Proposed Modified A* for Three-Dimensional Sphere Environment

Alia K. Abdul Hassan*, PhD (Asst Prof.)
hassanalia2000@yahoo.com

Duaa Jaafar Fadhil*
satomi.emoki111@gmail.com

Abstract: Robotic path-planning is a major problem in the robotic world, and one of the most used algorithms in this area is the A* algorithm. This paper presents a modified A* Algorithm that can work in a Three-Dimensional (3D) sphere environment and allows the moving of robot to reach its goal while avoiding obstacles on its path, solve local minima problems, safety rim and accurate convergence. The F(n)* evaluation function has been used in the proposed modified A*. Simulation results showed that the proposed modified A* guarantee to find near optimal, safe path from the start position to goal point in 3D static sphere environment with acceptable execution time.

Keywords: A*, path planning, 3D environment, sphere space.

* Computer Science Department, University of Technology, Baghdad, Iraq

1. Introduction

Path planning and obstacle avoiding are challenging topics for mobile robot's navigation where precise, efficient yet simple algorithms are needed to guarantee the robot reaching its goal while avoiding any collision along the path [1,2]. One of the best known path planning algorithms is the A* algorithm that was successfully implemented and tested as optimal, complete and easy path planning algorithm for a mobile robot. Originally the A* algorithm works on 2D environment, in this paper a proposal is presented to develop the A* version to work in a 3D environment where the robot has more freedom to move up and down along with the used to left, right, front and back directions.

A primary problem in the establishment of autonomous robots is the motion planning problem. In order for the robot to reach the final state without colliding with other objects or robots along the way, the motion planning problem determines the sequence of movements that the robot should follow with minimum possible amount of time [3,4].

The term path planning was used in a large scope of areas such as robotics, Artificial Intelligence (AI) and Control theory. So each area uses its own definition for this term. In robotics, path planning deals with a problem of moving the robot from one point to another [5,6]. Path planning problem is a basic simplified motion planning problem that deals with the geometric issues only. In such problem assuming that the only moving entity in the space is the robot ignoring any dynamic properties of the robot itself, also cutting down motions to non-contact motions, and by considering the robot as a single rigid object with its points being fixed in distance to each other. By those assumptions the robot's movement will be constrained only by the obstacles [3].

In the motion planning problem, the robot is normally represented as a rigid object in the space which is called the work space of the problem, while in the configuration space the robot is represented as a point in the space and the obstacles are mapped in that space. So, instead of planning the motion of a dimensional robot, this configuration transforms the problem into planning the motion of a point. Those assumptions that transformed the motion planning problem in work space to a basic path planning problem in the

configuration space, which simplify the problem by considering the basic issues of finding feasible path and work that out instead of digging in the additional difficulties [3].

In sphere space, the whole configuration space is bounded by a sphere with known center-point, and bound each obstacle by sphere shape. By knowing the center point and the radius of the space and each obstacle it can accurately compute the distance to the border and obstacles each time the robot moves to make sure that the robot is contained in the space of the problem and the path will -if found- is completely collision free [7,8].

3D path planning nowadays is urgently needed where environments tend to be unstructured, complex and full of uncertain factors such as underwater or forests where path planning in simple 2D algorithm is not efficient [9-11]. In 3D environments the Robot can move freely up and down in addition to the right, left, front and back traditional moves.

In this paper, 3D space is considered so the robot may choose one of 26 directions instead of the 8 directions that are experienced in the 2D space. The difficulty of the 3D environment falls in adding an additional axis to the coordinate system resulting in three parameters X,Y,Z to define the point in the space in the format: P(x,y,z), instead of the P(x,y).

Section 2 of this paper highlights the related work for the proposed subject. In section 3, the traditional A* algorithm is explained, and section 4 presents the proposed modified version of the A* algorithm showing in details all the modifications have been made for this algorithm, while section 5 discuss the experimental results of running the proposed algorithm in a Matlab simulated test environments.

2. Related Work

Many researches have been made in the field of path planning generally in both 2D and 3D environments, The most recent works with A* algorithm are discussed next:

In 2011, Jiahai Liang proposed a 3D environment A* planner where the hill climbing method was used under the constraints of the grid, using a cost model to estimate the running cost for the grid from the starting point to the target [12]. In 2012, S.M. M. R. Al-Arif, et al, proposed a water

based rescue system that is based on AI algorithms including the A* algorithm to choose the best most suited algorithm that would work with minimum cost and least time resource [13]. In 2013, the A-r-Star pathfinder which outperforms the A* in uniform gridded sparse world and was developed by Daniel O., et al. [14].

Modified A* algorithm for mobile robot was represented by Frantisek D., et al. in 2014 where they focuses on improving the computational time and path optimality [5]. In 2016, a proposal of an enhanced A* algorithm was made by Priyanka S. and Velappa G., where the new version was developed to work efficiently in unknown environments having the ability to search for optimal path by keeping track of the turns that the robot makes while moving [15].

The proposed paper is a modification of the A* to work for path planning in three dimensional environment where the modified version is proposed to solve the challenges of planning in 3D environments in the shape of spheres, which was not covered and focused on in the mentioned related works.

3. Traditional A* algorithm

One of the most efficient and commonly used algorithms is the A* algorithm [13]. The objective of this algorithm is to compute the shortest free path available from the given start point to the given goal point [16]. Choosing the next point to move to the path is determined by the f function's value, the point with the minimum f value will be chosen as the next step on the path. The f function represents the cost value and can be calculated for each point in the configuration space by the equation:

$$F_{(n)} = H_{(n)} + G_{(n)} \quad (1)$$

where, n is the current point (node) in the configuration space, H(n) is the cost of moving from the start point to the current point, G(n) is the cost of moving from the current point to the goal point, F(n) is the f function, which is the total cost of the current point.

The cost of both H(n) and G(n) is usually computed as the Euclidean distance that can be computed using the next equation :

$$Distance = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \quad (2)$$

where x1, x2, y1, y2 are the position parameters for both points.

The A* algorithm uses two main lists to keep track of the work, those are:

- Open List: stores the extensions of each point that are not explored yet.
- Close List: The close list stores all the points that are already been explored including the optimal path points.

Each point on those lists are linked to their parent point (in case the point is connected to two different parent points then ones that result in minimum cost are chosen to link to). During the algorithm's execution, if the open list was empty at any time, then there is no valid path in that map.

The traditional A* algorithm can be explained in the following steps:

1) Defining the start and goal points.
2) Mapping the obstacles in the space and load them in the close list.
3) Add the start point to both the close list and open list.
4) Extract the extensions of the current point and add them to the open list –if they already exist in the list then update them according to the minimum f value- after making sure that they are not included in the close list already or out of the boundaries of the space, and ignore them otherwise.
5) If the open list is empty at this stage then there is no available path.
6) If the goal point added to the open list as extension of another point, then the path is found and can be calculated by the linked parents' points.
7) If the goal was not found yet then the next step will be the point with minimum f value in the open list, add the selected point to the close list and repeat from step (4) [13].

4. Modified A* for Three-Dimensional sphere environment

The traditional A* algorithm is optimal, complete, easy and best suited in an offline static environment where the space of the problem is modeled as a graph and each node in the graph represent a robot's position [14]. In this paper, the modified version of this algorithm maintains those properties while working in a 3D environment by adding the following:

- 3D Environments, to work in 3D space it is needed to declare each point in the space as three parameter format (x,y,z) instead of the (x,y), so this modification will apply along all the algorithm whenever is needed to deal with a point within the space of work.

- 3D-sphere spaces, assuming that the whole configuration space is contained in a sphere shape and all the obstacles within the space are contained within sphere shapes as well, so the computations will be made using the center point and a radius of the sphere, and that makes the process of finding distance to each obstacle is unified along the way, which require a matrix to contain each obstacle's radius and center point to deal with them later in the steps of the proposed version.

- Safety rim, means adding extra space around each obstacle as an additional length to the radius of the obstacle that is not visual and does not really exist, but just added computationally to make the steps of the robot safer around the obstacles. Keeping in mind that {the radius + the rim >= step size} to ensure that the robot will not pass through the obstacle in any case.

- Calculating distance, the distance is used as the cost value for each node, the Euclidean distance in 3D is defied by equation 3:

$$\text{Distance} = \sqrt{(x1 - x2)^2 + (y1 - y2)^2 + (z1 - z2)^2} \quad (3)$$

- Open, Close lists, in proposed modified A* version will keep using those lists with slight change to make them work with the 3D concept, and here is an explanation for those changes:

- Open list, the only change in this list is extending the parameters to be 3D like, by storing the points' parameters as x,y,z in the list.

- Close list, in the traditional A* algorithm, the close list will contain all the obstacles as well as the points from the open list that are already been visited so, going over the same path more than once will not be done. In the proposed modified version, the close list contains only the points of the paths that are already explored including the final optimal path, while the obstacles will be saved in an individual matrix.

- Expand-list Function, this is a very important function that used to retrieve the current point's successors. To do so, it needs a series of checking for the successor's validity before including them in the open list. Those checks include:

1. Borders check, to ensure that any successor point it needs to extract falls within the borders of the configuration space.
2. Obstacles check, for each successor point need to test if the successor is not within or hitting an obstacle, and to do so, a loop is run for all the obstacles in the space and compute the distance between the successor and the center of the obstacle and make sure that distance is larger than the radius of that obstacle with the safety rim added to it.
3. Close-list check, to make sure that the robot will not go through the same point again, it is needed to check if the successor is already on the close list, and if so it is just ignored and skipped to the next one.
4. If the successor point passes all the three checks, then is it a valid point and can add it to the open list in the next step.

- Accurate convergence condition, to check if the robot comes to a convergence point and that point is accurately positioned on the given goal point, then a condition is added at each step checking the distance to the goal point if it was shorter than a threshold value (given as 1 in this work but can be changed), then the next point's cost will consider the G(n) value only (instead of the F(n)), and the step size will be equal to the threshold value divided by two. The threshold value chosen is 1 to make the robot lands as close as possible to the specified goal point. The procedure will be working as the following:

When the distance to the goal is equal to or less than the threshold (1), update the value of the threshold to be the current distance to the goal then the step will be half of this distance which is the threshold's updated value. This process will repeat until the robot is close enough to the goal

with tolerance value equal to 0.02 or any value that can be chosen by trial and error to suit the work aspects.

- Evaluation Function (F*(n)), to reduce the execution time for better performance, the original evaluation function F(n) is slightly changed by fixing the concept to take two times of the G(n) value against one time H(n), so the effect of the G(n) is heavier when choosing the successors, which cuts down the number of successors that are added to the open list along the execution that the algorithm need to go through and test each round. This will be shown as the following formula:

$$F^*_{(n)} = H_{(n)} + 2(G_{(n)}) \quad (4)$$

5. Experimental Results

The proposed modified A* algorithm was simulated using Matlab programming language in a 3D sphere environment with a 3D sphere shaped obstacles. The simulation was run on a laptop with Intel® Core™ i7-3517U CPU@1.90GHz , 15.9GB memory and Windows 8.1 Pro.

The executed environment is basically a cube of size (20×20×20) unit. Where the actual arena is a sphere shape space of a radius = 10 (the values were chosen for simulation purposes). The center of the sphere is at the center of the environment at the coordinates (0,0,0) as (x, y, z). The proposed modified A* tested to find a path from a specified start point to a specified goal point. Figure 1 (a) find a path in environment with single static obstacle, while multi-obstacles is shown in figure 1 (b) and crowded environment is shown in figure 1 (c).

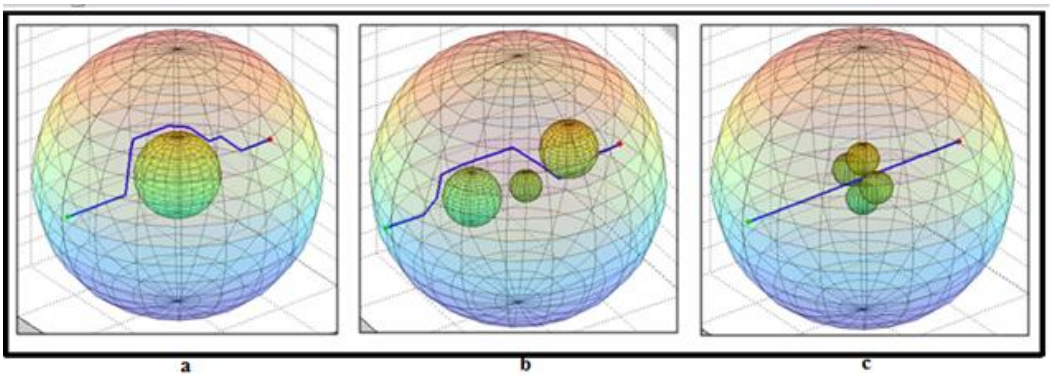

Figure1:Planned path using proposed algorithm a) single static obstacle b) multi static obstacles c) crowded environment

Figure 2 shows three cases of local minima that may occur when planning a path. The proposed modified algorithm succeeds to solve these problems and find a suitable short path.

The planned paths shown in figure 1 and 2 are collision free paths where the proposed modified A* add a safety rim around each obstacle to avoid the case when the distance to the obstacle that is against the robot's current position is smaller than the robot's step, then the robot will check the next position which passes through the obstacle's edge and move towards it ignoring the obstacle that's in the way and avoid collision with obstacles.

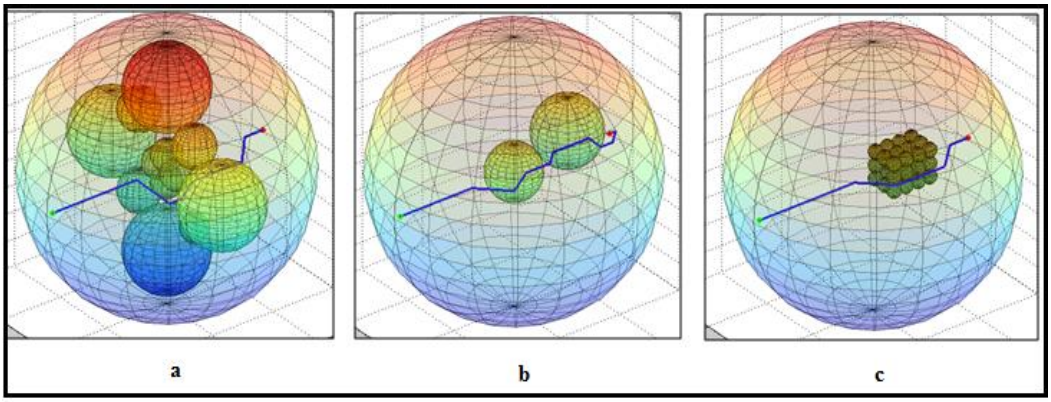

Figure 2 Local minima problems

Table 1 shows the execution time and path length for the running of the proposed modified A* algorithm for the environments showed in figure1 and 2. From table, 1 it can be seen that the proposed modified A* can solve most of the problems that occur with path planning in 3D environment and guarantee to find path if one found and in less time.

The planned path with proposed modified A* is near optimal path since it use the modified F(n)* Evaluation function that has positive affect on execution time as shown in table 1, also accurate convergence condition does not exist in the original A* algorithm, allows the float value to accurate position the element of the environment which will eventually affect the matching between the final point and the goal and that is where the condition plays to allow as accurate convergence as possible.

Table 1. Simulation results

Figure no.	Modified A* Algorithm With the original F(n)		Modified A* Algorithm With the modified F(n)*	
	Execution time (seconds)	Path Length (length units)	Execution time (seconds)	Path Length (length units)
1-a	3.9163	20.8137	0.0648	21.2566
1-b	1.0478	19.8407	0.0573	21.719
1-c	0.0642	17.5152	0.0631	17.5152
2-a	5.5347	19.8077	0.1123	19.8077
2-b	7.0019	20.8005	0.0604	20.9932
2-c	4.2476	19.1648	0.2597	19.1648

6. Conclusion

A proposed modified version for the traditional well known A* algorithm were developed in this paper. The proposed modified version works in 3D static known sphere environment where it has the ability to avoid 3D sphere obstacles and reaching the goal point. The proposed modified version is near optimal, of simple, complete and collision free with extra safety constrains provided by the safety rim, it also allow accurate convergence with the goal point with high speed execution time. As a future work, the algorithm can be developed further to work in unknown environments and the path can be smoothed.

References

[1] T. Weerakoon, K. Ishii and A. A. F. Nassiraei, **(2015)**, "AN ARTIFICIAL POTENTIAL FIELD BASED MOBILE ROBOT NAVIGATION METHOD TO PREVENT FROM DEADLOCK", JAISCR, Vol. 5, No. 3, pp. 1 89-203.

[2] S. V. Asl, Z. Davarzani and S. Staji, **(2015)**, "Planning Flying Robot Navigation in a Three-dimensional Space by Optimization Combining Q-learning and Monte Carlo Algorithms", International Journal of Hybrid Information Technology Vol.8, No.11 , pp.297-306.

[3] J. C. Latombe, **(1991).** "Robot Motion Planning", second printing, Kluwer Academic Publishers, NewYork, USA.

[4] Dr. A. K. A. Hassan, **(2014),** "Path Planning Method for Single Mobile Robot in Dynamic Environment Based on Artificial Fish Swarm Algorithm", Eng. &Tech. Journal, Vol. 32,Part (B), No.2.

[5] F. Ducho, A. Babinec, M. Kajan, P. Beno, M. Florek, T. Fico, L. Jurišica, **(2014),** "Path planning with modified A star algorithm for a mobile robot", ScienceDirect, Procedia Engineering 96 , 59 – 69.

[6] S. H. Shwail, A. Karim, **(2014),** " Probabilistic Roadmap, A*, and GA for Proposed Decoupled Multi-Robot Path Planning", IRAQI JOURNAL OF APPLIED PHYSICS, Vol. 10, No. 2, April-June.

[7] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki and S. Thrun, **(2005),** "Principles of robot motion: Theory, Algorithms, and Implementation", A Bradford Book, The MIT Press, Cambridge, England.

[8] T. A. Jaleel, A. K. A. Hassan, **(2016),** "Collision Avoidance Using Cat Swarm Algorithm for Multi Mobile Robot Path Planning in Dynamic Environment", Iraqi Journal of Science, Vol. 57, No.3C, pp:2348-2359.

[9] L. Yang, J. Qi, D. Song, J. Xiao, J. Han and Y. Xia, **(2016),** "Review Article, Survey of Robot 3D Path Planning Algorithms", Hindawi Publishing Corporation, Journal of Control Science and Engineering, Article ID 7426913.

[10] Sreeja Banerjee **(2014),** "A COMPARATIVE STUDY OF UNDERWATER ROBOT PATH PLANNING ALGORITHMS FOR ADAPTIVE SAMPLING IN A NETWORK OF SENSORS", University of Nebraska – Lincoln.

[11] L. Yang, J. Xiao, J. Qi, L. Yang, L. Wang and J. Han, **(2016),** "GART: An environment-guided path planner for robots in crowded environments under kinodynamic constraints", International Journal of Advanced Robotic Systems ,November-December: 1–18, DOI: 10.1177/1729881416671111.

[12] J. Liang, **(2011),** "A Path Planning Algorithm of Mobile Robot in Known 3D Environment", Procedia Engineering, Published by Elsevier Ltd. Open access under CC BY-NC-ND license.

[13] S.M. M. R. Al-Arif, A. H. M. I. Ferdous and H. S. Nijami, **(2012),** "Comparative Study of Different Path Planning Algorithms: A Water based Rescue System", International Journal of Computer Applications (0975-8887), Volume 39 – No.5.

[14] D. Opoku, A. Homaifar, E. Tunstel, **(2013),** "The A-r-Star (A*r) Pathfinder", International Journal of Computer Applications (0975 – 8887), Volume 67– No.8, April.

[15] P. Sudhakara and V. Ganapathy, (2016), "Trajectory Planning of a Mobile Robot using Enhanced A-Star Algorithm",Indian Journal of Science and Technology, Vol 9(41), DOI: 10.17485/ijst/2016/v9i41/93816, November.

[16] S. H. Shwail, A. Karim, S. Turner, **(2012),** "Probabilistic Multi Robot Path Planning in Dynamic Environments: A Comparison between A* and DFS", International Journal of Computer Applications (0975 8887).

مقترح نموذج معدل لخوارزمية A* لتخطيط مسار الروبوت في بيئة ثلاثية الأبعاد ذات الشكل الكروي

أ. م. د. علياء كريم عبد الحسن* دعاء جعفر فاضل*

المستخلص: تخطيط مسار الروبوت يعتبر من المشاكل الأساسية في عالم الروبوت، وواحدة من اكثر الخوارزميات المستخدمة في هذا المجال هي خوارزمية A*. هذا البحث يقترح نموذج معدل من خوارزمية A* قادرة على العمل في بيئة ثلاثية الأبعاد ذات شكل كروي حيث تقود الخوارزمية الروبوت للوصول الى الهدف دون التصادم مع أي من العوائق الموجودة على الطريق، موفرة حل لمشاكل Local Minima، مضيفة شريط الحماية بالإضافة الى تقارب دقيق مع الهدف واستخدامها لدالة تقييم معدلة (F(n)*) لتحسين الأداء. نتائج محاكاة النظام وتطبيقه برمجيا تظهر ان النسخة المقترحة من خوارزمية A* تضمن ايجاد اقرب افضل حل، بحيث يكون آمن من موقع الاقلاع بنقطة البداية حتى الهدف في بيئة ثابتة ثلاثية الابعاد ذات شكل كروي وبوقت تنفيذ جيد.

الكلمات المفتاحية: A*، تخطيط المسار، بيئة ثلاثية الأبعاد، مساحة كروية.

* قسم علوم الحاسوب , الجامعة التكنولوجية ، بغداد، العراق