An Algorithm for Swarm Robot to Avoid Multiple Dynamic Obstacles and to Catch the Moving Target

Quoc Bao Diep[1], Ivan Zelinka[1(✉)], and Roman Senkerik[2]

[1] Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava, 17. listopadu 15, Ostrava, Czech Republic
diepquocbao@gmail.com, ivan.zelinka@vsb.cz
[2] Department of Informatics and Artificial Intelligence, Faculty of Applied Informatics, Tomas Bata University in Zlín, Nad Stráněmi 4511, 76005 Zlín, Czech Republic
senkerik@utb.cz

Abstract. This paper presents a method for swarm robot to catch the moving target and to avoid multiple dynamic obstacles in the unknown environment. An imaginary map is built, including the highest mountain, some small hills, and a lowest lying land, respectively corresponding to the starting position of the robot, the detected obstacles, and the target. The robot is considered as a flow of water flowing from high to low. The flow of water is the robot trajectory that is divided into a set of points created by an algorithm called Self-organizing migrating algorithm. Simulation results are also presented to show that the obstacle avoidance and catching target task can be reached using this method.

Keywords: Self-organizing migrating algorithm · Obstacle avoidance · Swarm robot · Path planning

1 Introduction

One of the most important issues for swarm robotics applications is catching up with moving targets and avoiding multiple dynamic obstacles. It's complicated in that it requires an algorithm to work in real time to avoid obstacles that are standing or moving in an unknown environment where the robot does not know their position until detecting them by sensors arranged on the robot.

Besides the long-standing methods such as potential field method [8], and the vector field histogram [1], several new methods such as follow the gap method [10], and barrier function [2], or artificial intelligence methods such as genetic algorithm [7], and neural network [6] also demonstrate their effectiveness. Among the methods of artificial intelligence used to solve the problem as a function optimization problem, the self-organizing migrating algorithm (SOMA) emerges as a fast, powerful and efficient algorithm [4,12,13].

L. Rutkowski et al. (Eds.): ICAISC 2019, LNAI 11509, pp. 666–675, 2019.
https://doi.org/10.1007/978-3-030-20915-5_59

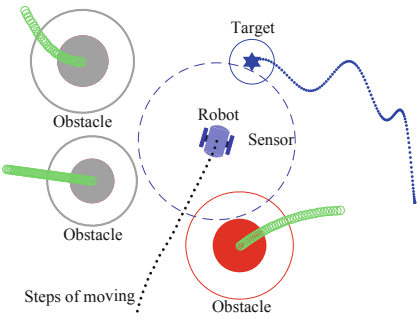

Fig. 1. The robot model and obstacles.

In this paper, the authors propose a method to guide the robot to catch the moving target without colliding with any dynamic obstacles based on the construction of an imaginary map and application of the SOMA.

2 Techniques

2.1 The Principle

The final goal of the robot is catching the moving target without colliding with any dynamic obstacles in the unknown environment. The assumptions outlined below ensure that the robot can work well under certain circumstances to ensure that the goal is achieved.

- *Robot*: For simplicity, all the physical dimensions of the robot are enclosed by a circle of radius r_{robot}. The sensors on the robot can accurately measure the distance to the obstacles within the sensors range, see Fig. 1. The robot is offered about the position of the target and can be controlled to reach a furthest desired position called moving step, i.e., the robot moves from the current position to a given furthest position without any difficulty. Moving step is a given parameter depending on the physical structure of the robot.
- *Obstacles*: Obstacles are considered as circles of radius r_{obs}. The obstacles can move at any speed and the robot does not know about the obstacles position until they are detected by the sensors.
- *Velocity*: The velocity of the robot must be greater than the velocity of each obstacle because if the opposite happens, the robot will not be able to avoid these dynamic obstacles. Similarly, to catch the target, the robot must move with the velocity greater than the target's velocity.

A method that can be visualized as *high mountain flowing water* is proposed to solve the issue. In this method, an imaginary map is built, in which the starting position of the robot considered as the top of the high mountain, the target considered as the lowest lying land, the obstacles considered as small hills,

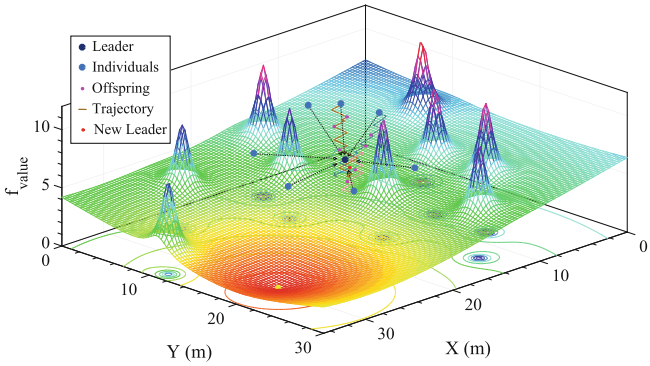

Fig. 2. The imaginary map.

and the robot considered as water, flowing from the top of the mountain to the lowest lying land, flowing around small hills without flowing back into them as natural law, see Fig. 2. When the target and obstacles move, the small hills and the lowest lying land also move respectively.

To build this map, a mathematical model will be shown in the next subsection.

2.2 The Imaginary Map

The starting position of the robot, the moving target, and the dynamic obstacles are three components that build the map, respectively corresponding to the top of the mountain, the lowest lying land, and the small hills. The lowest lying land and the small hills are not fixed things, but they will move based on the location of the target and the detected obstacles with the size of small hills depending on the distance and the size of obstacles.

The map is constructed using (1). Based on the idea presented above, the equation will contain two components: a sunken zone of the target created by the first component, depending on the distance (location) of the target that robot known; a raised zone of obstacles created by the second component, depending on the size and location of the obstacle.

$$f_{value} = \alpha \; dis_{tar} + \sum_{n=0}^{num_{obs}} \frac{c + r_{obs}}{b + dis_{obs}^2} \tag{1}$$

where:

- f_{value}: the fitness value,
- α: the sunken coefficient of the target,
- b: the influential coefficient of obstacle,
- c: the dynamic influential coefficient of obstacle,
- num_{obs}: the number of obstacles,

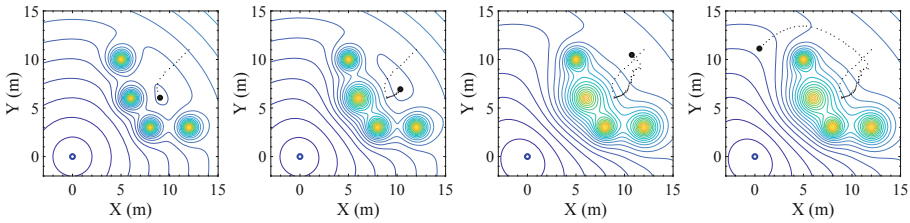

Fig. 3. The robot is trapped between obstacles.

- r_{obs}: the radius of obstacle,
- dis_{tar}: the distance from robot to target,
- dis_{obs}: the distance from robot to each detected obstacle.

The obstacles avoidance problem and catching target have now become an optimization problem, in which the optimal value is the fitness value in (1). At each processing times, SOMA will create a next point that robot must pass through. Two successive points are called moving step of robot, in which the distance from the current point to the next point must be less than or equal to the moving step, predetermined based on the physical structure of robot, see Sect. 2.1.

In case the robot is held by surrounding obstacles and the gaps between the obstacles is not enough for the robot to pass as shown in Fig. 3, the coefficient c will be changed to move the robot away from the hold. This coefficient plays the same role as the coefficient c in [5].

The robot trajectory is considered as a set of points created by SOMA that are presented in the next section.

3 Self-organizing Migrating Algorithm

SOMA was inspired by the competition-cooperation behavior of a group of intelligent insects while looking for food [12,13]. This algorithm has been applied to solve many different problems such as [9,11]. In the obstacles avoidance problem, SOMA acts as an algorithm that creates the next point for the robot move to. In other words, each individual in a population of SOMA is an imaginary point on the map.

At the start of the algorithm, a population containing all individuals is initialized around the current position of the robot based on (2). The population is then evaluated by the fitness function (1). A leader that contains the best fitness value is chosen for the current migration loop.

$$Pos_i = Pos_{actual} + rand_{-1 \to 1}\, Pos_{max} \tag{2}$$

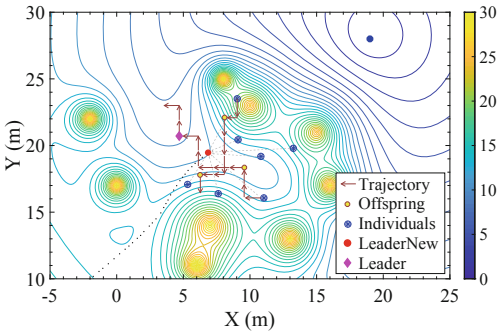

Fig. 4. The principle of SOMA.

where:

- Pos_i: the position of the i^{th} individual,
- Pos_{actual}: the actual position of the robot,
- Pos_{max}: the maximum position,
- $rand_{-1 \to 1}$: random number from -1 to 1.

Then, the mutation process takes place in preparation for the perturbation movement of individuals [12,13]. A random number created in $[0, 1]$ is compared with a given number of SOMA (called PRT) to perform the perturbation movement, see (3). Different from other evolutionary algorithms, the mutation process is performed before the crossover process. In order words, the $PRTVector$ is created and changed at each individual jump, before an individual performs its movement to ensure diversity of the population and increase the efficiency of the algorithm. Therefore, the individual trajectory is not straight, but the ladder, see Figs. 2 and 4.

$$if\ rand < PRT;\ PRTVector = 1;\quad else,\ 0. \tag{3}$$

The offspring in SOMA are generated by the crossover process to create a set of new individuals by moving the parent individual perturbatively [3,12,13]. In other words, parents move towards the leader in trajectories created earlier by $PRTVector$ in the mutation process, see (4).

$$P_{os}^{new} = P_{current} + (P_{leader} - P_{current})\ t\ PRTVector \tag{4}$$

where:

- P_{os}^{new}: the offspring position in new migration loop,
- $P_{current}$: the offspring position in current migration loop,
- P_{leader}: the leader position in current migration loop,
- t: jumping step, from 0 to $PathLength$.

Finally, the new leader is chosen not only to have the lowest fitness value but also to have the distance between the actual position of the robot and new leader must be smaller than the moving step given before.

4 Simulation Results

4.1 Setup

Matlab has been used to implement the proposed algorithm with the environment presented below.

- The robot used in the simulation was placed at the position $(1, 1)$m, with a radius $r_{robot} = 0.8$ m. The sensors can detect obstacles within a radius $r_{sensor} = 3.5$ m. The moving step of the robot $r_{movingstep} = 0.3$ m.
- The size and the initial positions of the obstacles are given in Table 1. In this simulation, the obstacles were placed in 9 different positions, including three standing obstacles (from 7^{th} to 9^{th}), six remaining dynamic obstacles moving in different trajectory and given in Table 2.
- The trajectory of the target is given in Table 2.
- The parameters of SOMA are given in Table 3.

Table 1. The initial position and radius of obstacles

$Obstacle_i$	1	2	3	4	5	6	7	8	9
x_i (m)	03	17	08	34	15	15	08	25	13
y_i (m)	10	08	22	10	03	17	16	04	26
r_i (m)	2.1	2.4	2.0	2.3	2.0	1.8	2.2	2.5	2.3

Table 2. The trajectory of obstacles and the target

$Obstacle_i$	$x(m)$	$y(m)$
1	$2sin(t/13)^2 + t/20$	$-t/10$
2	$t/22$	$sin(t/19) + t/11$
3	$t/15$	$-t/110$
4	$-sin(t/19) - t/11$	$cos(t/20)$
5	$-t/13$	$-sin(t/13) + t/11$
6	$t/60$	$0t$
$Target$	$-t/900$	$0.25sin(t/13)^2 - t/750$

Table 3. The parameters of SOMA

PopSize	Migration	PathLength	PRT	Step
40	10	2.7	0.1	0.11

4.2 Results

Figure 5 shows the simulation results. The dark color circles represent the dynamic obstacles, numbered 1 through 9 respectively, and they turn red to indicate that they are detected by the sensor of the robot. The black and blue dotted line show the trajectories of the robot and the target respectively. The green circle dotted line shows the trajectory of the obstacles. The figures were captured at 15^{th}, 30^{th}, 50^{th}, 69^{th}, 100^{th} and 123^{rd} s.

At the beginning of time, no obstacles were detected, so in the map, there was only an element of the sunken zone of the target. At the 15^{th} s, the first obstacle

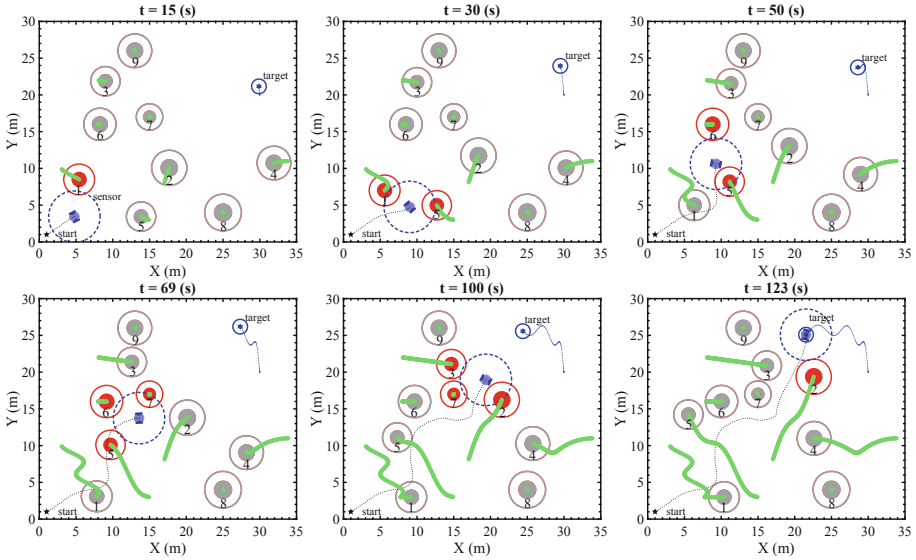

Fig. 5. The moving process of robot at different times. (Color figure online)

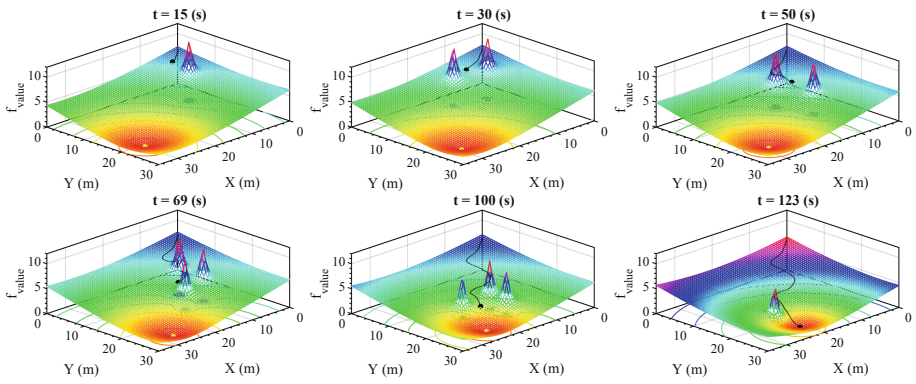

Fig. 6. The detailed trajectory of the robot in 3D map.

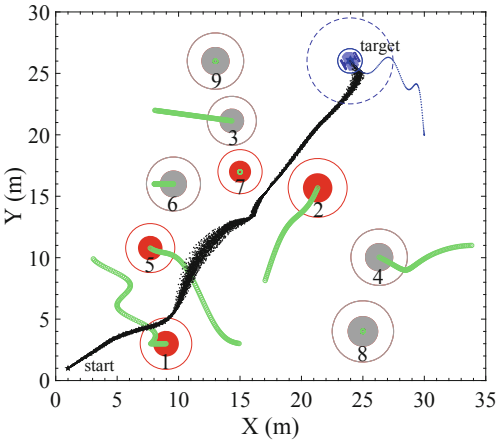

Fig. 7. Simulations were repeated 100 times.

was detected, a small hill appeared on the map (see Fig. 6), which influenced the trajectory of robot. At 30^{th} s, two obstacles were detected, the 1^{st} and the 5^{th}, so two small hills appeared in the map. Since the gap between the two obstacles is wide enough, the robot moves through the gap to catch the target, as can be seen in Fig. 6. This process continues until the robot catches the moving target. Since the speed of the robot is greater than the speed of the obstacles (assumed in Sect. 2.1), the robot can completely avoid them. In case the robot is trapped by surrounding obstacles, the coefficient c will be changed, and the size of the small hill will become larger, leading the robot away from the hold, as shown in Fig. 3 and [5].

In this simulation, it took 123 s for the robot to reach the target. During the movement of the robot, the 4^{th}, 8^{th}, and 9^{th} obstacles were undetected, in other words, they did not affect the movement of the robot, not appear on the map, although they still existed in reality.

Figure 6 shows the entire process of moving the robot in the form of a 3D map at the same time as shown in Fig. 5. The small hills rising upwards indicate that obstacles have been detected. The black dot is the position of the robot, the yellow dot is the position of the target, and it moves with the trajectory given in Table 2, the sunken zone also changes accordingly. In this map, the robot as a stream flows from high to low, moving round the small hills. When the hills change their position and size, the water is pushed to a lower land, as natural law. It means that the robot always tends to move to the target and not move into the obstacles.

To ensure the stability of the algorithm, the simulation was repeated 100 times, in which, the moving step of the robot changed to $r_{movingstep} = 0.4$ m, the parameters of simulation were preserved.

Figure 7 shows the trajectories of the robot, target, and obstacles after 100 simulation times. The black dotted line demonstrates the stability of the algo-

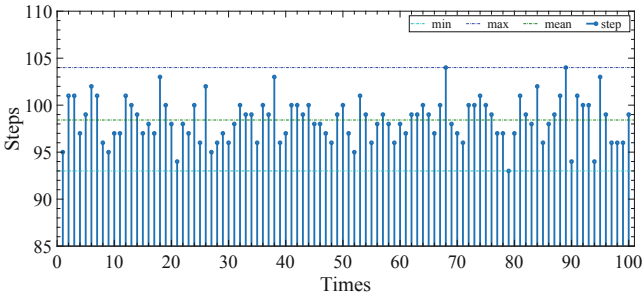

Fig. 8. Number of moving steps in 100 simulations.

rithm. Figure 8 shows the number of moving step that the robot needs to reach the target. The minimum number is 93 steps at the 79^{th} simulation, maximum number is 104 steps at the 68^{th} and the 89^{th} simulation.

5 Conclusion

The paper has solved the whole problem of catching the moving targets and avoiding multiple dynamic obstacles. The paper also successfully built a 3D map which depicts the robot, target, obstacles corresponding to the top of the mountains, small hills, and the lowest lying land. The trajectory of the robot was divided into a set of points, and the SOMA algorithm was proposed to find a reasonable trajectory for the robot. The paper also suggested ways leaving the robot moving away from the trapped area when the robot be held by the surrounding obstacles. The simulation results have been shown to demonstrate that the method is effective.

Acknowledgment. The following grants are acknowledged for the financial support provided for this research: Grant of SGS 2019/137, VSB-Technical University of Ostrava. The Author Roman Senkerik would like to acknowledge the support of the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014), further the European Regional Development Fund under the Project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089.

References

1. Borenstein, J., Koren, Y.: The vector field histogram-fast obstacle avoidance for mobile robots. IEEE Trans. Robot. Autom. **7**(3), 278–288 (1991)
2. Chen, Y., Peng, H., Grizzle, J.: Obstacle avoidance for low-speed autonomous vehicles with barrier function. IEEE Trans. Control Syst. Technol. **99**, 1–13 (2017)
3. Davendra, D., Zelinka, I.: Optimization of quadratic assignment problem using self organising migrating algorithm. Comput. Inform. **28**(2), 169–180 (2012)

4. Davendra, D., Zelinka, I., et al.: Self-organizing migrating algorithm. New Optimization Techniques in Engineering (2016)
5. Diep, Q.B., Zelinka, I.: Obstacle avoidance for swarm robot based on self-organizing migrating algorithm. In: XIIIth International Symposium Intelligent Systems, INTELS 2008, St. Petersburg, Russia (2018, accepted, in print)
6. Duguleana, M., Mogan, G.: Neural networks based reinforcement learning for mobile robots obstacle avoidance. Expert Syst. Appl. **62**, 104–115 (2016)
7. Hu, Y., Yang, S.X.: A knowledge based genetic algorithm for path planning of a mobile robot. In: 2004 Proceedings of IEEE International Conference on Robotics and Automation, ICRA 2004, vol. 5, pp. 4350–4355. IEEE (2004)
8. Koren, Y., Borenstein, J.: Potential field methods and their inherent limitations for mobile robot navigation. In: 1991 Proceedings of IEEE International Conference on Robotics and Automation, pp. 1398–1404. IEEE (1991)
9. Senkerik, R., Zelinka, I., Davendra, D., Oplatkova, Z.: Utilization of SOMA and differential evolution for robust stabilization of chaotic logistic equation. Comput. Math. Appl. **60**(4), 1026–1037 (2010)
10. Sezer, V., Gokasan, M.: A novel obstacle avoidance algorithm: "follow the gap method". Robot. Auton. Syst. **60**(9), 1123–1134 (2012)
11. Tomaszek, L., Zelinka, I.: Analysis of SOMA algorithm using complex network. In: Zelinka, I., Chen, G. (eds.) Evolutionary Algorithms, Swarm Dynamics and Complex Networks. ECC, vol. 26, pp. 115–129. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-55663-4_5
12. Zelinka, I.: SOMA-self-organizing migrating algorithm. New Optimization Techniques in Engineering. Studies in Fuzziness and Soft Computing, vol. 141, pp. 167–217. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-39930-8_7
13. Zelinka, I.: SOMA—self-organizing migrating algorithm. In: Davendra, D., Zelinka, I. (eds.) Self-Organizing Migrating Algorithm. SCI, vol. 626, pp. 3–49. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-28161-2_1