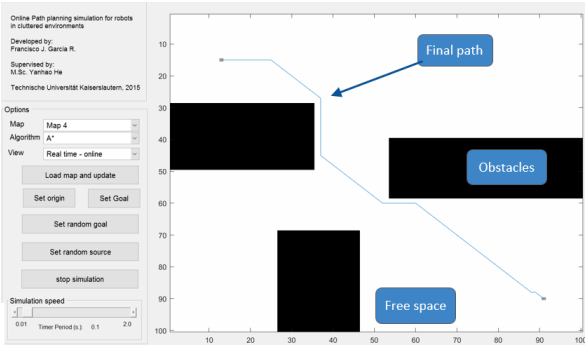


Fig. 7. GUI of simulator

The procedure for on-line simulation (simulator's algorithm) can be summarized as:

- The user selects a map, type of algorithm (A* - RRT - PRM - APF), initial and goal position of the robot
- Loads map and update the location of robot
- Add uncertainties to its position in X and Y (*)
- Use selected algorithm for computing the path
- Repeat from second step until goal is reached or stop condition is given

The simulator considers uncertainties around the robot position due to the top-view camera localization assumption, the figure 8 shows an estimated robot's uncertain region, that adds the errors due to pixel accuracy of the camera into simulation. Robot can be placed randomly inside of uncertain area plotted as a gray region.

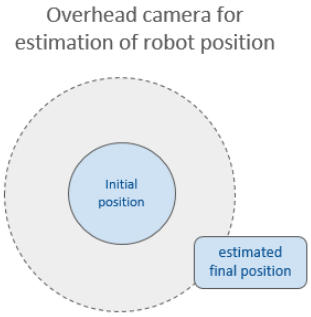


Fig. 8. Robot motion uncertainties due to camera pixel accuracy

Using the simulator were tested many different scenarios with all possible combinations but we have chosen only two different scenarios in order to clarify and explain the main features of all four algorithms:

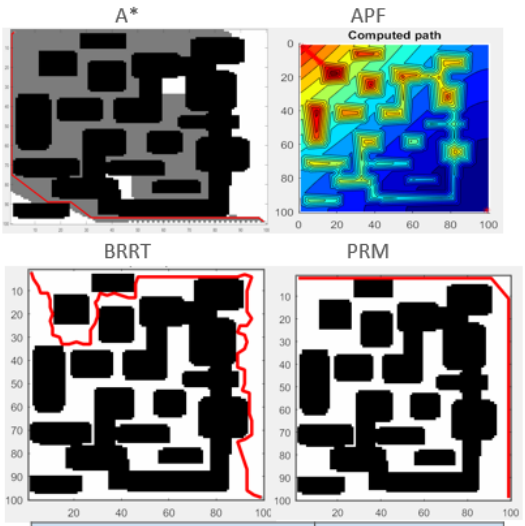
A. *Scenario 1*

It consider a map full of narrow passages and small obstacles, with initial position placed on the top-left side and goal point located into the bottom-right part. After simulation of this scenario we conclude that the best paths are given by A* and PRM but the computation time was also higher for this two algorithms. APF fails due to the narrow passages and the complexity of this map. In case of A*, the algorithm must cover almost the whole operational space before can find a optimal solution making the process slower than other approaches.

It can be observed that path generated by PRM is smoother and shorter than BRRT's path, and considering that both of them are probabilistic approaches, this results might appear as a circumstantial example but in fact, on one hand, the BRRT path grows "locally" with predefined maximum steps and is blinded to see the final goal, limiting a further path optimization, on the other hand, PRM has spread random samples in the entire operational space and can find the shortest distance based on its random points. Of course and due to the randomized nature of this algorithms, it might be possible to find some cases in which BRRT's path is shorter

than PRM's path but it was not the case for all experiments that we have made using the simulator.

Simulation results are shown in figure 9



Algorithm	Computation time (ms)
A*	**1192**
Potential field	**3**
Bidirectional RRT	**402**
Probabilistic Roadmap	**613**

Fig. 9. Path planning on scenario 1

B. Scenario 2

This scenario considers a smaller number of obstacles and adds some round objects into the map. We can observe on figure 10 that the best paths are given by A* and PRM, with a higher computation time than others, but now might be a better option to choose BRRT because even if the path is not as smoother as A*, its computation time was much more smaller.

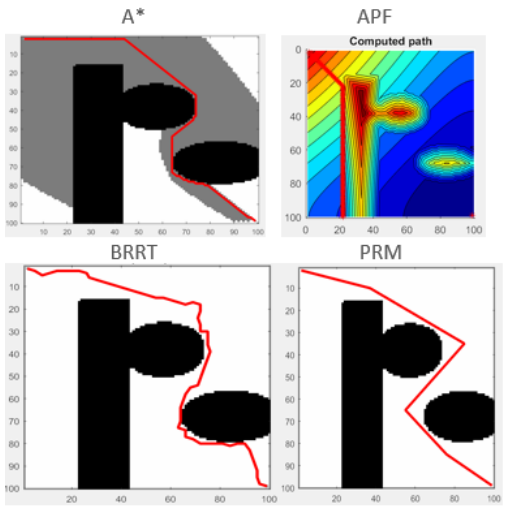
It is also important to point out that APF approach has failed in both scenarios 1 and 2 due to the presence of many local minima on the maps, given by multiples objects with complex shapes and small passages. It can be implemented a more advance algorithm of APF that can avoid local minima as we mentioned before on section II.

Simulation results for scenario 2 are shown in figure 10

C. Scenario 3

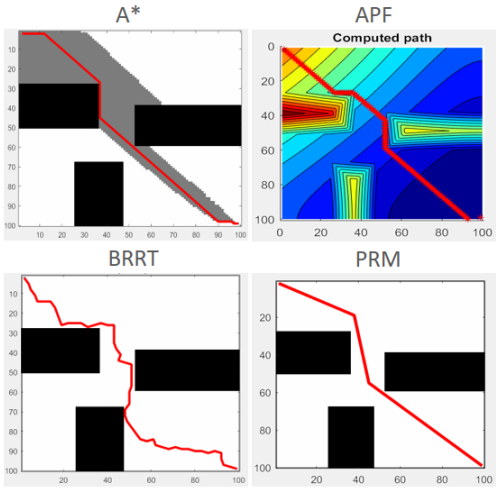
This scenario includes three rectangular objects with a big gap between them, and it tries to simulate a dynamic object crossing between the robot and its goal, as we can observe on figure 11, the computation time of APF and BRRT was smaller than other algorithms and even better than previous scenarios due to the absence of narrow passages. It suggest us that this algorithms have a good performance for local collision avoidance.

Simulation results for scenario 3 are shown in figure 11



Algorithm	Computation time (ms)
A*	**1716**
Potential field	**4**
Bidirectional RRT	**68**
Probabilistic Roadmap	**1504**

Fig. 10. Path planning on scenario 2



Algorithm	Computation time (ms)
A*	**809**
Potential field	**5**
Bidirectional RRT	**36**
Probabilistic Roadmap	**1814**

Fig. 11. Path planning on scenario 3

IV. CONCLUSIONS

After considering different scenarios and combinations between algorithms, maps and (initial-goal) robot's positions, it is possible to summarize the main observations in the following list:

- A* algorithm is usually slower than randomized ones but it can guarantee an optimal (smooth) path. We can speed up the computation time by reducing resolution of the map, but if its resolution is too small the objects are blurred and A* can not longer guarantee optimality

of the final path and even the feasibility of a solution.

- Planners based on random variables are suitable for high dimensional configuration spaces like robotic arms (many degrees of freedom), because they do not require to explore the full configuration space in order to find a suboptimal solution.
- For mobile robots, it is advisable to implement A*-based or RRT-based algorithms for path planning, because they can usually find a suitable path and their implementation is relative easy. It is also important to add a local collision avoidance algorithm using for instance APF into the control loop, because the global path planners are usually too slow for dynamic environments. A local collision avoidance algorithm should work from 10 to 30 hz and a global path planners requires commonly from 1 to 3 hz, depending on how fast change the robot's environment.
- There are multiple variations of this global path planners, because they usually do not fulfill all requirements at the same time, meaning that there is no universal global planner. Instead, each of this algorithm has special properties that make it suitable for specific problems.

The simulator's code was released as open source using license GPL and can be downloaded from `https://github.com/francisc0garcia/Path_Planning_Simulator`. Part of the source code was developed based on a project published by Rahul Kala at Robotics and Artificial Intelligence Laboratory from Indian Institute of Information Technology [10].

REFERENCES

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[2] Q. Zhu, Y. Yan, and Z. Xing, "Robot path planning based on artificial potential field approach with simulated annealing," in *Sixth International Conference on Intelligent Systems Design and Applications*, vol. 2. IEEE, 2006, pp. 622–627.

[3] H. Adeli, M. Tabrizi, A. Mazloomian, E. Hajipour, and M. Jahed, "Path planning for mobile robots using iterative artificial potential field method," *IJCSI International Journal of Computer Science Issues*, vol. 8, no. 4, 2011.

[4] M. G. Park and M. C. Lee, "A new technique to escape local minimum in artificial potential field based path planning," *KSME International Journal*, vol. 17, no. 12, pp. 1876–1885, 2003. [Online]. Available: http://dx.doi.org/10.1007/BF02982426

[5] Q. Zhu, Y. Yan, and Z. Xing, "Robot path planning based on artificial potential field approach with simulated annealing," in *Sixth International Conference on Intelligent Systems Design and Applications*, vol. 2, Oct 2006, pp. 622–627.

[6] L. Zhou and W. Li, "Adaptive artificial potential field approach for obstacle avoidance path planning," in *Computational Intelligence and Design (ISCID), 2014 Seventh International Symposium on*, vol. 2, Dec 2014, pp. 429–432.

[7] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[8] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug 1996.

[9] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[10] R. Kala. (2014) Code for robot path planning using a* algorithm. [Online]. Available: http://rkala.in/codes.php