On the Performance Analyses of a Modified Force Field Algorithm and Neural Network Approach for Obstacle Avoidance in Swarm Robotics

Girish Balasubramanian¹ · Senthil Arumugam Muthukumaraswamy¹ · Xianwen Kong²

Received: 24 December 2020 / Accepted: 10 March 2021 / Published online: 8 April 2021
© The Author(s) 2021

Abstract

Obstacle avoidance is a major hurdle when implementing mobile robots and swarm robots. Swarm robots work in groups and therefore require an efficient and functional obstacle avoidance algorithm to stay collision free between themselves and their surroundings. This paper reviews previous research in obstacle avoidance implementation using the force field method (FFM), also known as potential field method (PFM) and a neural network approach. Moreover, this paper aims to execute simulations using a modified force field algorithm and a neural network approach and compare them. The obtained results are analyzed to identify the performance characteristics and the time taken to perform tasks using a singular mobile robot against a swarm robot environment consisting of four and ten robots, respectively, in both simulation cases. Simulations showed that the algorithm was successful in navigating obstacles for both single and swarm robot environments. A single robot was found to take up to 340% longer to arrive at the required location compared to the first robot in the experiment. Moreover, it was found that the neural network approach showed ~27% improvement over the modified force field algorithm when it comes to cases where more than four robots are being used.

Keywords Swarm robotics · Proximity sensor · Obstacle avoidance · Force field algorithm · Performance · Multi-robot systems

Introduction

Researchers have developed multiple methods in tackling a major hurdle that is obstacle avoidance when it comes to development of various robots. Autonomous robots need to be able to navigate through and around obstacles to be able

to carry out tasks efficiently. This can be split into two tasks, path planning and obstacle avoidance [1, 2]. Path planning relies on global information and knowledge of the area that needs to be explored. Obstacle avoidance relies on local information in the present obtained through sensors. While path planning is important to minimize time taken to reach a destination, good obstacle avoidance is critical to ensure that the planned path can be carried out with few hindrances.

One of the most used algorithms in the real world for obstacle avoidance is the artificial potential field algorithm (APF) [3]. In this algorithm, T represents the target where the robot moves to and O represents the obstacles which the robot avoids. This is like how magnets apply attractive and repulsive forces. A combination of these forces around the object make up the path of the robot to reach its destination. One of the common problems with a generic force field algorithm is usually an oscillation problem in narrow spaces [3].

ANNs can be used to aid in obstacle avoidance for mobile robots. Neural networks can be used recognize environments surrounding robots and accordingly control speed to guide the robot to the required destination [17]. In this paper, two

This article is part of the topical collection “Soft Computing and its Engineering Applications” guest edited by Kanubhai K. Patel, Deepak Garg, Atul Patel and Pawan Lingras.

✉ Girish Balasubramanian
gb29@hw.ac.uk

Senthil Arumugam Muthukumaraswamy
m.senthilarumugam@hw.ac.uk

Xianwen Kong
X.Kong@hw.ac.uk

¹ Heriot-Watt University Dubai Campus, Dubai Knowledge Village, Dubai, UAE

² Heriot-Watt University, Campus The Avenue, Edinburgh EH14 4AS, UK

methods of procuring and processing the local sensor data to carry out obstacle avoidance are explored. The first method uses a modified force field algorithm. The second method uses a neural network to perform obstacle avoidance. Other algorithms for obstacle avoidance include gap method [4], particle swarm optimization [5], genetic algorithm [6], and fuzzy logic [7]. However, they were not considered as part of the scope of this paper.

A modified force field method (also called APF) and a neural network approach are evaluated in this paper. In terms of robot application for the modified APF algorithm, each proximity sensor is given a weight based on the trajectory at which the robot would face the obstacle on forward movement [8]. The obstacles are given repulsive forces, while the target destination is given an attractive force [9]. In the case of no destination, there are only repulsive forces present. The sum of the repulsive forces and the sensor readings aim to provide a sense of direction to the robot. Changes were made to the more commonly used force field method to better adapt for narrow spaces in this paper.

In the neural network approach, a multilayer feed-forward neural network is used. The network takes in sensor inputs and approximates a trajectory for the robot to follow to reach the end position [10, 11]. The neural network was trained with a dataset consisting of possible obstacle cases that could be encountered in a real-world environment.

This paper starts by reviewing existing work related to the force field method and artificial neural networks in the context of obstacle avoidance. Next, the experimental methods used to test the novel modified force field method and the artificial neural network method are presented. The results of the experiments are then discussed, and analysis is carried out. Finally, the paper gives closing thoughts and draws conclusions based on the obtained results.

Review of Previous Literature

Seyyed et al. proposed two modified versions of an artificial potential field (APF) algorithm [3]. The first algorithm proposed was called Bug1. In this algorithm, the robot completely bypasses an object and covers it. The weakness of the algorithm is its inefficiency. An example of the Bug1 algorithm pathing can be seen in Fig. 1.

Seyyed et al. managed to overcome this problem by introducing another modified algorithm called Bug2 [3]. In the Bug2 algorithm, the robot tries to maintain a path directly from start to end while circling any obstacles. When the robot can get back to the line path, the robot continues to move straight to the target location. An example of this pathing can be seen in Fig. 2. One of the major downsides of the APF algorithm includes the inability to handle scenarios in cases where the robot is unable to get back on path or if the robot is surrounded by obstacles. The robot is therefore

Fig. 1 Bug1 algorithm showcasing pathing of a single robot [3]

Fig. 2 Bug2 algorithm showcasing direct pathing of a single robot [3]

trapped. The proposed algorithm successfully tackled this problem and they were able to show simulation results for verification.

However, no practical testing was performed to validate the results. Moreover, both algorithms shown did not account for cases in which obstacles are entirely blocking the path to a target and assumes that there is a target location to begin with. This algorithm could be further tested in situations where a robot is set to free roam to look for certain objects in applications such as search and rescue. The algorithms shown also did not account for cases in which multiple robots are made to navigate an area at once. These algorithms could be further tested in a swarm robotics environment.

Anh et al. proposed a force field method that used rotational force fields and repulsive forces to better avoid more complex shapes [12]. Both clockwise and anti-clockwise forces were analyzed. Simulations were carried out for multiple robots as part of a swarm robot capable of taking either V-shaped or circular-shaped positions while avoiding obstacles and trying to maintain on course for the target. They had one robot in the swarm as a leader robot which was selected using an algorithm, as shown in Fig. 3.

The leader is set to determine the target tracking control algorithm function, while the members of the swarm follow the function. In cases where the leader gets trapped or cannot maintain shape, the leadership role is passed onto the nearest free member. The robots follow a V-shaped

Update data: The actual position of robots p_i ($i = 1, \dots, N, i \neq l$), obstacle's information, the target's position p_t , the actual position of the leader ($p_\xi = p_l$).

if time $t = 0$ (at initial time) then
 Compute the shortest distance from the robot p_i to the target p_t in order to determine the leader as

$$d_{i_1}^t = \min\{\|p_i - p_t\|, i = 1, \dots, N\}; p_l = p_{i_1}$$

else
if the current leader meets obstacle or is broken then
 the leadership is transferred to a member that is free and has the closest distance to the target **then**

$$d_{i_2}^t = \min\{\|p_i - p_t\|, i = 1, \dots, N, i \neq \xi, \text{free}\}$$

$$p_l = p_{i_2}$$

else
 Maintain the leadership of the current leader:

$$p_l = p_\xi.$$

end
end

Fig. 3 Leader Selection of Obstacle Avoidance Algorithm to determine target tracking control [12]

Fig. 4 **a** Swarm robots in a V-shape formation. **b** Swarm robots in a circular formation [12]

path until the tracked object trajectory is identified. The robots then assume a circular shape to surround the object while maintaining relative velocity. This can be seen in Figs. 4 and 5.

The algorithm successfully demonstrated the function. However, a few critical areas were missed in the work. The performance of this algorithm was not verified with respect to time. While moving objects were tracked by rotational means, many obstacles were of a static nature, and therefore, performance of this algorithm in a dynamic environment is unknown. Moreover, the requirement of a leader in the

algorithm meant that robots cannot adapt and carry out individual tasks if needed.

F Shamsfakr et al. proposed a neural network approach to obstacle avoidance in dynamic and unknown environments for a mobile robot. They used a multilayer feed-forward neural network to approximate an appropriate trajectory between the initial position and the target position [13]. They used two different neural networks to achieve better performance. There were two path planning strategies in their approach, this included normal action and wall-following action [14, 15]. In their approach, they were successfully able to showcase two scenarios of obstacle avoidance and path planning in both static and dynamic environments. However, their approach did not cover multi robot setups such as those that would be required in swarm robotics.

Experimental Methods

In this paper, the modified force field method of obstacle avoidance was evaluated in a single robot and swarm environments. The number of robots was varied between one, four, and ten depending on the experiment. The experiments were all carried out in Simulink using MATLAB and results were recorded in the form of time taken to achieve a certain goal. Each robot was made to have eight forward infrared proximity sensors. The experiment was carried out using the Robotics Toolbox differential drive algorithm provided by MathWorks [16]. The algorithms were modified to incorporate the modified force field algorithm.

The modified force field method assumed simplified weightings as opposed to existing methods. Depending on the expected type of navigation required, each sensor can be assigned a weight. For example, in the case of exploration in an open area with less narrow spaces, weights are assigned uniformly for all sensors. However, if the robots are expected to travel in narrow spaces, less weights can be given to side sensors to reduce oscillation. An example of a single robot with eight forward sensors is shown in Fig. 6.

In the case of the proposed force field algorithm, each sensor is given a weight from -1 to 1 for the direction of x and y based on the expected path. For the purposes of both evaluations, the weights were made the same and can be found in Table 1. The theory behind the selection of these weights are not covered in the scope of this paper.

The sum of the sensor readings multiplied by the contribution is then obtained for both x and y . Using these data, the velocity of the robot is calculated. This is given by:

$$\text{Sum_x} = \sum_{k=0}^7 |\text{SensorReading}(k_x)| \quad (1)$$

$$\text{Sum_y} = \sum_{k=0}^7 |\text{SensorReading}(k_y)|. \quad (2)$$

Fig. 5 Path followed by swarm robots using rotational force field algorithm [12]

Fig. 6 A swarm robot depiction with eight forward sensors

A random noise was also added to reduce oscillation in cases where the pathway around obstacles are too tight. This algorithm was designed in MATLAB and incorporated into the robotics toolbox to carry out the test. A flow-chart of the algorithm used can be found in Fig. 7.

For the first evaluation, an open area with multiple target points was selected. A single robot was made to move through the 2-D space until all checkpoints were identified and passed through. The path of the robot was tracked as well to draw other conclusions based on navigation. The purpose of this evaluation was to test the functionality of the tested algorithm and the divergence distance from the object (waypoint in this case). An example of the area used can be seen in Fig. 8.

The map consists of multiple checkpoints and the program records the time taken for the robot to cover all the

different checkpoints. The robot is considered successful once all checkpoints are reached.

In the second evaluation, 2-D maps were also used. However, this time, moving objects were introduced into the simulation to act as obstacles along with the robots themselves. The purpose of the second evaluation is to test how the algorithm performs in a dynamic environment in cases where one, four, or ten robots are used. The goal was for this experiment was to have a robot leave the map. The tests were also carried about for 500 iterations to normalize the effect of having a robot spawn too close to the exit and to get a more average result. The average reading of the results is then recorded. A 2-D map was created in which these robots were made to maneuver. This test environment can be seen in Fig. 8.

In this second evaluation, the robots were placed in random areas and moving objects were made to try and block the path. The timer stopped when any one robot was able to cross the exit out of the grid.

In the third evaluation, a multi-layered feed-forward neural network was used to determine the trajectory of the robots. Each robot uses an independent neural network which consists of inputs from each of the sensors as well as the location of another robot in the setup. The neural network used to determine the trajectory can be seen in Fig. 9.

R/x and R/y represent the Cartesian co-ordinates of a particular swarm robot in a 2-D plane with reference to one robot. This means one robot input will always be $[0,0]$,

Table 1 Sensor weights chosen for proposed modified force field method

Sensor directions	0	1	2	3	4	5	6	7
x	-1	-0.5	0	0.5	1	0.5	0	-0.5
y	0	0.5	1	0.5	0	-0.5	-1	-0.5

Fig. 7 Algorithm showcasing modified force field algorithm applied robot simulation for the purpose of evaluation 1

Fig. 8 2-D environment for robot travel created in MATLAB for evaluation 1

but the other robots will have different co-ordinates that change based on the relative position. Sensors one through eight represent the eight sensors present on a robot. It was designed specifically for a swarm robot consisting of four robots. This can be scaled to n number of robots as needed. Back propagation was done to carry out training of the neural network. The data set used was obtained by carrying out simulations of the FFM and obtaining the various turn angles. The obstacle avoidance algorithm was then implemented in MATLAB to carry out simulations to find the efficacy of the algorithm.

Fig. 9 Multilayered feed-forward neural network used to approximate the trajectory of the robots based on the location of the other robots and the sensor data

The neural network was then applied to the robots used for evaluations 1 and 2 to obtain simulation results for the time taken to reach the exit target similarly to how it was carried out for evaluation 2. 2-D maps were used like in Fig. 10. The purpose of the third evaluation is to identify how well the neural network approach performs in comparison to the modified force field algorithm. This evaluation was also carried out for one, four, and ten robots, respectively. The tests were also carried about for 500 iterations to normalize the effect of having a robot spawn too close to the exit and to get a more average result. The average reading of the results is then recorded.

Results and Analysis

In the first experiment, the modified force field algorithm was simulated to run in an open area and was tasked with crossing four checkpoints. This was performed for 100 iterations to obtain more reliable results. An example of the result for the force field method experiment with respect to pathing can be seen in Fig. 11.

The divergence from each waypoint after the first was recorded to evaluate functionality, and this can be seen in Fig. 12.

It was observed that the simplified modified force field method had a bigger divergence from the waypoint path following a much looser restriction. However, due to this nature of the algorithm, this was expected. Moreover, the robot was successfully able to navigate around the test environment proving the functionality of this algorithm.

In the second experiment, the modified force field algorithm was run in an open area with multiple robots and tasked to find the exit. This was done to evaluate how different number of robots perform in a dynamic environment. An example of the simulation results of this experiment being run with four robots can be seen in Fig. 13.

Fig. 10 2-D environment for robot travel created in MATLAB for evaluation 2

Fig. 11 Pathing of robot using force field method in Evaluation 1

The performance with respect to time taken was also recorded to identify the time taken for every robot to exit via the red goal. This result over 500 iterations was averaged out and recorded in Fig. 14 for three cases where one, four, and ten swarm robots were navigating the space.

The red line crossing through the moving objects was added as a guide for clarity to determine the moving path. The red crosses serve as guides for the robots to reach the destination. For example, in Fig. 13, robot 3 can be seen successfully leaving the area.

From analysis of Fig. 14, one swarm robot takes the most time to exit with 89.6 s compared to adding multiple robots into the same experiment. This was the expected result since random placements onto the escape grid and lack of robot interaction reduces the ability of the robot to

Fig. 12 Average divergence observed from robot to waypoint over 100 iterations

Fig. 13 Pathing of four robots using force field method in Evaluation 2

Fig. 14 Average time taken over 500 iterations for each robot to exit space

navigate the space. In contrast, the experiment with four and ten swarm robots take 26.4 and 29.4 s, respectively. This was again expected as adding more robots provide a multi-robot environment in where the robots can interact with each other and exit the environment. However, in the ten-robot scenario, the initial robot took longer than for four robots which was inconsistent with the expectation. This can be attributed to the size of the experiment space; an observation was made where the correct number of swarm robots needs to be selected depending on the available space for the experiment.

The results obtained from the third evaluation also included the identification of the time taken to exit the red goal, as shown in Fig. 13. These results are, however, shown for when the multi-layered feed-forward neural network is applied to the robots. The average time taken over 500 iterations for each robot to exit the space can be seen in Fig. 15.

Fig. 15 Average time taken over 500 iterations for each robot to exit space using the neural network approach

From analysis of Figs. 14 and 15, the neural network approach can identify the exit and path more efficiently as compared to the modified force field algorithm. This was expected, since the neural network is trained with many scenarios so can optimize the pathing on the fly. Whereas the modified force field algorithm uses a set of specified weights. The difference for one robot was 89.6 s for the modified force field algorithm and 76.2 s for the neural network approach which is an improvement of ~15%. The difference is even more noticeable when you add more robots. The time taken for the first exit with ten robots in the modified force field algorithm was found to be 29.4 s. However, the time taken was found to be 18.5 s for the neural network approach. This is an improvement of ~58%. The improvement diminishes as the robots that exit move up. However, it was found that the overall aggregate improvement was approximately ~27%.

Conclusion

In this paper, the performance and functionality of a modified yet simplified force field algorithm and a neural network approach for obstacle avoidance was analyzed. The experiments and simulations used infrared sensors and tested three scenarios where there were one, four, or ten robots present at a given time. The algorithms were successful in navigating an open environment and a dynamic environment in the MATLAB simulation environment.

In open areas, the algorithm with force field methods was shown to have a moderate divergence (0.23–0.44 cm) which could be improved with a different sensor type. In more dynamic areas, the infrared algorithm with force field methods was shown to be slower by about 340% over 500

iterations with only one robot due to hindrances caused by obstacles and the lack of robots to interact. However, the swarm robotics environments showcased much faster results (26.4–29.4 s) compared to a single robot which was in line with what was expected. Moreover, the neural network approach showed an even further improvement in performance as compared to the modified force field algorithm. The neural network approach showed an improvement by approximately ~27%.

Funding This study was self-funded by the corresponding author (Girish Balasubramanian).

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Mane SB, Vhanale S. Genetic algorithm approach for obstacle avoidance and path optimization of mobile robot. In: Iyer B, Nalbalwar SL, Pathak NP, editors. Computing, communication and signal processing. Singapore: Springer; 2019. p. 649–59.
- Chan SH, Xu X, Wu PT, Chiang ML, Fu LC. Real-time obstacle avoidance using supervised recurrent neural network with automatic data collection and labeling. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). IEEE. 2019. (pp. 472–477).
- Sangaiah AK, Wang J, Liu X. Obstacle avoidance of mobile robots using modified artificial potential field algorithm. EURASIP J Wireless Com Network. 2019;2019(1):70.
- Sezer V, Gokasan M. A novel obstacle avoidance algorithm: follow the gap method. Rob Auton Syst. 2012;60(9):1123–34.
- Chen X, Li Y. Smooth path planning of a mobile robot using stochastic particle swarm optimization. In: 2006 IEEE International Conference on Mechatronics and Automation, IEEE. 2006. (pp. 1722–1727).
- Tu J, Yang SX. Genetic algorithm based path planning for a mobile robot. In: 2003 IEEE International Conference on Robotics and Automation ICRA'03. vol. 1. IEEE. 2003. (pp. 1221–1226).
- Reignier P. Fuzzy logic techniques for mobile robot obstacle avoidance. Rob Auton Syst. 1994;12(3–4):143–53.
- Jiang Y, Yang C, Ju Z, Liu J. Obstacle avoidance of a redundant robot using virtual force field and null space projection. In: Haibin Yu, Liu J, Liu L, Zhaojie Ju, Liu Y, Zhou D, editors. International Conference on Intelligent Robotics and Applications. Cham: Springer; 2019. p. 728–39.
- Borenstein J, Koren Y. Real-time obstacle avoidance for fast mobile robots in cluttered environments. In: Proceedings of the 1990 IEEE International Conference on Robotics and Automation. IEEE. 1990. (pp. 572–577).
- Peng Y, Qu D, Zhong Y, Xie S, Luo J, Gu J. The obstacle detection and obstacle avoidance algorithm based on 2-d lidar. In: 2015 IEEE international conference on information and automation. IEEE. 2015. (pp. 1648–1653).
- Singh NH, Thongam K. Neural network-based approaches for mobile robot navigation in static and moving obstacles environments. Intel Serv Robot. 2019;12(1):55–67.
- Dang AD, La HM, Nguyen T, Horn J. Formation control for autonomous robots with collision and obstacle avoidance using a rotational and repulsive force-based approach. Int J Adv Rob Syst. 2019;16(3):1729881419847897.
- Shamsfakhr F, Bigham BS. A neural network approach to navigation of a mobile robot and obstacle avoidance in dynamic and unknown environments. Turk J Elec Eng Comp Sci. 2017;25(3):1629–42.
- Ghorpade D, Thakare AD, Doiphode S. Obstacle detection and avoidance algorithm for autonomous mobile robot using 2D LiDAR. In: 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA). Pune; 2017. (pp. 1–6).
- Hutabarat D, Rivai M, Purwanto D, Hutomo H. Lidar-based Obstacle Avoidance for the Autonomous Mobile Robot. In: 2019 12th International Conference on Information & Communication Technology and System (ICTS). IEEE. 2019. (pp. 197–202).
- Mathworks.com. (2018). Mobile robotics simulation toolbox—file exchange—MATLAB Central. [online] <https://www.mathworks.com/matlabcentral/fileexchange/66586-mobile-robotics-simulation-toolbox>. Accessed 8 Dec 2018.
- Duguleana M, Mogan G. Neural networks based reinforcement learning for mobile robots obstacle avoidance. Expert Syst Appl. 2016;62:104–15.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.