Int J Syst Assur Eng Manag (October 2021) 12(5):990–1000
https://doi.org/10.1007/s13198-021-01186-9

ORIGINAL ARTICLE

Unmanned aerial vehicle path planning based on A* algorithm and its variants in 3d environment

Dilip Mandloi[1] · **Rajeev Arya**[1] · **Ajit K. Verma**[2]

Received: 15 March 2021 / Revised: 10 May 2021 / Accepted: 28 June 2021 / Published online: 9 July 2021

Abstract Finding a safe and optimum path from the source node to the target node, while preventing collisions with environmental obstacles, is always a challenging task. This task becomes even more complicated when the application area includes Unmanned Aerial Vehicle (UAV). This is because UAV follows an aerial path to reach the target node from the source node and the aerial paths are defined in 3D space. A* (A-star) algorithm is the path planning strategy of choice to solve path planning problem in such scenarios because of its simplicity in implementation and promise of optimality. However, A* algorithm guarantees to find the shortest path on graphs but does not guarantee to find the shortest path in a real continuous environment. Theta* (Theta-star) and Lazy Theta* (Lazy Theta-star) algorithms are variants of the A* algorithm that can overcome this shortcoming of the A* algorithm at the cost of an increase in computational time. In this research work, a comparative analysis of A-star, Theta-star, and Lazy Theta-star path planning strategies is presented in a 3D environment. The ability of these algorithms is tested in 2D and 3D scenarios with distinct dimensions and obstacle complexity. To present comparative performance analysis of considered algorithms two performance metrices are used namely computational time which is a measure of time taken to generate the path and path length which represents the length of the generated path.

Keywords UAV · A-star · Theta-star · Lazy Theta-star · 3D environment

1 Introduction

Recently, the demand for UAVs in the field of civil applications, military applications, nuclear power plants, artificial intelligence, and other hazardous environments are increasing rapidly. A UAV is an aircraft that can navigate without requiring an on-board human pilot (Pandey et al. 2018). The main challenge in the development of such intelligent UAVs is to generate a navigation path from the source node to the target node. Path planning signifies finding a route from a source node to a target node while avoiding collision with any object. To create a safe and practical path, the path planning algorithm should be capable enough to improve computational time, system effectiveness, should have environmental parameters incorporated as well, to acquiesce with the mission requirements (De et al. 2012).

In the present paper, literature survey of pathfinding has been thoroughly studied and different algorithms and their variants are proposed. These algorithms are commonly known as Graph search-based algorithms, Sampling-based algorithms, Bioinspired algorithms, and Numerical optimization algorithms (LaValle 2006; Yang et al. 2014; Gonzalez et al. 2016). In Graph search-based method, pathfinding problem is solved in two steps: In the first step,

✉ Rajeev Arya
rajeev.arya@nitp.ac.in

Dilip Mandloi
dilipm.phd19.ec@nitp.ac.in

Ajit K. Verma
ajitkumar.verma@hvl.no

[1] Department of ECE, National Institute of Technology Patna, Patna, , Bihar 800005, India

[2] Faculty of Engineering and Natural Sciences, Western Norway University of Applied Sciences, Haugesund, Norway

a graph is created by discretizing the continuous environment. In the second step, a path is generated by navigating along the created graph using a graph-search technique (Rabin 2019). Breadth-first search and depth-first search algorithms are the basic graph search algorithms (Quan et al. 2020). Examples of Graph search-based algorithms include Dijkstra's algorithm (LaValle 2006), A-star, Theta-star (Nash et al. 2007), Lazy Theta-star (Nash et al. 2010), Lifelong Planning A-star (LPA) (Koenig et al. 2004), Anytime repairing A $*$ (ARA $*$), and Anytime D $*$ (AD $*$) (Likhachev et al. 2008), Dynamic A* (D*) (Saranya et al. 2016), D*-Lite (Al-Mutib et al. 2011), etc. The concept of sampling-based algorithms relies on a random sampling of configuration space and searching inside it for connectivity (Gonzalez et al. 2016). The Probabilistic Roadmap Method (PRM) (Yan et al. 2013) and the Rapidly exploring Random Tree (RRT) (Zammit et al. 2020) are the two-fundamental sampling-based algorithms. Bioinspired algorithms (XSYang2020) are used as powerful optimization tools, these algorithms have originated from the natural biological evolution and social behavior of species. Numerical optimization techniques aim to minimize or maximize a function that is subject to various restricted variables (Gonzalez et al. 2016). The efficiency of any path finding algorithm depends on the various environmental constraints, therefore, care must be taken while choosing an algorithm to solve a path finding problem.

Among all the above mentioned strategies, A-star is the path planning strategy of choice because of its simplicity in implementation and promises of optimality. However, A-star algorithm guarantees to find the shortest path on graphs but does not guarantee to find the shortest path in a real continuous environment. Both Theta-star and Lazy Theta-star are variants of the A-star algorithm that can overcome this shortcoming of the A-star algorithm at the cost of increased computational time (Sartori et al. 2019; Nash and Koenig 2013).

This paper presents a comparative analysis of path planning algorithms such as A-star, Theta-star, and Lazy Theta-star in a 3D environment. To demonstrate the ability of considered algorithms in different 3D environmental scenarios with distinct dimensions, performance metrics parameters considered are computational time and path length. The comparative analysis presented in this paper is based on simulation results obtained through MATLAB and literature.

This paper includes six sections. Section 2 defines existing work to solve path finding problems in 2D and 3D environment. Problem formulation is described in Sect. 3. In Sect. 4, the A-star algorithm and its variants, i.e., Theta-star and Lazy Theta-star are briefly explained with their operation principles. Section 5 presents simulation results and comparative analysis of the algorithms. conclusive remarks and future scope are presented in Sect. 6.

2 Related work

In this section, existing studies proposed to solve the path-finding problems in different scenarios are discussed.

Application area of UAV includes missile launching, photography, parcel delivery, rescue scenarios, etc. To complete these tasks UAV needs to move along an aerial path from the source node to the target node thereby requiring a 3D path planning technique (Pandey et al. 2018). Path planning problems are considered as optimization problems where the ultimate goal is to create an optimum set of waypoints that connects the source node to the target node while avoiding obstacle and collision (Goel et al. 2018). In the literature, we found that noticeable research activities have been performed by researchers to explore the path planning algorithms in a 3D scenario. In (Yang et al. 2014, 2016; Quan et al. 2020), authors have presented a survey on 3D path planning algorithms which includes sampling-based algorithms, graph search-based algorithms, bioinspired algorithms, and numerical optimization algorithms.

Previously, several researchers have applied these algorithms to solve path planning problems. In (LaValle 2006) S. M. LaValle presented an extensive study on path planning algorithms but his work was limited to the 2D scenario perspective; a 3D scenario was not addressed by him in detail. The graph search-based algorithms have widely been used in the field of robotics and video game applications due to the ease of their implementation and low computational cost. As the visibility graph method could produce the shortest path in a 3D environment effectively, in Omar and Gu (2010), R. Omar et al. used this method as a 3D path planning algorithm. A. Nash et al. presented their work in Nash et al. (2010), on the Lazy Theta-star algorithm along with its two variants i.e., Lazy Theta-star-R and Lazy Theta-star-P in a 3D environment extensively. They focused on four parameters such as path length, expansions, line-of-sight checks, and computational time to make a comparative analysis of the Lazy Theta-star algorithm with A-star and Theta-star algorithms. Based on their analysis they indicated that paths found by A-star algorithm generated longer path than the path generated by the Theta-star and Lazy Theta-star algorithms. They also demonstrated experimentally that Lazy Theta-star outperformed Theta-star in terms of computational time. Methods proposed in Nash et al. (2010) were further explored by authors L. D. Filippis et al. In (De et al. 2012), authors presented an application of basic Theta-star, a variant of the A-star algorithm in 3D path planning. They made a

comparison of Theta-star and A-star algorithm concerning various parameters and found that Theta-star performed better as compared to the A-star. In addition to that A. Nash and S. Koenig (Nash and Koenig 2013) discussed Any-angle path-planning algorithms in both 2D and 3D environments. Some studies included other heuristic-based algorithms also, such as, the authors V. Jeauneau et al. (Jeauneau et al. xxxx) tested Genetic Algorithm (GA), and A-star methods using 16 different scenarios, J. Carsten et al. (Carsten et al. 2006) presented Field D* algorithm in a 3D environment. In (Aine and Likhachev 2013) the authors S. Aine and M. Likhachev combined two algorithms Anytime D* and truncation (TD* Lite) to develop a new algorithm known as anytime truncated D* in both 2D and 3D scenarios.

To solve path finding problems in high-dimensional spaces, sampling-based algorithms are preferred as compared to graph search-based algorithms. In (Silva Virk Tokhi Malheiro Ferreira Guedes 2017), the authors A. Dias et al. focused their study on two real-time path planning algorithms, i.e., Grid Path Planning Roadmap Planning (GPRM), and the Particle Probabilistic Roadmap (PPRM) which are based on a search-based algorithm PRM to reduce per iteration median time. J. Han (Han 2019) proposed a path planning method, "Critical Obstacles and Surrounding Point Set" (COSPS) by comparing it with PRM and Wavefront algorithm in terms of path length and computational time. Another commonly used sampling-based algorithm, RRT, and its variant RRT* in a 3D environment was addressed by the authors of Pharpatara et al. (2017). They indicated that the integration of artificial potential fields with RRT* improved its performance significantly in terms of the number of iterations. The comparative analysis of graph search-based algorithm and sampling-based algorithm were also presented in literature. In (Zammit and Kampen 2018), authors C. Zammit et al. considered path length and computational time as the performance metrics to present a comparative performance analysis of a graph search-based technique A-star and sampling-based technique RRT and its variants in 3D environments. They found that A-star outperformed RRT in terms of computational time with a guarantee of optimality.

Several latest researches also include bioinspired algorithms such as glowworm swarm optimization and particle swarm optimization for path planning in a 3D environment with static and dynamic obstacles. In (Pandey et al. 2018) the authors P. Pandey et al. applied a glowworm swarm optimization algorithm to solve 3D path planning problems and they found a reduced path length and better obstacle avoidance capability compared to Dijkstra, PSO, IBA, and BBO algorithm. Similar to Pandey et al. (2018), authors of Goel et al. (2018) U. Goel et al. applied glowworm swarm optimization in a 3D environment with dynamic obstacles. In (BFilipicEMinisciMVasile2020) authors A. Mirshamsi et al. showed that in a three-dimensional environment PSO algorithm suffered from a slow convergence rate which could be overcome by implementing a parallel approach and improving termination conditions. Multi-fusion Based Algorithms, which are designed by combining multiple algorithms, were also applied in recent studies. In (Albaghdadi and Ali 2019), authors A. F. Albaghdadi and A. A. Ali combined a potential field algorithm with a Genetic algorithm and simulated in multiple static and dynamic 3D environments to obtain an optimal path from the source node to the target node.

In literature, authors have explored A-star, Theta-star, and Lazy Theta-star algorithms to generate a path from the source node to the target node in a 3D environment but their studies are limited. Authors of De et al. (2012) have presented the comparative analysis of A-star and Theta-star algorithms, but they have not included Lazy Theta-star in their studies. They have also not shown the effect of the gain factor on path length, instead, only the effect on the computational time has been shown. The authors of (Nash et al. 2010; Nash and Koenig 2013) have explained A-star, Theta-star, and Lazy Theta-star algorithms in detail, but they have presented theoretical results only, thereby leaving simulation results missing in their studies.

3 Problem formulation

Solving the path finding problem in a real-life environment represented by a 3D coordinate space (x, y, z) is always a challenging task because it includes several obstacles with uneven shapes and sizes. To generate a safe and practical path from the source node to the target node in a complex 3D environment, these obstacles must be detected and avoided. The role of a path planning algorithm is to complete this task optimally and create a set of waypoints between the source and the target node. In this paper, the solution of the path finding problem in complex 3D environment has been presented. The solution is based on the application of A-star algorithm and its two variants, namely Theta-star algorithm and Lazy Theta-star algorithm. The contribution can be summarized as follows:

1. To formulate a path finding problem two 3D environment map which include multiple static obstacles of different dimensions, as mentioned in Tables 1 and 2, are created. To detect the obstacles, the points which are located inside the obstacle boundary are treated as unsafe points by assigning numerical value "one" to them while all remaining points are treated as safe

Table 1 Summary of the related works

S.N	Reference	Path planning strategy adopted	Description of Main contribution	3D Map	Weighted cost function	Metric considered	
						Path length	Computational time
1	Filippis et al. (De et al. 2012)	A-star, Theta-star	Comparative analysis of A-star and Theta-star algorithms has presented	Yes	No	Yes	Yes
2	Nash et al. (Nash et al. 2007)	Basic Theta-star, Angle-Propagation Theta-star	The correctness and completeness of adopted strategies has proved on the 2D grids	No	Yes	Yes	Yes
3	Nash et al. (Nash et al. 2010)	Lazy Theta-star	Comparative analysis of Theta-star and Lazy Theta-star has presented	Yes	No	Yes	Yes
4	Nash et al. (Nash and Koenig 2013)	A-star, Theta-star, Lazy Theta-star	Analytically presented the comparative analysis of the adopted strategies	Yes	No	Yes	No
5	Zammit et al. (Zammit et al. 2020)	A-star, RRT	Adopted strategies has validated for real time path planning	Yes	No	Yes	Yes
6	Tan et al. (Tan et al. 2016)	A-star, Artificial potential field	Solution of the path finding problem has proposed by combining the two adopted strategies	Yes	No	Yes	No
7	Pharpatara et al. (Pharpatara et al. 2017)	RRT-star, Artificial potential field	Artificial potential field and RRT-star algorithm has Integrated to increase the convergence speed	Yes	No	Yes	No
8	Pandey et al. (Pandey et al. 2018)	GSO	Modified GSO algorithm has proposed to solve path finding problems in 3D environment	Yes	No	Yes	Yes
9	Yan et al. (Yan et al. 2013)	PRM	octree algorithm has incorporated to PRM algorithm	Yes	NO	No	No
10	Goel et al. (Goel et al. 2018)	GSO	Solution of the path finding problem has been provided in an environment having moving obstacle	Yes	No	Yes	Yes
11	This Paper	A-star, Theta-star, Lazy Theta-star	Solution of the path finding problem has presented in complex 3D environment	Yes	Yes	Yes	Yes

Table 2 Description of terminology used in algorithms

Term	Description
OpenList	List of nodes to be visited during path finding
ClosedList	List of visited nodes
Target node (P_{target})	coordinates of target node i.e. (x_t, y_t, z_t)
Source node (P_{source})	coordinates of source node i.e. (x_s, y_s, z_s)
Current node ($P_{current}$)	coordinates of the node with lowest f_{value} in OpenList
Adjacent node (P_{adj})	coordinates of adjacent node to current node
Current node ($P_{current}$)	coordinates of the node with lowest f_{value} in OpenList
Adjacent node (P_{adj})	coordinates of adjacent node to current node
Line-of-sight	Line of sight path exist when $P_{current}$ and P_{adj} are visible to each other

points by assigning numerical value "zero" to them. The environment maps are illustrated in Figs. 1 and 2.

2. The solution to the path finding problem has been presented in two different scenarios by adopting three previously mentioned path finding strategies with weighted cost function. A set of paths is generated to demonstrate the effect of the gain factor included with the heuristic value of the cost function on the performance metrics.
3. To outline the advantage and disadvantages of the three adopted strategies a comparative analysis is presented

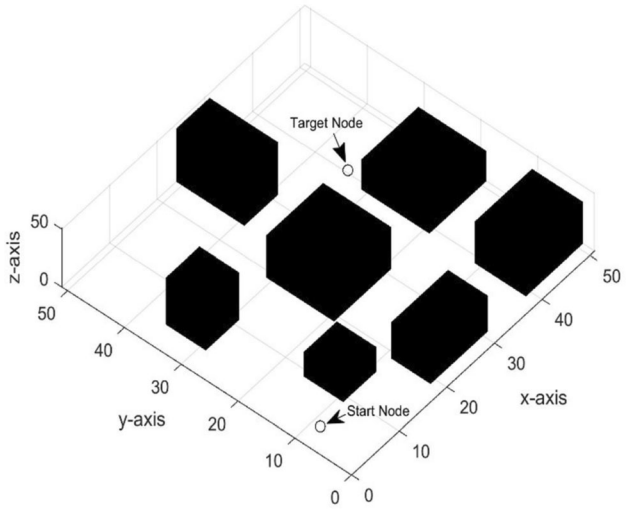

Fig. 1 Environment map for scenario-I

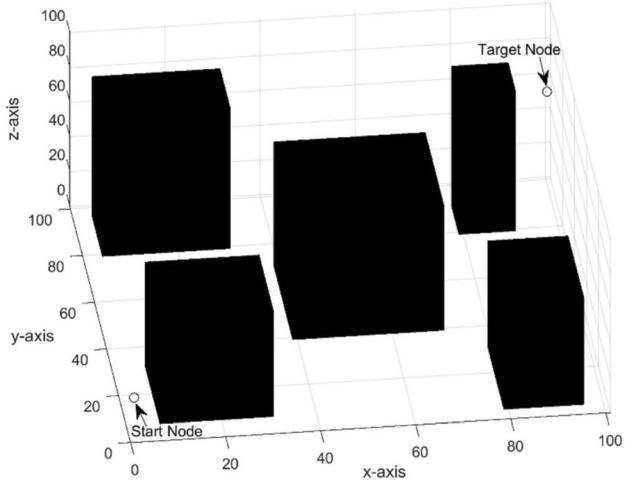

Fig. 2 Environment map for scenario-II

in terms of length of the generated path and time to compute that path.

4 The methodology

In this section, three path planning algorithms such as A-star, Theta-star, and Lazy Theta-star, are briefly explained with their operation principle.

4.1 The A-star algorithm

A-star, the most commonly used graph search-based path planning algorithm proposed in Hart et al. 1968 as an extension of Dijkstra's algorithm. A-star algorithm provides better computational time performance as compare to the Dijkstra's algorithm by including heuristic value (Koubaa Bennaceur Chaari Trigui Ammar Sriti Alajlan Cheikhrouhou Javed 2018).

In A-star algorithm, the process of finding shortest path from a source node to a target node relay on the constant f_{value}. The f_{value} is the cost of moving from the start node to the current node and it is evaluated by the following formula

$$f_{value} = g_{value} + g_f * h_{value}$$

where:

g_{value}: represents the total movement cost from the start node to the current node.

g_f: A constant number multiplied to the heuristic value to improve the algorithm performance is case of ties.

h_{value}: It represents the estimated movement cost from the current node to the target node.

Algorithm 1 presents the principal of A-star algorithm. To understand the algorithm, the description of terminologies used in algorithms is provided in Table 2.

Algorithm 1	
	//Initialize parameters: Mapsize, Obstacle coordinates, Coordinates of the Start node (P_{start}) and Target node (P_{target}), gain factor (g_f)
1	create an OpenList, put P_{start} as first entry of the list
2	create an empty ClosedList
3	while (OpenList is not empty && the P_{target} has not reached)
4	find node with lowest f value and consider it current node ($P_{current}$)
5	if ($P_{current}$) equals the (P_{target})
6	path found; break
7	remove ($P_{current}$) from OpenList and add it to the ClosedList
8	for (all the P_{adj} of $P_{current}$)
9	check the existence of (P_{adj}) on the map
10	set adjacent_node_cost equals sum of g value of $P_{current}$ and distance from the $P_{current}$ to the P_{adj}
11	if P_{adj} is in the OpenList
12	adjacent_node_cost < g (P_{adj})
13	remove P_{adj} from OpenList
14	if P_{adj} is in the ClosedList
15	adjacent_node_cost < g (P_{adj})
16	move P_{adj} from the ClosedList to the OpenList
17	else if P_{adj} is not in both lists
18	add the P_{adj} to the OpenList

4.2 The theta-star algorithm

Theta-star algorithm was proposed by authors of Nash et al. (2007) for path planning of any angle. Algorithm 2 shows the principle of Theta-star algorithm. Theta-star algorithm finds shorter paths compared to the A-star algorithm. The principle of theta star is same as A-star algorithm. The difference is that in case of A-star algorithm, parent node of a current node is a neighbor but in case of Theta-star it is not necessary for parent to also be a neighbor, rather, it can be any node. Theta-star algorithm checks line-of-sight between the node to generate the path.

Algorithm 2

```
//Initialize parameters: Mapsize, Obstacle coordinates, Coordinates of the Start node (P_start) and Target node (P_target), gain factor (g_f)
create an OpenList, put P_start as first entry of the list
create an empty ClosedList
while (OpenList is not empty & the P_target has not reached)
    find node with lowest f value and consider it as current node (P_current)
    if (P_current) equals the (P_target)
path found; break
remove (P_current) from OpenList and add it to the ClosedList
for (all the P_adj of P_current)
    check the existence of (P_adj) on the map
    set adjacent_node_cost equals sum of g value of P_current and distance from the P_current to the P_adj
    if P_adj is in the OpenList
        adjacent_node_cost < g (P_adj)
        remove P_adj from OpenList
    if P_adj is in the ClosedList
        adjacent_node_cost < g (P_adj)
        move P_adj from the ClosedList to the OpenList
    else if P_adj is not in both lists
add the P_adj to the OpenList
    if Line-of-sight between parent of P_current and P_adj
    if (g value of P_adj>sum of g value of parent of P_current and distance from parent of P_current to P_adj)
    set P_adj parent to the parent of P_current
    else if (g (P_adj)>sum of g (P_current) and distance from P_current to P_adj)
    set P_adj parent to the P_current
```

4.3 The lazy theta-star algorithm

The Lazy Theta-star algorithm is an extension of Theta-star algorithm proposed by the authors of Nash et al. (2010). The Lazy Theta-star algorithm generates path of same length as that of Theta-star in lesser amount of time and is comparable to the A-star algorithm. The Lazy Theta-star generate faster path than the Theta-star Because the Lazy Theta-star algorithm use lazy evaluation technique for expansion at each node so a single line-of-sight check is performed at each node. Algorithm 3 shows the principle of Lazy Theta-star algorithm.

Algorithm 3

```
//Initialize parameters: Mapsize, Obstacle coordinates, Coordinates of the Start node (P_start) and Target node (P_target), gain factor (g_f)
create an OpenList, put P_start as first entry of the list
create an empty ClosedList
while (OpenList is not empty && the P_target has not reached)
    find node with lowest f value and consider it as current node (P_current)
    if (P_current) equals the (P_target)
path found; break
remove (P_current) from OpenList and add it to the ClosedList
for (all the P_adj of P_current)
    check the existence of (P_adj) on the map
    set adjacent_node_cost equals sum of g value of P_current and distance from the P_current to the P_adj
    if P_adj is in the OpenList
        adjacent_node_cost < g (P_adj)
        remove P_adj from OpenList
    if P_adj is in the ClosedList
        adjacent_node_cost < g (P_adj)
        move P_adj from the ClosedList to the OpenList
    else if P_adj is not in both lists
add the P_adj to the OpenList
    if no line-of-sight between parent of P_current and P_adj
    set parent of P_current to P_adj with min (sum of g (P_adj) and distance from the P_current to the P_adj
    elseif (g value of P_adj>sum of g value of parent of P_current and distance from parent of P_current to P_adj)
    set P_adj parent to the parent of P_current
```

5 Simulation results

In this section, simulation results are provided to demonstrate the ability of the three algorithms described in the previous section.

This work presents the application of A-star, Theta-star and Lazy Theta-star algorithms with weighted cost function to find safe and real path from source node to target node in real life environment. To demonstrate the ability of these algorithms, a comparative analysis has presented in

terms of length of the generated path and time taken to compute that path.

To carry out the experiment, a two 3D environment scenario with different dimensions have been created. The details of map size, coordinates of the start node, coordinates of the target node, and coordinates to define obstacle position for two scenarios have been provided in Table 3. Here obstacles are created in the form of cuboids, however in the real-life environment UAV encounters obstacles with uneven shape and size. To detect the presence of any obstacle, the points which are located inside the obstacle boundary are treated as unsafe points by assigning numerical value one to them while all remaining points are treated as safe points by assigning numerical value zero to them. After defining the obstacle region on the environment map, source node and target node are mapped on the map using their co-ordinate points (x_s,y_s,z_s) and (x_t,y_t,z_t) respectively. The A-star, Theta-star and Lazy Theta-star algorithms with weighted cost function have been implemented to find a route from the source node to the target node. In each scenario, to demonstrate the effect of the gain factor included with the heuristic value of the cost function on the performance metrics, set of simulations has been performed with different values of gain factor. Table 4 shows the algorithms performance in terms of computational time and path length for different values of gain factor g_f.

In this work, all the simulations are carried out in MATLAB version 9.7.0.1190202 (R2019b), running on a machine with Intel i5 processor, 8 Gb RAM, 256 Gb SSD and 1 Tb HDD hard drive and Windows 10 OS.

Figures 3, 4, 5 shows the path generated from the source node (3, 8, 5) to the target node (35, 30 35) using A-star, Theta-star, and Lazy Theta-star algorithms with the value of gain factor equal to 2 in the map defined by scenario-I. The value of the length of the generated path and time taken to generate that path i.e., the computational time for three algorithms with distinct values of the gain factor shown in Table 4. The values mentioned in Table 4 are the average values obtained by repeatedly performing simulation ten times with each algorithm individually for the map defined by the scenario-I. Results show that among the three algorithms, the A-star algorithm requires less time to generate a path from the source node to the target node compared to the Theta-star and Lazy Theta-star algorithms. However, the A-star algorithm is faster than the Theta-star and Lazy Theta-star algorithms but, it generates a longer path than both the algorithms. The length of the generated path using Theta-star and Lazy Theta-star algorithms are the same but the Theta-star algorithm requires longer time to generate the path as compared to the Lazy Theta-star algorithm. The computational time of the Lazy Theta-star algorithm is comparable to the A-star algorithm, and its path length is the same as to Theta-star algorithm. So, the Lazy Theta-star algorithm is an optimum strategy to generate a path from a source node to a target node. Algorithm's performance was also checked for distinct values of gain factor. The results show that an algorithm becomes faster as the value of the gain factor increases. But this faster performance of the algorithm is obtained at the cost of a longer generated path. Figures 6, 7 and 8 shows the path generated from the source node (3, 18, 1) to the target node (98, 82 75) using A-star, Theta-star, and Lazy Theta-star algorithms with the value of gain factor equal to 2 in the map defined by scenario-II. The environment map defined by scenario-II is more complex than the map

Table 3 Details of environment map considered for simulation

Scenario No	Map size	UAV Coordinates		Obstacle Number	Obstacle Position (x_o, y_o, z_o)—(x_e, y_e, z_e)
		Start Node (x_s, y_s, z_s)	Target Node (x_t, y_t, z_t)		
I	50*50*50	3, 8, 5	35, 30, 35	1	(25, 40, 1)—(30, 50, 45)
				2	(10, 10, 1)—(15, 15, 20)
				3	(20, 20, 1)—(30, 30, 35)
				4	(40, 3, 1)—(50, 10, 35)
				5	(5, 30, 1)—(10, 35, 40)
				6	(40, 20, 1)—(50, 30, 25)
				7	(20, 3, 1)—(30, 8, 30)
II	100*100*100	3, 18, 1	98, 82, 75	1	(5, 5, 1)—(30, 30, 60)
				2	(5, 80, 1)—(30, 95, 80)
				3	(80,80, 1)—(90, 90, 80)
				4	(40, 40, 1)—(70, 70, 70)
				5	(80, 5, 1)—(95, 30, 60)

Table 4 Algorithms Performance in terms of computational time and path length for different values of gain factor g_f

Test No	Name of Algorithm	Gain factor gf	Scenario I		Scenario II	
			Computational time	Path Length	Computational time	Path Length
1	A-star	1.5	0.07733	58.24	0.13539	158.46
2	Theta-star		0.19379	55.04	1.13898	148.99
3	Lazy Theta-star		0.11503	55.04	1.04598	148.99
4	A-star	2	0.05583	58.83	0.11138	159.1
5	Theta-star		0.13241	55.25	0.23078	150.13
6	Lazy Theta-star		0.06602	55.25	0.207	150.13
7	A-star	5	0.04034	61.26	0.09926	159.92
8	Theta-star		0.08853	57	0.14833	150.13
9	Lazy Theta-star		0.05146	57	0.12243	150.13
10	A-star	10	0.03812	61.95	0.1118	161.43
11	Theta-star		0.07402	57.61	0.15734	150.13
12	Lazy Theta-star		0.04041	57.61	0.12163	150.13

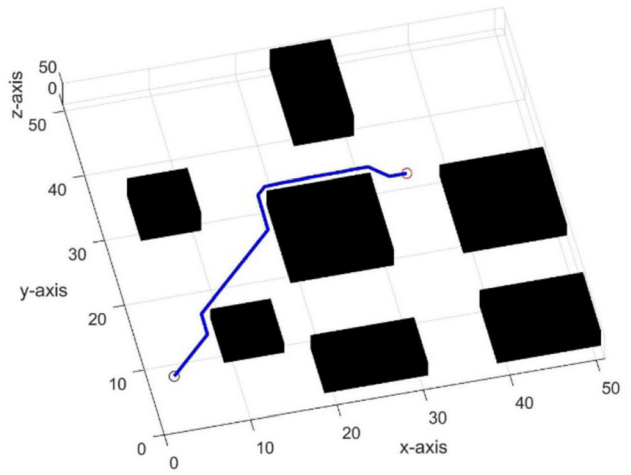

Fig. 3 Path generated in scenario-I map with $g_f = 2$ using A-star algorithm

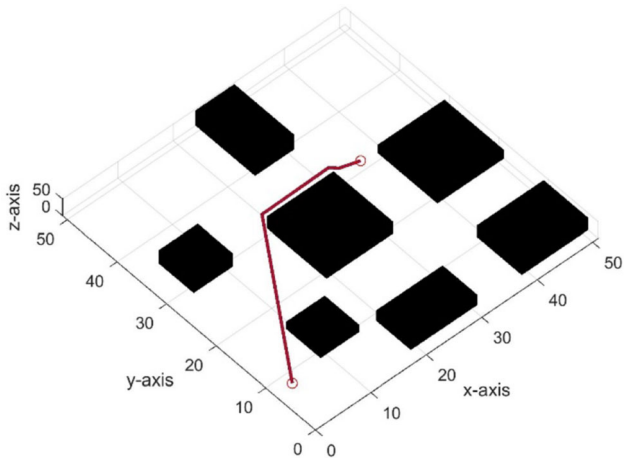

Fig. 4 Path generated in scenario-I map with $g_f = 2$ using Theta-star algorithm

defined by the scenario-I because in the case of scenario-I the map size is 50*50*50 so the number of nodes is 125000 while in the case of scenario-II the map size is 100*100*100 so the number of nodes is 1000000. In scenario-II, UAV encounters three obstacles to move from the source node to the target node while in the scenario-I it encounters two obstacles. The A-star algorithm generates a faster path in scenario-II as compared to the Theta-star and Lazy Theta-star algorithms. But the results shown in Table 3 indicate that as the complexity of the environment increases, the performance of the A-star algorithm becomes poorer while the performance of Theta-star and Lazy Theta-star improves. From the results shown it is also clear that the Theta-star and Lazy Theta-star algorithms generate a smooth path while the path generated by the A-star algorithm is not a smooth path.

In Figs. 9, 10 simulation results are presented to shows the comparison of the path generated by three algorithms in scenario-I and II. In the Figs. 9 and 10, only two paths are visible instead of three and the reason behind this is that the Theta-star and the Lazy Theta-star algorithms generate the same path so their paths are getting overlapped. Above results shows that the Theta-star and the Lazy Theta-star algorithms generates smooth and shorter path compared to A-star algorithm.

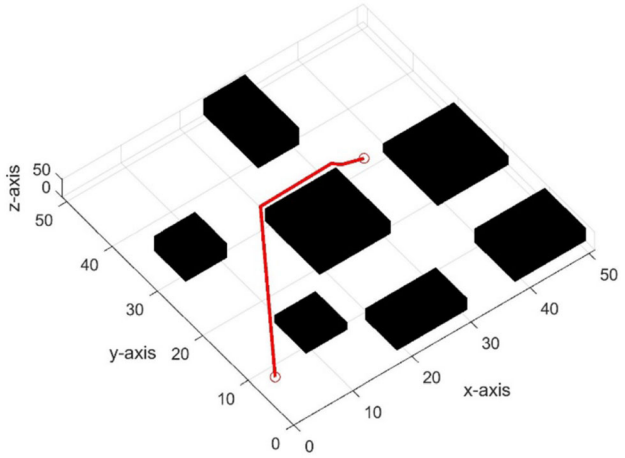

Fig. 5 Path generated in scenario-I map with $g_f = 2$ using Lazy Theta-star algorithm

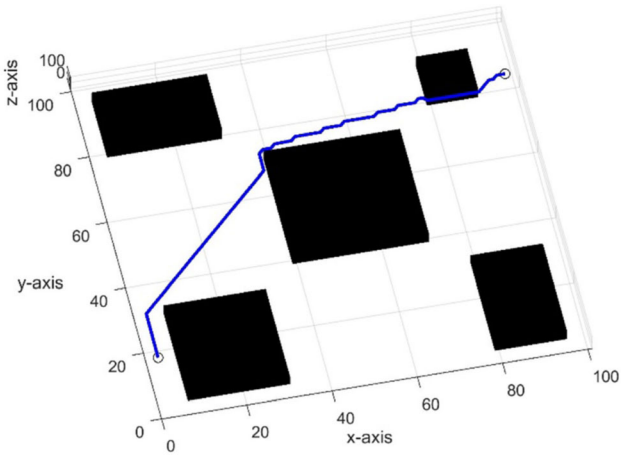

Fig. 6 Path generated in scenario-II map with gf = 2 using A-star algorithm

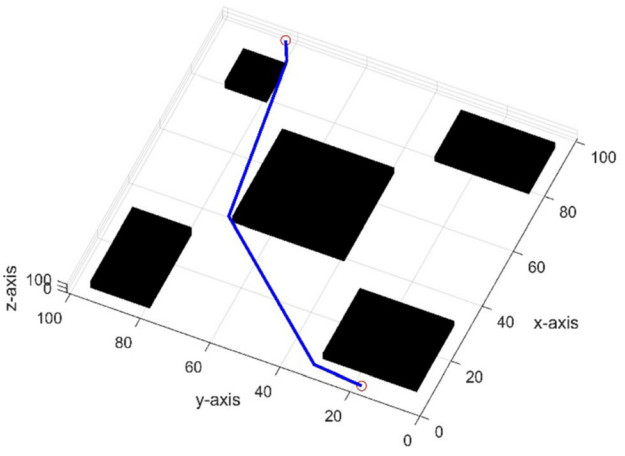

Fig. 7 Path generated in scenario-II map with $g_f = 2$ using Theta-star algorithm

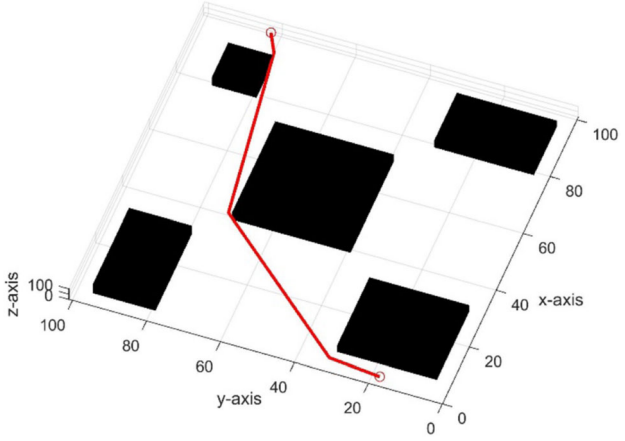

Fig. 8 Path generated in scenario-II map with $g_f = 2$ using Lazy Theta-star algorithm

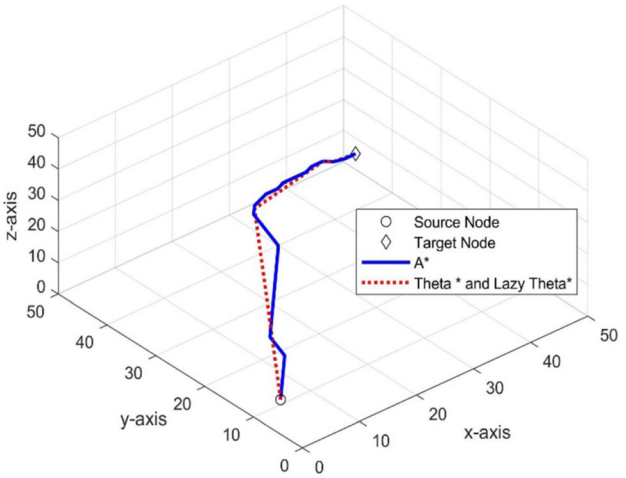

Fig. 9 Elevation of the path generated for Scenario-I using three algorithms with $g_f = 2$

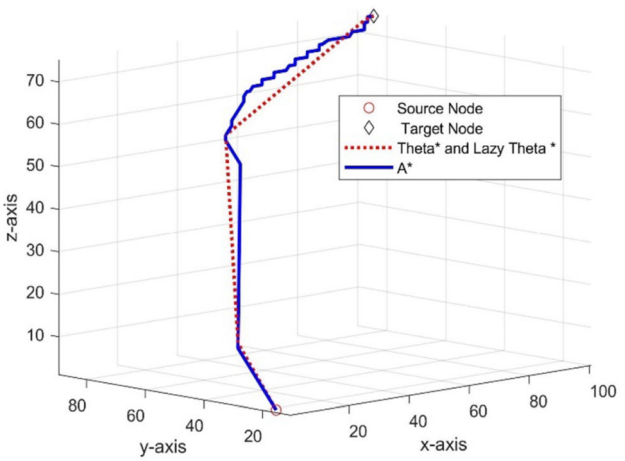

Fig. 10 Elevation of the path generated for Scenario-II using three algorithms with $g_f = 2$

6 Conclusion and future scope

In this paper, a comparative analysis of three path planning algorithms namely A-star, Theta-star, and Lazy Theta-star algorithms has been presented in the 3D environment. To demonstrate the ability of considered algorithms a 2D and a 3D scenario with distinct dimensions and obstacle density as described in Table 3 have been created. To compare the algorithms and to show the effect of the gain factor, two parameters, i.e., computational time and path length have been considered as performance metrics. The comparative analysis presented in this paper is based on the simulation results obtained through MATLAB and literature. Simulation results are shown in Table 4. They show that A-star finds a path faster than the Theta-star and Lazy Theta-star algorithms but it generates a longer path compare to the both algorithms. The Lazy Theta-star algorithms generate a path of the same length as Theta-star in less amount of time. So, the Lazy Theta-star algorithm is preferred to the A-star algorithm in terms of path length and to the Theta-star algorithm in terms of computational time. We also demonstrated how the path generation time of the algorithm can be improved by multiplying heuristic values of the cost function to a constant number, called a gain factor. Results shown in Table 4 indicate that an increase in the value of gain factor results in a reduction in the value of computational time at the cost of increased path length, so its value must be carefully selected to get optimum results.

We have provided a solution to the path finding problem in the scenario where a single UAV is moving in a known 3D environment having static obstacles. In the future, this work can be extended to include dynamic obstacles with multiple UAVs.

Funding There was no outside funding or grants received that assisted in this study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Aine S, Likhachev M (2013) Anytime truncated D*: anytime replanning with truncation. in Proceedings of the Sixth International Symposium on Combinatorial Search. 2–10

Albaghdadi AF, Ali AA (2019) 3D Path planning of fixed and mobile environments using potential field algorithm with Genetic algorithm. 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON) IEEE 115–119.

Al-Mutib K, AlSulaiman M, Emaduddin M, Ramdane H and Mattar E (2011) D* Lite Based Real-Time Multi-Agent Path Planning in Dynamic Environments, 3rd International Conference on Computational Intelligence, Modelling & Simulation, pp. 170–174.

Carsten J, Ferguson D, Stentz A (2006) 3D field D*: Improved path planning and replanning in three dimensions. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, 3381–3386

De FL, Guglieri G, Quagliotti F (2012) Path planning strategies for UAVS in 3D environments. J Intell Rob Syst 65(1):247–264

Filipic B, Minisci E, Vasile M (2020) Bioinspired optimization methods and their applications. Springer, Berlin

Goel U, Varshney S, Jain A, Maheshwari S, Shukla A (2018) Three-dimensional path planning for uavs in dynamic environment using glow-worm swarm optimization. Procedia Comput Sci 133:230–239

Gonzalez D, Perez J, Milanes V, Nashashibi F (2016) A review of motion planning techniques for automated vehicles. IEEE Trans Intell Transp Syst 17(4):1135–1145

Han J (2019) An efficient approach to 3D path planning. Inf Sci 478:318–330

Jeauneau V, Jouanneau L (2018) Path planner methods for UAVs in real environment. IFAC-Papers OnLine 51(22):292–297

Koenig S, Likhachev M, Furcy D (2004) Lifelong planning A*. Artif Intell 155:93–146

Koubaa A, Bennaceur H, Chaari I, Trigui S, Ammar A, Sriti MF, Alajlan M, Cheikhrouhou O, Javed Y (2018) Robot path planning and cooperation foundations. Algorithms and Experimentations, Springer, Berlin

LaValle S (2006) Planning algorithms, 1st edn. Cambridge University Press, Cambridge

Likhachev M, Ferguson D, Gordon G, Stentz A, Thrun S (2008) Anytime search in dynamic graphs. Artif Intell 172(14):1613–1643

Nash A, Koenig S (2013) Any-angle path planning. Artif Intell Mag 34(4):85–107

Nash A, Daniel K, Koenig S, Felner A (2007) Theta*: Any-angle path planning on grids. Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, Menlo Park, California

Nash A, Koenig S, Tovey CA (2010) Lazy. Theta*: Any-angle path planning and path length analysis in 3D. National Conference on Artificial Intelligence

Omar R, Gu D (2010) 3D path planning for unmanned aerial vehicles using visibility line-based method, In Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics. 80–85

Pandey P, Shukla A, Tiwari R (2018) Three-dimensional path planning for unmanned aerial vehicles using glowworm swarm optimization algorithm. Int J Syst Assur Eng Manag 9:836–852

Peter H, Nilsson N, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern 4(2):100–107

Pharpatara P, Herisse B, Bestaoui Y (2017) 3-D trajectory planning of aerial vehicles using RRT*. IEEE Trans Control Syst Technol 25(3):1116–1123

Quan L, Han L, Zhou B, Shen S, Gao F (2020) Survey of UAV motion planning. IET Cyber-Syst Robot 2(1):14–21

Rabin S (ed) (2019) Game AI Pro 360: guide to movement and pathfinding. CRC Press, Boca Raton

Saranya C, Unnikrishnan M, Ali SA, Sheela DS, Lalithambika VR (2016) Terrain based D_* algorithm for path planning. IFAC-PapersOnLine 49(1):178–182

Sartori D, Zou D, Yu W (2019) An efficient approach to near-optimal 3D trajectory design in cluttered environments for multirotor UAVs. in IEEE 15th International Conference on Automation Science and Engineering 1077–1022

Silva MF, Virk GS, Tokhi MO, Malheiro B, Ferreira P, Guedes P (2017) Human-centric robotics. World Scientific Press, Singapore

Tan J, Zhao L, Wang Y, Zhang Y, Li L (2016) The 3D Path Planning Based on A* Algorithm and Artificial Potential Field for the Rotary-Wing Flying Robot. 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC) 551–556

Yang XS (2020) Nature-inspired computation and swarm intelligence. Academic Press, Cambridge

Yang L, Qi J, Xiao J, and Yong X (2014) A literature review of UAV 3D path planning. in IEEE 11th World Congress on Intelligent Control and Automation, pp. 2376–2381

Yang L, Qi J, Song D, Xiao J, Han J, Xia Y (2016) Survey of robot 3D path planning algorithms. J Control Sci Eng 2016:1–22

Yan F, Liu YS, Xiao JZ (2013) Path planning in complex 3D environments using a probabilistic roadmap method. Int J Autom Comput 10:525–533

Zammit C, Kampen EJV (2018) Comparison between A* and RRT Algorithms for UAV Path Planning. AIAA Guidance, Navigation, and Control Conference, 1–23

Zammit C, Jan E, Kampen V (2020) Comparison of A* and RRT in real–time 3D path planning of UAVs. AIAA Scitech 2020 Forum