

Collision-free Random Paths between Two Points

Mohammad Ali H. Eljinini

Faculty of Information Technology, Isra University, Amman, Jordan
E-mail: ma.eljinini@iu.edu.jo

Ahmad Tayyar

Department of Computer Science, Jarash University, Amman, Jordan
E-mail: ahmad.tayyar@jpu.edu.jo

Received: 31 July 2019; Revised: 20 August 2019; Accepted: 12 September 2019; Published: 08 June 2020

Abstract—This paper proposes a collision-free path planning algorithm based on the generation of random paths between two points. The proposed work applies to many fields such as education, economics, computer science and AI, military, and other fields of applied sciences. Our work has spanned several phases, where in the first phase a novel computer algorithm to generate random paths between two points in space has been developed. The aim was to be able to generate paths between two points in real-time that cannot be predicted in advance. In the second phase, we have developed an ontology that describes the domain of discourse. The aim was two folds; firstly, to provide an optimized generation of best points that are closer to the target point. Secondly, to provide sharable, reusable ontological objects that can be deployed to other projects. We reinforced our solution by the initiation of several case studies that have been designed using and extending our work. One problem that we have faced in some cases is the existence of some obstacles between the starting and the ending point. For example, in our work towards the automation of a navigation system for drones, we faced some obstacles like trees, no flying zones, and buildings. This problem is also applicable to mobile robots and other unmanned vehicles, where collision-free mobility is necessary. In this phase, we have reworked the algorithm to generate random paths between two points $P_0(x_0, y_0)$, $P_n(x_n, y_n)$ with obstacles. Our generated random paths are placed within circles that are centered in P_n : c_1, c_2, \dots, c_{n-1} , which passes thru the points P_1, P_2, \dots, P_{n-1} respectively. Point P_i may approach P_n if it takes any position within circle c centered in P_n with radius P_iP_n and satisfies some constraints, discussed in detail in the paper, which insure that the selected paths do not fall within obstacles and reach the target point. We also classified the generated paths based on given properties such as the longest path, shortest path, and paths with some given costs. The resulted algorithms were very encouraging and leading to the applicability of real-life cases.

Index Terms—Random Paths, Obstacles, Mobile Robots, Ontology, Graph Theory.

I. INTRODUCTION

In the first phase of our work, we have developed a novel computer algorithm to generate random paths between two points in space [1]. A random path consists of a finite number of connected points that are generated randomly and satisfy the condition: $L(P_iP_n) < L(P_{i-1}P_n)$. Where $L(P_iP_n)$ is the length of the path between the two connected points P_i and P_n . The aim was to be able to generate paths between two points in real-time that cannot be predicted in advance. Our work applies to many fields such as education, economics, computer science, AI, military, industry, and other fields of applied sciences. In the second phase, we have developed an ontology that describes the domain of discourse. The aim was two folds; firstly, to provide an optimized generation of best points that are closer to the target point. Secondly, to provide sharable, reusable ontological objects that can be deployed to other projects [2].

We reinforced our solution by the initiation of several projects that have been designed using and extending our work. One problem that we have faced is the existence of obstacles in some cases between the starting point and the ending point. For example, in our work towards the automation of a navigation system for drones, we faced some obstacles like trees, no flying zones, buildings, among other obstacles. In this phase of our work, we have reworked the algorithm to generate random paths between two points $P_0(x_0, y_0)$, $P_n(x_n, y_n)$ with obstacles. Also, we have classified the generated paths based on given properties such as the longest path, shortest path, and paths with some given costs. The resulted algorithms were very encouraging and leading to the applicability of real-life cases. The following section presents related work. Section 3 discusses the structures and algorithm used in our work. In section 4, we provide a discussion on the assessment and evaluation of our work. The work is concluded in section 5, and future work is presented.

II. RELATED WORK

The generation of paths between two points is also called path planning, and increasingly being employed in the navigation of mobile robots and unmanned vehicles. Mobile robots are used in many automated environments such as servicing older adults, transferring goods in large factories, agriculture and farming, transportation, and military applications [3]. Researchers have worked on and proposed many algorithms for path planning in recent years. The task of generating free-collision, suitable paths in real-time is computationally challenging. Most early works reduced the problem into the two-dimensional grid, and transformed obstacles into blocked cells; graph-searching algorithms are then used on the resulted map to find suitable paths between the starting point and the target point [4, 5]. In many cases, Dijkstra's algorithm, a well-known, greedy-based algorithm, has been used to find the shortest path between the starting point and the target point. The computation cost of Dijkstra's algorithm becomes $O(N^2)$, where N is the number of nodes. Less efficient pathfinding algorithms, in this case, may run as high as $O(2^n)$ [6, 7]. Early approaches became inefficient with more complex and dynamic environments. Other approaches utilize sensor-based models found in mobile robots equipped with cameras and other sensors. The movement is performed in a straight line toward the target point until an obstacle is confronted. Then, it tries to move around the edge of the obstacle, in an attempt of finding another suitable path. This approach works with unknown static environments. However, it becomes difficult when confronted with moving obstacles. Another problem is finding the optimal path when required. Later, researchers turned into evolutionary approaches such as Genetic Algorithms, Ant Colony Optimization, and Particle Swarm Optimization [8, 9, 10]. Evolutionary approaches model biological behavior found in insects, birds, and mammals. In recent years, hybrid models are being used where more than one approach is often combined to overcome their drawbacks. The system starts with the most straightforward approach, for example, moving towards the target point in a straight line. The system then steps up to another approach based on the current situation until reaching the target point. This is accomplished efficiently by the usage of hierarchal structures, where the higher level provides general directions (i.e., the location of the target point), and the lower level deals with obstacles, besides following the instructions from the higher level [11].

In our work, all points between the starting point and the target point are dynamically generated in real-time by a random function. We used a biased approach where each generated point is tested and validated based on some constraints, such as its location in relation with the target point and the existence of obstacles, which we will describe in detail in the following section.

In phase two of our work as described in [2], we have designed an Intelligence Consultation Unit (ICU) that guides the process of generating random paths between the starting point and the target point. We have used the

Object-Oriented Paradigm and UML (Unified Modeling Language) to build our model.

In this phase, we have extended our model to provide descriptions of which condition to consider based on the following criteria:

1. $L(P_i P_n) < L(P_{i-1} P_n)$, For $i = 1, 2, \dots, n-1$
2. $P_i \notin Z_i$, where Z_i is the set of points inside an obstacle.
3. $L(P_i P_{i+1})$ does not pass through an obstacle.

Let P_0 and P_n be two points. Initially, we generate a random displacement at the point P_0 ; the ICU considers the endpoint of that displacement is P_1 if it was acceptable. It means that our ICU decides if this displacement satisfies the above constraints and converges to P_n . Otherwise, a new random displacement is generated. The process is repeated at point P_1 to obtain the next point P_2 , and so on until a point with a specific distance from P_n is reached. When applying this scenario to mobile robots, points become the footsteps to follow one at a time. Once the robot reached a point, the remaining distance is re-evaluated, and a new decision that satisfies the above constraints are taken. The process of generating the paths is very fast and works well in complex and dynamic environments.

III. THE GENERATION OF RANDOM PATHS WITH OBSTACLES

In this section, we present the main contribution of this paper. We have extended our work to handle the existence of obstacles in order to provide collision-free paths. We have built a UML model that defines the various concepts that describe the domain of discourse. We have used geometrical entities like points, lines, circles, and squares.

Our generated random paths are placed within circles that are centered in P_n : c_1, c_2, \dots, c_{n-1} , which passes through the points P_1, P_2, \dots, P_{n-1} respectively. Point P_i may approach P_n if it takes any position within circle c centered in P_n with radius $P_i P_n$ and satisfies the following constraints:

1. $L(P_i P_n) < L(P_{i-1} P_n)$, For $i = 1, 2, \dots, n-1$
2. $P_i \notin z_i$, where z_i is the set of points inside an obstacle.
3. $L(P_i P_{i+1})$ does not pass through an obstacle.

The position of P_i is generated randomly based on the following equations:

$$x += \text{random}(2*L+1) - L; \quad (1)$$

$$y += \text{random}(2*L+1) - L; \quad (2)$$

A random displacement is generated at point $P_i(x_i, y_i)$ within a square with the length of sides equal to $(2L)$, and point P_{i+1} as the endpoint of that displacement if it

was located within the circle c_i . The generation process is repeated at P_{i+1} , and within the Circle c_{i+1} centered in P_n with radius $P_{i+1}P_n$ to obtain new point P_2 . This iteration comes to a halt when reaching a point with a specific distance from P_n . This process will result in obtaining a set of points $P_1, P_2 \dots P_n$, which satisfies the first constraint, and can also be written as the following:

$$L(P_{n-1}P_n) < L(P_{n-2}P_n) < \dots < L(P_1P_n) < L(P_0P_n) \quad (3)$$

Circles and squares play an important role in directing the generated paths to reach the target point. The length of the side of the square limits the distance between the accepted points.

The second constraint guarantees that the generated point does not belong to the set of points falling inside an obstacle. Obstacles are transformed into geometrical shapes. Treating an obstacle as a geometrical entity

allows us to compute the points that lay inside its boundaries; therefore, we can quickly check that line P_iP_{i+1} does not pass through an obstacle. An acceptable shape we used in our work is a circle. The assumption that an obstacle falls inside the boundaries of a circle allows us to generate collision-free paths with low computational cost as will be shown below.

Figure 1 presents our extended higher-level model, which uses the object-oriented paradigm, where the ICU unit, the generated path, and all shapes that we use are instantiated objects. We begin by initializing the starting point P_0 , the ending point P_n , L , and d . Then we create new objects for the path and for the ICU. The iteration starts with generating the first random point. The new point is then sent to the ICU unit to check for the constraints. If the point is accepted, we update our path, and a new random point is generated. We iterate through the process until the target point is reached.

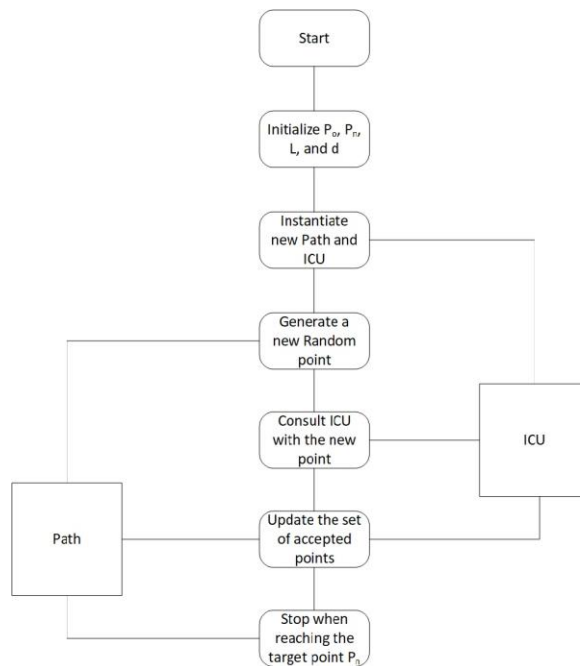


Fig.1. Path generation model

The ICU is a rule-based engine that is used to decide which points to accept based on certain thresholds that are explained previously.

Our extended algorithm consists of the following steps:

1. Initialize x_0, y_0, x_n, y_n, L, d
2. Let $P(x, y)$ be current position, then $x \leftarrow x_0, y \leftarrow y_0$
3. Calculate distance $L1(P, P_n)$ between P and P_n
4. Generate a random displacement at $P(x, y)$ within a square centered in P and its side length is $2L$, so $x \leftarrow x + Dx$ and $y \leftarrow y + Dy$
5. Calculate the new distance $L2(P, P_n)$
6. If $(L2 < L1$ and $P(x, y)$ does not belong to any obstacle) then
 - a. Link the points P and P_0

b. Rename the new position by $P_0(x_0 \leftarrow x, y_0 \leftarrow y)$

c. Consider $L1=L2$

7. If $L2 < d$ then stop

8. Go to step number 4

To test the algorithm, we have implemented a system in C-Sharp. In the implementation of the proposed algorithm, to ensure that our code is structured, we placed steps 4 to 7 inside a do-while loop, with the condition $(L2 \geq d)$. For experimentations, the initial values for the starting point and the target point are entered by the user at the beginning of the run. In addition, the user may insert any number of obstacles and sets their sizes and locations.

We have performed our testing and evaluations in two sets; the first set was done without obstacles and the

second set with obstacles.

A. Testing without obstacles

In this set, we have tested the algorithm using 9 cases, where we generate random paths in groups of 10, 100, 1000, 10000, 20000, 40000, 60000, 80000, and 100000 iterations. We recorded all the results in table 1. Figure 2 shows the graphs of the generated random paths without obstacles for the first four cases.

We have chosen the locations for $P_0(x_0, y_0)$ to be (300, 300) and for $P_n(x_n, y_n)$ to be (1000, 300) during all tests in this set. This is to ensure objectivity when performing

the comparisons. In these experiments, we have used the Cartesian Plane with the point (0,0) located in the top left corner, and the units used are in pixels. Therefore, the distance between P_0 and P_n is equal to 700, based on the following equation:

$$Distance(P_0, P_n) = \sqrt{(X_n - X_0)^2 + (Y_n - Y_0)^2} \quad (4)$$

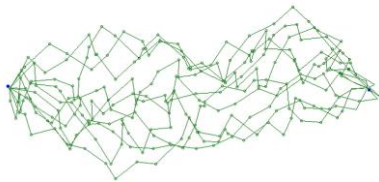
We computed the length of each random path, in addition to the minimum, the maximum, and the average distance for each case as shown in table 1.

Table 1. Length Of Each Path With Different Iterations

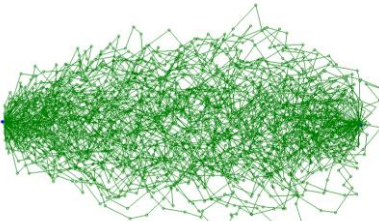
Case	Number of Iterations	Length of Straight Path	Length of Min Generated Path	Length of Max Generated Path	Length of Average Generated Path
1	10	700	809	1339	1066
2	100	700	871	1403	1049
3	1000	700	803	1542	1056
4	10000	700	794	1536	1054
5	20000	700	780	1566	1055
6	40000	700	780	1592	1055
7	60000	700	761	1606	1056
8	80000	700	742	1627	1056
9	100000	700	733	1633	1055

The most important observation in this table is that the greater the number of iterations, the length of the shortest path is approaching the value of 700 which represents the straight path between P_0 and P_n . Also, the length of the longest path is going above 1600.

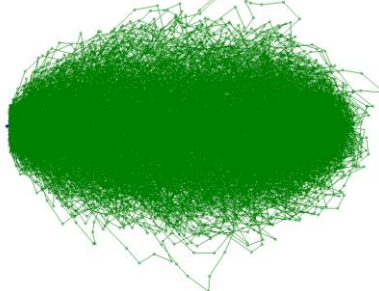
Case 1



Case 2



Case 3



Case 4

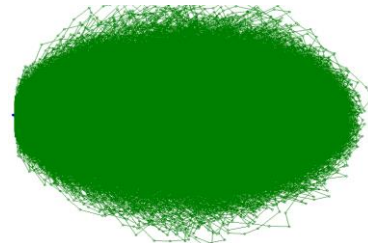


Fig.2. Four cases of random paths without obstacles

The plotted path in figure 3 represents the results with no obstacles, where we can also see the average with the higher number of iterations is converging to around the 1000 line, and the minimum distance, with the larger number of iterations, is approaching the 700 lines.

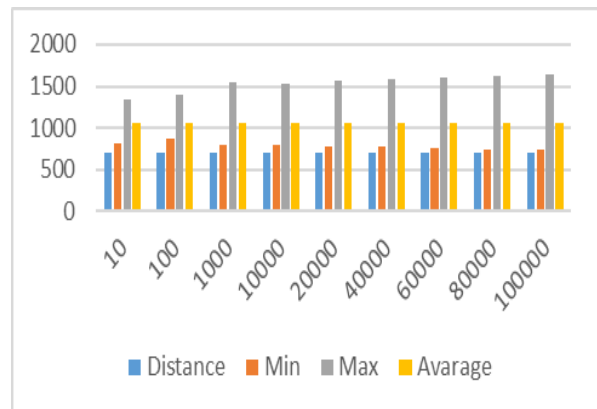


Fig.3. Results with no Obstacles

B. Testing with obstacles

We have tested the algorithm using a different number of obstacles, sizes, and locations, and recorded all the results. Figure 4 shows 4 cases, where each case presents a set of obstacles and many generated paths, connecting the starting point with the target point. Obstacles are presented as yellow circles.

To ensure the generation of %100 collision-free paths, we have placed safe zones around each obstacle, which are shown in red. In theory, where well-defined mathematical principles are used, the safe zone can be eliminated. However, in real life with mobile objects, for example, robots, drones, and moving obstacles, a safe zone may become a lifesaver.

In the first two cases, we have set the distance between adjacent points to 10, while in the third and fourth cases, we have set the distance between adjacent points to 50. The generated paths in the first two cases are smoother, more focused, and shorter in comparison with the last two cases. In all cases, the generated paths converged to the target point, avoiding all obstacles. Moreover, the condensation of the paths, shown in a darker color, have shown that most paths favored the shortest path between the starting point and the target point. The time complexity of our algorithm is $O(n)$, running one loop equal to the number of the generated points between the starting point and the target point. Most of the operations done in this algorithm is generating points, calculating distances, and deciding which one is closer to the target point that satisfies the constraints described above. The analysis of our algorithm is presented in detail in the next section.

We chose the same locations for $P_0(x_0, y_0)$ to be (300, 300) and for $P_n(x_n, y_n)$ to be (1000, 300) during all tests. The actual distance between P_0 and P_n is also equal to 700.

We calculated the length of every edge and path for every case. The results are shown in Table 2. The maximum path length shown in the table is close to 1500, and the minimum is close to 1000, where the average is approximately 1170.

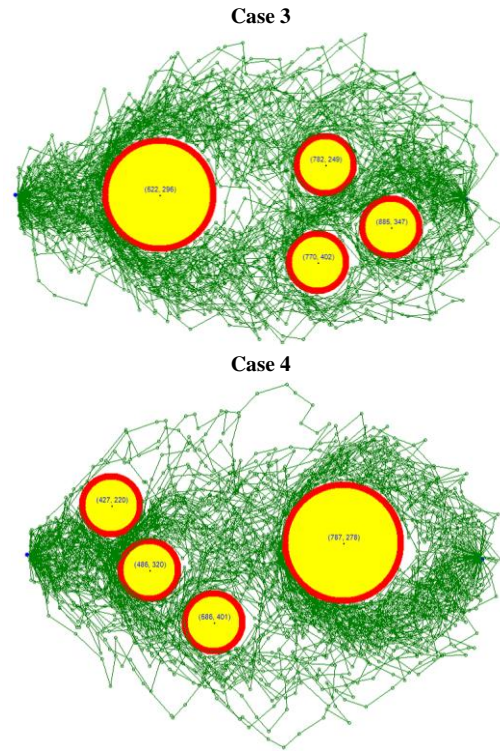
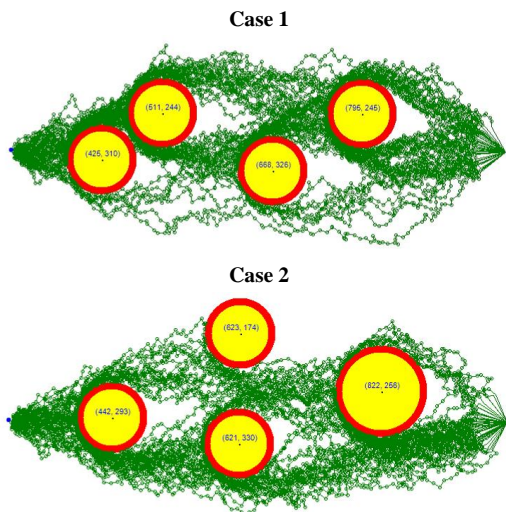


Fig.4. Four cases of random paths with obstacles

Table 1. Results obtained from the four cases

	Case 1	Case 2	Case 3	Case 4
Number of Paths	50	50	100	100
Starting Point	300,300	300,300	300,300	300,300
Target Point	1000,300	1000,300	1000,300	1000,300
$L(P_i P_{i+1})$	10	10	50	50
Actual length	700	700	700	700
Shortest path	943	919	880	892
Longest path	1164	1106	1469	1396
Average Length	1019	1004	1118	1102

We took one case this time and tested the algorithm again with the distance between two points is 50. This time we randomly generated paths in groups of 1, 10, 100, 1000, and 10000 iterations. We computed the length of each random path, in addition to the average, the minimum, and the maximum distance for each group as shown in Table 3.

Table 2. Length of each path with different iterations

Number of Iterations	Length of path	Average	Max	Min
1	700	1013	1013	1013
10	700	995	1131	884
100	700	1069	1308	878
1000	700	1075	1441	825
10000	700	1074	1507	789

Looking at table 3, we see that the average stayed in the range of 995 and 1075, while the minimum distance ranged from 789 to 1013. The plotted path in figure 5

presents our findings, where we can see the average stayed around 1000, and the minimum distance, with the larger number of iterations, is converging towards the distance of 700.

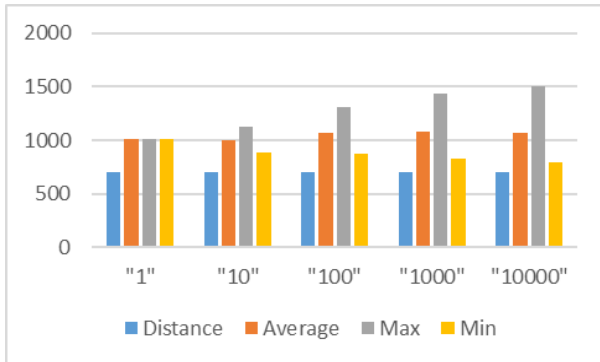


Fig.5. Results with Obstacles

IV. DISCUSSION AND EVALUATION

The main objective of this work is the addition of obstacles and the generation of free-collision random paths. We have tested the algorithm with many cases; each has a different set of constraints with a variety of obstacles. Our obstacles have different sizes and locations. Moreover, we have generated random paths up to one hundred thousand paths, covering many possibilities. In all cases, the generated paths converged to the target point, avoiding all obstacles. Moreover, tests have shown that paths favored the average-length path between the starting point and the target point.

From the literature review shown in section 1, the time complexity of all the algorithms that we have reviewed ranged from $O(n^2)$ to $O(2^n)$. The time complexity of our algorithm is $O(n)$, running one loop equal to the number of the generated points between the starting point and the target point. Most of the operations done in this algorithm are generating points, calculating distances, and deciding which one is closer to the target point that satisfies the constraints described above. The analysis of our algorithm is follows:

Steps of the Algorithm	Running Time
<i>Initialize</i> x_0, y_0, x_n, y_n, L, d	C_0
<i>Let</i> $P(x, y)$ <i>be current position, then</i> $x \leftarrow x_0, y \leftarrow y_0$	C_1
<i>Calculate distance</i> $L1(P, P_n)$ <i>between</i> P <i>and</i> P_n	C_2
LOOP (while $L2 > d$) BEGIN:	
<i>Generate random displacement at</i> $P(x, y)$ <i>within square centered in</i> P <i>and side length is</i> $2L$, <i>so</i> $x \leftarrow x + Dx, y \leftarrow y + Dy$	$3n + C_3$
<i>Calculate the new distance</i> $L2(P, P_n)$	$n + C_4$
<i>If</i> $(L2 < L1)$ <i>and</i> $P(x, y)$ <i>does not belong to any obstacle) then</i>	$3n + C_5$
a. Link the points P and P_0	

b. Rename the new position by
 $P_0(x_0 \leftarrow x, y_0 \leftarrow y)$

END OF LOOP

$T(n)$

$7n + C$

As presented above, the sum of all constants C_0 to C_5 is equal to C , a constant. Thus, the time complexity of our novel algorithm yields to $O(n)$.

V. CONCLUSION AND FUTURE WORK

We have proposed and implemented an algorithm, described in section 2, for generating collision-free random paths between two points (P_0, P_n). We have tested the algorithm under all possible constraints, and the generated paths were able to converge to the target point avoiding all obstacles, which have different sizes and locations. The algorithm is very fast, with time complexity of $O(n)$.

This work can be applied in many cases. During the second phase of our work, as described in [2], we have initiated several projects utilizing the algorithm. Extending the work to deal with obstacles was very helpful in solving problems faced us in some of these projects, as described in the following subsections.

A. Towards the Automation of Drone Navigation Systems

Unmanned aircraft (Drones) are becoming very popular these days. This is due to their low cost and the various applications they provide for humanity in many fields like farming, industry, transporting products, research, and military.

Most drones today use pilot-controlled navigation systems. Current technologies for automating the drone navigation systems is still in its infancy stages and open for research [12, 18, 19, 20].

Probably, one of the most faced challenges, in the task of automating the navigation of drones is dealing with obstacles such as buildings, trees, and other objects. This becomes much harder with moving obstacles, such as vehicles, and other drones. Planning such paths in a complex and dynamic environment becomes computationally very costly.

Over the last few years, due to its low cost and low risk of casualties, many countries around the world are utilizing drones in their military and civil operations. While it is easy to remotely pilot a single drone, controlling many drones while conducting various tasks becomes a major objective.

Current-generation drones face the limitations of flying at low altitudes and slow speed compared to manned aircraft. In hostile zones, flying drones from one point to another in straight lines are vulnerable to anti-air defense systems and therefore are easy targets.

The generation of random paths between two points in real-time, which cannot be predicted in advance, can be of great value to flying drones. This method would greatly minimize the loss of drones over disputed regions.

Due to the limitations of the current-generation drones, most countries are using them for surveillance purposes only.

To the authors' knowledge, all current-generation drones are remotely piloted drones. While some developed countries are advancing the state-of-the-art drone technology in many directions, the development of self-piloted drones is still in its infant stage.

This research can be valuable work in this direction. We are applying our research findings towards the automation of drone navigation systems.

B. Crawling the World-Wide-Web

The World-Wide-Web is a huge dynamic digraph. Vertices in the graph are webpages, where each may contain textual data, images, and other multimedia elements. Hyperlinks are the edges that connect webpages together. Search engines use crawlers that traverse the web collecting and processing all types of data. Crawlers use breadth-first graph traversal algorithm to collect data and store it in databases.

Is it possible to crawl the web in random order? What kind of results may we get? Can we consider unwanted webpages as obstacles? How can we set up a starting webpage and a target webpage within this large dynamic graph?

These are some questions that we are working on in this project. One problem we are facing is dealing with geometrical shapes like circles and squares, which we have used to move closer to the target point. In this research, we are using costs instead, which are generated from some properties of the webpages.

Some scholars are using random walk theories to visualize subsets of the web and collect information in order to study web properties. Crawling the web in random order is an unbiased process, while our algorithm is biased because we must converge to the target point. Random walk models have been used in biology, physics, ecology, medicine, computer science, and other scientific disciplines [15, 16, 17].

REFERENCES

- [1] A. Tayyar, "Generating Random Paths between Two Points in Space: Proposed Algorithm". *Proceedings of the International Conference on Computer Science, Computer Engineering, and Social Media*, Thessaloniki, Greece, 2014.
- [2] A. Tayyar, M.A. Eljinini, "Ontology-Based Generation of Random Paths between Two Points". *Journal of Applied and Theoretical Information Technology*, vol. 96, no. 15, pp. 4984-4905, 2018.
- [3] P. Raja, S. Pugazhenthii, "Optimal Path Planning of Mobile Robots: A Review". *International Journal of Physical Sciences*, vol. 7, no. 9, pp. 1314-1320, 2012.
- [4] D. Payton, J. Rosenblatt, D. Keirse, "Grid-based mapping for autonomous mobile robot". *Journal of Robotics and Autonomous Systems*, vol. 11, no. 1, pp. 13-21, 1993.
- [5] O. Hachour, "Path planning of autonomous mobile robot". *International Journal of Systems Applications, Engineering and Development*, vol. 4, no. 2, pp. 178-190, 2008.
- [6] P. Frana, "An Interview with Edsger W. Dijkstra". *Communications of the ACM*, vol. 53, no. 8, pp. 41-47, 2010.
- [7] W. Huijuan, Y. Yuan, Q. Yuan. "Application of Dijkstra Algorithm in robot path planning". *Proceedings of the Second International Conference on Mechanical Automation and Control Engineering*, pp. 1067-1069, 2011.
- [8] I. Al-Taharwa, A. Sheta, M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment". *Journal of Computer Science*, vol. 4, no. 4, pp. 341-344, 2008.
- [9] M. Garcia, O. Montiel, O. Castillo, R. Sepulveda, P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost evaluation". *Journal of Applied Soft Computing*, vol. 9, pp. 1102-1110, 2009.
- [10] Q. Zhang, G. Gu, "Path planning based on improved binary particle swarm optimization algorithm". *Proceedings of the IEEE International Conference on Robotics, Automation and Mechatronics*, China, pp. 462-466, 2008.
- [11] Y. Zhu, T. Zhang, J. Song, X. Li, "A new hybrid navigation algorithm for mobile robots in environments with incomplete knowledge". *The Journal of Knowledge-Based Systems*, vol. 27, pp. 302-313, 2012.
- [12] T. Krajn'ik, V. Vonasek, D. Fi'ser, J. Faigl. "AR-drone as a platform for robotic research and education". *Proc. Research and Education in Robotics: EUROBOT*, 2011.
- [13] L. Kleinrock, *Queueing Systems*. John Wiley & Sons, 1975.
- [14] L. Allen, G. Jackson, J. Ross, S. White, "What counts is how the game is scored: One way to increase achievement in learning mathematics". *Simulation & Games*, vol. 9, pp. 371-389, 1978.
- [15] EA. Codling, MJ. Plank, s. Benhamou, "Random walk models in biology". *Journal of the Royal Society Interface*, vol 5, no. 25, pp. 813-834, 2008.
- [16] PM. Kareiva, N. Shigesada, "Analyzing insect movement as a correlated random walk". *Oecologia*. vol. 56, pp. 234-238, 1983.
- [17] A. Okubo, S. Levin, *Diffusion and Ecological Problems: Modern Perspectives*. Springer Science & Business Media, New York, 2nd Ed., 2013.
- [18] D. Floreano, R.J. Wood, "Science, technology and the future of small autonomous drones". *Nature*, vol. 521, pp. 460-466, 2015.
- [19] M. Hassanalian, A. Abdelkefi, "Classifications, applications, and design challenges of drones: A review". *Progress in Aerospace Sciences*, vol. 91, pp. 99-131, 2017.
- [20] M. Funk, "Human-drone interaction: let's get ready for flying user interfaces. Interactions". *Interactions*, ACM, New York, vol. 25, no. 3, pp. 78-81, 2018.

Authors' Profiles



Mohammad Ali H. Eljinini was born in Amman, Jordan in 1961. He received B.Sc. and M.Sc. degrees from the USA, in 1990 and 1992, respectively. In 1992, he worked with UNITEL Inc. in the USA, as a Systems Director. In 1996, he joined the Faculty of Information Technology at Isra University, in Amman, Jordan, as an Instructor of Computer Science.

He received the Ph.D. degree in Health Informatics from City University, in the UK, in 2007. He spent the 2007-2012 academic years as the chair of the department of the Computer Information Systems at Isra University. He served as the Dean of the Faculty of Information Technology during the 2013-2016 academic years.

He is currently an Associate Professor at the Department of Computer Information Systems, in the faculty of Information Technology, at Isra University, Amman, Jordan. He has published over twenty refereed journals and conference papers in the areas of E-Health, E-Learning, Ontology and Knowledge Representation, and the Semantic Web. In addition, he has authored several books in Computer Science.



Ahmad Tayyar was born in Aleppo, Syria, in 1954. He received B.Sc. from Aleppo University, Aleppo, Syria in 1981. He received the Ph.D. degree from Hute Alsace University, in Mulhouse, France, in 1991.

He worked at Tichreen University, Aleppo University, and Isra University during the years of 1991 to 2004, as Assistant Professor.

He worked at Isra University and Aleppo University as Associate Professor during the academic years of 2004 to 2016. He moved to Jarash University in 2016, where he still there to this date.

During his career in academia, he authored several books in computer science. He also published many scientific research articles in refereed journals and conference proceedings. He also supervised many Masters and Ph.D. students. His research interests include Algorithms, Graphics, Shortest Paths, and Random Trajectories.

How to cite this paper: Mohammad Ali H. Eljinini, Ahmad Tayyar, "Collision-free Random Paths between Two Points", International Journal of Intelligent Systems and Applications(IJISA), Vol.12, No.3, pp.27-34, 2020. DOI: 10.5815/ijisa.2020.03.04