Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 177 (2020) 57–63

Procedia Computer Science

www.elsevier.com/locate/procedia

The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2020)
November 2-5, 2020, Madeira, Portugal

Path Planning Algorithm for Unmanned Ground Vehicles (UGVs) in Known Static Environments

Eman Almoaili[a,*], Heba Kurdi[a,b]

[a]*Computer Science Department, King Saud University, Riyadh, Saudi Arabia*
[b]*Mechanical Engineering Department, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA*

Abstract

Unmanned ground vehicles (UGVs) have been utilized in many civilian fields in addition to the traditional military field. This is due to their increasing capabilities in terms of performance, power, and tackling risky missions. Many applications require the UGV to autonomously navigate static environments while taking into consideration obstacle avoidance. Autonomous path planning is one of the key challenges and issues related to UGVs. Generally, robotic path planning is an optimization search problem that comes in different forms. Some of its forms have been solved by different classical algorithms such as A*, but these algorithms are computationally inefficient. In contrast, the emerging nature-inspired algorithms outperform the classical ones since the computational overhead is reduced. Nature-inspired algorithms are among the most common heuristic algorithms. This paper proposes a near-optimal algorithm to find a feasible path for UGV in a static environment. The performance of the proposed algorithm was compared with other well-established algorithms in path planning literature such as A* using a simulator developed for this purpose. The simulator tests three performance measures path length, farness from obstacles, and running time. The simulator results revealed that the length of the path generated by this algorithm is near-optimal, however, the generated path is kept far from obstacles.

© 2020 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the Conference Program Chairs.

Keywords: Nature-Inspired algorithms; Unmanned ground vehicles; UGV; path planning; offline path planning

* Corresponding author.
E-mail address: ealmoaili@ksu.edu.sa

1877-0509 © 2020 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the Conference Program Chairs.
10.1016/j.procs.2020.10.011

1. Introduction

Autonomous unmanned ground vehicles (UGVs) are capable of functioning independently without human intervention [1]. The first generations of UGVs were motivated by military applications. In the past few years, the applications of autonomous UGVs have grown dramatically and have shown considerable diversity. In addition to military applications, many civil applications have emerged, such as examination of power lines, surveillance systems, detection of mines, data collection, imaging, security, agriculture, deep exploration in oceans, and scientific exploration in space, just to name a few [1][2]. Generally, UGVs are able to carry out tasks that are dangerous, distant, costly, or difficult for humans to execute [1][3].

These applications demand the UGV to navigate autonomously and avoid different obstacles at the same time. Due to this, path planning is considered one of the key significant challenges for UGVs. Path planning is used to find a feasible path for the UGV during its navigation from the starting location to the target location while avoiding obstacles and optimizing some predefined criteria. These criteria or performance measures, such as minimization of time, distance, and controlling effort, are used to identify the best plan [3][4]. Therefore, a path planning algorithm usually suggests addressing some performance measures such as the path length, the running time, or the number of explored nodes.

Recent decades have witnessed dramatic advancements in artificial intelligence and robotics. As a result, UGVs are an active and growing research area [1][5]. This has led to developments in path planning algorithms. Many algorithms have been proposed to address the issues of UGV path planning. Based on its solution method, path planning algorithms can be classified as either exact (deterministic) methods or approximate methods. Exact methods are enumerative and follow systematic steps to find the solution [6][7]. The majority of classical/conventional algorithms fall in this category [4][8]. Approximate methods can be divided into two categories, approximation algorithms, and heuristic algorithms. Approximation algorithms produce "provable solution quality and provable run-time bounds [7]." Heuristic algorithms provide good solutions for large-instance problems in a reasonable amount of time. Therefore, accepted performance is obtained within an acceptable cost. Heuristic algorithms are further classified into meta-heuristics and problem-specific heuristics [6][7]. Meta-heuristics are applicable to a wide range of optimization problems without any adaptation to the specific problem [6][7], such as ant colony optimization (ACO) and simulated annealing (SA). Problem-specific heuristics are designed to find a solution to a specific problem [6][7].

Nature-inspired algorithms, which are considered meta-heuristic, belong to a class of algorithms that take advantage of the perfection of natural systems. Thus, nature-inspired algorithms are capable of efficiently solving very complex problems. They can be categorized into physics-inspired, biology-inspired, and chemistry-inspired algorithms [6][9].

This paper proposes a near-optimal path planning algorithm for a single UGV taking into consideration obstacle avoidance in static environments. The proposed algorithm uses two heuristic values; therefore, it is named H2A (Two-Heuristic Algorithm). The performance of the proposed algorithm is compared with the A* algorithm's performance, which is the most well-known algorithm used in static environments.

The rest of this paper is structured as follows. Section 2 presents and describes the H2A algorithm. The evaluation methodology is illustrated in section 3. Section 4 discusses the results of the comparison between the H2A algorithm and the A* algorithm. The last section concludes the paper and suggests guidelines for future work.

2. The Proposed Algorithm (H2A)

Before describing the design of H2A, it is important to make some assumptions. The UGV navigates in a two-dimensional environment that is represented by a two-dimensional grid. The UGV occupies one grid cell, and it can move in any direction in the environment. That is, the eight-direction movement is possible. The obstacle size is equal to one grid cell, and it is stationary in the static environment.

Since the environment is static, the H2A is executed offline. Thus, the path is calculated offline before the UGV navigates. The algorithm tries to optimize two performance measures simultaneously, the path length and the farness from obstacles. One of the contributions of this algorithm is that it generates two cells toward the solution in each iteration. This can minimize the running time of the algorithm.

The H2A suggests two heuristic values, H1 and H2, to guide the search. Starting from the start position as the current position, the algorithm finds the best combination of H1 and H2 for the current cell based on an objective function.

That is, the best H1 and H2 give the minimum value of $f(i)$ among all possible values for H1 and H2 in the current iteration step. As a result, H1 and H2 constitute two movements calculated in one iteration step. The next iteration step takes H2 as the current position. Then the same procedure of finding the best combination of H1 and H2 based on the objective function is applied. The algorithm stops when either reaching the goal position or when the complete path cannot be found. Table 1 summarizes the general process of the proposed algorithm. The flowchart of the algorithm is presented in Figure 1.

Table 1. H2A process.

The Input	1. The start position of the UGV. 2. The goal position. 3. The obstacles positions.
The output of Each Iteration Step	H1 and H2: The next two feasible cells toward the goal position.
Complete Output	The feasible path from the start to the goal positions.

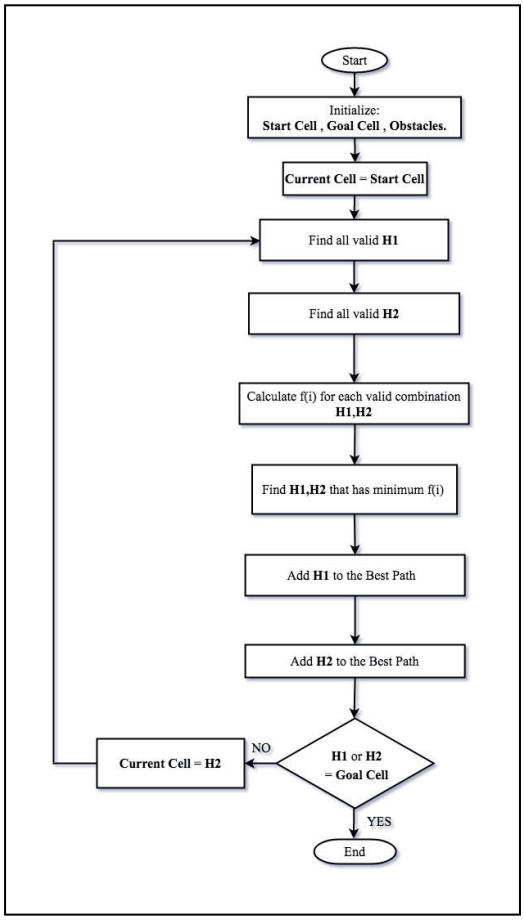

Fig. 1. H2A flowchart.

3. Evaluation Methodology

To test the performance of the H2A, a systematic simulator has been implemented in C++ to randomize the start position, goal position, and obstacle positions using environments of size 16×16, 64×64, and 256×256 with obstacle ratios 20%, 40%, and 60%. In addition, a GUI simulator has been implemented using C++ and wxWidgets library, which provides GUI components that can be integrated within the C++ code. The GUI simulator is useful to illustrate the farness from obstacles. The code was implemented under Linux OS. The work in [10] has been extended and built upon when designing these simulators.

Figure 2 shows the interface of the simulator. The environment is modeled as a two-dimensional workspace with three possible dimensions 8×8, 16×16, and 32×32. The start position is represented as a red square, the goal position is represented as a green square, and the obstacles are represented as black squares. When an algorithm generates the path, it is shown in blue squares.

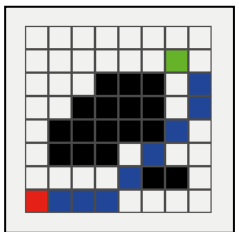

Fig. 2. GUI simulator.

As crucial performance measures for path planning algorithms basically concern with the path length, running time, and obstacle avoidance, this paper introduces a systematic evaluation framework to test the H2A based on these measures. The independent variables are the environment size, the obstacles ratio, start position, goal position, and obstacles positions. The hypothesis is that the H2A will generate near-optimal path length, decrease the running time, and keep the generated path far from obstacles. To test this hypothesis, both simulators were used within the following evaluation framework:

1. In the systematic simulator, the environment size was fixed at 16×16, and the obstacle ratio was varied with obstacle ratio of 20%, 40%, and 60%.
2. The simulator randomly generated three scenarios of start, goal, and obstacles positions.
3. Based on these settings, A* and H2A were run, and the resulted path lengths and running times were saved in two Excel files, one for A* results and the other for H2A results.
4. The same steps 1-3 were repeated for the environment size 64×64 and 256×256. However, for the environment size 256×256, the obstacle ratio used was 10%, 20%, and 30%.

Table 2 presents the path lengths for both A* and H2A in the different settings described above, while the running time for both algorithms for the same settings is presented in Table 3.

Table 2. Path length for A* and H2A in different scenarios (S1, S2, S3), obstacles ratio, and environments size.

		16*16 Path Length					64*64 Path Length					256* 256 Path Length		
		S1 S=222 G=131	S2 S=63 G=2	S3 S=229 G=96			S1 S=517 G=95	S2 S=3058 G=1913	S3 S=1259 G=3246			S1 S=45601 G=13982	S2 S=38615 G=39608	S3 S=43714 G=19458
20%	A*	12	14	9	20%	A*	28	20	32	10%	A*	132	32	193
	H2A	14	14	9		H2A	30	20	34		H2A	183	32	227
40%	A*	12	14	9	40%	A*	27	19	32	20%	A*	130	32	193
	H2A	12	14	23		H2A	29	22	37		H2A	180	37	225
60%	A*	12	14	9	60%	A*	27	19	32	30%	A*	131	32	193
	H2A	12	14	11		H2A	27	19	33		H2A	171	35	223

S1: Scenario1, S2: Scenario2, S3: Scenario3, S: Start position, G: Goal position.

Table 3. Running time for A* and H2A in different scenarios (S1, S2, S3), obstacles ratio, and environments size.

		16*16 Run Time(ms)					64*64 Run Time(ms)					256* 256 Run Time(ms)		
		S1 S=222 G=131	S2 S=63 G=2	S3 S=229 G=96			S1 S=517 G=95	S2 S=3058 G=1913	S3 S=1259 G=3246			S1 S=45601 G=13982	S2 S=38615 G=39608	S3 S=4371 4 G=194 58
20%	A*	2.462	2.622	0.932	20%	A*	9.325	1.956	4.476	10%	A*	9.325	1.956	4.476
	H2A	0.441	0.324	0.266		H2A	1.298	0.907	1.908		H2A	1.298	0.907	1.908
40%	A*	0.547	0.707	0.223	40%	A*	2.938	1.444	3.979	20%	A*	2.938	1.444	3.979
	H2A	0.494	0.346	1.037		H2A	0.895	1.175	2.458		H2A	0.895	1.175	2.458
60%	A*	0.522	0.518	0.311	60%	A*	3.553	1.822	4.647	30%	A*	3.553	1.822	4.647
	H2A	0.350	0.414	0.253		H2A	1.425	0.926	2.084		H2A	1.425	0.926	2.084

S1: Scenario1, S2: Scenario2, S3: Scenario3, S: Start position, G: Goal position.

4. Results and Discussion

In the following, the performance of H2A is compared with A* based on path length, running time, and farness from obstacles as key performance measures.

4.1. Path Length

Figure 3 illustrates the path length of both A* and H2A in the 16×16 environment, while Figures 4 and 5 show the algorithms' path length in the 64×64 and 256×256 environments, respectively. In the 16×16 environment, the majority of generated paths by both algorithms have the same length, as shown in Figure 3. In Figure 4, in which the environment is 64×64, the majority of paths generated by H2A are slightly longer than the paths generated by A*, and in few cases, both algorithms generated equal path length. However, in Figure 5, H2A generates quite longer path length compared to A* as the environment increases to 256×256. These figures reveal that H2A does not always generate the shortest path; however, it can generate near-optimal path length.

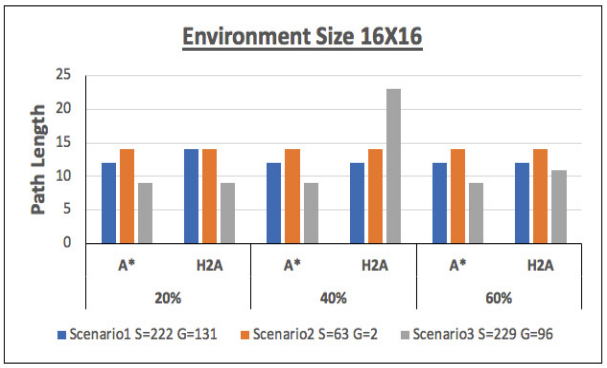

Fig. 3. Path length in 16×16 environment for different settings.

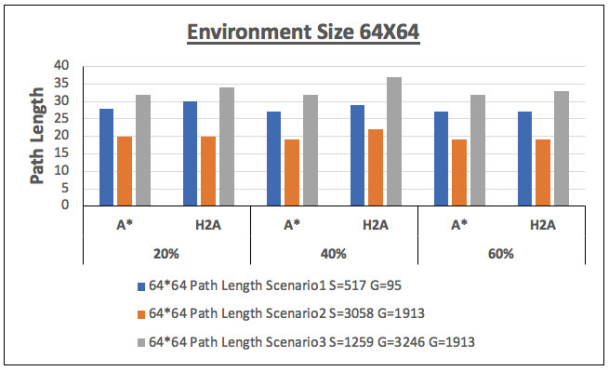

Fig. 4. Path length in 64×64 environment for different settings.

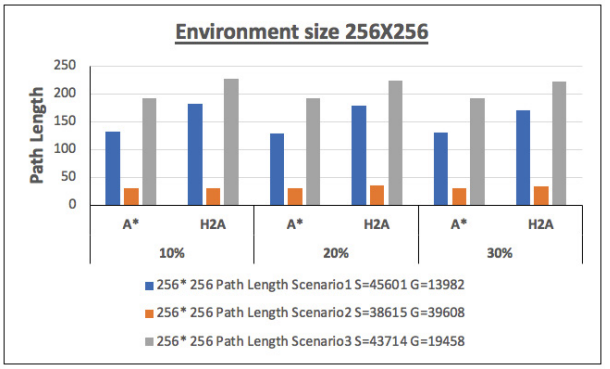

Fig. 5. Path Length in 256×256 environment for different settings.

4.2. Running Time

Figure 6 shows the running time in milliseconds in 16×16 environment for A* and H2A, whereas their running time in 64×64 and 256×256 environments are shown in Figures 7 and 8. In Figure 6, as an overall trend, the running time of H2A in the 16×16 environment is less than A* running time. As the environment size grows in Figure 7 and Figure 8, 64×64, and 256×256, respectively, H2A exhibits considerably faster running time than A*. That is, compared to A* running time, H2A reduces the computational overhead, especially when the environment size increases.

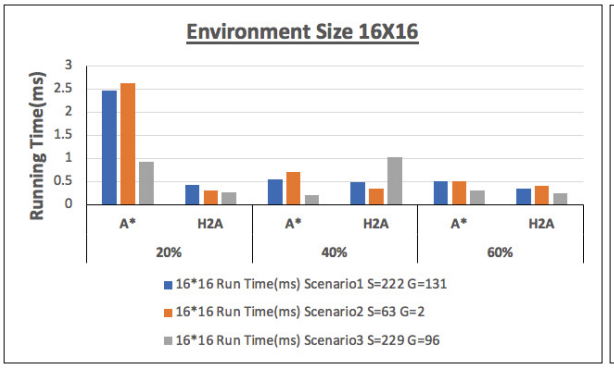

Fig. 6. Running time in 16×16 environment for different settings.

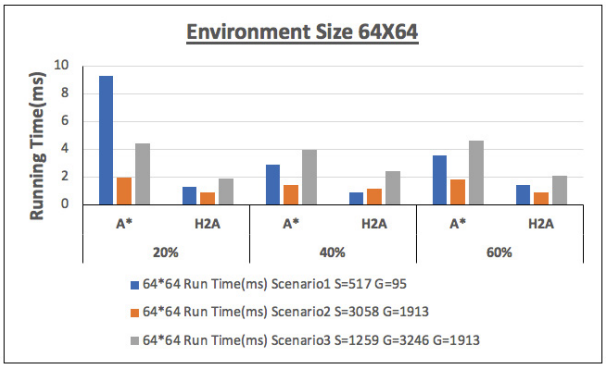

Fig. 7. Running time in 64×64 environment for different settings.

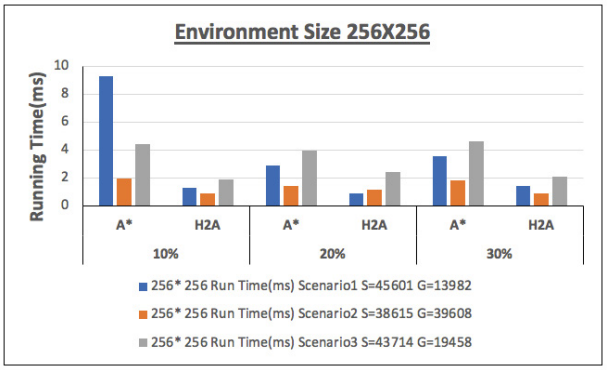

Fig. 8. Running time in 256×256 environment for different settings.

4.3. Farness from Obstacles

Figure 9 shows how H2A is able to generate a path that is far from obstacles. In this Figure, the left grid shows the path generated by A*, while the right grid presents H2A's path. Both paths have the same length; however, H2A's path is farther from obstacles.

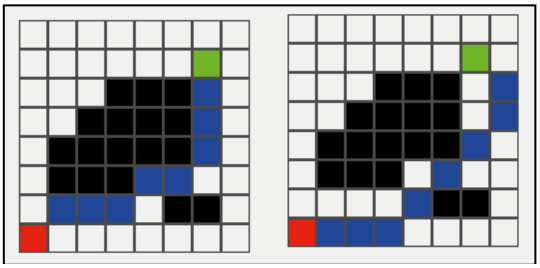

Fig. 9. A* and H2A paths' farness from obstacles.

5. Conclusion

Autonomous navigation of UGVs is increasingly required for many applications in recent years. Therefore, autonomous path planning is a significant challenge in the field of autonomous UGVs. The classical algorithms of path planning, such as A*, involve computational overhead that makes them practically efficient. In contrast, heuristic algorithms are more efficient and have been growingly utilized in path planning. This paper proposed a heuristic path planning algorithm (H2A) for UGVs in static environments. Even though H2A did not always generate the shortest path, it generates the shortest farthest path from obstacles. The performance of H2A was compared with the performance of the A* algorithm. The results showed that H2A does not always generate the shortest path; however, it is able to find the near-optimal path. Most importantly, the results prove that H2A is faster than A* and can generate a path that is farthest from obstacles.

As future work, H2A will be enhanced to generate the shortest path. The running time will be improved as well. Another performance measure will be considered, which is the number of explored nodes in the environment. Moreover, the size of the environment will be enlarged. Most importantly, dynamic environments and moving obstacles will be considered in the future. Integration with ROS will also be a significant aim so that H2A will become practically efficient.

References

[1] E. Yagdereli, C. Gemci, and A. Z. Aktas, “A study on cyber-security of autonomous and unmanned vehicles,” *J. Def. Model. Simul.-Appl. Methodol. Technol.-JDMS*, vol. 12, no. 4, pp. 369–381, Oct. 2015, doi: 10.1177/1548512915575803.

[2] E. Erdem, D. G. Kisa, U. Öztok, and P. Schüller, “A General Formal Framework for Pathfinding Problems with Multiple Agents,” 2013.

[3] M. Murillo, G. Sanchez, L. Genzelis, and L. Giovanini, “A Real-Time Path-Planning Algorithm based on Receding Horizon Techniques,” J. Intell. Robot. Syst., vol. 91, no. 3–4, pp. 445–457, Sep. 2018, doi: 10.1007/s10846-017-0740-1.

[4] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, “Heuristic approaches in robot path planning: A survey,” Robot. Auton. Syst., vol. 86, pp. 13–28, Dec. 2016, doi: 10.1016/j.robot.2016.08.001.

[5] J. Chen, X. Zhang, B. Xin, and H. Fang, “Coordination Between Unmanned Aerial and Ground Vehicles: A Taxonomy and Optimization Perspective,” IEEE T. Cybern., vol. 46, no. 4, pp. 959–972, Apr. 2016, doi: 10.1109/TCYB.2015.2418337.

[6] I. Boussaïd, J. Lepagnot, and P. Siarry, “A survey on optimization metaheuristics,” Information Sciences, vol. 237, pp. 82–117, Jul. 2013.

[7] E.-G. Talbi, Metaheuristics: From Design to Implementation. WILEY, 2009.

[8] B. B. K. Ayawli, R. Chellali, A. Y. Appiah, and F. Kyeremeh, “An Overview of Nature-Inspired, Conventional, and Hybrid Methods of Autonomous Vehicle Path Planning,” *J. Adv. Transp.*, p. 8269698, 2018, doi: 10.1155/2018/8269698.

[9] N. Siddique and H. Adeli, *Nature-Inspired Computing Physics and Chemistry-Based Algorithms*, 1st ed. CRC Press, 2017.

[10] I. Chaari, A. Koubaa, H. Bennaceur, A. Ammar, M. Alajlan, and H. Youssef, “Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments,” *Int. J. Adv. Robot. Syst.*, vol. 14, no. 2, Apr. 2017, doi: 10.1177/1729881416663663.