JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# Improved RRT and A* Path Exploration Algorithms for Mobile Robots

**[1]N. Rama Devi, [2]Dr. K. Shyamala**
[1]Professor, [2]Professor and Head
[1]Department of CSE, [2]Department of CSE
[1]CBIT, Hyderabad, India, [2]Osmania University, Hyderabad, India

***Abstract :*** Robotics is a broad field with many facets and active area of research. With the rise of artificial intelligence, an increasing amount of research and development is being dedicated toward making robots more intelligent and helpful. In today's fast-paced environment, almost all robotic applications need efficient, speedy, and secure communication among robot agents. Mobile robots can help people in a number of circumstances and have a broad range of uses. The document proposes research on cooperative behavior of robots in path planning in an environment. Here, two improvised path-finding algorithms, A* and RRT, in a robotic environment are employed for path finding. Furthermore, robotic cooperation is characterized by a cluster-forming technique. The approaches are compared on various performance metrics, such as the length of the path, smoothness and processing time. In order to make robotic communication safe and secure, Rijndael encryption is utilized. The purpose of this study is to provide a proposal for more research into the cooperative behavior of robots when it comes to route planning in an environment. The utility of two path-finding algorithms in a robotic environment, A* and Rapidly-Exploring Random Tree (RRT) is demonstrated through Matlab Robotic Simulator. Experimental results have been performed to find the shortest paths in a number of scenarios. A comparative analysis is also presented.

***IndexTerms*** **- Robotic simulation, path-finding algorithms, obstacle detection, A* algorithm, RRT algorithm, heuristic exploration.**

## I. INTRODUCTION

A robot is a machine, most often one that can be programmed by a computer, that is capable of autonomously performing a complex set of actions. An external control device may be used to lead a robot, or the control can be built inside the robot. While some robots are designed to imitate human forms, the vast majority of robots are task-oriented machines that prioritize functionality above emotional aesthetics. Swarm robotics is a technique for coordinating a large number of robots via the use of a large number of very simple physical robots [1]. It is expected that the robots' interactions with one another and with their surroundings result in the desired aggregate behavior. This approach evolved from research on artificial swarm intelligence, as well as developmental biology of insects, ants, and other naturally occurring swarming systems. Researchers have presented some innovative ways and means to simulate the behavior of robots [1-4]. Mobile robots are, now-a-days, grabbing the positions of humans by proving their performance in risky and sensitive applications. Again the effectiveness of deploying robots could be improved by putting them in networks so that the application to which it is dedicated will be executed. When the robots are engaged in a network, information sharing is a must and so the communication between the robots should be barrierless, delayless, reliable and secure.

Here improvements to two algorithms for path finding are presented in this paper based on A* and RRT approaches. Hereby, we discuss these algorithms to shoulder the problem of routing in networks of robots. Subsequently, in order to achieve security at the time of information passage, an efficient cryptographic scheme, Rijndael algorithm is also preented here. So that the robot network will exhibit high resistance against information hacking. Eventually, the network of robots will be a delayless and a secure network that is suitable for co-operative applications.

## II. BACKGROUND WORK

The robot's goal is to acquire as much data as possible about its surroundings. The issue is solved using forward simulation methods with an open-loop approximation[1]. The basic challenge in simulation is termed "forward" or "direct dynamics." To solve this issue, determine the forces and torques delivered to actuators to generate the desired motion [2]. Deep learning approaches are expanded to train a robotic grasping system to grab new things from raw monocular RGB pictures in a simulated environment [3]. Based on mobile autonomous indoor robots, actionable building data can be gathered in real time for further analysis and decision-making. They can cover large areas with a modest number of sensors on mobile indoor robots [4]. Communication is an important aspect. When it comes to being able to engage with one another, robots' capacity to communicate with one another is crucial. The ability of these systems to function properly is heavily reliant on the data exchange mechanism. When considering a scenario involving a driverless vehicle, it is possible that the vehicle may communicate with and receive data from other vehicles in order to make judgments. When deciding whether to use the brakes or the accelerator, the sophisticated technology built into the automobile will first detect and evaluate the surrounding environment before making a choice on which to employ.

Determinations are made on the basis of information gathered from one's immediate surroundings as well as from personal experience of interaction among robots. It primarily relies on the data sharing mechanism. For example, in a driverless car scenario, a car may share and receive data from other vehicles to make decisions. In order to make a decision on whether to apply the brake or accelerator, the intelligent system embodied in the car will perceive its environment. Based on the data it receives from the environment, decisions are made. Using decentralized, partly observable Markov decision processes to make decisions for multiple agents in a sequence has been studied recently and has shown promising results [5]. These strategies are based on interactions and cooperation in the local area. Yan et al. [6] provide a survey on multi-robot communication and coordination. The challenges of ubiquitous computing [7] can be addressed with the unification of other technologies like blockchain [8].

Path finding is critical in many sorts of robotic applications, from communication to physical exploration. It is also important in autonomous vehicles. It is basically reliant on a robust optimum route finding algorithm that may adopt a multihop graph-based method [9] [10] to deliver packets from a source to a destination in a reliable and delayless way. Path finding is essential in both motion-based planning [11] and task-based planning [12], and it is particularly important in motion-based planning.

## III. Robotic Ecosystem : Characteristics and Classification

A robotic ecosystem is formed with various elements. Here, we consider six elements as part of the ecosystem, including number of robots, robotic planning, decision-making, environment, coordination and communication [6]. This ecosystem has been depicted in Figure 1.

### 3.1 *Single-robot vs. Multi-robot Systems*

Single-robot system is made up of only one robot that can represent itself, its surroundings, and its interactions. In such a system, the robot is often built to do a job on its own. Multiple sensors are often combined into such robots, necessitating a complicated mechanism and a highly intelligent control system. On the other hand, there is a multi-robot system (MRS). Whether the group is homogeneous or heterogeneous, an MRS has more than one individual robot. Using an MRS over a single-robot system has various potential benefits, including greater geographical distribution, robustness, and overall performance.

### 3.2 Robotic Planning

The effort of devising a series of activities to attain a goal is referred to as "planning" [34]. Planning may be utilized in MRS to organize robots in order to complete the team task. Unfortunately, determining the best MRS strategy is usually an NP-hard job. As a result, the present task is to guarantee that planning is manageable and results in excellent results.

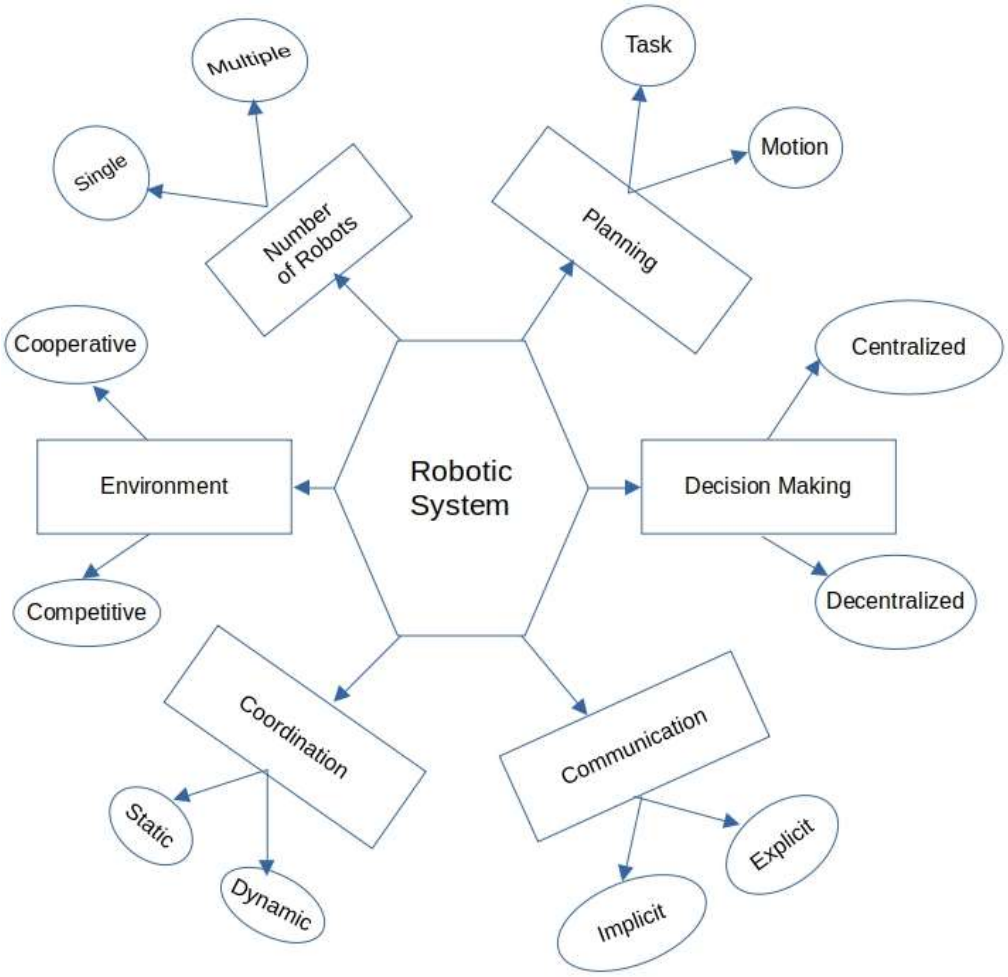



**Figure 1: Essential Elements of a Robotic Ecosystem**

### 3.3 Robotic Decision Making

Determining the best course of action from a variety of options is a cognitive process that results in the selection of one of many options. Every decision-making process comes to a conclusion at the end of it. Decision-making based on planning may be centralized or decentralized in the MRS, depending on the group architecture of the robots participating in the task.

### 3.4 Robotic Environment

The cooperative robots have a common objective that is divided into sub-goals. Robots compete against one other to better satisfy their own self-interests in a condition known as competition. Robots with opposing utility functions, on the other hand, are in competition with one another. The polar opposite of cooperative conduct is competitive behavior.

### 3.5 Robotic Coordination

The coordination in a robotic environment is either static or dynamic. Static coordination is governed by a set of rules that have been established. Coordination that is based on dynamic rules, on the other hand, is more difficult to achieve. Additionally, dynamic coordination may be divided into two types: explicit coordination and implicit coordination.

### 3.6 Robotic Communication

In multi-robot situations, communication as a way of coordination often arises as a reasonable behavior. In fact, communication is a kind of robot interaction. On the one hand, robots may communicate location information, environmental data, and sensor data with other robots. It can facilitate a robot to learn about the intents, objectives, and behaviors of other robots via this interaction. The mechanism for the direct transmission of information between robots is known as explicit communication, and it may take the form of unicast or broadcast deliberate messages. Implicit communication is the process through which a robot obtains knowledge about other robots in the system from its surroundings.

## IV. PROPOSED METHODOLOGY AND FRAMEWORK

### 4.1 Path Finding Algorithms

Depending on the nature of the assignment, there are a variety of path-finding algorithms to choose from [13-19]. The algorithms may be split into two categories: single-source-finding algorithms and all-source-finding algorithms. They may only work with negative weights, or they may only work with positive weights. In the literature, there are several path-finding algorithms to choose from. Dijkastra's algorithm, the Wallman Ford Algorithm, the A* Search, and other path-finding algorithms are among the most widely used. These approaches may also include metaheuristics, such as ant colony optimization (ACO), gray wolf optimization (GCO), the cuckoo search, and other variants of the search algorithm.

### 4.1.2 Improved A* Path Finding Algorithm

The A* algorithm is a popular technique to find the route between two nodes called source and destination. It works on the principle of various other well-known algorithms such as Dijkstra, Breadth First Search to find the shortest algorithms. It is very flexible and can be used in various contexts such as path planning, motion planning of Robots and many more. A* provides the optimal solution for the specific problems.

A* algorithm finds the optimal path which starts from the starting node, called source node, denoted by $X_{source}$ and passes through all the nodes of the graph to reach the destination node called $X_{target}$. It finds the adjacent nodes and selects the next node based on a cost function denoted by f(c). It calculates the total cost for moving from node $X_i$ to $X_j$ based on g(c)+h(c).

$$f(c) = g(c) + h(c) \quad (1)$$

In equation (1), f(c), g(c) and h(c) are all functions, wherein g(c) denotes the actual cost or distance between source node to the next traversing node and h(c) denotes the heuristic cost from next node to the destination. Optimality and the accuracy of the shortest path depend on the accuracy of the heuristic values. Our cost function should be more accurate to predict the accurate heuristic values. Nodes which are traversed in the graph are called exposed nodes and nodes which are yet to be traversed are called unexposed nodes.

Every time A* finds the successor node of the previous node in the graph, calculate its cost and move ahead in the graph to reach the destination.

Shortest path from source to destination is shown in figure 2 and figure 3. Figure 4 and figure 5 are the examples of shortest paths in dense environments where more numbers of interconnected nodes are available with more obstacles.

**Improved A* Path Finding Algorithm**

1. Assign Xsource as a source node in the graph
2. Assign Xtarget as a destination node in the graph
3. Assign Xunexposed as an open node which must be exposed in the graph.
4. Calculate the D(Xsources → Xunexposed) Distance of adjacent Xunexposed nodes denoted by Ni from the Xsource node.
5. Find the smallest value of D(XsourcesXunexposed), add this node, n, and move with it.
6. If D(XsourcesXunexposed) is empty
7. There is no path exist // there is an obstacle
8. While (n=! Xtarget ) // Destination is not reached
9. Mark Node n exposed node.
10. Perform the FindSuccessor() function
11. Re-calculate the D(XiXj) for all the successors nodes and mark them as Unexposed nodes
12. If D(XiXj) is empty // there is an obstacle
13. There is no path exist
14. Else If D(XiXj) is lesser than when ni was exposed
15. Successor exposed nodes of Ni are remarked as unexposed
16. End
17. End
18. Parent node n is exposed (Path Calculated)

**4.1.3 Improved Rapidly-Exploring Random Tree (RRT) Path Finding Algorithm**

There are many algorithms which are used for path planning and address the navigation problem. Rapidly-Exploring Random Trees (RRT) is one of those algorithms which can be used for creating the graphs and finding the path between two mentioned nodes of the graph. Figure 2 in the next section shows the working principle of the RRT algorithm.

The RRT algorithm selects the nodes randomly till nodes are available to explore in the graph. It generates a path in a unidirectional manner by avoiding the obstacles till the destination node is reached. In contrast to A* algorithm, RRT algorithm is significantly faster. The RRT algorithm lacks optimality with respect to the cost function. Path generated by the RRT algorithm are longer than A* algorithm. Manhattan and Euclidean distance measurement methods also affect the RRT algorithm. Path generated by the RRT algorithm is shown in figure 6.

**Improved Rapidly-Exploring Random Tree (RRT) Path Finding Algorithm**

1. Assign source node, Xsource, to the Tree T
2. Assign a destination node, Xtarget
3. **Repeat** for each edge of the graph
4. **While** ( Xtarget NOTϵ T
5. Find random node Xrandom ϵ Xavailable
6. Find the closest node Xnearest and connect Enearest Xrandom
7. **If** cost (Xnearest to Xrandom) > cost [Enearest Xrandom ]
8. Then, Xnew, Dist = cost [ Xnearest →Xrandom ]
9. **If** Node Xnearest and Xnew be connected to each other without any obstacle
10. Then add Xnew in the tree T
11. **End**
12. **Else** define Xnew,, dist = cost [ Xedge Xrandom ]
13. **If** Xedge and Xnew, be connected to each other without any obstacle
14. Then Add Xnew in tree T
15. **End**
16. **End**
17. **If** cost [Xnew Xtarget ] < dist AND ( Xnew and Xtarget ) connected to each other without obstacle
18. add Xtarget to tree T
19. **End**
20. **End**
21. **If** Xtarget is in the tree T
22. Path is found, show the path
23. **Else** no path found
24. **End**

**4.2 The Cryptographic Scheme for Networked Mobile Robotic Systems**

Cryptographic systems are required in a broad variety of situations. Because of the increasing growth of the internet, there is a continuing demand for effective ciphers. Cryptography is used in a variety of applications, such as email certification, identity verification, copyright protection, electronic watermarks, biometrics, and so on. Cryptography is used at a lower level to provide security for embedded devices, packet cables, and other systems.

Because of the widespread use of cryptographic systems, proper security in the transmission of robotic networks is required, and no robots other than the source and destination robots in the network may access the information. From numerous cryptographic techniques, Rijndael is selected as the cryptographic strategy for dealing with the security problems of a network of robots. Compared to the other algorithms, Rijndael meets the NIST security standards while using the least amount of hardware, hence it is chosen. Rijndael can encrypt and decrypt communications quickly, and it performs well on a variety of systems, from smart cards to 64-bit computers. It also requires a small amount of memory, and when compared to other algorithms, its implementation in hardware yields the best results [20].

The Rijndael (pronounced "rain-dahl") technique was recognized as a candidate for the Advanced Encryption Standard (AES) by the US National Institute of Standards and Technology (NIST) (AES). On the first list, there were more than 15 submissions. The field was subsequently whittled down to five finalists, with Rijndael becoming victorious. Vincent Rijmen and Joan Daemen, two Belgian cryptologists, invented this approach. The name of the cipher is a combination of the founders' surnames. The two cryptologists had previously worked together in the square, where Rijndael had begun his career.

The Rijndael algorithm is a next-generation symmetric block cipher that processes data in 128-bit blocks and supports key sizes of 128, 192, and 256 bits. The block sizes may, however, be equal to the key sizes, surpassing the AES design condition. Based on key/block sizes [21], Rijndael uses a variable number of rounds, as given below:

- For a 128-bit key/block, 9 rounds are utilized.
- A total of 11 rounds are utilized for a 192-bit key/block.
- 13 rounds are needed for a 256-bit key/block.

Rath et al. [20] discussed a security protocol with the IDS framework for a robotic ad hoc-network. Various cyber-physical threats for mobile robotic systems have been described in [21] and [22].

The following are some of the most significant characteristics for an encryption algorithm[23]:

a) resilience to all known vulnerabilities
b) speed and code compactness across different platforms
c) shortness of design

The non-linearity of the key expansion, which nearly prevents the possibility of similar keys, and the inverted usage of separate components, which virtually removes the possibility of weak and semi-weak keys, as occurs for DES, are two aspects of the new cipher's design that stand out. These are the two important benefits of the new encryption system [24].

## V. EXPERIMENTAL RESULTS

Path planning algorithms were implemented in robotic simulation set up in Matlab.

### 5.1 Experimental Setup

The experiments were done out on a machine running the Windows 10 operating system and equipped with 16 GB of RAM and a Ryzen CPU to get the findings. The software stack included Matlab R2021b, which was utilized for the calculations. In order to put these ideas into practice, the mobile robotic simulation toolbox in Matlab was employed.

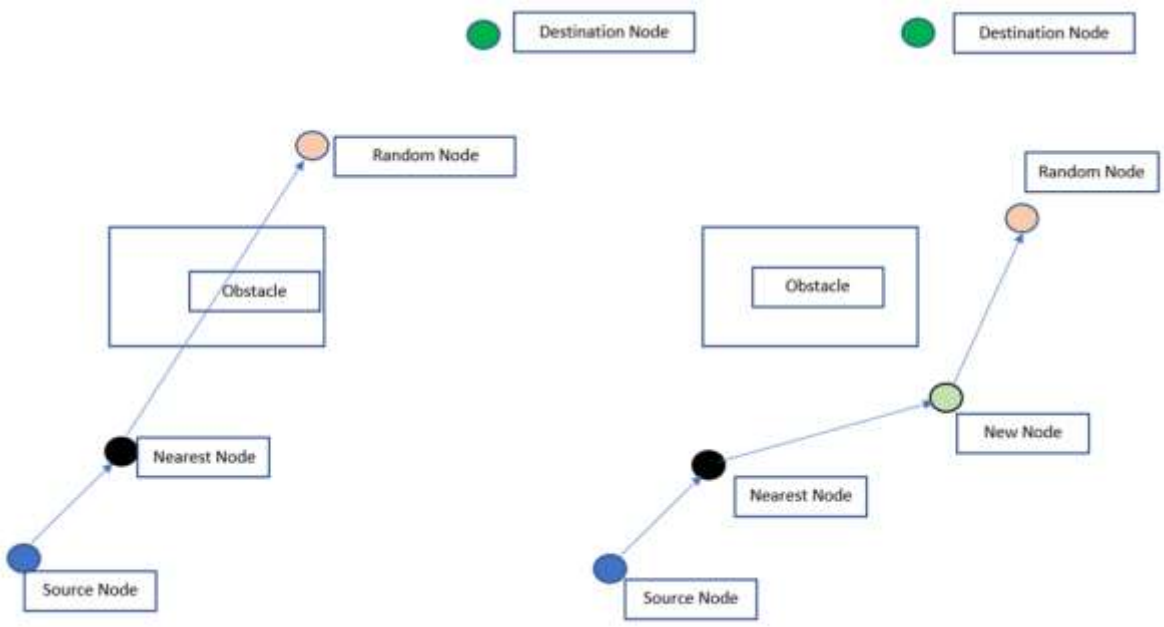
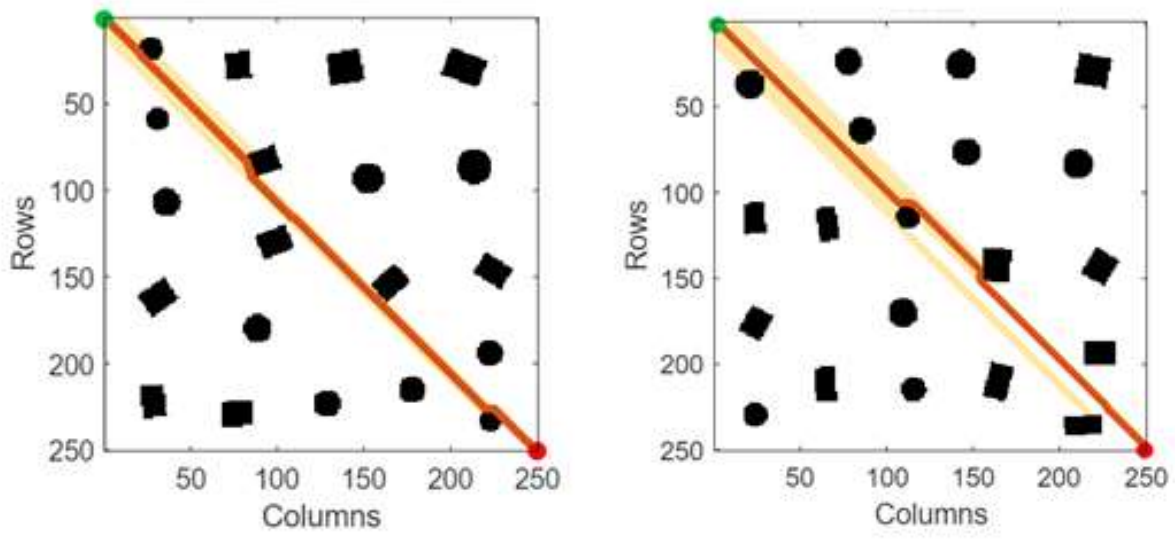



**Figure 2:** Successive iterative steps of Improved RRT algorithm to reach the destination

### 5.2 Path Finding Results of Improved A*

The results of the A* algorithm are shown in figures 3(a) and 3(b). In Figure 2(a), Xsource = [2,3] is the coordinates of the source node and Xtarget = [250,250] is the destination node. The A* algorithm finds the shortest path from source to destination by avoiding the obstacles. Similarly, in figure 3(b), the coordinates of the source and target nodes are the same but obstacles in the path are suffelled. Again, the A* algorithm finds the route between source and target node by avoiding obstacles.




(a) (b)

**Figure 3:** Shortest path from source to destination with various obstacles.

The A* algorithm applied to a dense network compared to Figures 3(a) and 3(b). In Figure 4(a), Xsources = [0,0] and Xtarget = [700,700]. After successfully executing the algorithm, we got the result mentioned in figure 4(a). The algorithm computed the shortest path between source and target nodes and avoided all the obstacles. In Figure 4(b), we changed the coordinates of the source node, Xsource = [0,0] and the target node, Xtarget = [400,700]. Algorithms again generated the shortest path between the mentioned nodes.

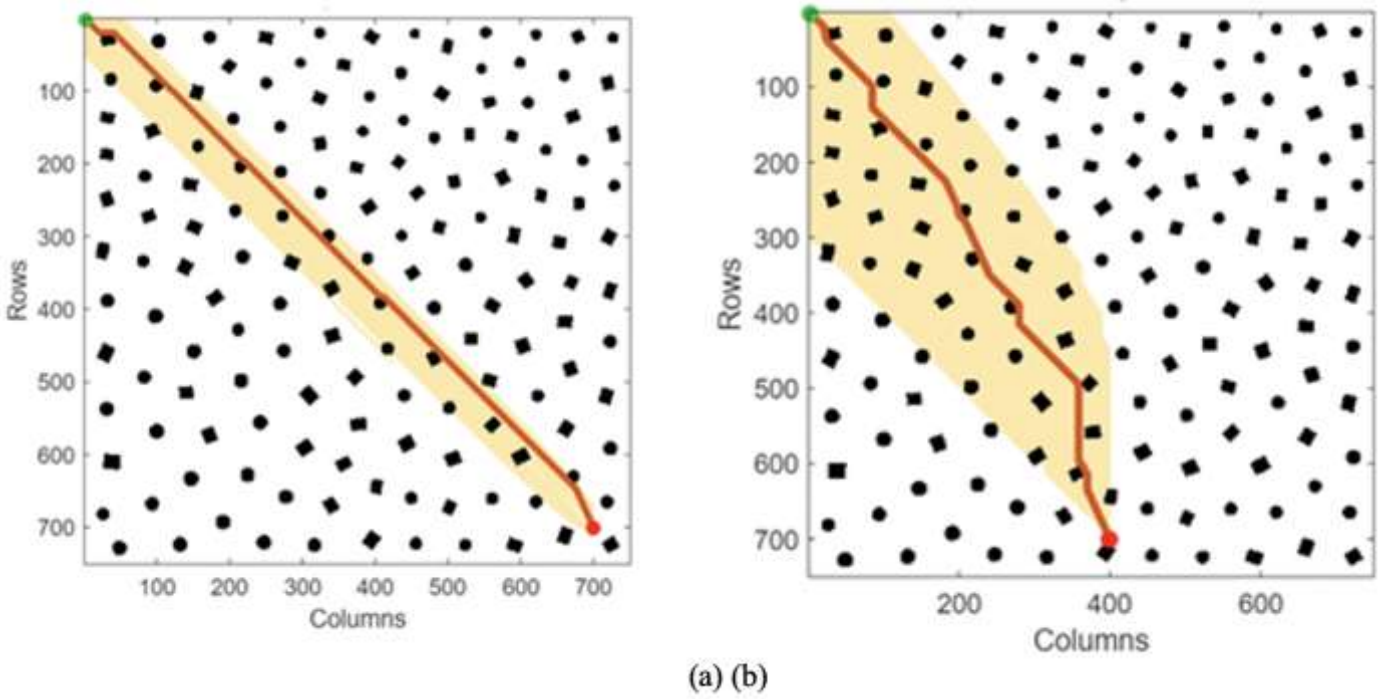



**Figure 4:** Shortest path in a dense environment with two different destinations.

### 5.3 Path Finding Results of Improved RRT

An improved RRT algorithm is executed to find the shortest path between source and destination with the mentioned coordinates Xsource = [0.5, 0.5] and Xtarget = [2.5, 0.5]. A few parameters are not tuned correctly, so the algorithm ignores the obstacles and finds the path between the source and target nodes.

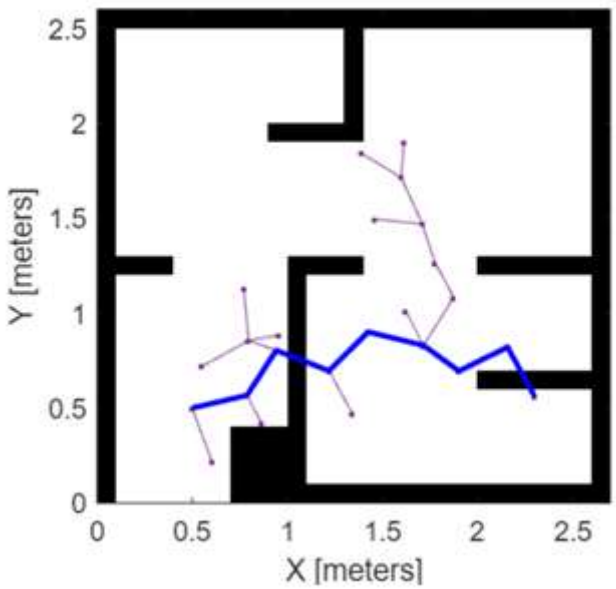
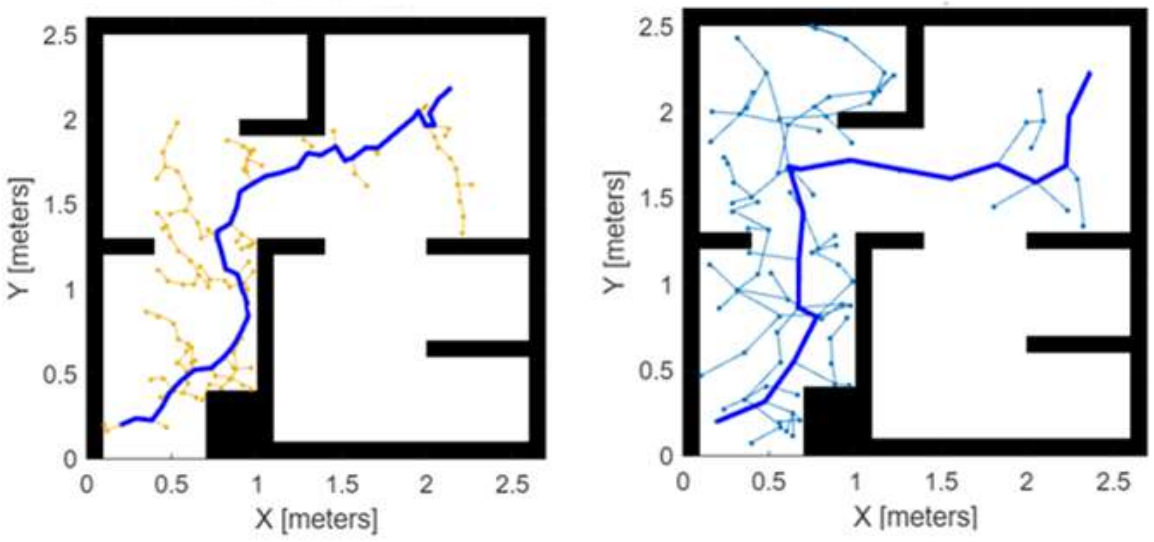



**Figure 5:** Shortest path between source and destination without considering the obstacles

Later, those parameters are tuned correctly and, based on the two different values, the algorithm generates two results for each value. Input values for Xsources = [0.20, 0.20], Xtarget = [2.25, 2.25]. This time, the algorithm generates the shortest path between source and target nodes and considers the obstacles too.




**(a) (b)**

**Figure 6 :** Improved performance of the RRT algorithm with tuned parameters to find shortest path between source and destination with obstacle consideration

**5.4 Comparative Analysis**

A* and RRT are both algorithms used to get the shortest path between two nodes in a definite time. In reference to the literature, the A* algorithm produces almost the optimal solution, while the solution generated by the RRT algorithm is not always optimal. Both the algorithms differ to each other in terms of their processing time.

In this paper, we improved both the algorithms to get the optimal results in output and get the resulting path in a definite time. It is also mentioned in Figure 6 that when some parameters are not correctly tuned, then RRT produces non-optimal results. Once there is some improvement in the used parameters, the algorithm generates a solution that is close to the optimal solution. Results generated by the RRT algorithms, shown in figures 6(a) and 6(b), are near to the optimal solution and in definite time also.

| Parameters | A* Algorithm | RRT Algorithm |
|---|---|---|
| ***Time Complexity*** | O(E) to O(b^d), Heuristic dependent, | O(n) |
| ***Space Complexity*** | O(b^d) | O(n logn) |
| ***Optimality*** | Provides Optimal Solution | Provides less Optimal solutions as compared to A* algorithm |
| ***Completeness*** | Provide complete path source to target node. | RRT also provides a complete path between source and target node. |
| ***Accuracy*** | High in comparison with RRT algorithm | Low |
| ***Processing Speed*** | Slower in comparison with RRT algorithm | Faster than A* algorithm |
| ***Path Length*** | Path between source to destination is Lower than RRT | Path between source and destination is higher to compared to A* |

Table 5.1 Performance Parameters for the Improvised A* and RRT Algorithms

In the A* time complexity, E is the number of edges, b is the branching factor and dis the depth.

**VI. CONCLUSION**

In the path planning problem, a feasible path from initial state to the goal region has to be found in the least amount of time possible. Identifying the shortest route between a source and a destination is always the most important consideration in path planning challenges. In such challenges, the length of the route and the amount of time required are the two parameters that must be optimized. A* and RRT are two methods that are used to identify the shortest route between two nodes, referred to as the source and destination, in a certain amount of time. A* and RRT are both deterministic algorithms. In this study, several parameters are modified in order to increase both the algorithms' performance in terms of optimum solution and their performance

in terms of running time. When compared to the RRT method, A* constructs the route in a shorter amount of time. The A* algorithm searched just the regions required for route creation, but the RRT method searched the path environment as well. The A* and RRT algorithms are also helpful for real-time route planning. These algorithms may be used in conjunction with sensors in a dynamic environment to determine the route between two nodes between two points. Because of its optimality and short computing time, A* is suitable for route planning in both static and dynamic situations. It is also effective with shuffling or dynamic barriers. When working in a 3D environment, RRT is very beneficial for path-planning. Various factors such as dynamic environment, moving obstacles, speed of the obstacles, size and shape of the obstacles, path optimality in a long distance path and time to generate the path will confirm the use of improved A* and RRT algorithm for autonomous robotic movement.

## REFERENCES


**[1]** Lauri, M., & Ritala, R. (2016). Planning for robotic exploration based on forward simulation. *Robotics and Autonomous Systems*, *83*, 15-31.

**[2]** Lilly, K. (2012). *Efficient dynamic simulation of robotic mechanisms* (Vol. 203). Springer Science & Business Media.

**[3]** Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., ... & Vanhoucke, V. (2018, May). Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 4243-4250). IEEE.

**[4]** Mantha, B. R., Menassa, C. C., & Kamat, V. R. (2018). Robotic data collection and simulation for evaluation of building retrofit performance. *Automation in Construction*, *92*, 88-102.

**[5]** Matignon, L., Jeanpierre, L., & Mouaddib, A. I. (2012, July). Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. In *Twenty-sixth AAAI conference on artificial intelligence*.

**[6]** Yan, Z., Jouandeau, N., & Cherif, A. A. (2013). A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, *10*(12), 399.

**[7]** Chibani, A., Amirat, Y., Mohammed, S., Matson, E., Hagita, N., & Barreto, M. (2013). Ubiquitous robotics: Recent challenges and future trends. *Robotics and Autonomous Systems*, *61*(11), 1162-1172.

**[8]** Ferrer, E. C. (2018, November). The blockchain: a new framework for robotic swarm systems. In *Proceedings of the future technologies conference* (pp. 1037-1058). Springer, Cham.

**[9]** Jung, J. H., Park, S., & Kim, S. L. (2010). Multi-robot path finding with wireless multihop communications. *IEEE Communications Magazine*, *48*(7), 126-132.

**[10]** Yu, J., & LaValle, S. M. (2013, May). Planning optimal paths for multiple robots on graphs. In *2013 IEEE International Conference on Robotics and Automation* (pp. 3612-3617). IEEE.

**[11]** Hönig, W., Kumar, T. S., Cohen, L., Ma, H., Xu, H., Ayanian, N., & Koenig, S. (2016, March). Multi-agent path finding with kinematic constraints. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*.

**[12]** Panov, A. I., & Yakovlev, K. (2017). Behavior and path planning for the coalition of cognitive robots in smart relocation tasks. In *Robot Intelligence Technology and Applications 4* (pp. 3-20). Springer, Cham.

**[13]** Cui, X., & Shi, H. (2011). A*-based pathfinding in modern computer games. *International Journal of Computer Science and Network Security*, *11*(1), 125-130.

**[14]** Bajrami, X., Dermaku, A., Demaku, N., Maloku, S., Kikaj, A., & Kokaj, A. (2016, June). Genetic and Fuzzy logic algorithms for robot path finding. In *2016 5th Mediterranean Conference on Embedded Computing (MECO)* (pp. 195-199). IEEE.

**[15]** Ashish, D. D. V., Munjal, S., Mani, M., & Srivastava, S. (2021). Path finding algorithms. In *Emerging Technologies in Data Mining and Information Security* (pp. 331-338). Springer, Singapore.

**[16]** Mohanty, P. K., & Parhi, D. R. (2016). Optimal path planning for a mobile robot using cuckoo search algorithm. *Journal of Experimental & Theoretical Artificial Intelligence*, *28*(1-2), 35-52.

**[17]** Brand, M., Masuda, M., Wehner, N., & Yu, X. H. (2010, June). Ant colony optimization algorithm for robot path planning. In *2010 international conference on computer design and applications* (Vol. 3, pp. V3-436). IEEE.

**[18]** Mac, T. T., Copot, C., Tran, D. T., & De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, *86*, 13-28.

**[19]** Raja, P., & Pugazhenthi, S. (2012). Optimal path planning of mobile robots: A review. *International journal of physical sciences*, *7*(9), 1314-1320.

**[20]** Rath, M., & Pattanayak, B. K. (2019). Security protocol with IDS framework using mobile agent in robotic MANET. *International Journal of Information Security and Privacy (IJISP)*, *13*(1), 46-58.

**[21]** Ahmad Yousef, K. M., AlMajali, A., Ghalyon, S. A., Dweik, W., & Mohd, B. J. (2018). Analyzing cyber-physical threats on robotic platforms. *Sensors*, *18*(5), 1643.

**[22]** Basan, A., Basan, E., & Gritsyni, A. (2019, October). Overview of Information Security Issues for a Robotic System. In *2019 IEEE 19th International Conference on Communication Technology (ICCT)* (pp. 1275-1279). IEEE.

**[23]** Penchalaiah, N., and R. Seshadri. "Effective Comparison and evaluation of DES and Rijndael Algorithm (AES)." *International journal of computer science and engineering* 2.05 (2010): 1641-1645.

**[24]** Zammit C, Van Kampen EJ. Comparison between A* and RRT algorithms for UAV path planning. In2018 AIAA guidance, navigation, and control conference 2018 (p. 1846).