Path Planning Algorithms For Unmanned Aerial Vehicle: Classification, Performance, and Implementation

Ali Haidar Ahmad, Oussama Zahwe, Abbass Nasser, Benoit Clement

Proc. of the International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME 2023)
19-20 July 2023, Tenerife, Canary Islands, Spain

Path Planning Algorithms For Unmanned Aerial Vehicle: Classification, Performance, and Implementation

Ali Haidar Ahmad
Lab-STICC UMR CNRS 6285
ENSTA Bretagne
Brest, France
ali.haidar@ensta-bretagne.fr

Oussama Zahwe
ICCS-Lab, Computer Science Department
American University of Culture and Education
Beirut, Lebanon
osamazahwi@auce.edu.lb

Abbass Nasser
ICCS-Lab, Computer Science Department
American University of Culture and Education
Beirut, Lebanon
abbassnasser@auce.edu.lb

Benoit Clement
Lab-STICC UMR CNRS 6285
ENSTA Bretagne
Brest, France
benoit.clement@ensta-bretagne.fr

***Abstract*—— Path planning and obstacle avoidance form the basis of the UAV operations. The objective of an Unmanned Aerial Vehicle (UAV) is to navigate an optimal path towards its destination while ensuring the avoidance of the obstacles along the way. Several algorithms have been proposed by many researchers to achieve this objective. In this paper, we focused on Global path-planning algorithms for UAVs with obstacle avoidance. We compare various algorithms by highlighting their characteristics, advantages, and limitations. In addition, this paper implements four of the most well-known methods that tackle environmental challenges. Our results offer practical insights and guidance for researchers seeking to develop more effective path planning algorithms for UAVs.**

Keywords— UAV, Obstacle Avoidance, Path planning algorithms, Classification, Global path planning

I. INTRODUCTION

The use of unmanned aerial vehicles (UAVs) has gained popularity in various industries such as search and rescue, environmental tracking, mapping, surveillance, military operations, and more.[1]

One of the major challenges facing UAVs operations is path planning, which directly affects their power consumption. As a result, the primary objective is to ensure that the UAV arrives safely and in the shortest time possible. To this end it is necessary to use efficient path planning algorithms that can optimize the UAV's direction while avoiding obstacles and conserving energy. UAV path planning can be classified into two categories: global path planning and local path planning [2]. In addition, each of these categories can be further divided into subcategories. Moreover, we can distinguish two types of the flying environments a static environment where all conditions (obstacles, destination, weather…) remain constant over time, and a dynamic environment, where these conditions are subject to change [3]. However, the adopted path planning method should be adapted to the environment conditions so as the UAV can accomplish its mission successfully.

To ensure that the aerial vehicle is adequately tested prior to physical experimentation, simulation has been proposed as a reliable and efficient technique.. One such simulator, JSBSIM, is a versatile and object-oriented Flight Dynamics Model (FDM) that has been integrated into our approach, as suggested by [4].

In this paper, we classify UAVs path planning, while mentioning the optimal environment which they can be used for. To better illustrate the practical utility of these methods, we conduct experiments on four different path planning algorithms, each tailored to a specific environment and task. For example, one of the tasks is docking of a UAV on a mobile landing station.

II. UAV GLOBAL PATH PLANNING

The task of identifying the optimal direction while avoiding obstacles relies heavily on the algorithm's ability to optimize the route. For instance, compared to wheeled mobile robots, planning a path for UAVs is much more intricate due to their high-speed operating environment and significant dynamic challenges, which include obstacles, wind, and rain. Consequently, more sophisticated and precise algorithms are necessary to tackle the obstacle avoidance and path planning difficulties for UAVs Fig 1. focuses on Global Path Planning (GPP) algorithms, GPP aims to identify the optimal route for a UAV to travel from its starting point to its destination within a given environment, while avoiding obstacles and complying with any constraints, such as restricted airspace or minimum altitude requirements. As mentioned by Fig. 1, Four subgroups

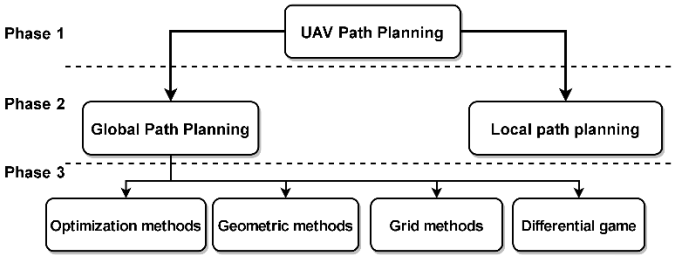

Figure 1 UAV Path Planning Classification Categories

of GPP can be distinguished: optimization methods, geometric methods, grid methods, and differential games.

A. *Optimization based methods*

The optimization-based methods form a subgroup of GPP that uses optimization techniques to find the best path through a given environment. They formulate the path planning problem as an optimization problem with an objective function that reflects desired characteristics of the path. Optimization is carried out using mathematical programming [5]or numerical optimization techniques [6]. These methods are useful in handling complex environments with multiple constraints and can generate optimal or near-optimal paths. In the following, we will present some Optimization-based methods.

Nonlinear programming (NLP): a method for optimizing solutions to problems that have nonlinear objective functions and/or constraints. It involves techniques such as quadratic programming, mixed-integer programming, and nonlinear least squares optimization [7]

Linear programming (LP): a mathematical optimization method used to find the optimal solution to a problem that involves linear objective functions and linear constraints [8]

Genetic algorithm (GA): an optimization method inspired by the process of natural selection. It involves generating a population of potential solutions to a problem, evaluating their fitness, and iteratively improving the solutions through a process of selection, crossover, and mutation [9]

Nature-inspired: methods use mathematical models to simulate the collective behavior of natural systems such as swarms of birds or colonies of ants. Particle Swarm Optimization (PSO) [10] and Ant Colony Optimization (ACO) which is inspired by the behavior of ants are examples of nature-inspired methods.

Simulated annealing: a probabilistic optimization method inspired by the process of annealing in metallurgy. It involves a random search through the solution space that allows for occasional "uphill" moves to avoid getting stuck in local minima [10].

Tabu search: is a metaheuristic optimization method that involves a search through the solution space while avoiding previously visited solutions by using a 'tabu list' to keep track of previous solutions and prevent revisiting them [11].

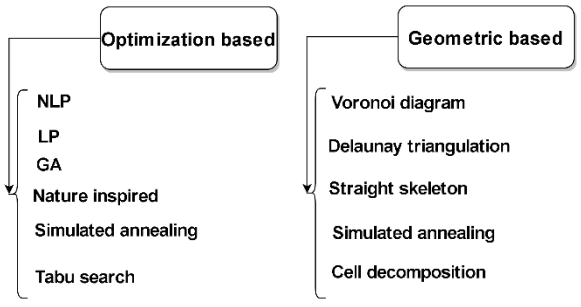

Figure 2 Subcategories of Optimization and Geometric based methods.

B. *Geometric based methods*

A subcategory of global path planning in which the environment is represented as a geometric space consisting of obstacles and free areas. It emphasizes the use of geometric concepts and algorithms to facilitate effective and safe autonomous navigation [12]. The following methods are considered subcategories of geometric based methods.

Voronoi diagram based: Generate a graph of the environment by constructing a Voronoi diagram based on the distance to obstacle positions. The graph consists of nodes representing the obstacles and edges connecting adjacent regions. The method finds a path between the start and goal positions while avoiding obstacles by searching this graph[13].

Visibility graph-based: the visibility graph-based method involves constructing a graph that represents the visibility relationships between the obstacles in the environment, and then using graph algorithms to find the shortest path between two points in the environment [14].

The Delaunay triangulation-based: A geometric path planning approach that uses a triangulation of the environment's free space. Each point in the triangulation is equidistant to its three nearest neighbor points, and the edges of the triangles form a network of potential paths. A pathfinding algorithm is then used to find the shortest path.

Medial axis-based: A method that generates a skeleton that represents the centers of the largest circles that can be inscribed within the obstacles in the environment. This skeleton can also be used as a basis for path planning algorithms to find the shortest path while avoiding obstacles.

Cell decomposition-based: A method for path planning called cell decomposition divides the environment into simpler cells, which can be connected to form a path. The cells can be of any shape but squares or rectangles are commonly used [15].

C. *Grid based methods*

Grid-based are a type of UAV path planning that divide the planning space into a grid and create a graph to find the shortest or most optimal path for the UAV to follow while avoiding obstacles and meeting objectives. They are commonly used because of their computational efficiency and ability to handle complex environments. The following sections will provide more information on these methods Fig 3.

Dijkstra's algorithm: used to solve the shortest path problem of a graph with non-negative edge costs, it gives the shortest path tree. For a given vertex of the graph, the algorithm starts by finding the costs of the shortest path between a source vertex and a destination vertex. Once the shortest path to the destination vertex has been found, the algorithm stops [16].

A* algorithm: An algorithm finds the shortest path between two nodes in a graph by assigning tentative distance to each node and calculating a heuristic value based on the estimated distance to the goal node. It then selects the node with the lowest total distance and explores its neighbors until the goal node is reached or all possible paths have been explored. A* algorithm is faster than Dijkstra's algorithm because it uses a heuristic function. (1), where f(n) is the estimated path to reach the goal node throw the

node n, g(n) is the cost to reach node n from the starting point, h(n) is the heuristic cost to reach goal node from node n. However, the quality of the heuristic function greatly influences the algorithm's performance. The A* algorithm is guaranteed to find the shortest path if the heuristic function is admissible and consistent. In [17], the authors utilized the A* algorithm in conjunction with GA (Genetic Algorithm) to find the minimum path and for obstacle avoidance. They also utilized ray tracing for calculating coverage.

$$f(n) = g(n) + h(n) \qquad (1)$$

Dynamic A* (D* lite): A real-time planning algorithm that updates the path as the environment changes. It uses a graph search approach and maintains a local map of the environment to update the cost of the path. The algorithm is efficient and can handle real-time planning applications.

Iterative Deepening A* (IDA): A search algorithm that uses depth-first search memory efficiency and A* optimality. It gradually increases the depth limit until it finds the goal node, using a heuristic function to estimate the cost at each depth limit. The algorithm terminates when the goal node is found, or the depth limit exceeds a pre-determined threshold.

Breadth-First Search (BFS): Graph traversal algorithm used to explore all the vertices or nodes of a graph in breadth-first order, i.e., visiting all the nodes at a given depth before moving on to the nodes at the next depth. BFS can be used to find the shortest path between two nodes in an unweighted graph.

Depth-First Search (DFS): An algorithm used to traverse and search through a graph or tree data structure. It starts at a given node and explores as far as possible along each branch before backtracking. The algorithm does this recursively until all nodes have been visited, marking each visited node as it progresses. DFS uses a stack data structure to keep track of the nodes to be visited and is commonly implemented using recursion.

D. Differential games based methods:

A mathematical framework for analyzing strategic interactions between multiple decision-makers over time. It models the dynamics of the system as a set of differential equations and studies the optimal strategies of each player under various conditions. The objective is to find a solution that maximizes each player's gain while taking into account the actions of the other players. In the following sections, we discuss methods that are considered as Differential games methods Fig 3.

Homicidal chauffeur (HC): A car with small turning radius with target to catch a pedestrian who tries to escape. Of course, the car can move faster than the pedestrian, but the pedestrian has better maneuverability. The main idea is what is the ideal strategy that each player can use to increase their own score? The car must choose a strategy that will ensure the capture of the pedestrian under different circumstances and the Pedestrian use a strategy to escape and in case that he can't escape he will try to maximize the capture time on the other hand the car want to minimize this time. This simple game considered as a classical 1-persuer-1- evader problem used in many military applications so for example we can assume that the car is anti-aircraft missile, and the pedestrian is the enemy plane.

Target interception (TI): A method used to calculate the future position of a moving object and direct a UAV to intercept it at that position. This is achieved by utilizing the target's heading angle and speed to determine its future location.

Two cars Two cars with the same turn radius limitation engaged in a pursuit evasion game, each car want to catch the other, according to Isaacs the two cars Differential game it is just like the HC except the evader too suffers the constraint of bounded curvature [18]

Lady in the lake: A man runs around a circle and a woman moves from the circle center to the boundary of the circle. When the woman reaches the boundary, the distance between the man and the woman is measured by the angle between the segments connecting their position and the circle center. The man tries to minimize this angle, but the woman tries to maximize it. The objective of the game is to design policies for both players[19]

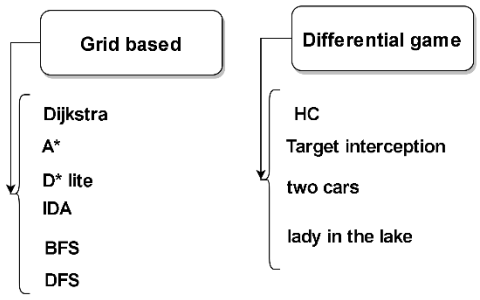

Figure 3 Subcategories of Grid and Differential game based methods

III. IMPLEMENTATION

The choice of path planning techniques depends on the required task [2], so to cover most known scenario that can face a UAV during the path planning, we make two implementations. The first one deal with static environment and fixed destination using two-grid methods (Dijkstra and A*), both methods use offline calculation to provide an optimal path, so we can compare their results. The second implementation deal with dynamic environment with moving destination using two Differential games methods (HC and TI), both methods can be used in a dynamic environment and apply to the scenario where a drone is trying to catch a moving landing station.

G*rid-based Implementation:*

In the Grid-based implementation, we use python scripts, and 2D map in presence of static obstacles. The used map was divided into 900 cells, with 30 columns and 30 rows. The aim of this implementation is to monitor the limitation of these two methods by comparing their performance in terms of number of searched nodes and the path length. We change the location of obstacles and the cartesian location of the destination.

We use the Euclidean distance between the current node and the destination node (goal node) as the heuristic function. In both Dijkstra and A* methods, the UAV will choose the offline path before starting the journey due to the static destination and the absence of moving obstacles or any other influencing factors on the UAV.

In the first implementation, we use an obstacles-free map, and we use a destination node with Cartesian location (12, 12)

(see Fig 4) in order to compare the number of searched nodes between the two methods. The results show that both methods generate the same path which is the optimal path. However, there is a huge difference between the numbers of the searched nodes between the two methods. With A*, only 12 nodes were searched, while 237 nodes were searched by Dijkstra to get the same path as A*. This difference refers to that A* use heuristic function to estimates the distance from each explored node to the goal node while Dijkstra explores all the neighbors of each node.

In the second implementation of Fig 5, we add obstacles to the map and change the destination Cartesian location to (14, 12). The results show that the length of the path is the same in the two methods (18 nodes). However, the path itself is different and the number of searched nodes in Dijkstra equal to 215 and 204 in A*. The difference between the number of nodes searched is significantly lower than the first implementation. These results indicate that the heuristic function used is not ideal for this map.

In the third implementation of Fig 6 we change the destination Cartesian location to (14, 18). The A* failed to find a path to the destination due to the inefficiency of using Euclidean distance as a heuristic function in a complex environment., while Dijkstra was able to find the path in 50 steps and the number of searched nodes is 593 nodes.

The three implementations show that the effect of the heuristic function on path generation is direct, and that the heuristic function does not always generate the optimal path.

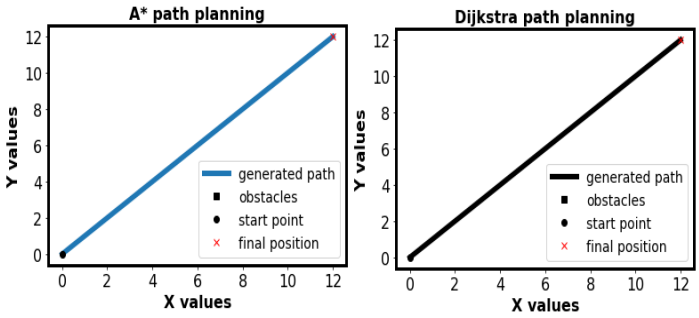

Figure 4 A* and Dijkstra Path Planning without obstacles

A. *Differential gameImplementation:*

The second implementation focuses on two differential game-based methods: HC and TI. This implementation is designed to address the challenge of docking a drone onto a mobile station in a 2D dynamic environment with wind effects and no obstacles. We consider this use case as a cooperative scenario, meaning that neither the landing station nor the drone would take any premeditated actions to prevent the goal from being reached.

For this implementation, we used a Python script to generate paths based on differential game algorithms (HC, TI). We also used JSBSIM FDM [4], to simulate the wind effects on a real airplane. We select a Cessna c172 airplane for the simulation, with an altitude hold autopilot set to 4000 feet (1219 meters) to simplify the implementation to a 2D environment. To simulate the wind effects, we added a wind speed of 23 feet per second, which is considered above average at 4000 feet. During the implementation, we varied the heading angle and starting location of the landing station (with the UAV starting coordinates used as a reference point at (0, 0), and we changed the wind direction to cover different possibilities (east, west), allowing for variations in the aircraft's path. In all implementations, we maintain a constant landing speed of 15 m/s for the landing station and 51 m/s for the aircraft. We considered the plane to have caught the landing station when the remaining time to catch it became less than 1 second and the distance was less than 51 meters.

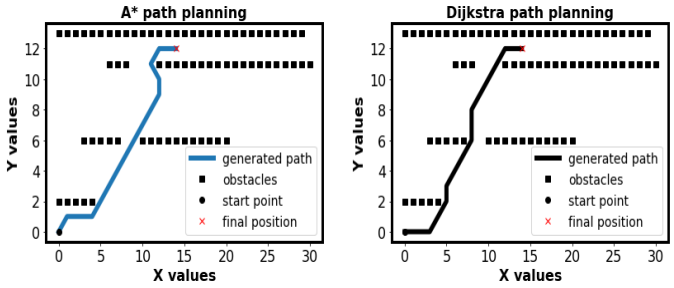

Figure 5 A* and Dijkstra Path Planning in presence of obstacles and catching location (14,12)

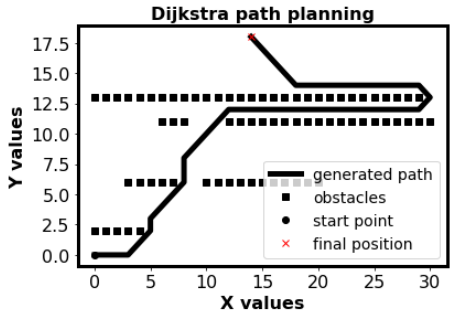

Figure 6 Dijkstra Path Planning in presence of obstacles and destination coordination (14,18)

To implement HC, we assume that the UAV would act as the pursuer and the landing station would act as the evader, both modeled as points moving under the following assumptions. The UAV moves at a constant speed and can change its heading angle by instantaneously selecting the radius of curvature of its trajectory above a given threshold. The landing station moves at a constant speed and with a constant heading angle. The capture occurs when the distance between the pursuer and the evader reaches a given threshold. It is typically assumed that the pursuer has an advantage in speed but a disadvantage in maneuverability. Fig 7 shows a dynamic model of the movements of the pursuer (UAV) and evader (landing station). Let (x1, y1) be the Cartesian coordinates of the UAV with respect to a fixed frame O, X, Y. Let W1 denote the speed of the landing station and θ denote its heading angle, measured clockwise from the Y axis. Also, let R denote the minimum radius of curvature of the UAV's trajectory and φ denote the ratio of the minimum radius of curvature to the actual radius of curvature. Then the motion of the UAV is modeled by (2), (3) and (4)

Table 1 lists specification of some grid and differential games-based methods.

Algorithm	Online/offline Calculation	Connection between UAV and destination	Optimal solution	Dynamic/static environment	limitation
Dijkstra	**Offline**	**Yes**	**Yes**	**Static**	**cannot handle negative edge weights[20]**
A*	**Offline**	**Yes**	**Depend On Heuristic Function**	**Static**	**Heuristic function is the key for finding the optimal solution. [24]**
D*	**Online**	**Yes**	**No**	**Dynamic**	**require a significant amount of computation time and resources to update the cost-to-come values for each cell in the grid[21]**
IDA	**Offline**	**Yes**	**Yes**	**Static**	**Can be slow if branching factor is high or the heuristic function is not informative[22].**
BFS	**Offline**	**No**	**Yes**	**Static**	**Very memory-intensive if the branching factor is high or the search space is large [23]**
DFS	**Offline**	**No**	**No**	**Static**	**Possible to get stuck in an infinite loop [23]**
HC	**Online**	**Yes**	**No**	**Dynamic**	**Doesn't take into consideration the environment effects (wind,rain,…) [18]**
TI	**Online**	**Yes**	**Depend On Environment**	**Dynamic**	**The environment plays a major role in the success or failure of the method [12].**

$$\dot{x}_1 = W_1\, Sin\theta \qquad \dot{x}_2 = W_2\, Sin\theta \qquad (2)$$

$$\dot{y}_1 = W_1\, Cos\theta \qquad \dot{y}_2 = W_2\, Cos\,\theta \qquad (3)$$

$$\dot{\theta} = \frac{W_1}{R}\emptyset \qquad |\emptyset| \leq 1 \qquad (4)$$

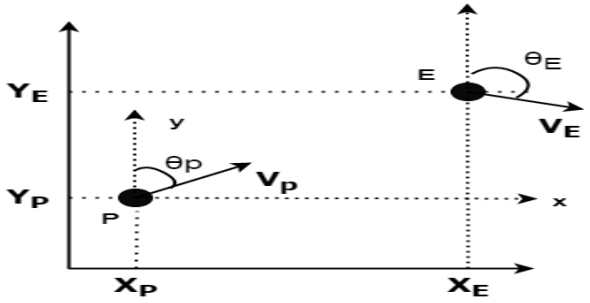

Figure 7 Dynamic model of the motions of HC

In [18], Isaacs provides two examples of the simplest pursuit game involving two players, P and E, with speeds W and w, respectively, where W>w. The players move in the plane, each with simple motion, along a straight line connecting their initial positions, with E fleeing and P pursuing. The payoff is the time of capture, and in Isaacs' example, the question was why P cannot extrapolate E's future position and calculate a collision point, enabling P to go straight to it and reduce the number of steps required to catch E in from 15.5 steps to 4.3 steps. Isaacs points out that P's prediction has no basis since the game allows E to change direction abruptly. If E remains unaware of P until two-time units have elapsed, E will belatedly assume their optimal strategy and flee directly away from P. In our case, the landing station and the drone will not take premeditated actions, so we use the simple method Target Interception (TI). In order detect any external influences on the drone, we compare the actual location with the assumed location by using a constant speed and heading to ensure the drone is on the right trajectory.

1) Change landing station heading angle

First implementation we use the same starting location for the landing station (X=17000, Y=17000) which is 24042 meters away from the UAV and we change the heading angle $(\frac{\pi}{3}, \frac{\pi}{4}, \frac{\pi}{5}, \frac{\pi}{6})$.We repeat the same experiment but in the second time we change the direction of wind from west to east.

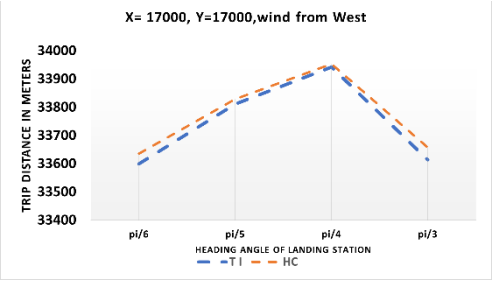

Figure 8 The effect of the heading angle of the landing station on the total travel distance with a wind direction from the west

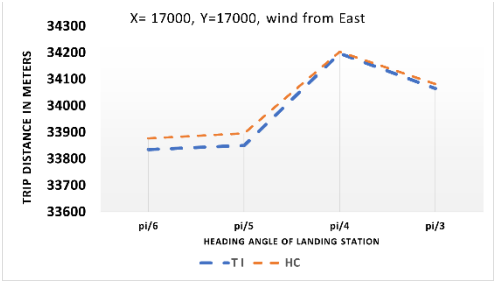

Figure 9 The effect of the heading angle of the landing station on the total travel distance with a wind direction from the east

The implementation results show that TI was the best method with the shortest distances to reach the landing station over the four trials, however the HC method results was very close when the heading angle equal to $\frac{\pi}{4}$. In addition to the

heading angle of the landing station, the wind direction influences the travel distance Fig 8 Fig 9

2) Modify the initial distance:

During the second implementation we use the same heading angle in all the tests, but we change the coordination of landing station at t0 randomly and incrementally, which leads to changing in the initial distance d0 between landing station and the UAV, it is therefore normal to see that the traveled distance (TD) at evolves incrementally. Table 2. The results show that the drone using TI reaches the mobile landing station by a shorter path.

TABLE 2 UAV TRAVELED DISTANCE WHILE CHANGING THE LANDING STATION START POSITION.

	Method	**Distance at the initial state t0, in meters**			
		32311	***35511***	***37947***	***56921***
Traveled distance in m	***HC***	44702	47666	52076	78188
	TI	44632	47608	51990	78143

3) Results

The results showed that the TI approach performed better than the HC method. However, in some cases, there were only slight differences between the results of the two methods, and the differences observed were not statistically significant. Despite this, the overall trend indicates that the TI approach is more effective than the HC approach.

IV. CONCLUSION

In this paper, we presented the global path planning of UAVs classification categories and its subcategories, while focusing on their advantages and limitations and the environment in which they can be used.

We implemented four methods, two of them considered as UAV grid based methods whish generate a path offline in static environment in presence of obstacles, the other two methods considered as Differential game methods whish deal a dynamic task which is docking of a UAV on a mobile landing station with the presence of wind as an external influence factor.

As a future work, we plan to extend our implementation to include a wider range of trajectory planning methods that address different environmental challenges. Our goal is to develop a comprehensive dataset of trajectory planning solutions that can be used to train a supervised learning model for UAV trajectory planning using machine learning techniques.

V. REFERENCES

[1] A. Heidari, N. Jafari Navimipour, M. Unal, and G. Zhang, "Machine Learning Applications in Internet-of-Drones: Systematic Review, Recent Deployments, and Open Issues," ACM Comput. Surv., vol. 55, no. 12, pp. 1–45, Dec. 2023, doi: 10.1145/3571728.

[2] Y. Yang, X. Xiong, and Y. Yan, "UAV Formation Trajectory Planning Algorithms: A Review," Drones, vol. 7, no. 1, p. 62, Jan. 2023, doi: 10.3390/drones7010062.

[3] Z. Xu, D. Deng, Y. Dong, and K. Shimada, "DPMPC-Planner: A real-time UAV trajectory planning framework for complex static environments with dynamic obstacles," in 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 250–256. doi: 10.1109/ICRA46639.2022.9811886.

[4] T. Vogeltanz, "A Survey of Free Software for the Design, Analysis, Modelling, and Simulation of an Unmanned Aerial Vehicle," Arch. Comput. Methods Eng., vol. 23, no. 3, pp. 449–514, Sep. 2016, doi: 10.1007/s11831-015-9147-y.

[5] B. Pérez, R. Mínguez, and R. Guanche, "Offshore wind farm layout optimization using mathematical programming techniques," Renew. Energy, vol. 53, pp. 389–399, May 2013, doi: 10.1016/j.renene.2012.12.007.

[6] J. Nocedal and S. J. Wright, Numerical optimization, 2nd ed. New York: Springer, 2006.

[7] F. Borrelli, D. Subramanian, A. U. Raghunathan, and L. T. Biegler, "MILP and NLP Techniques for centralized trajectory planning of multiple unmanned air vehicles," in 2006 American Control Conference, Minneapolis, MN, USA, 2006, p. 6 pp. doi: 10.1109/ACC.2006.1657644.

[8] Y. Chen, J. Han, and X. Zhao, "Three-dimensional path planning for unmanned aerial vehicle based on linear programming," Robotica, vol. 30, no. 5, pp. 773–781, Sep. 2012, doi: 10.1017/S0263574711000993.

[9] R. Shivgan and Z. Dong, "Energy-Efficient Drone Coverage Path Planning using Genetic Algorithm," in 2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR), Newark, NJ, USA, May 2020, pp. 1–6. doi: 10.1109/HPSR48589.2020.9098989.

[10] J. L. Foo, J. Knutzon, J. Oliver, and E. Winer, "Three-Dimensional Path Planning of Unmanned Aerial Vehicles Using Particle Swarm Optimization," in 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, Virginia, Sep. 2006. doi: 10.2514/6.2006-6995.

[11] B. Tong, J. Wang, X. Wang, F. Zhou, X. Mao, and W. Zheng, "Optimal Route Planning for Truck–Drone Delivery Using Variable Neighborhood Tabu Search Algorithm," Appl. Sci., vol. 12, no. 1, p. 529, Jan. 2022, doi: 10.3390/app12010529.

[12] H. Choset, Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, 2005.

[13] H. Tong, W. W. chao, H. C. qiang, and X. Y. bo, "Path Planning of UAV Based on Voronoi Diagram and DPSO," Procedia Eng., vol. 29, pp. 4198–4203, 2012, doi: 10.1016/j.proeng.2012.01.643.

[14] A. Majeed and S. Lee, "A Fast Global Flight Path Planning Algorithm Based on Space Circumscription and Sparse Visibility Graph for Unmanned Aerial Vehicle," Electronics, vol. 7, no. 12, p. 375, Dec. 2018, doi: 10.3390/electronics7120375.

[15] S. K. Debnath et al., "Different Cell Decomposition Path Planning Methods for Unmanned Air Vehicles-A Review," in Proceedings of the 11th National Technical Seminar on Unmanned System Technology 2019, vol. 666, Z. Md Zain, H. Ahmad, D. Pebrianti, M. Mustafa, N. R. H. Abdullah, R. Samad, and M. Mat Noh, Eds. Singapore: Springer Nature Singapore, 2021, pp. 99–111. doi: 10.1007/978-981-15-5281-6_8.

[16] E. J. Dhulkefl, A. Durdu, and Electrical and Electronic Engineering, "Path Planning Algorithms for Unmanned Aerial Vehicles," Int. J. Trend Sci. Res. Dev., vol. Volume-3, no. Issue-4, pp. 359–362, Jun. 2019, doi: 10.31142/ijtsrd23696.

[17] N. Bolourian and A. Hammad, "LiDAR-equipped UAV path planning considering potential locations of defects for bridge inspection," Autom. Constr., vol. 117, p. 103250, Sep. 2020, doi: 10.1016/j.autcon.2020.103250.

[18] R. Isaacs, "Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit," Jan. 1965.

[19] P. Cheng, "A short survey on pursuit-evasion games." 2003.

[20] N. Makariye, "Towards shortest path computation using Dijkstra algorithm," in 2017 International Conference on IoT and Application (ICIOT), Nagapattinam, India, May 2017, pp. 1–3. doi: 10.1109/ICIOTA.2017.8073641.

[21] M. C. Reeves, "Master of Science in Electrical Engineering".

[22] R. E. Korf, "Depth-first iterative-deepening," Artif. Intell., vol. 27, no. 1, pp. 97–109, Sep. 1985, doi: 10.1016/0004-3702(85)90084-0.

[23] R. E. Korf, "Linear-space best-first search," Artif. Intell., vol. 62, no. 1, pp. 41–78, Jul. 1993, doi: 10.1016/0004-3702(93)90045-D.

[24] U. K. G. Z. Aditya Chatterjee, "Publication, Open-Source, Computer Science,," 2017. [Online]. Available: iq.opengenus.org. [Accessed September 2017].