Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.777647

# OBSTACLE DETECTION AND PATH PLANING FOR ROBOTS

**Govinda. K**
SCOPE, VIT, Vellore
kgovinda@vit.ac.in

**Chintalapati Akhil**
SCOPE, VIT, Vellore, India
chintalapati.akhil2020@vitstudent.ac.in

## Abstract
Obstacle detection bot is a smart robot that can identify barriers in its path and rapidly choose a new path to deliver / move safely and without interruption. These robots can be seen in modern restaurants, hospitals, and other congested areas where items must be delivered quickly and safely. Topics linked to robotics have been one of the most researched subjects in recent years. Meanwhile, intelligent mobile robots are popular, but controlling and navigating them is challenging, and the inability to deal with permanent impediments and avoid them through safe and secure routing is a key necessity of these systems. Motion planning, also known as path planning (sometimes known as the navigation issue or the piano mover's problem), is a computational problem that involves determining a set of viable configurations for moving an object from one location to another. Computational geometry, computer animation, robotics, and computer games all utilize the phrase. The computation of a continuous path that connects a start configuration S and a goal configuration G while avoiding collision with known obstacles is a basic motion planning problem. In a 2D or 3D workspace, the robot and obstacle geometry are defined, and the motion is represented as a path in (potentially higher-dimensional) configuration space. The subject of optimal path planning is a well-studied topic in academics that has found applications in a variety of industries. Google Maps is a widely used application that mainly relies on path finding algorithms, image processing, satellite navigation, and sensor-based systems, among other things. Autonomous vehicles are futuristic technologies that will be on the road in all developed countries within the next five years. All of the challenges listed above necessitated the development of speedier algorithms for finding the shortest path from a source to a target or multiple goals while avoiding static or moving impediments. So the proposed work aims to compare four different path planning algorithms, namely Bug 1 Algorithm, Bug 2 Algorithm, D* Algorithm and A * algorithm and through an thorough comparative study conclude which of them is the best algorithm.
Keywords: Route, Distance, Left, Right, Grid.

## Introduction
A mobile robot is a vehicle that can move around on its own. Mobile robots have been widely used in indoor environments such as cleaning large buildings, transportation in industry, surveillance in large buildings, and outdoor environments such as agriculture, forest, space,

underwater, military, firefighting, sewage tubes, and mining as robotic technologies have progressed.

Autonomous mobile robots are being developed for a variety of applications where long-term capabilities are advantageous. The primary goal of mobile robotics is to develop entirely autonomous robots that can execute tasks without the need for human involvement.

Mobile robots' industrial and technical applications are growing in importance, particularly in terms of reliability (uninterrupted and reliable execution of monotonous tasks such as surveillance), accessibility (inspection of sites that are inaccessible to humans, such as tight spaces, hazardous environments, or remote sites), and cost (transportation systems based on autonomous mobile robots can be cheaper than standard track-bound systems).

An autonomous mobile system should be able to plan its route to its destination, identify and avoid obstacles along the way, and arrive at the destination with a reasonable degree of precision. The ability to avoid obstacles is critical in any mobile robotic system. The robot must be trusted to carry out its duties without endangering itself or others. The prerequisites for a better obstacle avoidance algorithm include speed, robustness, and independence from past knowledge of the surroundings.

Mobile Robots (MRs) are now widely employed in a variety of industries, including military, industrial, agricultural , and many others. Robot route planning, which enables the MR go from the start point to the target point under obstacle limitations while obtaining the shortest path with minimal energy consumption and the shortest running time, is one of the most concerning topics in the field of robotics. In reality, there are two forms of MR path planning: global and local.

The Global path planning methodology is a static motion planning method in which the MR's trajectory is determined before the MR moves. When the environment is well-known and the terrain is static, the MR's trajectory is created (no dynamic obstacles). Although this motion planning type ensures that the goal point will be achieved or that the target point is inaccessible, it also aids in determining if the goal point will be reached. Unfortunately, when using the global path planning type, two requirements, namely a known environment and a static terrain, only exist in perfect circumstances. It is difficult to obtain a reliable map of the obstacles, and there is no guarantee that these obstacles will remain static in the face of several other environmental variables and occurrences.

The other type, known as local path planning , is a dynamic motion planning strategy in which the MR's trajectory is produced online based on current information observed by the MR's onboard sensors. Because of its quick response to changing forms and obstacle positions in a dynamic environment, this motion planning method is known as a more flexible and reliable method than the previous type.

As we all know, autonomous MR navigation is required for a variety of applications, including rescue robots searching for and rescuing persons trapped within collapsed buildings during disasters . Additionally, when some events occur unexpectedly on the MR's predetermined trajectory, the global path planning is unavailable.

However, we discovered that the local path planning method has limitations: I Without a global information environment, the local path planner cannot guarantee global convergence to the objective point . (ii) As a result, it may become stuck at some "local minima spots" and require recalculation of waypoints as the MR moves .

Humanity's future is in the hands of robots. It is critical to understand the algorithms used to create them. Robots have the ability to replace all restaurant servers, delivery employees, and a variety of other hazardous tasks in nuclear power plants, among other places. As a result, the creation of such robots will improve technology while also reducing manual involvement in all subjects.
So there are various path planning algorithms to make obstacle detection robot. In this project we aim to study four such algorithm. We have tried to understand the algorithm and the code to implement the algorithms. Then we have tried to implement 2 of the most widely used algorithms for path planning namely A * and D *. Both of these algorithms have their own advantages and disadvantages. Both are useful in different scenarios. So in the end we have concluded the project by giving a comparative report of the merits and demerits of the 4 chosen algorithms. Also we have given our conclusion about which among them is best for obstacle detection and path planning.

Path planning algorithms are used by mobile robots, unmanned aerial vehicles, and autonomous cars in order to identify safe, efficient, collision-free, and least-cost travel paths from an origin to a destination. Choosing an appropriate path planning algorithm helps to ensure safe and effective point-to-point navigation, and the optimal algorithm depends on the robot geometry as well as the computing constraints, including static/holonomic and dynamic constrained systems, and requires a comprehensive understanding of contemporary solutions. The goal of this paper is to help novice practitioner's gain an awareness of the classes of path planning algorithms used today and to understand their potential use cases—particularly within automated or unmanned systems. To that end, we provide broad, rather than deep, coverage of key and foundational algorithms, with popular algorithms and variants considered in the context of different robotic systems. The definitions, summaries, and comparisons are relevant to novice robotics engineers and embedded system developers seeking a primer of available algorithms.

## LITERATURE REVIEW

In this research paper they have studied about BUG algorithms for robot navigation and path planning. However, our conclusion according to the project is that A* algorithm is the best algorithm because it gives the shortest path between the start and the end point so it is more optimal and efficient for path planning robots compared to the various BUG algorithm they have studied here [1].

In the research paper they have concluded that A star algorithm is best for search and rescue which is correct because A* is fast and efficient. But when search operations are taking place, we do not know the position of all the obstacles. So, our conclusion according to the project is D * algorithm is the perfect algorithm in this case as this where there may be moving obstacles as D * algorithm modifies itself whereas A* starts from scratch [2].

In this research paper titled "Modified bug-1 algorithm based strategy for obstacle avoidance in multi robot system" they have used the Bug 1 algorithm for their path planning project.

However according to our observation in our project we have seen that the bug 1 algorithm is actually an exhaustive search algorithm, in this algorithm the robot takes an entire revolution around the hurdle's surface until it finds the point of departure with minimum distance from the goal point. This method of obstacle detection and path planning requires large memory and is less efficient. Thus, we find that it is better to use A * algorithm as it is faster and more efficient [3].

In this project they have built a path planning robot using Arduino uno and ultrasonic sensor. However the code that they have uploaded in the Arduino IDE is not that of A * algorithm so though it plans paths and avoids obstacles, it not give the shortest path hence consumes a lot of time [4].

In this research paper titled "D-Star Algorithm Modification", the D-star algorithm has been modified and been used for object detection and path planning of the robot. The D-Star algorithm has been proposed as the best algorithm for path planning. However, according to the results that we got from our project we have concluded that is the position of the obstacles are already known then the A-Star algorithm serves as a much better alternative for the path planning as it is faster and more effective than the D-Star algorithm[5].

## PROPOSED METHOD

### BUG 1 ALGORITHM

- Robot proceeds along the shortest path between robot position X and destination location until it comes upon a hurdle.
- When it comes face to face with an obstacle, it rotates around its surface and calculates the distance to the goal point.
- It determines the point of departure that is closest to the target after one complete revolution.
- Depending on the distance between the departure point and the hit point, it maintains or changes the direction of motion.

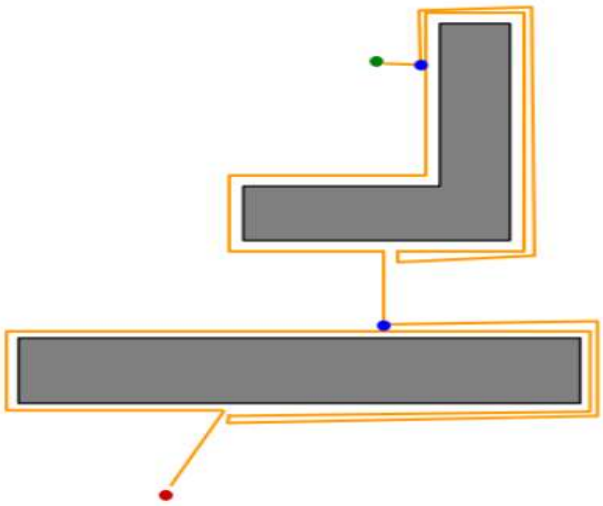

Fig1. Movement in Bug1 Algorithm

**ALGORITHM**

- Let $q^L_0 = q_{start}$; i = 1
- repeat
  - repeat
    - from $q^L_{i-1}$ move toward $q_{goal}$
  - until goal is reached or obstacle encountered at $q^H_i$
  - if goal is reached, exit
  - repeat
    - follow boundary recording pt $q^L_i$ with shortest distance to goal
  - until $q_{goal}$ is reached or $q^H_i$ is re-encountered
  - if goal is reached, exit
  - Go to $q^L_i$
  - if move toward $q_{goal}$ moves into obstacle
    - exit with failure
  - else
    - i=i+1
    - continue

**BUG 2 ALGORITHM**

- The Bug 2 algorithm is a modified version of the Bug 1 method.
- Instead of looking for the shortest distance, the Bug 2 algorithm concentrates on keeping the motion in the same direction as the goal.
- It determines the slope of the line connecting the starting point and the goal location. When a robot comes across an obstacle, it begins moving along the obstruction's edge until it finds a place with the same slope.
- It begins to move along the line that connects the point of departure and the point of destination.

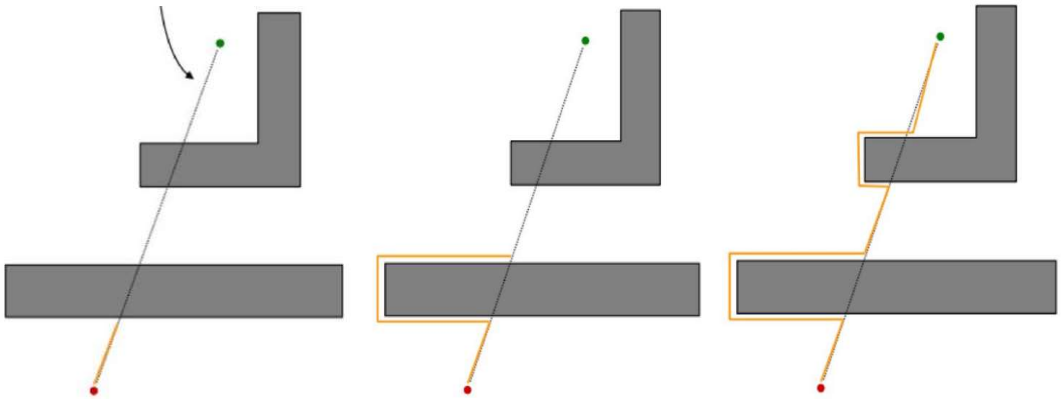

Fig2. Movement in Bug2 Algorithm

**ALGORITHM**

- Let $q^L_0 = q_{start;}\ i = 1$
- repeat
  - repeat
    - from $q^L_{i-1}$ move toward $q_{goal}$ along the m-line
  - until goal is reached or obstacle encountered at $q^H_i$
  - if goal is reached, exit
  - repeat
    - follow boundary
  - until $q_{goal}$ is reached or $q^H_i$ is re-encountered or m-line is re-encountered, x is not $q^H_i$, $d(x,q_{goal}) < d(q^H_i,q_{goal})$ and way to goal is unimpeded
  - if goal is reached, exit
  - if $q^H_i$ is reached, return failure
  - else
    - $q^L_i = m$
    - i=i+1
    - continue

**D * ALGORITHM**

- D star Lite is a pathfinding technique that helps a programmer to determine the best path between two points in a known or partially known environment.
- D* starts by going backwards in time from the goal node. Each expanded node has a back pointer that points to the target's next node.
- When an obstruction is found along the intended path, all affected sites are re-added to the OPEN list. The algorithm investigates its neighbours to see if it can minimise the cost of the node.
- If not, all descendants of the node, that is, nodes with backpointers to it, will be propagated.

**A * ALGORITHM:**

- A * algorithm is a path-finding algorithm that looks for the shortest path between the starting and ending states.
- It's utilised in a variety of applications, particularly those that use maps. There are three parameters in the A* algorithm:
- g: This is the cost of travelling from the starting cell to the current one. That is, the total number of cells visited since the source was removed.

- h: This is the projected cost of transferring from the current to the end cell.
- f: It is equal to the product of g and h. f = g + h is the formula. This f value will be used by the algorithm to make judgments.
- The path with the least f value among all potential paths is chosen.
- This decision is made after each cell has been advanced.

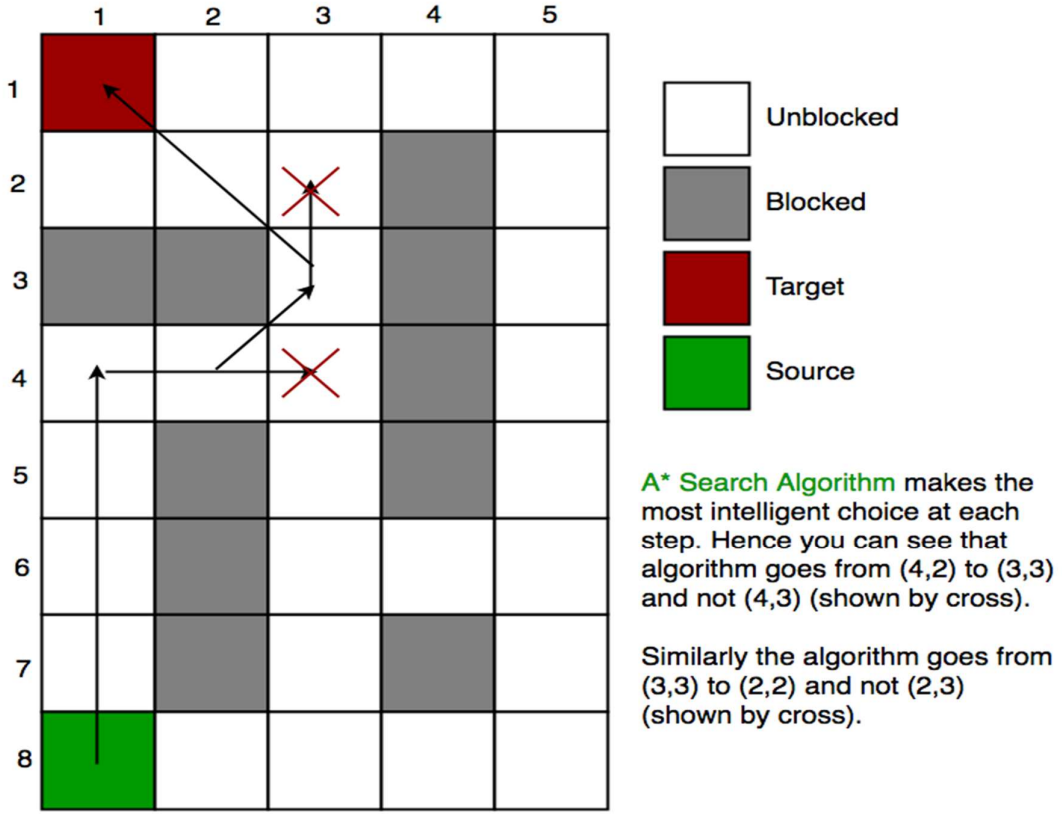


Fig3. Movement in A* Algorithm

**PSEUDOCODE**

- We must first initialise both the open and closed lists before proceeding to the main function. Both start out as an equal and open list of empty nodes.
- So, in this code, the maze, as well as the start and finish nodes, are provided to the astar function first. The initial node is now inserted to the open list first.
- We'll now execute a loop till the open list isn't empty.
- The current node is the one with the lowest f value at the start. After that, that value is moved from the open list to the closed list.

**Results and Discussion**

The following are some graphs used to compare the about stated algorithm:

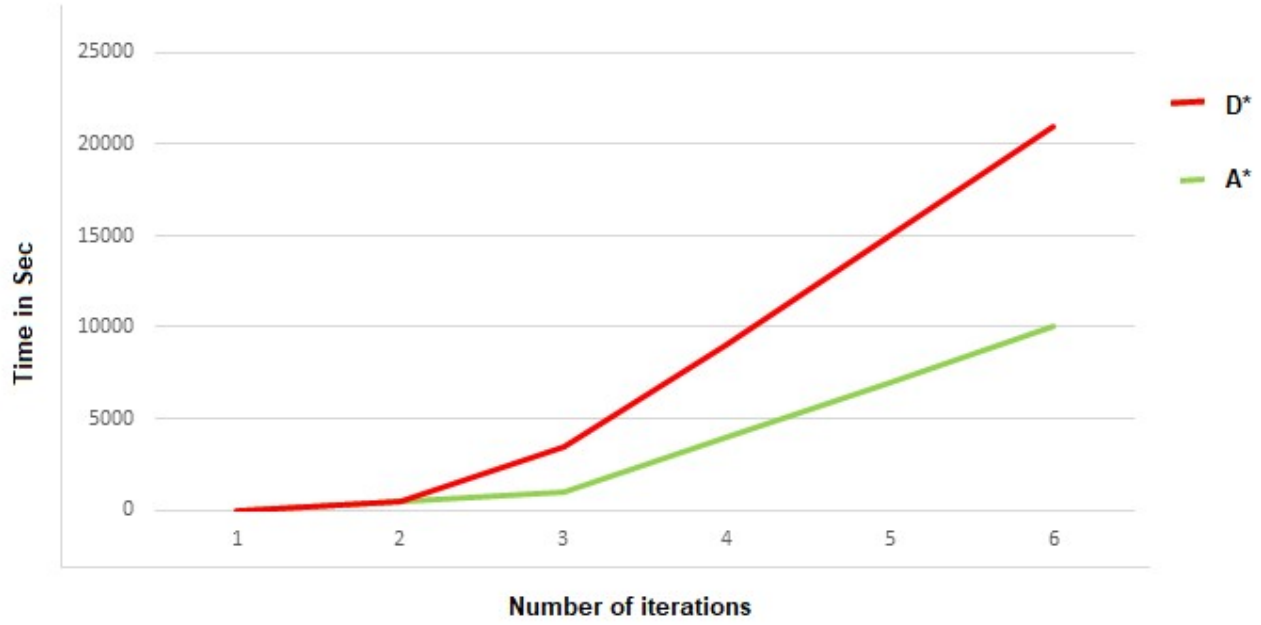


Fig4. Runtime Analysis of A* and D*

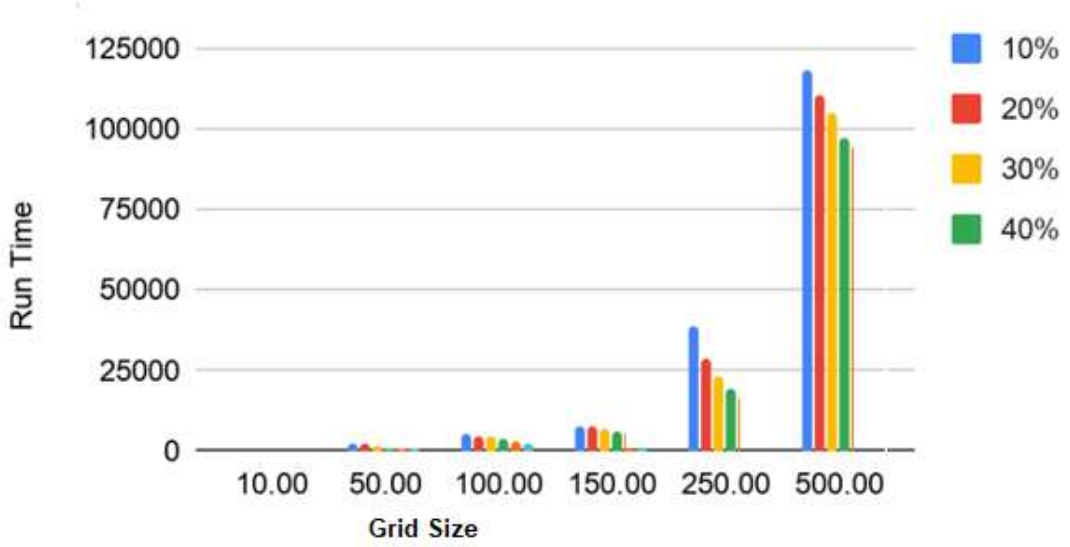


Fig5. Grid Size Vs Runtime of Algo's

Conclusion

So out of the above algorithms both BUG 1 and BUG 2 are outdated and they take a lot of time for planning the path and avoiding obstacles. Also, they do not give the shortest path between the starting point and the ending point. Whereas A * and D * provide the shortest path along with avoiding obstacles. So, these two algorithms are chosen over BUG 1 and BUG 2. Among these two the only difference is that for A * the path starts from the start to the end and then once the path is completed, it backtracks and gives the path whereas D * starts from the end and goes to the starting position. Also, A * algorithm takes much less time than D * but when the robot is in motion and the orientation of the obstacle is suddenly changed, then A *

algorithm starts from scratch once again whereas D* algorithm modifies its path as it is in motion. Hence when the robot moves in path where the position of obstacle is known then A * algorithm being the fastest algorithm and is the best. However if the robot moves in a path where the obstacles can change anytime then D * algorithm is much faster and hence the best choice

REFERNCES

1. K. N. McGuire1∗, G.C.H.E. de Croon1and K. Tuyls21Delft , A Comparative Study of Bug Algorithms for Robot Navigation
University of Technology, The Netherlands2University of Liverpool, United Kingdom.
2. Xiang Liu , A Comparative study on A star algorithm for search and rescue School of Electronic Information and Control Engineering Beijing university of Technology Beijing 100124, China
3. Kandathil, Jom & Mathew, Robins & Hiremath, Somashekhar. (2018). Modified bug-1 algorithm based strategy for obstacle avoidance in multi robot system. MATEC Web of Conferences. 144. 01012. 10.1051/matecconf/201714401012.
4. Kadry, Seifedine & Alferov, Gennady & Fedorov, Viktor. (2020). D-Star Algorithm Modification. International Journal of Online and Biomedical Engineering (iJOE). 16. 108. 10.3991/ijoe.v16i08.14243