Automatic Pathfinding of the Robotic Arm by A* Algorithm on 2-Dimension in MATLAB Simulink

Changrui Cai

Northeastern University, Boston, United State

cai.cha@northeastern.edu

Abstract. This project will give people who are new to A* algorithm a general idea about how robotic arms works in Simulink by applying the A*algorithm. Since A*algorithm combined advantage of two effective algorithm, this project decides to choose it as the main algorithm of the robot arms. In this project, the research question is whether a 2-joint robot arm passes through a single obstacle in 2D space to reach its destination and it selects the shortest path. The answer will be found by applying the A* algorithm on the robot arm. By analyzing the GHF values and trajectory graphs generated by MATLAB and Simulink. It has been confirmed that the A* algorithm works on the robot arm. Finally, success rate was calculated to be 81.7%, confirming that the 2-joint robot arm could pass through a single obstacle in 2D space to reach its destination and it selects the shortest path. The result of this project ca be more convincing if it run in 3-dimension.

Keywords: MATLAB, Simulink, A* algorithm, robotics, auto pathfinding.

1. Introduction

It has been a long time since the first robot was built. The first robot was named “Ultimate,” a reprogrammable manipulator in the 1950s. During these eighty years, the increasing development of technology has made people move their attention from manual to machines and robots. Compared with the work done by a human, the precision and accuracy of the action performed by human-controlled robots are much higher than humans, especially in simple work and action. Therefore, this motivation makes people use the robot in important work positions. In advance, people even pursue robot work automatically in some areas—for example, the Sparrow from Amazon. Sparrow is a robot system designed by amazon and is currently in use, and the Sparrow can automatically detect, select, and handle without crushing, so Amazon can pick and pack over 13 million packages daily. As a result, to improve efficiency, the increasing demand for robotic atomization is predictable and inevitable.

Human-like behavior is equally significant as the human-like appearance of robots. As a result, some researchers are trying to make the robot more like a human. For example, a research group developed a human-like motion planner of a robot arm based on features of the motion of human arms, which is called "human arm motion patterns (HAMPs)" [1]. According to research, they find a HAMPs value that can represent humans and use it to derive a program framework. Then, the researcher applied the developed planner to the robot and found that the robot could act like a human [1]. The article is typical research on how to design robots to act like a human. However, in this article, the research group is going to focus on efficiency, like the experiment shown in the research [2-4]. These researchers achieve their experiment by applying an algorithm to their experiment subject.

The A* algorithm is widely used. For example, the most noticeable was the application in vehicles. In the research, the author’s group developed a planning program of path-finding based on the A* algorithm [5]. Applying for this program allows an Automated Guided Vehicle (AGV) to reach the destination without a large turning angle [5]. This example shows a small part of how the A*algorithm helps the development of the vehicle. Not only for the actual vehicles, but the algorithm also works for virtual things. The A* algorithm was applied to Non-Playable Characters in some games, especially pathfinding games [6]. The appearance of NPC with this algorithm can improve the player’s experience since they act as a strong player (they always know how to find the shortest way). In research, Ade’s group uses Artificial Intelligence with A* algorithm to see the performance of it

playing tree-plant games, and the result shows the AI finished the game with an average process time of 0.0732s [6]. As a result, it is reasonable why people decide to use such an algorithm in the game's development.

Influenced by this trend, the research problem is related to the manipulator, which is whether or not a 2-joint robot arm passes through a single obstacle in 2D space to reach its destination and selects the shortest path. In order to find out the answer, some difficulties need to be addressed. Firstly, making every robot arm work in an empty room is impossible, so how to let a robot arm pass through an obstacle in a complex environment is a question that needs to be considered. Secondly, besides passing through an obstacle, it is essential to confirm that the robot follows the shortest path to improve the robot's efficiency. Such two complex tasks are hard to accomplish under human control, so that a program would be composed. With these in mind, the best-fit algorithm should be the A* algorithm. A* algorithm is good at pathfinding because it simultaneously utilizes the information derived from Dijkstra's Algorithm and Greedy Best-First-Search [7, 8]. The equation of the A* algorithm is listed below:

$$f(v) = g(v) + h(v) \tag{1}$$

Where $f(v)$ is the total distance, $g(v)$ is the length of the path from start point to a current point, and $h(v)$ is heuristic distance of the cell to the goal state [9]. After compared all the $f(v)$, the lowest value will be chosen as next input of the program [9]. Moreover, $g(v)$ can be seen as the information derived from Dijkstra's Algorithm [8]. Dijkstra's Algorithm is a greedy algorithm which was always applied on shortest way finding [2]. It will calculate the remaining distance between current position to the destination [3]. $h(v)$ can be seen as the information derived from Greedy Best-First-Search [8]. Greedy Best-First-Search is an algorithm, always depict as a tree map, that find path based on its evaluation function – select the best point with the aim of reaching the goal [4]. By combining these two widely used algorithm (Dijkstra's Algorithm and Greedy Best-First-Search), our group believe A* algorithm will be the best choice for our program, and this article will discuss the standard A*algorithm.

In this research, whether the program of the A* algorithm works will be examined first. Then, the robot arm's success rate be recorded. The success rate is the times a robot arm successfully passes through an obstacle and selects the shortest path divided by the total times. Finally, a conclusion can be drawn about the research question by evaluating the success rate.

2. Methods

In this research, some proposed hypotheses are required to determine the answer to the research question. The first hypothesis is that the robot arm can only move in a 2D space, and the second hypothesis is that there is only one obstacle during each simulation. Specifically, the third is the most important: if the success rate is higher than 80%, the robot arm passes the test, and the research question answer is that the robot arm can pass through the obstacle and select the shortest path. Furthermore, all the simulations and programming are manipulated in MATLAB and Simulink, and the moving space of the robot arm is rasterized.

As mentioned in the previous section, the A* algorithm needs to be applied to the robot arm, and the robot arm will move based on the information derived from the $f(v)$ value. In order to find the lowest $f(v)$ value, the created function will first measure and record three sets of $g(v)$ and $h(n)$ before making a decision. Then, after comparing the recorded data, the function will select the appropriate set of $g(v)$ and $h(v)$, and the program will produce and send a vector to the robot arm, which is the direction and distance it should move. At last, the function will repeat the above steps until it reaches the final point.

Since the robot needs to pass through an obstacle, a collision-detecting function is necessary, and there are two sections of this collision-detecting function. The first section enables the robot to forecast that its two sticks will not touch the obstacle, and once it touches it, it will eliminate the

current road and find a new one. To clarify, when the robot arm touches the obstacle, the obstacle will give no g(v) and h(v) values, so the robot arm will not record these values. The second section is designed to prevent errors since there is no guarantee that no errors will occur. Therefore, the second section of the collision-detecting function will end the process when the robot touches the obstacle because of a bug. Moreover, the second function will end instantly for some exceptional conditions where there is no space for the robot arm to pass. These two types of ends can be seen as a failure during the experiment.

For each experiment, the robot arm will start at the same point. After running the program, MATLAB will generate all data sets, which are recorded and selected g(v), h(v), and f(v) values and the decided vector. Moreover, MATLAB will show the animation of the robot arm moving and its trajectory. Finally, to examine whether the A* algorithm works or not, the selected g(v), h(v), and f(v) will be evaluated.

3. Results

The recorded and selected g(v), h(v), and f(v) are listed in the next three pages below:

Table 1. The recordd and selected GHF values and the robot arm's x and y movement (Beginning)

G	H	F	G	H	F
1	**Recorded**		**4**	**Recorded**	
0.005657	13.4078335	13.41349	0.0056569	13.4078335	13.41349
0.004	13.3701159	13.37412	0.004	13.3701159	13.37412
0.005657	13.3330417	13.3387	0.0113137	13.2774244	13.28874
1	**Selected**		0.0096569	13.2400906	13.24975
0.005657	13.3330417	13.3387	0.0096569	13.2966161	13.30627
x-direction movement		-0.02	0.0169706	13.1472431	13.16421
y-direction movement		0.02	0.0153137	13.1103013	13.12562
2	**Recorded**		0.0153137	13.1673839	13.1827
0.005657	13.4078335	13.41349	0.0169706	13.2608446	13.27782
0.004	13.3701159	13.37412	0.0226274	13.0172962	13.03992
0.011314	13.2774244	13.28874	0.0209706	12.980755	13.00173
0.009657	13.2400906	13.24975	0.0226274	12.9448832	12.96751
0.011314	13.2034087	13.21472	0.0209706	13.0384048	13.05938
0.009657	13.2966161	13.30627	0.0226274	13.1320219	13.15465
2	**Selected**		**4**	**Selected**	
0.011314	13.2034087	13.21472	0.0226274	12.9448832	12.96751
x-direction movement		-0.02	x-direction movement		-0.02
y-direction movement		0.06	y-direction movement		0.14
3	**Recorded**		**5**	**Recorded**	
0.005657	13.4078335	13.41349	0.0056569	13.4078335	13.41349
0.004	13.3701159	13.37412	0.004	13.3701159	13.37412
0.011314	13.2774244	13.28874	0.0113137	13.2774244	13.28874
0.009657	13.2400906	13.24975	0.0096569	13.2400906	13.24975
0.009657	13.2966161	13.30627	0.0096569	13.2966161	13.30627
0.016971	13.1472431	13.16421	0.0169706	13.1472431	13.16421
0.015314	13.1103013	13.12562	0.0153137	13.1103013	13.12562
0.016971	13.07402	13.09099	0.0153137	13.1673839	13.1827
0.015314	13.1673839	13.1827	0.0169706	13.2608446	13.27782
0.016971	13.2608446	13.27782	0.0226274	13.0172962	13.03992
3	**Selected**		0.0209706	12.980755	13.00173
0.016971	13.07402	13.09099	0.0209706	13.0384048	13.05938
x-direction movement		-0.02	0.0226274	13.1320219	13.15465
y-direction movement		0.1	0.0282843	12.8875909	12.91588
			0.0266274	12.8514591	12.87809
			0.0282843	12.8160056	12.84429
			0.0266274	12.9096863	12.93631

0.0282843	13.0034611	13.03175
5	**Selected**	
0.0282843	12.8160056	12.84429
x-direction movement		-0.02
y-direction movement		0.18

Table 2. The recorded and selected GHF values and the robot arm's x and y movement (A few seconds after the beginning)

G	H	F	G	H	F
8	**Recorded**		0.020970	12.980755	13.0017
0.005656	13.40783	13.4134	0.020970	13.038404	13.0593
0.004	13.37011	13.3741	0.022627	13.132021	13.1546
0.011313	13.27742	13.2887	0.028284	12.887590	12.9158
0.009656	13.24009	13.2497	0.026627	12.851459	12.8780
0.009656	13.29661	13.3062	0.026627	12.909686	12.9363
0.016970	13.14724	13.1642	0.028284	13.003461	13.0317
0.015313	13.11030	13.1256	0.033941	12.758134	12.7920
0.015313	13.16738	13.1827	0.032284	12.722421	12.7547
0.016970	13.26084	13.2778	0.032284	12.781236	12.8135
0.022627	13.01729	13.0399	0.033941	12.875169	12.9091
0.020970	12.98075	13.0017	0.039598	12.628935	12.6685
0.020970	13.03840	13.0593	0.037941	12.593649	12.6315
0.022627	13.13202	13.1546	0.037941	12.653062	12.691
0.028284	12.88759	12.9158	0.039598	12.747156	12.7867
0.026627	12.85145	12.8780	0.045254	12.5	12.5452
0.026627	12.90968	12.9363	0.043598	12.465151	12.5087
0.028284	13.00346	13.0317	0.043598	12.525174	12.5687
0.033941	12.75813	12.7920	0.045254	12.619429	12.6646
0.032284	12.72242	12.7547	0.050911	12.371337	12.4222
0.032284	12.78123	12.8135	0.049254	12.336936	12.3861
0.033941	12.87516	12.9091	0.050911	12.303251	12.3541
0.039598	12.62893	12.6685	0.049254	12.397580	12.4468
0.037941	12.59364	12.6315	0.050911	12.491997	12.5429
0.037941	12.65306	12.691	**9**	**Selected**	
0.039598	12.74715	12.7867	0.050911	12.303251	12.3541
0.045254	12.5	12.5452	x-direction movement		-0.02
0.043598	12.46515	12.5087	y-direction movement		0.34
0.045254	12.43100	12.4762	**10**	**Recorded**	
0.043598	12.52517	12.5687	0.005656	13.407833	13.4134
0.045254	12.61942	12.6646	0.004	13.370115	13.3741
8	**Selected**		0.011313	13.277424	13.2887
0.045254	12.43100	12.4762	0.009656	13.240090	13.2497
x-direction movement		-0.02	0.009656	13.296616	13.3062
y-direction movement		0.3	0.016970	13.147243	13.1642
9	**Recorded**		0.015313	13.110301	13.1256
0.005656	13.40783	13.4134	0.015313	13.167383	13.1827
0.004	13.37011	13.3741	0.016970	13.260844	13.2778
0.011313	13.27742	13.2887	0.022627	13.017296	13.0399
0.009656	13.24009	13.2497	0.020970	12.980755	13.0017
0.009656	13.29661	13.3062	0.020970	13.038404	13.0593
0.016970	13.14724	13.1642	0.022627	13.132021	13.1546
0.015313	13.11030	13.1256	0.028284	12.887590	12.9158

0.015313	13.16738	13.1827	0.026627	12.851459	12.8780
0.016970	13.26084	13.2778	0.026627	12.909686	12.9363
0.022627	13.01729	13.0399	0.028284	13.003461	13.0317
0.020970	12.98075	13.0017	0.033941	12.758134	12.7920
			0.032284	12.722421	12.7547
			0.032284	12.781236	12.8135
			0.033941	12.875169	12.9091
			0.039598	12.628935	12.6685
			0.037941	12.593649	12.6315
			0.037941	12.653062	12.691
			0.039598	12.747156	12.7867
			0.045254	12.5	12.5452
			0.043598	12.465151	12.5087
			0.043598	12.525174	12.5687
			0.045254	12.619429	12.6646
			0.050911	12.371337	12.4222
			0.049254	12.336936	12.3861
			0.049254	12.397580	12.4468
			0.050911	12.491997	12.5429
			0.056568	12.242957	12.2995
			0.054911	12.209013	12.2639
			0.056568	12.175795	12.2323
			0.054911	12.270289	12.3252
			0.056568	12.364869	12.4214
			10	**Selected**	**0**
			0.056568	12.175795	12.2323
			x-direction movement		-0.02
			y-direction movement		0.38

Table 3. The recorded and selected GHF values and the robot arm's x and y movement (An instant point)

G	H	F	G	H	F	G	H	F
26	**Recorded**		0.0735391	11.8595953	11.9331344	0.1357645	10.6230881	10.7588526
0.00566	13.40783	13.41349	0.07188	11.82709	11.89897	0.14142	10.35809	10.49951
0.00400	13.37012	13.37412	0.07188	11.89033	11.96221	0.13976	10.33247	10.47224
0.01131	13.27742	13.28874	0.07354	11.98541	12.05895	0.13976	10.40481	10.54457
0.00966	13.24009	13.24975	0.07920	11.73243	11.81163	0.14142	10.50190	10.64333
0.00966	13.29662	13.30627	0.07754	11.70043	11.77797	0.14708	10.23572	10.38280
0.01697	13.14724	13.16421	0.07754	11.76435	11.84189	0.14542	10.21078	10.35620
0.01531	13.11030	13.12562	0.07920	11.85960	11.93879	0.14542	10.28397	10.42939
0.01531	13.16738	13.18270	0.08485	11.60560	11.69045	0.14708	10.38123	10.52831
0.01697	13.26084	13.27782	0.08320	11.57411	11.65730	0.15274	10.11385	10.26659
0.02263	13.01730	13.03992	0.08320	11.63873	11.72192	0.15108	10.08960	10.24068
0.02097	12.98075	13.00173	0.08485	11.73414	11.81899	0.15108	10.16366	10.31474
0.02097	13.03840	13.05938	0.09051	11.47911	11.56962	0.15274	10.26109	10.41383
0.02263	13.13202	13.15465	0.08885	11.44814	11.53700	0.15839	9.99250	10.15089
0.02828	12.88759	12.91588	0.08885	11.51347	11.60232	0.15674	9.96895	10.12569
0.02663	12.85146	12.87809	0.09051	11.60905	11.69956	0.15674	10.04390	10.20064
0.02663	12.90969	12.93631	0.09617	11.35297	11.44914	0.15839	10.14150	10.29989
0.02828	13.00346	13.03175	0.09451	11.32254	11.41705	0.16405	9.87168	10.03573
0.03394	12.75813	12.79208	0.09451	11.38859	11.48310	0.16239	9.84886	10.01125
0.03228	12.72242	12.75471	0.09617	11.48434	11.58050	0.16239	9.92472	10.08711
0.03228	12.78124	12.81352	0.10182	11.22720	11.32902	0.16405	10.02247	10.18652
0.03394	12.87517	12.90911	0.10017	11.19732	11.29749	0.16971	9.75141	9.92112
0.03960	12.62894	12.66853	0.10017	11.26410	11.36427	0.16805	9.72934	9.89739
0.03794	12.59365	12.63159	0.10182	11.36002	11.46184	0.16805	9.80612	9.97417
0.03794	12.65306	12.69100	0.10748	11.10180	11.20928	0.16971	9.90404	10.07375
0.03960	12.74716	12.78675	0.10582	11.07249	11.17831	0.17536	9.63172	9.80708

0.04525	12.50000	12.54525	0.10582	11.14002	11.24584	0.17371	9.61041	9.78412
0.04360	12.46515	12.50875	0.10748	11.23610	11.34358	0.17371	9.68814	9.86184
0.04360	12.52517	12.56877	0.11314	10.97679	11.08993	0.17536	9.78621	9.96158
0.04525	12.61943	12.66468	0.11148	10.94806	11.05954	0.18102	9.51262	9.69364
0.05091	12.37134	12.42225	0.11148	11.01635	11.12783	0.17936	9.49210	9.67146
0.04925	12.33694	12.38619	0.11314	11.11261	11.22574	0.17936	9.57079	9.75015
0.04925	12.39758	12.44684	0.11879	10.85219	10.97098	0.18102	9.66902	9.85004
0.05091	12.49200	12.54291	0.11714	10.82405	10.94118	0.18668	9.39415	9.58082
0.05657	12.24296	12.29953	0.11714	10.89312	11.01025	0.18502	9.37443	9.55945
0.05491	12.20901	12.26392	0.11879	10.98954	11.10833	0.18502	9.45410	9.63912
0.05491	12.27029	12.32520	0.12445	10.72800	10.85245	0.18668	9.55249	9.73916
0.05657	12.36487	12.42144	0.12279	10.70047	10.82326	0.19233	9.27631	9.46865
0.06223	12.11487	12.17709	0.12279	10.77033	10.89312	0.19068	9.25743	9.44811
0.06057	12.08139	12.14196	0.12445	10.86692	10.99137	0.19233	9.23959	9.43192
0.06057	12.14331	12.20388	0.13011	10.60424	10.73435	0.19068	9.33809	9.52877
0.06223	12.23806	12.30028	0.12845	10.57733	10.70579	0.19233	9.43663	9.62896
0.06788	11.98708	12.05496	0.12845	10.64800	10.77646	**26**	**Selected**	**0**
0.06623	11.95408	12.02030	0.13011	10.74477	10.87487	0.19233	9.23959	9.43192
0.06623	12.01666	12.08288	0.13576	10.48094	10.61670	x-direction movement		-0.02
0.06788	12.11156	12.17945	0.13411	10.45466	10.58877	y-direction movement		1.34

The data in the Table 1, Table 2, and Table 3 is the data measured and recorded by the function, and the robot arm's decision (x&y movement). These three tables also show how the robot arms move after selecting the lowest f value. Moreover, from the above three tables, it is evident that the number of sets of g(n), h(n), and f(n) values increases with running time. There are almost thousands of data recorded near the final point.

In order to find the robot arm's behavior, some data of sample points are listed in the table below:

Table 4. Selected GHF value of two instant point

Selected			Selected		
0.611009235	6.103277808	6.714287043	0.322842712	0.322842712	6.503076195
x-direction movement		0.02	x-direction movement		-0.023
y-direction movement		-0.02	y-direction movement		-0.023

MATLAB also generated some graphs showing the robot arm's trajectory and its position before and after an instant listed in Table 4, and the graphs are listed below and next page:

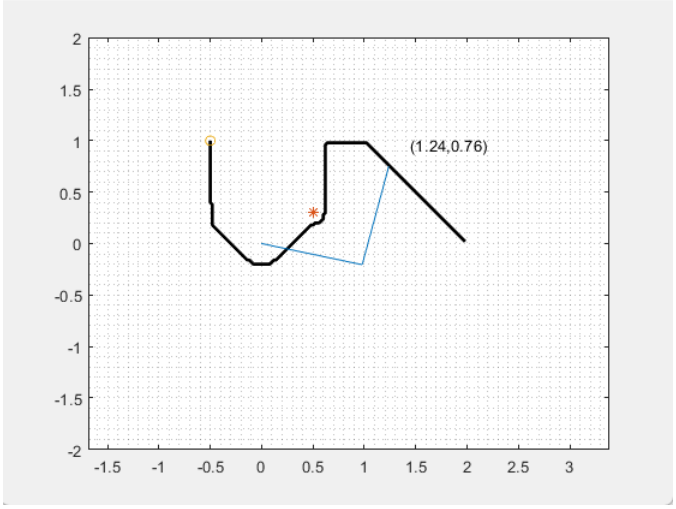


Figure 1. The robot arm's trajectory and its position before making decision (Sample 1)

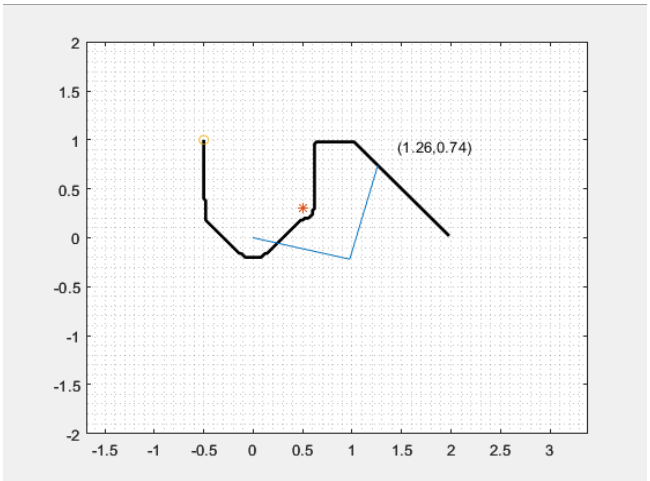


Figure 2. The robot arm's trajectory and its position after making decision (Sample 1)

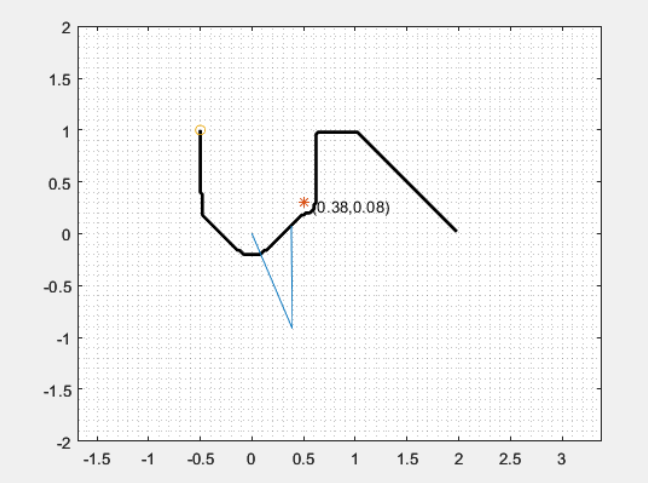


Figure 3. The robot arm's trajectory and its position before making decision (Sample 2)

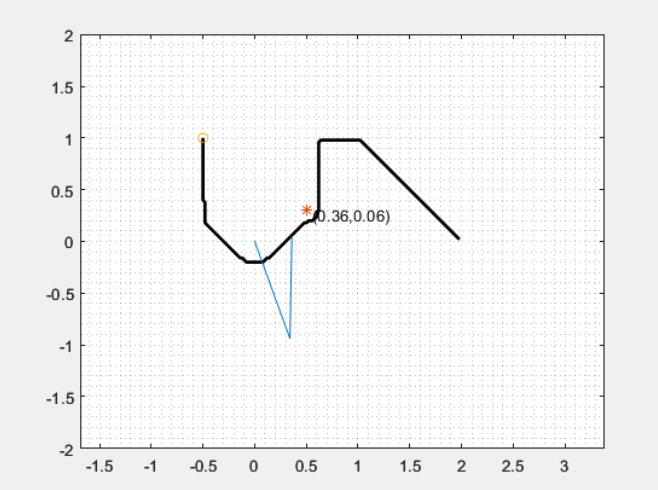


Figure 4. The robot arm's trajectory and its position after making decision (Sample 2)

Fig. 1, Fig. 2, Fig. 3, and Fig. 4 are captured in the same trajectory with different time. The position of the point (1.24, 0.76) after adding a vector (0.02, -0.02) is (1.26, 0.74), which is the same as the position Fig. 2 shows. Likewise, the position of the point (0.38, 0.08) after adding a vector (-0.023, -0.023) is (0.36, 0.06), which is the same as the position Fig. 4 shows. These two-point movements prove that the robot arm follows the correct direction.

Furthermore, the number of the robot arm that successful and failed to pass through an obstacle is listed in the table on the next page:

Table 5. Success rate

Number of test	Pass/Fail	Number of test	Pass/Fail	Number of test	Pass/Fail	Number of test	Pass/ Fail	Number of test	Pass/Fail	Number of test	Pass/Fail
1	P	11	**F**	21	P	31	P	41	**F**	51	P
2	P	12	P	22	P	32	P	42	P	52	P
3	P	13	P	23	P	33	P	43	P	53	P
4	P	14	**F**	22	P	34	P	44	**F**	52	P
5	P	15	P	25	**F**	35	P	45	P	55	**F**
6	**F**	16	P	26	**F**	36	P	46	P	56	P
7	**F**	17	P	27	P	37	**F**	47	P	57	**F**
8	P	18	P	28	P	38	P	48	P	58	P
9	P	19	P	29	P	39	P	49	P	59	P
10	P	20	P	30	P	40	P	50	P	60	P
RATE	0.816666667										

The experiments are repeated 60 times, and the bold font shows how many failures occurred during the project (Table 5).

4. Analysis

In order to get evaluate the A* algorithm work, the meaningless obstacle needs to eliminate first. Meaningless means the obstacle does not influence the robot arm; in this condition, neither the code work nor the robot arm takes action while meeting the obstacle. The meaningless trajectories and a good trajectory are listed below.

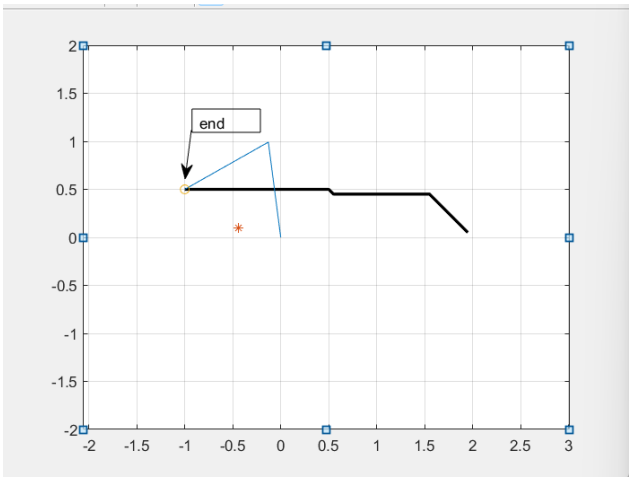


Figure 5. Meaningless obstacle and robot arm's trajectory

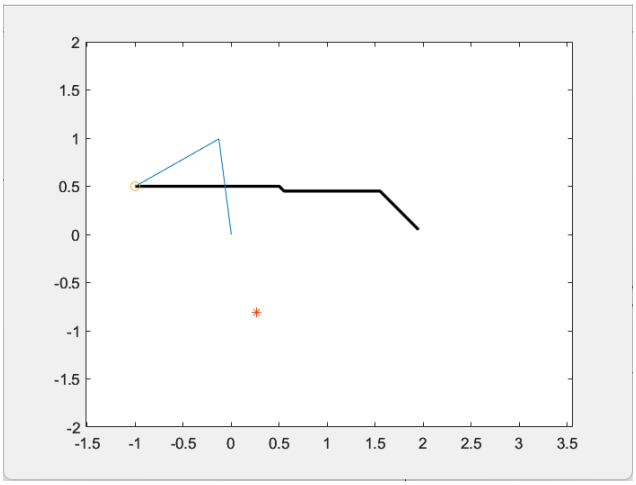


Figure 6. Meaningless obstacle and robot arm's trajectory (Evident one)

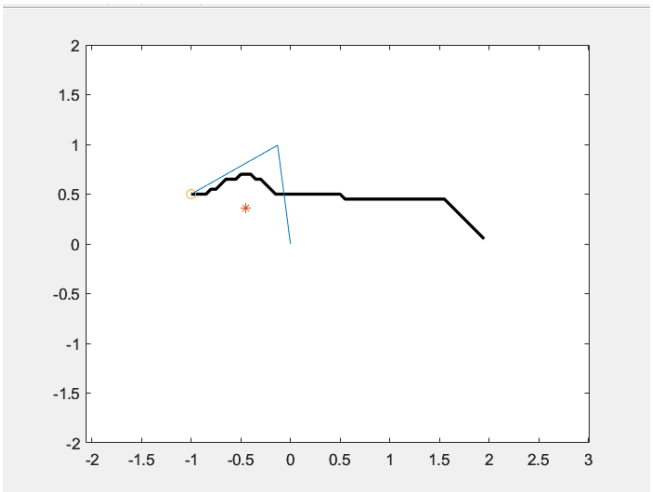


Figure 7. A good robot arm's trajectory

It may not show very clearly in Fig. 5, but it is more prominent in Fig. 6. The meaningless obstacle will always cause the trajectory to be the same. In order to eliminate meaningless obstacles, there is a solution to set a fixed start point and endpoint. When the robot arm moves between a fixed start point and end point, the shortest way would be the same every time, like the path in Fig. 5 and Fig. 6. However, in Fig. 7, a shaking path at the end of trajectory is observed, which differs from the previous two paths. Consequently, this finding supports that the robot arm takes action when it meets an obstacle.

Moreover, by manually analyzing the GHF values provided in Table 1, Table 2, and Table 3, the research group can confirm that the robot arm selects the correct GHF values, which is same as the smallest value sorted by Microsoft Excel, and selects the shortest path. Similarly, by analyzing Table 4 and Fig. 1, 2, and 3, the research team can confirm that the robot arm follows the direction derived after selecting the correct GHF value. In summary, it proves the A* algorithm works in this experiment.

The success rate is the times a robot arm successfully passes through an obstacle and selects the shortest path divided by the total times. By evaluating the success rate of 81.7%, which higher than 80%, it is believed that it can show the robot arm can pass through a single obstacle in 2D space to reach its destination and selects the shortest path.

5. Discussion

Before starting this experiment, it was believed that the trajectory curve would be a curve line, not a winding line shown in Fig.1, 2, 3, and 4. The explanation is that when people try to reach an object through an obstacle, they chose the most comfortable motion and trajectory without considering the distance. In contrast, the robot arms' behavior was strictly based on the A* algorithm, which means it chooses the shortest way. Therefore, it was believed the difference between these two lines was between the objective of focusing on human-like and focusing on efficient robot arms.

The A* algorithm can be improved by combing other algorithms and techniques or revising the standard A* algorithm. For example, a derived algorithm, combined jump-A* and dynamic window approach, shows better smoothness and turning point performance in pathfinding than the standard A* algorithm [10]. In their research, the research team applies not only a jump point search method but also a new distance evaluation function to optimize the standard A* algorithm [10]. These A*algorithm and its derivatives set the distance as their cost, and their mission is to find the shortest distance. In advance, some research treats angle or other properties as cost and optimizes the A*algorithm by lowering this cost. For example, Nam Kyu Kang's group let their A*algorithm evaluate the angle in addition to the distance [11]. To sum up, there are many ways to optimize the A* algorithm based on what people value. Therefore, it is significant to identify all the potential costs

when designing this algorithm. Furthermore, the simulation result can be more persuasive if a 3-D dimension environment and multiple obstacles are applied.

6. Conclusion

After analyzing the tables and data, it has been confirmed that the robot arm selects the correct GHF values and follows the correct direction derived after selecting the correct GHF value, which means the A* algorithm can work well on the robot arm. Furthermore, by evaluating the success rate of 81.7%, it is believed that it can be evidence that a 2-joint robot arm can pass through a single obstacle in 2D space to reach its destination and it selects the shortest path.

References

[1] J. Zhao, C. Wang, and B. Xie, "Human-like motion planning of robotic arms based on human arm motion patterns," Robotica, vol. 41, no. 1, 259–276, Sep. 2022.

[2] D. Wahyuningsih and E. Syahreza, "Shortest Path Search Futsal Field Location with Dijkstra Algorithm," IJCCS (Indonesian Journal of Computing and Cybernetics Systems), vol. 12, no. 2, 161.

[3] Y. Zhang and M. Chen, "Performance evaluation of Software-Defined Network (SDN) controllers using Dijkstra's algorithm," Wireless Networks, Aug. 2022.

[4] C. Frăsinaru and M. Răschip, "Greedy Best-First Search for the Optimal-Size Sorting Network Problem," Procedia Computer Science, vol. 159, 447–454, 2019.

[5] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment," IEEE Access, vol. 9, 59196–59210, 2021.

[6] A. Candra, M. A. Budiman, and R. I. Pohan, "Application of A-Star Algorithm on Pathfinding Game," Journal of Physics: Conference Series, vol. 1898, no. 1, 012047, Jun. 2021.

[7] P. O. N. Saian, Suyoto, and Pranowo, "Optimized A-Star algorithm in hexagon-based environment using parallel bidirectional search," 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE), Oct. 2016.

[8] A. Patel, "Introduction to A*", Stanford.edu, 2019. http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

[9] F. Duchoň et al., "Path Planning with Modified a Star Algorithm for a Mobile Robot," Procedia Engineering, vol. 96, 59–69, 2014.

[10] L. Liu et al., "Global Dynamic Path Planning Fusion Algorithm Combining Jump-A* Algorithm and Dynamic Window Approach," IEEE Access, vol. 9, 19632–19638, 2021.

[11] N. K. Kang, H. J. Son, and S.-H. Lee, "Modified A-star algorithm for modular plant land transportation," Journal of Mechanical Science and Technology, vol. 32, no. 12, 5563–5571, Dec. 2018.