Path Planning using Deep Q-learning Network and Artificial Potential Fields for a Robot Formation

by

Yang Yang
Master of Engineering, University of Victoria, 2022

A Project Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Mechanical Engineering, University of Victoria

Yang Yang, 2022
University of Victoria

All rights reserved. This thesis may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.

Supervisory Committee

Path Planning using Deep Q-learning Network and Artificial Potential Fields for a Robot Formation

by

Yang Yang
Master of Engineering, University of Victoria, 2022

Supervisory Committee

Dr. Daniela Constantinescu, Professor, Mechanical Engineering
Supervisor

Dr. Caterina Valeo, Professor, Mechanical Engineering
Departmental Member

Abstract

A common path planning in robotics is the artificial potential field method. The artificial potential field is the superposition of the attractive potential field generated by the target and the repulsive potential field generated by the obstacles. The total force on a robot moving in the artificial potential field is the sum of the attractive force from the attractive field and the repulsive force from the repulsive potential field. The robot then moves in the direction of the total force, whose direction is along the negative gradient of the artificial potential field. If the artificial potential field has a unique minimum at the target, the robot will reach it without hitting obstacles. However, if the artificial potential field has multiple minima, the robot may arrive at a location with locally minimum potential. The total force on the robot is zero at such a local minimum and drives the robot towards it in its neighbourhood. Therefore, the robot becomes stationary and cannot arrive at the target.

Deep Q-learning network has been proposed to overcome the local minima problem of robot path planning based on artificial potential field. This project investigates the impact of combining deep Q-learning network with an artificial potential field, as proposed in [1], to achieve path planning for a robot formation. Specifically, it uses a deep Q-learning network to guide the robot formation to the target in an artificial potential field created by an environment with multiple targets and obstacles. Deep Q-learning network is a type of deep reinforcement learning, that is, it combines reinforcement learning and deep learning. As in [2], a black-hole potential field is also added in the artificial potential field. Simulation results show that deep Q-learning network can good results in the fixed artificial potential field. Out of 40 tests, a robot reaches the target without hitting any obstacles in 37 tests. However, the deep Q-learning network does not improve path planning performance in different artificial potential fields. During training in four different artificial potential fields, the robot can not achieve collision-free path planning in two of them. Out of four tests, the robot achieves collision-free path planning in three tests. Similarly, the robot formation successfully finds the target in the fixed artificial potential field. Out of eight tests, all tests are successful.

Table of Contents

Supervisory Committee ii
Abstract iii
Table of Contents iv
List of Tables v
List of Figures vi
1. Introduction 1
1.1 Background 1
1.2 Objective 2
1.3 Approach 2
1.4 Organization of the Report 3
2. Literature Review 4
3. Artificial Potential Field for Robot Path Planning 6
3.1 Attractive Potential Field 6
3.2 Repulsive Potential Field 7
3.3 Black-hole Potential Field 8
3.4 Gradient Descent Method 11
4. Deep Reinforcement Learning 13
4.1 Reinforcement Learning 13
4.2 Convolutional Neural Network 15
4.3 Training Convolutional Neural Network 20
4.4 Deep Q-learning Network 21
5. Path Planning for Robots Formation 23
6. Experiments 24
6.1 Path Planning using Gradient Descent in an Artificial Potential Field 24
6.2 DQN Training for Specific Artificial Potential Field 26
6.3 DQN Training for General Artificial Potential Field 30
6.3.1 Basic Learning 31
6.3.2 Curriculum Learning 31
6.4 DQN Robot Formation 38
7. Conclusion and Recommendations 41
Bibliography 42

List of Tables

Table 1: Next position of the robot 14
Table 2: Artificial potential field parameters 24
Table 3: DQN with fixed artificial potential field parameters 27
Table 4: DQN with general artificial potential field parameters 30

List of Figures

Figure 1: Example artificial potential field for an environment with one target (red dot) and one obstacle (white rectangle). The path of the robot (blue dot) is shown in yellow..... 2
Figure 2: Deep Reinforcement Learning 3
Figure 3: Example attractive potential field of a target with x and y coordinates equal to 20 6
Figure 4: Example repulsive potential field of a rectangular obstacle..... 8
Figure 5: The attractive potential field of an environment with three targets without a black-hole potential field 8
Figure 6: Path planning in an environment with three targets whose attractive potential fields combine into an overall potential field with a global minimum at a position other than any target without a black-hole potential field 9
Figure 7: Example black-hole field potential field of a target with x and y coordinates equal to 20 10
Figure 8: The black-hole potential field of an environment with three targets 10
Figure 9: Path planning in an environment with three targets whose attractive potential fields and black-hole potential fields combine into an overall potential field with three local minima at the three targets..... 11
Figure 10: Example artificial potential field resulting from the superposition of attractive, repulsive, and the black-hole potential fields 12
Figure 11: Example observation 13
Figure 12: Structure of reinforcement learning in path planning 15
Figure 13: Convolutional neural network..... 15
Figure 14: Example observation *O* (5x5 matrix) and convolution filter *C* (3x3 matrix)..... 16
Figure 15: Example feature matrix M..... 17
Figure 16: Feature map matrix F 18
Figure 17: Two example two feature maps..... 18
Figure 18: Max-pooling process of the two 3x3 feature maps in Fig. 17 with a 2x2 pool..... 18
Figure 19: Flattening of the two feature maps from Fig. 18..... 19
Figure 20: The fully connected layer structure..... 19
Figure 21: Deep Q-learning Network 22
Figure 22: Examples of failed path planning using artificial potential field for an environment with one target (red dot) and three obstacle (white places). The path of the robot (blue dot) is shown in yellow. 25
Figure 23: Examples of successful path planning using artificial potential field for an environment with one target (red dot) and three obstacle (white places). The path of the robot (blue dot) is shown in yellow..... 26
Figure 24: Examples of path planning in a fixed artificial potential field, with obstacles obstructing the direct way of the robot to the target. 28
Figure 25: Examples of paths planned in a fixed artificial potential field, without obstacles between the robot and the target. 29
Figure 26: Example of failed path planning. 30
Figure 27: Paths planned by the deep Q network during basic learning. 31

Figure 28: Paths planning by the deep Q network during curriculum learning: the robot reaches the target in the environments i(a) and (b), and does not reach the target in the environment in (c) and (d). 32
Figure 29: Path planning in specific artificial potential fields: the robot reaches the target in all four cases. 33
Figure 30: Paths planes in arbitrary potential fields after curriculum learning: the robot reaches the target in the environment in (a), (c), and (d), but not in the environment in (b)..... 34
Figure 31: Test of failed path planning in Fig. 30(b) with shorter distance between the target and the robot start position, and test paths planning in Fig. 30(a), (c), and (d) using longer distance between the target and the robot start position 35
Figure 32: Tests of robot start positions around the robot end position in Fig. 30(b) 36
Figure 33: Paths planned by the gradient descent method in the same environment and for the same initial robot position as in Fig. 30: the robot cannot find the target in any of the four tested environment. 37
Figure 34: Paths planned by the robot formation in general artificial potential fields: as in the case of single robot path planning, the formation finds the target in the environment in (a), (c), and (d), but fails to reach the target in the environment in (b). 38
Figure 35: Path planning for a three-robot formation a in fixed artificial potential field, when the obstacles obstruct the direct path between the formation and the target. 39
Figure 36: Path planning for a three-robot formation a in fixed artificial potential field, when no obstacles obstruct the direct path between the formation and the target. 40

1. Introduction

1.1 Background

The artificial potential field method is a robot path planning method, which was proposed by Khatib in 1986 [3]. The artificial potential field is the superposition of the attractive field generated by the target and the repulsive field generated by the obstacles. The total force on a robot moving in an artificial potential field is the superposition of the attractive force generated by the attractive field and the repulsive force generated by the repulsive field. When the distance between the target and the robot increases, the attractive potential energy of the robot and the attractive force on the robot increase. When the distance between the target and the robot decreases, the attractive potential energy of the robot and the attractive force on the robot decrease. When the robot reaches the target, the attractive potential energy of the robot and the total force on it are zero. When the distance between the obstacles and the robot increases, the repulsive potential energy of the robot and the repulsive force on the robot decrease. When the distance between the obstacles and the robot decreases, the repulsive potential energy of the robot and the repulsive force on the robot increase. When the robot hits the obstacles, the repulsive potential energy of the robot is infinite, and the artificial potential field applies an infinite repulsive force on the robot.

Gradient descent is a first-order optimization algorithm that computes the direction of motion of the robot at its current position as the opposite direction of the gradient of the artificial potential field, and defines the movement distance in the movement direction. The movement direction is the direction of the total force. The movement distance can be set to a small distance. If the artificial potential field has a unique minimum at the target, the robot will reach the target without hitting obstacles.

The superposition of the attractive potential field of the target and the repulsive potential field(s) of the obstacle(s) can create an overall potential field with multiple minima. According to the gradient descent technique, the total force on the robot is zero at all minima because the gradient of the artificial potential field is zero at all its minima. Therefore, if the robot reaches a local minimum of the artificial potential field, the total force on the robot is zero and the robot cannot move from the minimum. If the minimum of the artificial potential field reached by the robot is not at the target, the robot cannot arrive the target. Fig.1 shows the path (yellow line) planned by the gradient descent technique for a robot (blue dot) in an environment with one target (red dot) and a rectangle obstacle (in the center of the figure). The artificial potential field has two minima, one at the target (above the obstacle) and one above the obstacle. The robot (blue dot) starts above the obstacle and moves along the negative gradient direction of the artificial potential field. Finally, the robot arrives at the local minimum, which it cannot leave.

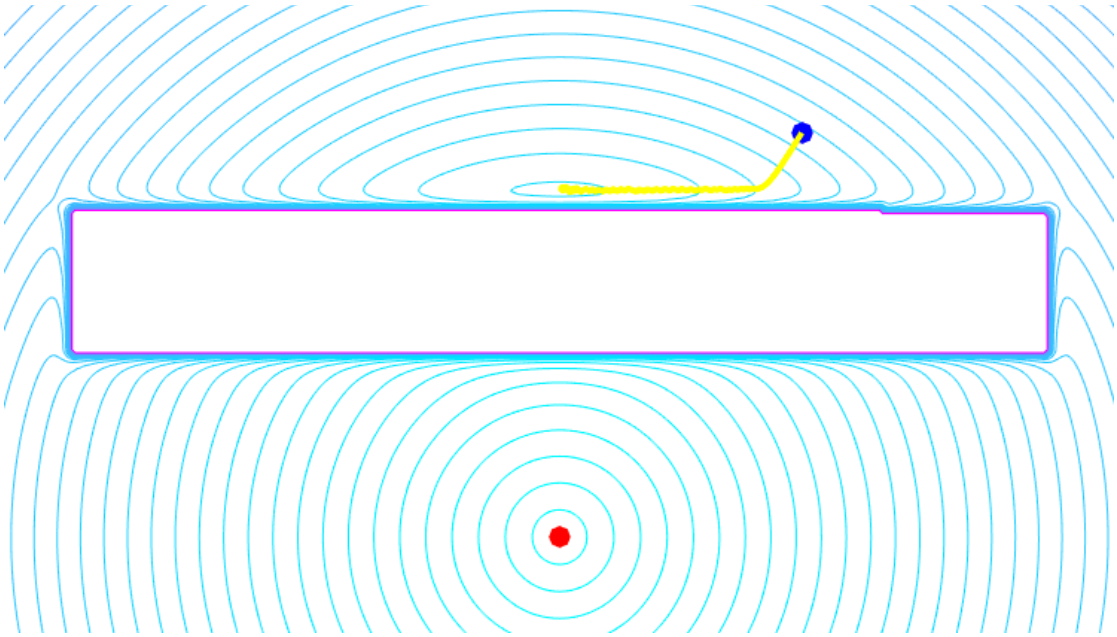


Figure 1: Example artificial potential field for an environment with one target (red dot) and one obstacle (white rectangle). The path of the robot (blue dot) is shown in yellow

1.2 Objective

The existence of multiple minima will prevent the gradient descent algorithm from guiding the robot to the target if the robot starts close to the local minimum. The objective of this report is to ease the local minimum problem using deep reinforcement learning algorithm and achieve path planning for a formation of three robots. When the robots are near the local minimum, the deep reinforcement learning algorithm should let the robots move to the target.

1.3 Approach

Deep reinforcement learning is an artificial intelligence method that combines reinforcement learning and deep learning. Deep reinforcement learning uses a reward mechanism to avoid local minimum problem. Fig. 2 shows an example of how deep reinforcement learning can solve the local minimum problem.

When the robot is at position a, the robot can go to right, stay there, or go to left. The position d is the obstacle. If the robot goes to right, the robot will get the maximum reward, which indicates that the robot hit the obstacle and the path planning is unsuccessful. If the robot remains at position a, the robot will get zero reward. If the robot goes to left, the robot will get a negative reward and arrive at position b. Then, if the robot goes from position b to position c, it will get a maximum positive reward. Therefore, the robot will get a positive reward from position a to position c. As a result, the robot chooses to go to left when the robot is at position a (local minimum) because the robot can get the largest reward for its three possible actions when the robot is at position a. Deep reinforcement learning is the strategy used to predict these rewards. According to these rewards, the robot can leave the local minimum.

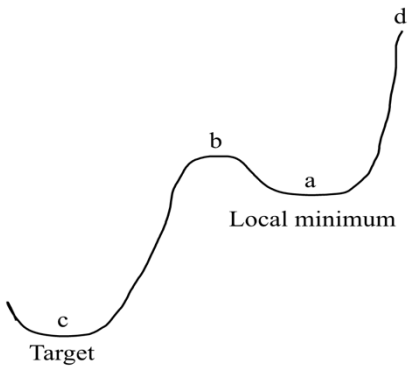


Figure 2: Deep Reinforcement Learning

When one robot (the leader) can achieve the path planning from the start point to the target, the other two robots (the followers) will follow it and the three-robot formation will reach the target together. As the leader moves, follower1 will remain within the communication range of the leader and follower2 will remain within the communication range of follower1. The three robots will move at the same time.

1.4 Organization of the Report

The rest of report is organized as follows: Section 2 overviews current technique for robot path planning; Section 3 introduces the technical details of the artificial potential field for robot path planning in terms of attractive, repulsive, and black-hole potential field; Section 4 introduces deep reinforcement learning for the path planning in terms of reinforcement learning, deep learning, and deep Q-learning network. Section 5 introduces technical details for the path planning of the three-robot formation; Section 6 presents the implementation of the artificial potential field and the deep Q-learning network methods for robot path planning; Section 7 concludes the report and presents recommendations for future work.

2. Literature Review

Path planning strategies for mobile robots can be classified into global and local methods [3]. Global techniques require that the robot be provided with information about the environment in the form of maps, cells, grids, etc. Local methods require only that the robot sense its environment [4]. Global path planning for mobile robots is a challenging problem due to its computational complexity. Namely, it is a non-deterministic polynomial-time hard (NP-hard) problem [5]. The disadvantage of local path planning is that the environment is unknown and local minima problem occur. Existing local path planning techniques include environmental modeling method that send environment information to the planning algorithm, optimization criteria, and path search algorithms that plan a collision-free route which must satisfy optimization criteria like path length, smoothness, safety degree, etc [3]. For path search, two types of algorithms exist: (1) the conventional type algorithms, which includes the artificial potential fields, behavior decomposition, cased-based learning, and rolling windows techniques; and (2) the artificial intelligence algorithms, among them the artificial neural network, genetic, ant colony optimization, particle swarm optimization, and simulated annealing algorithms.

To overcome the problem of local minima of artificial potential fields, [6] modify the conventional potential field algorithm as follows: when the robot gets trapped in a local minimum, the robot backtracks to someplace outside it and a new potential field is added to increase the potential energy at the local minimum and, therefore, to ensure that the robot will not be trapped at that local minimum again. However, the efficiency of this robot path planning algorithm needs to be improved to be used in real environment. A potential field method for path planning of mobile robots based on adapting attributes of animal motion is proposed in [7]. This method facilitates human-computer interaction because the path planning of the robot becomes more natural and intuitive for humans who can understand the intentions of the robot easier. Furthermore, due to its low computational complexity, this method has been successfully implemented in the embedded system of a mobile robot. Overcoming local minima setting virtual target points in the path of the robot has been proposed in [8]. Compared to the traditional artificial potential field algorithm, the algorithm in [8] has acceptable computational complexity. A modified artificial potential field algorithm that relies on simulated annealing to solve the problem of local minima has been introduced in [9]. However, complex artificial potential fields demand adjustments of the paraments of the simulated annealing algorithm. Three different improved artificial potential field methods are proposed [10], namely: the Virtual Force Field algorithm, which combines the potential field with a certainty grid to automatically detect the local minima; the Virtual Field Histogram method, which can guide robots through cluttered environments and unknown obstacles using a two-dimensional Cartesian Histogram; and the Harmonic Potential Function, which is used to build an obstacle avoidance potential field. The harmonic function is a good method to solve the problem of local minima. In [11], an ant colony algorithm improves the potential field method by changing the formula of the attractive potential field and by adding a new force on the robot to help it avoid becoming trapped at a local minimum. The ant colony algorithm finds new and better paths in the artificial potential field. In [12], a membrane evolutionary artificial potential field combines membrane computing with a genetic algorithm and the artificial potential field technique to plan paths both in static and in dynamic environments. The algorithm in [12] can find a suitable robot path in a short time and can achieve the minimum-length collision-free path. However, it cannot realize multi-robot

coordinated path planning and cannot work in three-dimensional space. In [13], an electrostatic potential field approach for mobile robot path planning avoids the problem of local minima by adding an electron potential field to each obstacle. The method in [13] can be applied in static and dynamic environments and also for multi-robot path planning. In [14], path planning in dynamic environments is based on a new potential field that incorporates the robot position, velocity, acceleration and physical dimensions into the calculations. The method in [14] solves the local minima problem and achieves reasonable planning. In [15], a dynamic robot path planning method solves the problem of randomly moving obstacles of the conventional artificial potential field techniques by employing the relative velocity and relative position of the robot and obstacles, and optimizes the attractive and repulsive potential fields using the particle swarm algorithm. It can generate smooth and safe paths.

Deep reinforcement learning is popular in artificial intelligence-based path planning algorithms. A path planning method which improves deep reinforcement learning by training a convolutional neural network is proposed in [16]. Compared with the traditional deep Q network method, the method in [16] has improved convergence speed, planning success rate, and path accuracy. In [17], multi-robot path planning method uses deep reinforcement learning to solve the problem of collisions among multiple robots simultaneously planning their paths. In [18], the deep Q network path planning method is improved by including two convolutional neural networks whose inputs are several frames of the current environment state information. The technique in [18] is suitable for path planning in static environments only. A deep Q network is used for goal path planning in [19]. They input complete map images with robot start position and with the target position on convolutional neural networks. Compared with conventional goal path planning algorithms (A* and Dijkstra), the deep Q network method in [19] has higher success rate. However, it does not work well in real city maps, and the deep Q networks need to be redesigned for such maps.

This project investigates the claim in [1] that a deep Q network can be combined with artificial potential fields to lessens the problem of the robot becoming trapped at a local minimum of the field when the robot plans it path using conventional gradient descent. The algorithm in [1] begin a local path planning algorithm is superior to global path planning algorithms in the sense that it is suitable both for static and for dynamic environment.

3. Artificial Potential Field for Robot Path Planning

3.1 Attractive Potential Field

In robot path planning using artificial potential fields, the target generateds an attractive potential field with unique minimum at the target. The robot is attracted by the target at any position on the map. When the robot is farther from the target and has higher attractive potential energy, it is subjected to a stronger attractive force from the target. The attractive force is the negative gradient of the attractive potential field of the target at the robot position. The attractive potential field formula and attractive force formula are given by:

$$PP_T = [(P_x - P_{Tx})^2 + (P_y - P_{Ty})^2]^{0.5} \quad (1)$$

$$U_{att}(\boldsymbol{P}) = k_{att} * \frac{{PP_T}^2}{2} \quad (2)$$

$$\boldsymbol{F}_{att}(\boldsymbol{P}) = -\nabla U_{att}(\boldsymbol{P}) \quad (3)$$

where PP_T is the distance between the robot and the target; $\boldsymbol{P}$ is the position of the robot on the map; P_x and P_y are the x and y coordinates of the position $\boldsymbol{P}$; $\boldsymbol{P}_T$ is the position of the target on the map; P_{Tx} and P_{Ty} are the x and y coordinates of the target; $U_{att}\,(\boldsymbol{P})$ is the attractive potential energy of the robot at position $\boldsymbol{P}$; k_{att} is the attraction coefficient; and $\boldsymbol{F}_{att}\,(\boldsymbol{P})$ is the attractive force on the robot at position $\boldsymbol{P}$.

Fig. 3 shows an example attractive potential field for a target with x and y coordinates equal to 20. The target has the lowest attractive potential energy.

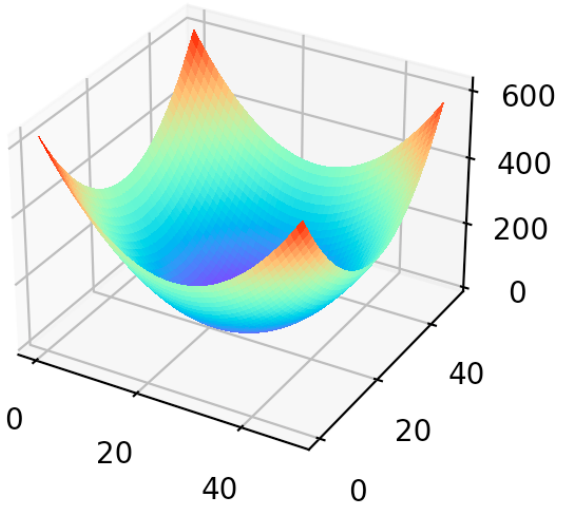


Figure 3: Example attractive potential field of a target with x and y coordinates equal to 20

3.2 Repulsive Potential Field

Repulsive potential fields are generated by obstacles. The obstacles have a repulsion range. When the distance between the robot and an obstacle is smaller than the repulsion range, the robot is repulsed by the obstacle. As the distance between the robot and the obstacle becomes smaller, the repulsive potential energy becomes higher and the robot is subjected to increasing repulsive force from the obstacle. When the robot is infinitely close to the obstacle, the repulsive potential energy value of the robot is close to the positive infinite. When the distance between the robot and the obstacle is greater than the repulsion range, the repulsive potential fields of the obstacle becomes zero and the robot is not repulsed by the obstacle. Both of the repulsive potential energy and the repulsive force on the robot are zero. The repulsive force is the negative gradient of the repulsive potential field at the robot position. The repulsive potential field and repulsive force are given by:

$$PP_O = [(P_x - P_{Ox})^2 + [(P_y - P_{Oy})^2]^{0.5} \tag{4}$$

$$U_{rep}(\boldsymbol{P}) = \frac{k_{rep}}{2} * \left(\frac{1}{PP_O} - \frac{1}{P_d}\right)^2 \qquad if\ PP_O \leq P_d \tag{5}$$

$$U_{rep}(\boldsymbol{P}) = 0 \qquad if\ PP_O > P_d \tag{6}$$

$$\boldsymbol{F}_{rep}(\boldsymbol{P}) = -\nabla U_{rep}(\boldsymbol{P}) \tag{7}$$

where PP_O is the distance between the robot and the obstacle; P_{Ox} and P_{Oy} are the x and y coordinates of the obstacle; P_d is the repulsion range, which is a constant; $U_{rep}\,(\boldsymbol{P})$ is the repulsive potential energy of the robot at position $\boldsymbol{P}$; k_{rep} is the repulsive coefficient; and $\boldsymbol{F}_{rep}\,(\boldsymbol{P})$ is the repulsive force on the robot at position $\boldsymbol{P}$.

Fig. 4 shows an example repulsive potential field, shaped like a frustum. The upper surface of the frustum is rectangular and has high (infinite) repulsive potential energy. This indicates that the rectangular area is the obstacle. The repulsive potential energy of the frustum decreases to zero and remains zero far from the obstacle. This indicates that, as the robot moves away from the obstacle and leaves repulsion range of the obstacle, its potential energy due to the repulsive potential field is zero.

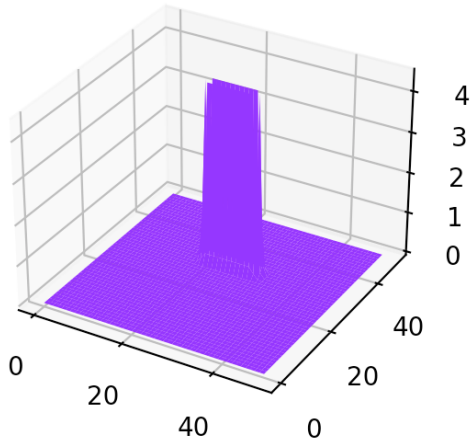


Figure 4: Example repulsive potential field of a rectangular obstacle

3.3 Black-hole Potential Field

Unlike the attractive and the repulsive potential field, the black-hole potential field serves to strengthen the attractive force that the target applies to the robot. The black-hole field can not solve the local minimum problem caused by the obstacles but can help the robot reach a target when multiple targets exist in the environment. Fig. 5 shows the attractive potential fields of three targets in the same environment, and the overall attractive potential field of the environment. Fig. 6 shows the level lines of the overall attractive potential field due to the three targets (red dots), as well as the path planned (yellow line) for the robot (blue dot) by the gradient descent algorithm. Because the overall attractive potential field has a global minimum that is not at any target, the gradient descent algorithm cannot drive the robot to any target. Instead, the robot moves towards to the position with minimum potential.

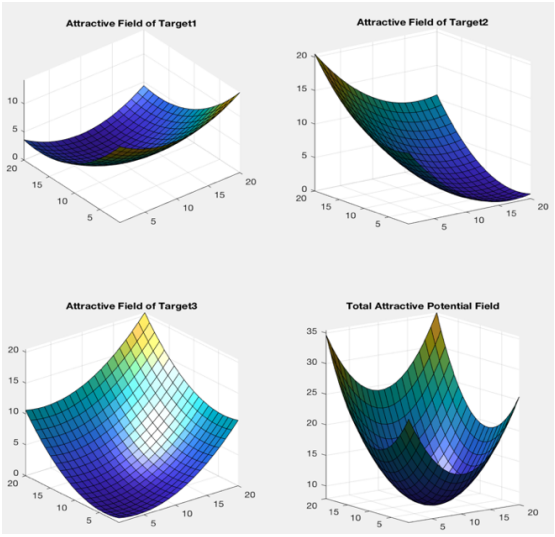


Figure 5: The attractive potential field of an environment with three targets without a black-hole potential field

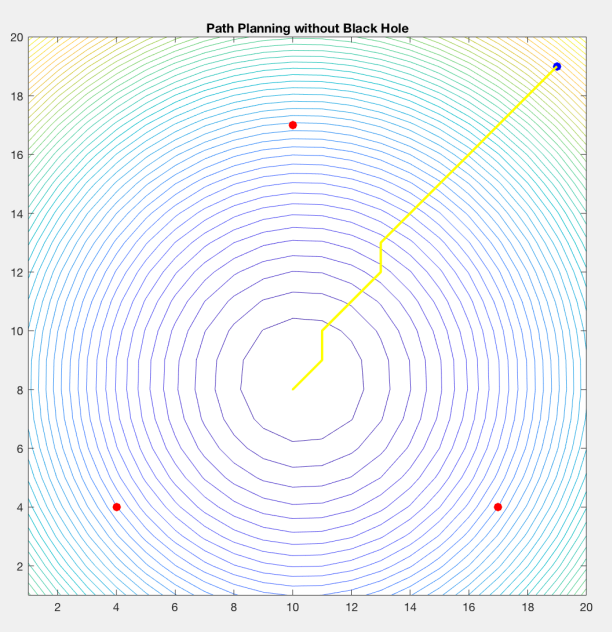


Figure 6: Path planning in an environment with three targets whose attractive potential fields combine into an overall potential field with a global minimum at a position other than any target without a black-hole potential field

The black-hole potential field is generated by the targets and has a certain range. When the distance between the robot and the target is smaller than the black-hole range, the robot is attracted by the target. As the distance between the robot and the target becomes smaller, the black-hole potential energy decreases and the target applies greater black-hole force on the robot. When the distance between the robot and the target is greater than the black-hole range, the robot feels no black-hole attraction from the target. The black-hole potential energy and force on the robot are zero. The black-hole force is the negative gradient of the black-hole potential field at the robot position. The black-hole potential field and black-hole force are given by:

$$U_{str}(\boldsymbol{P}) = -\frac{k_{str}}{2} * (P_S - PP_T)^2 \qquad if\ PP_T \leq P_S \qquad (8)$$

$$U_{str}(\boldsymbol{P}) = 0 \qquad if\ PP_T > P_S \qquad (9)$$

$$\boldsymbol{F}_{str}(\boldsymbol{P}) = -\nabla U_{str}(\boldsymbol{P}) \qquad (10)$$

where $U_{str}(\boldsymbol{P})$ is the black-hole potential energy at position $\boldsymbol{P}$; k_{str} is the black-hole coefficient; P_S is the black-hole range of the target, which is a constant; and $\boldsymbol{F}_{str}(\boldsymbol{P})$ is the black-hole force on the robot at position $\boldsymbol{P}$.

Fig. 7 shows an example black-hole potential field for a target with x and y coordinates equal to 20. The target has the lowest black-hole potential energy. The black-hole range is four. The

black-hole potential energy value is zero when the position of the robot is not in the black-hole range.

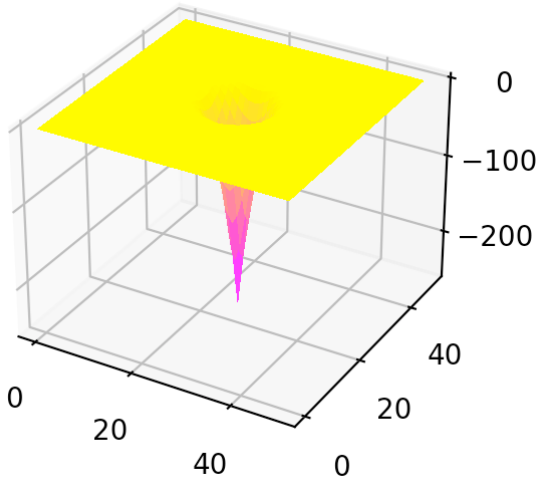


Figure 7: Example black-hole field potential field of a target with x and y coordinates equal to 20

After adding the black-hole field to the artificial potential fields of the three targets in Fig. 5, the problem caused by superposition of the attractive potential fields of the multiple targets can be solved. Fig. 8 shows the three black-hole fields of the targets and their superposition.

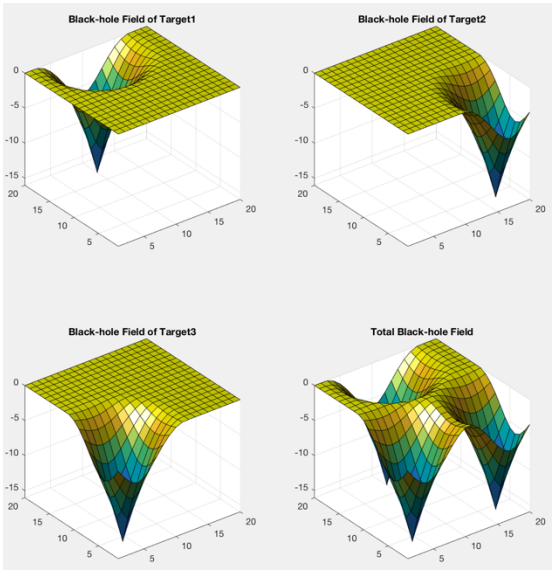
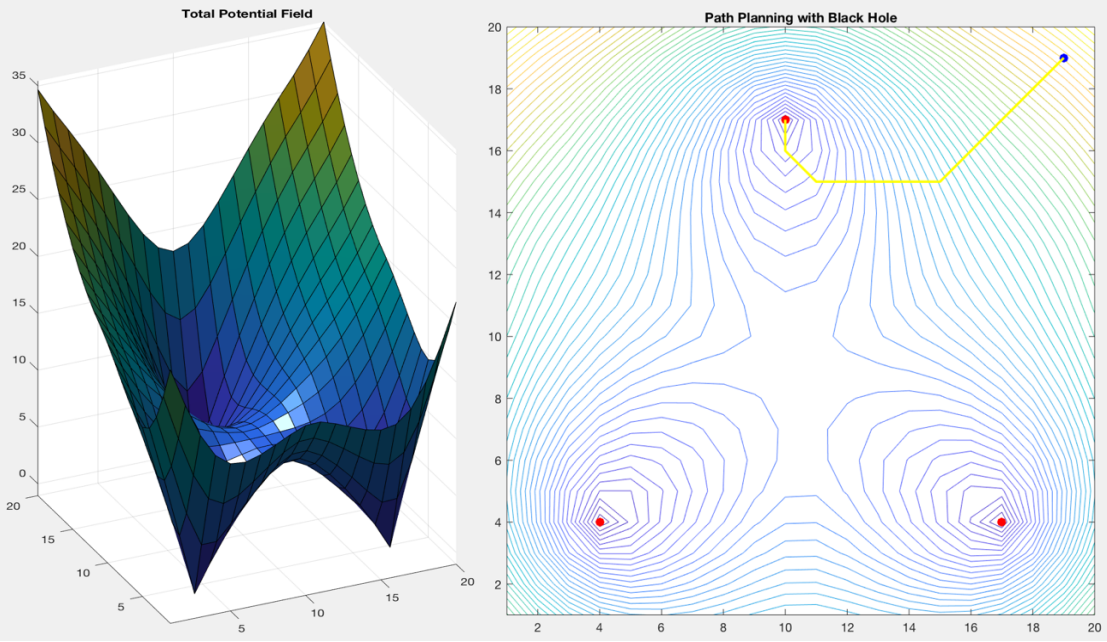


Figure 8: The black-hole potential field of an environment with three targets

Fig. 9 (a) shows the overall artificial potential field, which is the superposition of the three attractive potential fields and the three black-hole fields generated by the three targets. Fig. 9 (b) shows the level lines of the overall attractive potential field due to the three targets (red dots), as well as the path planned (yellow line) for the robot (blue dot) by the gradient descent algorithm.

Because the overall attractive potential field has three local minimums at the three targets, the gradient descent algorithm method can drive the robot to one of them. As a result, the robot successfully reaches the closest target.



(a) **(b)**

Figure 9: Path planning in an environment with three targets whose attractive potential fields and black-hole potential fields combine into an overall potential field with three local minima at the three targets

3.4 Gradient Descent Method

The artificial potential field of the environment in which the robot moves is the superposition of the attractive and black-hole potential field of all targets and the repulsive potential fields of all obstacles. The total force on the robot is the negative gradient of the artificial potential field. Thus, the artificial potential field $U_{total}(\boldsymbol{P})$ and the total force $\boldsymbol{F}_{total}(\boldsymbol{P})$ are given by:

$$U_{total}(\boldsymbol{P}) = U_{att}(\boldsymbol{P}) + U_{rep}(\boldsymbol{P}) + U_{str}(\boldsymbol{P}) \tag{11}$$

$$\boldsymbol{F}_{total}(\boldsymbol{P}) = -\nabla U_{total}(\boldsymbol{P}) \tag{12}$$

As an example, Fig. 10 shows the artificial potential field resulting from the superposition of attractive potential field in Fig. 3, the repulsive potential fields in Fig. 4, and the black-hole potential field in Fig. 7.

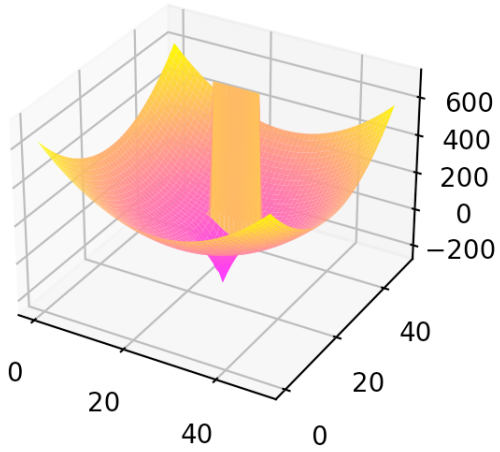


Figure 10: Example artificial potential field resulting from the superposition of attractive, repulsive, and the black-hole potential fields

Then, the gradient descent algorithm is used to update the position of the robot [20]. Specifically, the algorithm moves the robot along the direction of descent in the artificial potential field. The distance that the robot moves in one step is a constant positive scalar. Specifically, the gradient descent algorithm computes the next position of the robot by:

$$\boldsymbol{P}_{n+1} = \boldsymbol{P}_n - \alpha_n * \nabla U_{total}\,(\boldsymbol{P}_n) \tag{13}$$

$$\alpha_n = \frac{1}{\|\nabla U_{total}\,(\boldsymbol{P}_n)\|} \tag{14}$$

where $\boldsymbol{P}_n$ is the current position of the robot; $\boldsymbol{P}_{n+1}$ is the next position of the robot; and $\nabla U_{total}\,(\boldsymbol{P}_n)$ is the gradient of the artificial potential field at the current position of the robot. Note from the definition of α_n is that the algorithm moves the robot a unit distance in each step.

4. Deep Reinforcement Learning

4.1 Reinforcement Learning

Agent, observation, reward, action, and environment are components of reinforcement learning. In robot path planning, the robot is the agent. The energy of artificial potential field around the robot is the observation.

Fig. 11 shows an example observation, namely a five by five matrix of potential field values around a robot in the center of the observation (at the position with the artificial potential energy 10). In this observation, the two positions with very large artificial potential energy (100) are positions of obstacles. While the artificial potential energy of the obstacles should be infinite, its value is chosen a large finite value in practice (100 in this example) to sidestep the singularities that infinity would introduce in the calculations.

6	8	8	11	11
5	7	9	11	100
4	6	10	11	100
4	5	7	11	11
3	4	6	8	9

Figure 11: Example observation

The reward that the robot gets for moving from the current position to the next position is the step reward, and is computed by:

$$R_n = W_1 + W_2 + W_3 + W_4 \tag{15}$$

where W_1 is the negative constant penalty applied when the robot collides with an obstacle or when it moves outside the artificial potential field; W_2 is a large positive constant reward that the robot receives if it arrives the target; W_3 is the negative constant step penalty that encourage the robot to reach the target as soon as possible (it increases as the path length increases); and W_4 is a reward that encourage the robot to move to a position with lower potential energy (it is positive if the robot goes to a position $\boldsymbol{P}_{n+1}$ with lower potential energy, and is negative otherwise). The reward W_4 is given by:

$$W_4 = \beta * [U_{total}(\boldsymbol{P}_{n+1}) - U_{total}(\boldsymbol{P}_n)] \tag{16}$$

when β is constant and negative. In robot path planning, the robot can choose among nine actions at each step. Namely, it can go up (action1), down (action2), to the left (action3), to the right (action4), left up (action5), left down (action6), right up (action7), right down (action8), or

remain in place (action9). The action in reinforcement learning is the motion chosen by the robot. The next possible positions of the robot based on the nine actions are shown in Table 1.

Table 1: Next position of the robot

$\boldsymbol{P}_{n+1}$	a_{n1}	a_{n2}	a_{n3}	a_{n4}	a_{n5}	a_{n6}	a_{n7}	a_{n8}	a_{n9}
$\boldsymbol{P}_{n+1x}$	$\boldsymbol{P}_{nx}$+0	$\boldsymbol{P}_{nx}$+0	$\boldsymbol{P}_{nx}$-1	$\boldsymbol{P}_{nx}$+1	$\boldsymbol{P}_{nx}$-1	$\boldsymbol{P}_{nx}$-1	$\boldsymbol{P}_{nx}$+1	$\boldsymbol{P}_{nx}$+1	$\boldsymbol{P}_{nx}$
$\boldsymbol{P}_{n+1y}$	$\boldsymbol{P}_{ny}$+1	$\boldsymbol{P}_{ny}$-1	$\boldsymbol{P}_{ny}$+0	$\boldsymbol{P}_{ny}$+0	$\boldsymbol{P}_{ny}$+1	$\boldsymbol{P}_{ny}$-1	$\boldsymbol{P}_{ny}$+1	$\boldsymbol{P}_{ny}$-1	$\boldsymbol{P}_{ny}$

In this table: a_{n1} to a_{n9} are the nine possible actions, action1 to action9; P_{n+1x} and P_{n+1y} are the x and y coordinates of the robot at the next position; and P_{nx} and P_{ny} are the x and y coordinates of the robot at its current position.

Fig. 12 shows the structure of reinforcement learning in path planning. The reinforcement learning environment receives the current action a_n of the robot (the motion that the robot decides to take at the current step). Then, the learning environment computers and sends to the robot: (1) the observations at current $\boldsymbol{O}_n$ and next $\boldsymbol{O}_{n+1}$ robot positions (the 5x5 matrix of values of the artificial potential field around those robot positions); (2) the current action a_n of the robot, (3) the step reward for the current action of the robot, and (4) the searching information I_n at the next position, where is a flag I_n that indicates if the search finishes at the next step. The robot evaluates the nine possible actions and decides which action to take based on the observation $\boldsymbol{O}_n$ using the convolutional neural network. The details of how the robot evaluates its nine possible actions using the observation $\boldsymbol{O}_n$ will be introduced in Section 4.2, while the details of how the robot can improve the evaluation using $\boldsymbol{O}_n$, $\boldsymbol{O}_{n+1}$, a_n, and I_n are presented in Section 4.3. The search information I_n indicates that the search is over if the robot: (1) reaches the target; (2) collides with an obstacle; (3) collides with boundary of the artificial potential field; or (4) has taken the maximum number of steps *tt* during the current search. Otherwise, I_n indicates that the robot should continue its search for the target.

At each step and without knowing where the target is, the robot chooses one of the nine possible actions and moves to the next position $\boldsymbol{P}_{n+1}$. Because nine actions are possible at each step, many paths lead from the next robot position $\boldsymbol{P}_{n+1}$ to the last position of the search. On each path from the next position $\boldsymbol{P}_{n+1}$ to the last position of the search, the robot will receive step reward in every step. The sum of all step reward along a specific path is the total reward from the next position to the last position of the search along the specific path. There are many different specific paths that the robot can take and the total rewards of these specific paths are different. The maximum total reward among these total rewards is $R_{total}(\boldsymbol{P}_{n+1}/\boldsymbol{P}_{n/anj})$ and is unknown. $\boldsymbol{P}_{n/anj}$ indicates that the robot chooses the action a_{nj} at the current position $\boldsymbol{P}_n$. The maximum total reward from the current robot position to the last position when the robot chooses the action a_{nj} in one search $R_{total}(\boldsymbol{P}_{n/anj})$ is given by:

$$R_{total}\left(\boldsymbol{P}_{n/anj}\right) = R_{n/anj} + R_{total}\left(\boldsymbol{P}_{n+1}/\boldsymbol{P}_{n/anj}\right) \quad (17)$$

where $R_{n/anj}$ is the step reward for choosing action a_{nj} at the current position. At each position $\boldsymbol{P}_n$, the robot aims to get one of the nine total rewards $R_{total}(\boldsymbol{P}_{n/an1})$ to $R_{total}(\boldsymbol{P}_{n/an9})$ but this is not possible because the robot cannot know the total reward until it finishes this search. The real total reward that corresponds to the action a_n selected by the robot at the current position is $R_{total}(\boldsymbol{P}_n)$. The strategy that the robot uses to select action a_n is the greedy method and will be introduced in Section 4.3. After obtaining the observation $\boldsymbol{O}_n$ and $\boldsymbol{O}_{n+1}$ from the environment, the robot computes the nine predicted total rewards $R_{ptotal}(\boldsymbol{P}_{n/an1})$ to $R_{ptotal}(\boldsymbol{P}_{n/an9})$ using the convolutional neural network that will be introduced in Section 4.2. The predicted total reward corresponding to the action a_n taken by the robot is $R_{ptotal}(\boldsymbol{P}_n)$. Thus, if the robot chooses a_{n1}, then a_n is a_{n1}, the $R_{total}(\boldsymbol{P}_n)$ is $R_{total}(\boldsymbol{P}_{n/an1})$, the $R_{ptotal}(\boldsymbol{P}_n)$ is $R_{ptotal}(\boldsymbol{P}_{n/an1})$, and the R_n is $R_{n/an1}$.

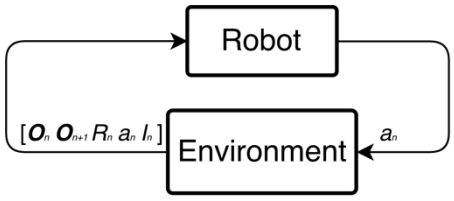


Figure 12: Structure of reinforcement learning in path planning

4.2 Convolutional Neural Network

Deep learning methods aim to simulate the human thinking. Convolutional neural network is one such deep learning method. In path planning, the robot uses a convolutional neural network to think about the observation of the environment. Fig. 13 shows the basic structure of the convolutional neural network proposed in [1] to enhance robot path planning using artificial potential fields.

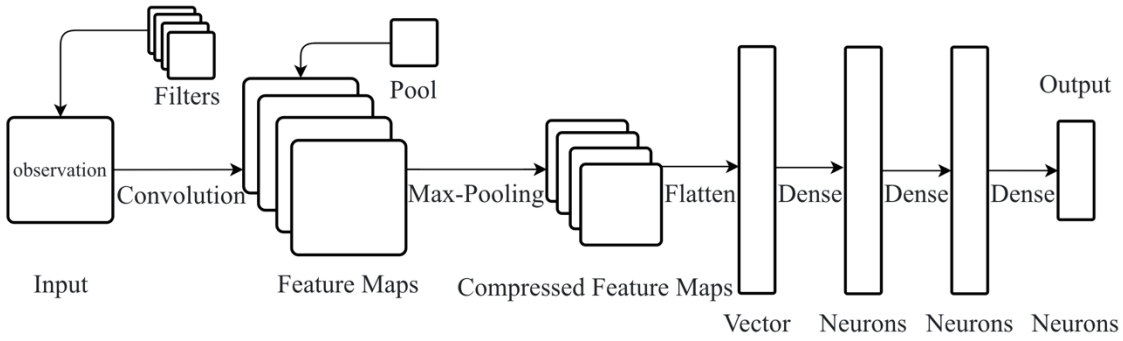


Figure 13: Convolutional neural network

Specifically, the convolutional neural network comprises: the convolution layer, the max-pooling layer, the flatten layer, and the fully connected layer. The structural design of convolutional neural networks impacts the success rate of path planning. A better structure of the convolutional neural network will lead to a higher success rate of the path planning. Fig. 13 shows one convolution layer, one max-pooling layer, one flatten layer, and three fully connected layers.

The first layer of the convolutional neural network is the convolution layer, which extracts features from its input, namely from the observation. The convolution filters are several matrices, all having the same size, smaller than the size of the observation. The entries in (weights of) the convolution filters are randomly generated initially and updated thereafter as introduced in Section 4.3. Each filter extracts specific feature of the observation, through the so-called convolution process, which is the same for all convolution filters. After convolution, the extracted feature forms a feature matrix. The feature matrix is processed by the activation function and turned into a feature map. The activation function provides nonlinear capabilities to the convolutional neural network. This project uses the Leaky ReLU activation function given by:

$$LeakyReLU(x) = max(\lambda * x, x) \tag{18}$$

Where λ is a constant value greater than zero and smaller than one; *x* is the input of Leaky ReLU activation function, which is the entry in the feature matrix, and all entries in the feature matrix are processed through the Leaky ReLU activation function. Fig. 14 shows an example 5x5 observation matrix ***O*** and one 3x3 convolution filter ***C***. The convolution of the observation with the filter then produces the feature matrix ***M*** in Fig. 15, with entries:

6	8	8	11	11
5	7	9	11	100
4	6	10	11	100
4	5	7	11	11
3	4	6	8	9

1	2	3
4	5	-6
7	8	-9

Figure 14: Example observation *O* (5x5 matrix) and convolution filter *C* (3x3 matrix)

The convolution process for one convolution filter is given by:

$$\begin{aligned} M_{11} &= 6*1+8*2+8*3+5*4+7*5+9*-6 \\ &\quad +4*7+6*8+10*-9 \\ &= 33 \end{aligned} \tag{19}$$

$$\begin{aligned} M_{12} &= 8*1+8*2+11*3+7*4+9*5+11*-6 \\ &\quad +6*7+10*8+11*-9 \\ &= 87 \end{aligned} \tag{20}$$

$$\begin{aligned} M_{13} &= 8*1+11*2+11*3+9*4+11*5+100*-6 \\ &\quad +10*7+11*8+100*-9 \\ &= -1188 \end{aligned} \tag{21}$$

$$M_{21} = 5*1+7*2+9*3+4*4+6*5+10*-6 \quad 7(22)$$
$$+4*7+5*8+7*-9$$
$$= 37$$

$$M_{22} = 7*1+9*2+11*3+6*4+10*5+11*-6 \quad (23)$$
$$+5*7+7*8+11*-9$$
$$= 58$$

$$M_{23} = 9*1+11*2+100*3+10*4+11*5+100*-6 \quad (24)$$
$$+7*7+11*8+11*-9$$
$$= -136$$

$$M_{31} = 4*1+6*2+10*3+4*4+5*5+7*-6 \quad (25)$$
$$+3*7+4*8+6*-9$$
$$= 44$$

$$M_{32} = 6*1+10*2+11*3+5*4+7*5+11*-6 \quad (26)$$
$$+4*7+6*8+8*-9$$
$$= 52$$

$$M_{33} = 10*1+11*2+100*3+7*4+11*5+11*-6 \quad (27)$$
$$+6*7+8*8+9*-9$$
$$= 374$$

33	87	−1188
37	58	−136
44	52	374

Figure 15: Example feature matrix M

The feature matrix in Fig. 15 is the result of a linear calculation. The last step in the convolution layer is to make its output non-linear by applying the Leaky ReLU activation function to each entry in the feature matrix:

$$F_{11} = F(M_{11}) = max(\lambda \cdot M_{11}, M_{11}) = max(\lambda \cdot 33,33) \quad (28)$$

Fig. 16 shows the output of the convolution layer the feature map and λ is 0.001. A network with multiple convolution filters will produce as many feature maps as many of convolution filters it has, all with the same size.

F_{11} =33	F_{12} =87	$F_{13} = -1.188$
F_{21} =37	F_{22} =58	$F_{23} = -0.136$
F_{31} =44	F_{32} =52	F_{33} =374

Figure 16: Feature map matrix F

The second layer of the convolutional neural network is the max-pooling layer. The max - pooling layer compresses the input feature maps and simplifies the computational complexity of the convolutional neural network. The pool in the max-pooling layer has size smaller than the size of the feature maps. The pool scans the feature maps and selects the largest number in the scanned area, which it assembles in the compressed feature maps. An example max-pooling of the two 3x3 feature maps shown in Fig. 17 with a 2x2 pool is illustrated in Fig. 18, step-by-step.

0	2	7
0	5	2
6	3	0

3	0	4
2	1	0
8	6	3

Figure 17: Two example two feature maps

0	2
0	**5**

2	**7**
5	2

3	0
2	1

0	**4**
1	0

0	5
6	3

5	2
3	0

2	1
8	6

1	0
6	3

5	**7**
6	**5**

3	**4**
8	**6**

Figure 18: Max-pooling process of the two 3x3 feature maps in Fig. 17 with a 2x2 pool

The next layer of the convolutional neural network is the flatten layer, which flattens the compressed feature maps into a vector. The flattening of the example two compressed feature maps from Fig. 18 is illustrated in Fig. 19.

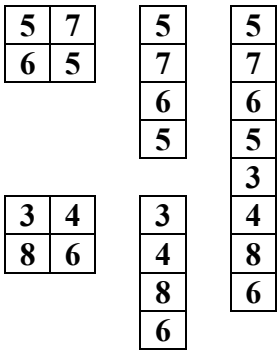


Figure 19: Flattening of the two feature maps from Fig. 18

The last layer of the convolutional neural network is the fully connected layer. Fig. 13 shows three fully connected layers. The fully connected layer acts as a classifier in a convolutional neural network. Each fully connected layer comprises a vector of neurons. Each neuron receives as inputs all the elements in the input vector. The output of the fully connected layer is a vector which is the classification result. Each entry in the output vector is a linear combination of all elements in the input vector filtered through an activation function. Fig. 20 shows an example calculation performed by one of the fully connected layers. The input is a 2x1 vector. The layer has three fully connected neurons, i.e., the size of the neurons vector is 3x1. The output of the layer is a 3x1 vector. The ReLU activation function is used in this example.

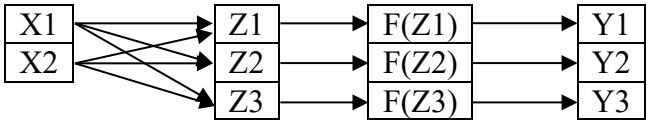


Figure 20: The fully connected layer structure

The convolution in this example fully connected layer is given by:

$$\mathrm{Z1} = W_{11} * \mathrm{X1} + W_{12} * \mathrm{X2} + b_1 \tag{29}$$

$$\mathrm{Z2} = W_{21} * \mathrm{X1} + W_{22} * \mathrm{X2} + b_2 \tag{30}$$

$$\mathrm{Z3} = W_{31} * \mathrm{X1} + W_{32} * \mathrm{X2} + b_3 \tag{31}$$

where W_{11}, W_{12}, W_{21}, W_{22}, W_{31}, and W_{32}, are weights $\boldsymbol{W}$; and the b_1, b_2, and b_3 are biases $\boldsymbol{B}$. The weights and bias are randomly generated initially; and are updated as introduced in the section 4.3 during the training of the network.

For path planning, the output of the last fully connected layer is a 9x1 vector, whose entries are the nine total predicted rewards $R_{ptotal}(\boldsymbol{P}_{n/an1})$ to $R_{ptotal}(\boldsymbol{P}_{n/an9})$ for the nine possible action a_1 to a_9 that the robot can take at the current position $\boldsymbol{P}_n$. If the robot hits an obstacle at one of the nine possible next positions, the predicted total reward for the respective action $R_{ptotal}(\boldsymbol{P}_{n/anj})$ will be the smallest possible reward (a large negative value). To ensure that the convolutional neural

network outputs this smallest possible value as the value of the predicted total reward for an action that corresponds to a collision, the network does not include activation functions in its last fully connected layer.

4.3 Training Convolutional Neural Network

The method to make the robot's judgment of the nine total rewards more accurate is to reduce the error between the total reward $R_{total}(\boldsymbol{P}_n)$ and the predicted total reward $R_{ptotal}(\boldsymbol{P}_n)$. This total reward $R_{total}(\boldsymbol{P}_n)$ is unknown until the robot finished the search at the next position. The robot uses the temporal difference method [1] to reduce the error between the total reward $R_{total}(\boldsymbol{P}_n)$ and the predicted total reward $R_{ptotal}(\boldsymbol{P}_n)$ without knowing $R_{total}(\boldsymbol{P}_n)$ by building the loss function $Loss(\boldsymbol{\theta})$ [1]. When the input of convolutional neural network (observation) is known, the convolutional neural network is a function of the filter weights **C**, weights $\boldsymbol{W}$, and the biases $\boldsymbol{B}$, all collected in $\boldsymbol{\theta}$. The predicted total reward at current position using the predicted total reward at next position and loss function are given by:

$$TQ = R_n + \gamma * Q(\boldsymbol{\theta}_n/\boldsymbol{P}_{n+1}) \tag{32}$$

$$Loss(\boldsymbol{\theta}_n) = [TQ - Q(\boldsymbol{\theta}_n/\boldsymbol{P}_n, a_n)]^2 \tag{33}$$

where $Q(\boldsymbol{\theta})$ is the so-called Q function of the convolutional neural network. For a given value of $\boldsymbol{\theta}$, $Q(\boldsymbol{\theta})$ outputs the nine predicted total reward from $R_{ptotal}(\boldsymbol{P}_{n/an1})$ to $R_{ptotal}(\boldsymbol{P}_{n/an9})$; $Q(\boldsymbol{\theta}_n/\boldsymbol{P}_{n+1}, a_{n+1})$ means that the robot is at $\boldsymbol{P}_{n+1}$, the action chosen by the robot is a_{n+1}, current values in the $\boldsymbol{\theta}$ is determined called $\boldsymbol{\theta}_n$; the output of $Q(\boldsymbol{\theta}_n/\boldsymbol{P}_{n+1}, a_{n+1})$ is predicted total reward $R_{ptotal}(\boldsymbol{P}_{n+1})$ at the next robot position with choosing action a_{n+1}; γ is the discount rate of the $Q(\boldsymbol{\theta}_n/\boldsymbol{P}_{n+1}, a_{n+1})$; γ is a constant value between zero and one; TQ is another kind of predicted total reward at the current robot position and the principle of TQ is same as the equation (17); the predicted total reward at the current robot position is the sum of the predicted total reward at the next robot position and the step reward the robot gets by moving from the current position to the next position; therefore, TQ is a predicted total reward and is not a function; $Q(\boldsymbol{\theta}/\boldsymbol{P}_n, a_n)$ is the convolutional neural network function for $\boldsymbol{\theta}$ when the robot is at current position $\boldsymbol{P}_n$ with choosing action a_n; $Q(\boldsymbol{\theta}/\boldsymbol{P}_n, a_n)$ means that the robot is at $\boldsymbol{P}_n$, the action chosen by the robot is a_n, current values in the $\boldsymbol{\theta}$ is not determined; therefore $Q(\boldsymbol{\theta}/\boldsymbol{P}_n, a_n)$ is a function when the current robot position $\boldsymbol{P}_n$ and the action a_n is known; if the values in the $\boldsymbol{\theta}$ is determined in the $Q(\boldsymbol{\theta}/\boldsymbol{P}_n, a_n)$, the output of $Q(\boldsymbol{\theta}/\boldsymbol{P}_n, a_n)$ is $R_{ptotal}(\boldsymbol{P}_n)$; different $\boldsymbol{\theta}$ values in $Q(\boldsymbol{\theta}/\boldsymbol{P}_n, a_n)$ can cause different value of $R_{ptotal}(\boldsymbol{P}_n)$; although TQ is not the $R_{total}(\boldsymbol{P}_n)$, the target of $Q(\boldsymbol{\theta}/\boldsymbol{P}_n, a_n)$ is to make its output close to the TQ by changing the values in the $\boldsymbol{\theta}$; and $Loss(\boldsymbol{\theta})$ is the loss function for the $\boldsymbol{\theta}$.

The robot selects its action based on the greedy method [1]. This method provides randomness in selection through a parameter ε called the exploration rate, which is a positive number smaller than one. In particular, each time the robot needs to choose an action, the greedy method generates r random number ρ between zero and one. If this random number ρ is larger than the current exploration rate ε, the robot will choose the action that corresponds to the largest predicted total reward in $R_{ptotal}(\boldsymbol{P}_{n/an1})$ to in $R_{ptotal}(\boldsymbol{P}_{n/an9})$. Otherwise, the robot chooses a random

action. Because the robot's judgment of the total reward $R_{total}(\boldsymbol{P}_{n/anj})$ becomes more accurate as the number of searches increases, the exploration rate ε decrease as the number of searches increases, until it reaches a specific value. An exploration rate that remained high would make it difficult for the robot to find the target in the search. The exploration rate is given by:

$$\varepsilon_s = \max\left(\varepsilon_i * \frac{ts - s}{ts}, \varepsilon_f\right) \tag{34}$$

where the total search times is *ts*; *s* indicates that the current search is the *s*th search; ε_i is initial exploration rate; ε_f is final exploration rate; and ε_s is the exploration rate in the *s*th search.

The loss function $Loss(\boldsymbol{\theta}_n)$ is computed as the mean square error given by:

$$Loss(\boldsymbol{\theta}_n) = \frac{1}{\mathrm{m}}\sum_{1}^{\mathrm{m}}[TQ - Q(\boldsymbol{\theta}/\boldsymbol{P}_n, a_n)]^2 \tag{35}$$

Where m is the constant batch size, that is the number of samples used for training. One sample includes the current observation $\boldsymbol{O}_n$, the next observation $\boldsymbol{O}_{n+1}$, the current action a_n, the step reward R_n, and the searching information I_n at the next position. These samples are drawn randomly from the experience pool. The experience pool is the place to store samples. The robot deposits one sample into the experience pool after each movement. The number of samples (ns) in the experience pool must be larger than batch size for the training to start.

The purpose of optimization is to minimize the loss function $Loss(\boldsymbol{\theta}_n)$ by optimizing the value of $\boldsymbol{\theta}$. The optimization of $\boldsymbol{\theta}$ is carried out the Adam optimizer [21], which is imported from TensorFlow. The TensorFlow is the Google Brain open-source software library for deep neural networks (DNN) [22]. The input of the Adam optimizer is set to be the loss function $Loss(\boldsymbol{\theta}_n)$ and its learning rate η is set constant as 0.001, which is a good default setting [21]. The Adam optimizer returns $\boldsymbol{\theta}_{n+1}$, which will be then used by the convolutional neural network to predict the total reward for the next robot position $\boldsymbol{P}_{n+1}$.

4.4 Deep Q-learning Network

Deep reinforcement learning is a combination of reinforcement learning and deep learning. Deep Q-learning network is a type of deep reinforcement learning. In reinforcement learning, the robot uses deep learning (convolutional neural network) to predict the total rewards corresponding to the nine possible actions. Fig. 21 shows the flow chat of the deep Q-learning network (DQN). When a search *s* begins, the start position of the robot and the target position are created in the reinforcement learning environment. The environment creates the observation $\boldsymbol{O}_n$ based on the robot start position. Then, the robot begins its first movement, and the movement time *t* is 1. If the random number ρ is smaller than the current exploration rate ε_s, the robot chooses a random action a_n. Otherwise, the convolutional neural network outputs the action a_n corresponding to the largest predicted total reward. The robot sends the action a_n to the environment. The environment sends the sample (which includes the observations $\boldsymbol{O}_n$, $\boldsymbol{O}_{n+1}$ at the current and next robot positions, the step reward R_n, the action a_n, and the search information I_n) to the experience pool.

If the number *ns* of samples in the experience pool is larger than batch size *m*, the *m* samples are used to build loss function $Loss(\boldsymbol{\theta}_n)$. The loss function $Loss(\boldsymbol{\theta}_n)$ is then sent to the Adam optimizer to optimize the neural network parameters $\boldsymbol{\theta}$. The optimizer sends $\boldsymbol{\theta}_{n+1}$ to the convolutional neural network. Then, the environment sends $\boldsymbol{O}_{n+1}$ to the robot and the robot begins the next movement ($t = 2$). If the movement times *t* equals to the maximum number of movements *tt*, the next search begins ($s = 2$). If the number of searches *s* equals the maximum number of searches *ts*, the training of the convolutional neural network ends.

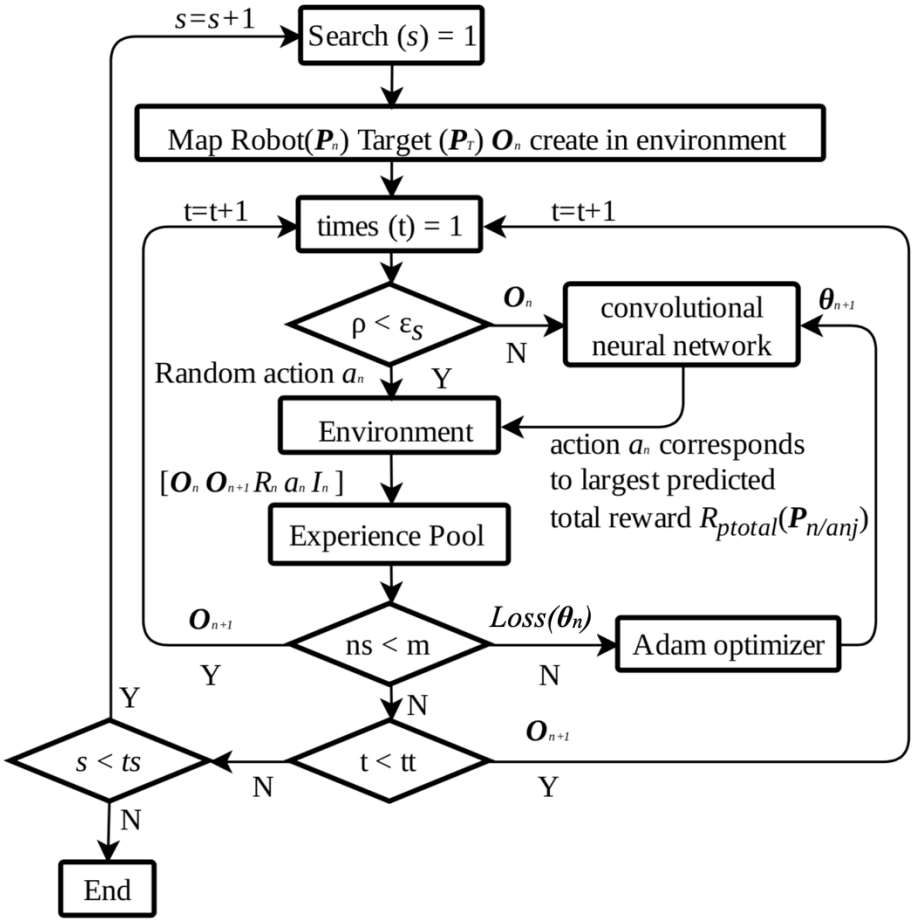


Figure 21: Deep Q-learning Network

5. Path Planning for Robots Formation

For a three-robot formation with one leader and two followers, the leader robot carries out the path planning for the team and the followers maintain the formation. In this project, the formation is chosen linear: follower1 moves so as to remain within communication distance of the leader and follower2 moves so as to remain within communication distance of the follower1.

The artificial potential field of leader includes attraction potential field and black-hole potential fields generated by the target, as well as the repulsive potential fields generated by the obstacles. A deep Q-learning network is used to achieve path of the leader in the artificial potential field.

The artificial potential field of follower1 includes attractive potential field generated by the leader and the repulsive potential fields generated by the obstacles and by follower2. The repulsive potential fields ensure that follower1 can avoid collisions both with obstacles and with follower2. Follower1 moves to the neighbouring position with the second lowest potential energy when moving to the neighbouring position with the lowest potential energy would make it collide with the leader. Otherwise, it moves to the neighbouring position with the lowest potential energy.

The artificial potential field of follower2 includes attractive potential field generated by the follower1 and repulsive potential fields generated by the obstacles and by the leader. In other words, the collision of follower2 with the leader is avoided by enabling the leader to repulse it. The follower2 chooses its action similarly to follower1.

6. Experiments

This section presents simulation results for path planning for a single robot and for a team of three robots. First, the path planning is carried out using gradient descent in the artificial potential field of the target and obstacles. Then, a deep Q-learning network is implemented to help the robots learn how to escape local minima in an artificial potential field. The path planning algorithms are implemented on an Intel Core i7-4770HQ machine with 16G RAM, in MATLAB_R2019b.

6.1 Path Planning using Gradient Descent in an Artificial Potential Field

This section simulates a robot searching for its target in an environment with three arbitrarily shaped obstacles. The robot uses a discretized map of the environment with size 20x20. The parameters of the attractive, repulsive and black-hole potential fields that compound to form the artificial potential field of the environment are shown in Table 2.

Table 2: Artificial potential field parameters

Parameter	k_{att}	k_{rep}	k_{str}	p_0	p_s	Robot Movs
Value	1	20	8	3	8	20

Fig. 22 and 23 illustrate robot paths (yellow lines) planned using gradient descent in the artificial potential field. The field level lines are shown in purple. The target is the red dot, and the robot is the blue dot. The robot moves along the negative direction of the gradient of the artificial potential field. In Fig. 22, the robot becomes trapped in a local minimum. In Fig. 23, the robot can reach the targets following the steepest descent direction because there is no local minimum in the path of the robot.

To help the robot learn to escape local minima in artificial potential fields. [1] proposes to use a deep Q-learning neural network. Therefore, the following sections investigate the effectiveness of such a neural network for robot path planning. Three cases are investigated: (1) path planning in a fixed artificial potential field, namely in an environment with fixed target and obstacles; (2) path planning in a general artificial potential field, namely in an environment with arbitrarily placed target and obstacles; and (3) path planning for a three-robot team in a fixed artificial potential field. The simulations are implemented in Python3.

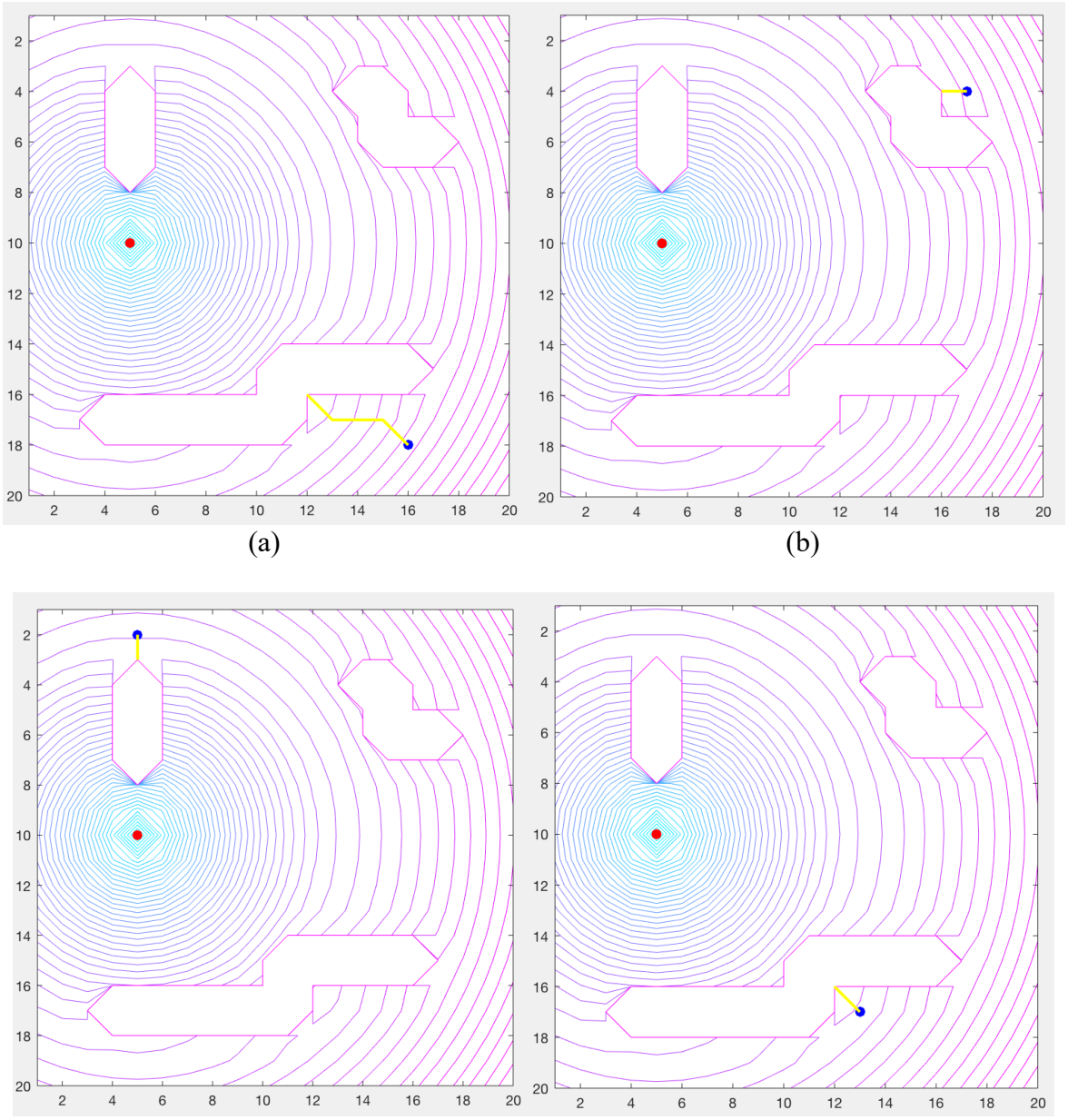


Figure 22: Examples of failed path planning using artificial potential field for an environment with one target (red dot) and three obstacle (white places). The path of the robot (blue dot) is shown in yellow.

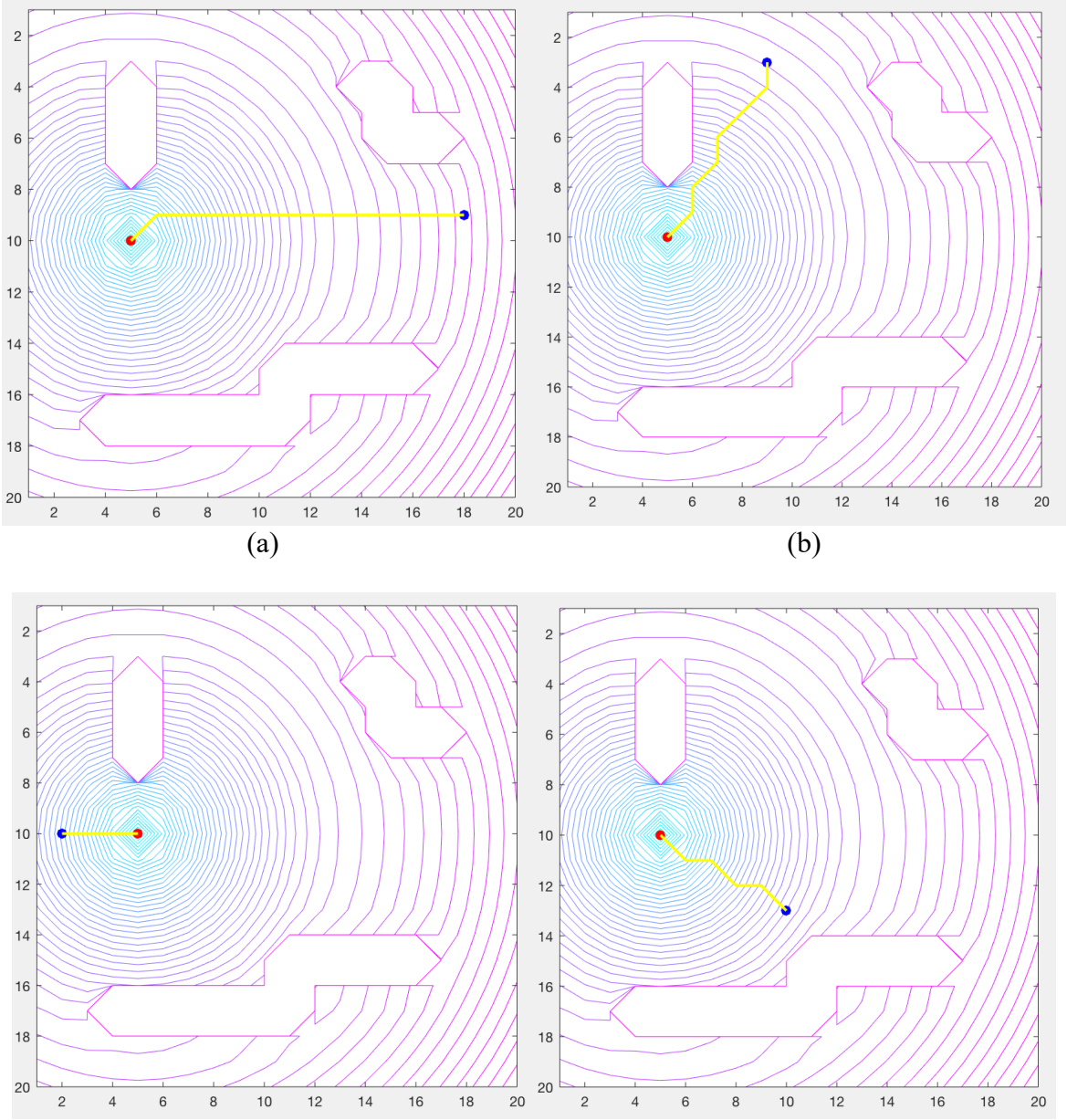


Figure 23: Examples of successful path planning using artificial potential field for an environment with one target (red dot) and three obstacle (white places). The path of the robot (blue dot) is shown in yellow.

6.2 DQN Training for Specific Artificial Potential Field

In this section, the robot uses the deep Q-learning network to plan its path to the target in a certain artificial potential field. The target and the obstacles are always at the same positions and, thus, the artificial potential field does not change during the training and during experiments. The

control parameters for path planning using deep Q-learning network in the fixed artificial potential field from Section 6.1 above are shown in Table 3. Because [1] does not provide all parameters of the deep Q-learning network, not all values in Table 3 match those in [1]. Specifically, the number of filters in convolutional layers and the number of convolutional layers are designed by observing the path planning results after training. The step reward paraments are adjusted from the parameters in [1]. Lastly, the parameters in the third column are designed by observing the path planning results after training because [1] does not train the deep Q-learning network in a fixed artificial potential field.

Table 3: DQN with fixed artificial potential field parameters

Parameter		Parameter		Layer		
k_{att}	2	s	30000	Layer 1	convolutional	Filters size: 2x2
k_{rep}	20	tt	100	Layer 2	max-pooling	Pool size: 2x2
k_{str}	8	ε_i	0.95	Layer 3	convolutional	Filters size: 2x2
p_0	0.5	ε_f	0.05	Layer 4	max-pooling	Pool size: 2x2
p_s	8	m	7	Layer 5	convolutional	Filters size: 2x2
$\boldsymbol{O}_n$	11x11	η	0.001	Layer 6	flatten	
W_1	-1000			Layer 7	fully connected	Neurons: 32
W_2	1000			Layer 8	fully connected	Neurons: 32
W_3	-1			Layer 9	fully connected	Neurons: 9
β	-4					
γ	0.95			Layer 1	Filters number:10	
Map	20x20			Layer 3	Filters number:10	
				Layer 5	Filters number:20	

Fig. 24 shows that the robot (blue dot) can plan a path (yellow line) around the obstacles to reach its target (red dot) when it uses the deep Q-learning network. In Fig. 24(a), the robot does not move towards to the local minimum in the beginning, but it moves so as to leave the local minimum. It then moves towards the targets along the obstacle and finally goes to the target directly. Similarly, in Fig. 24(b), the robot starts by moving so as to leave the local minimum and then moves to the target directly. In Fig. 24(c), the robot starts by moving towards position with lower potential energy until it finds the obstacle. It then moves along the obstacle and finally goes to the target without straight way. Lastly, Fig. 24(d) shows another example where the robot moves to escape a local minimum of the artificial potential field and succeeds to find its target.

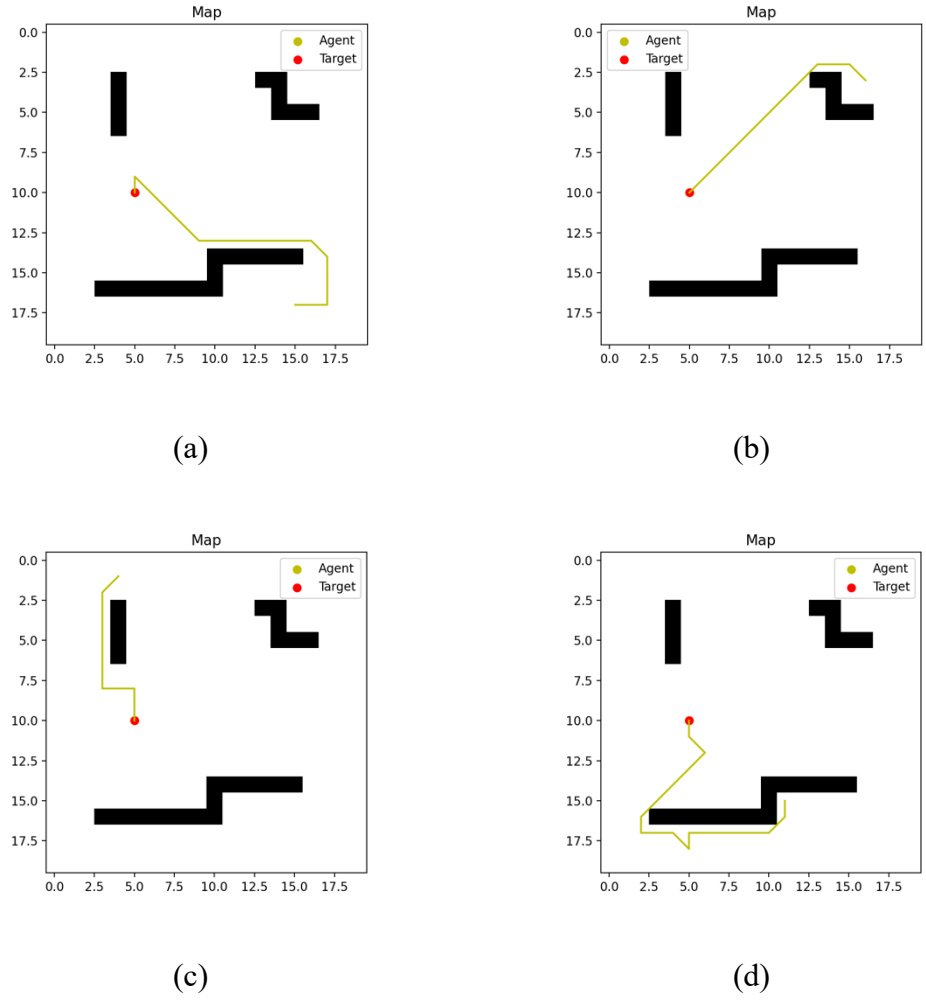


Figure 24: Examples of path planning in a fixed artificial potential field, with obstacles obstructing the direct way of the robot to the target.

In Fig. 25 shows motions planned when the obstacle does not impede the robot from moving toward the target. In Fig. 25(a), the robot moves towards to lower potential energy position, but the robot cannot move to the target directly. In Fig. 25(b), the robot moves towards to the target directly overall. In Fig. 25(c) and (d), the robot can move to the target without straight way.

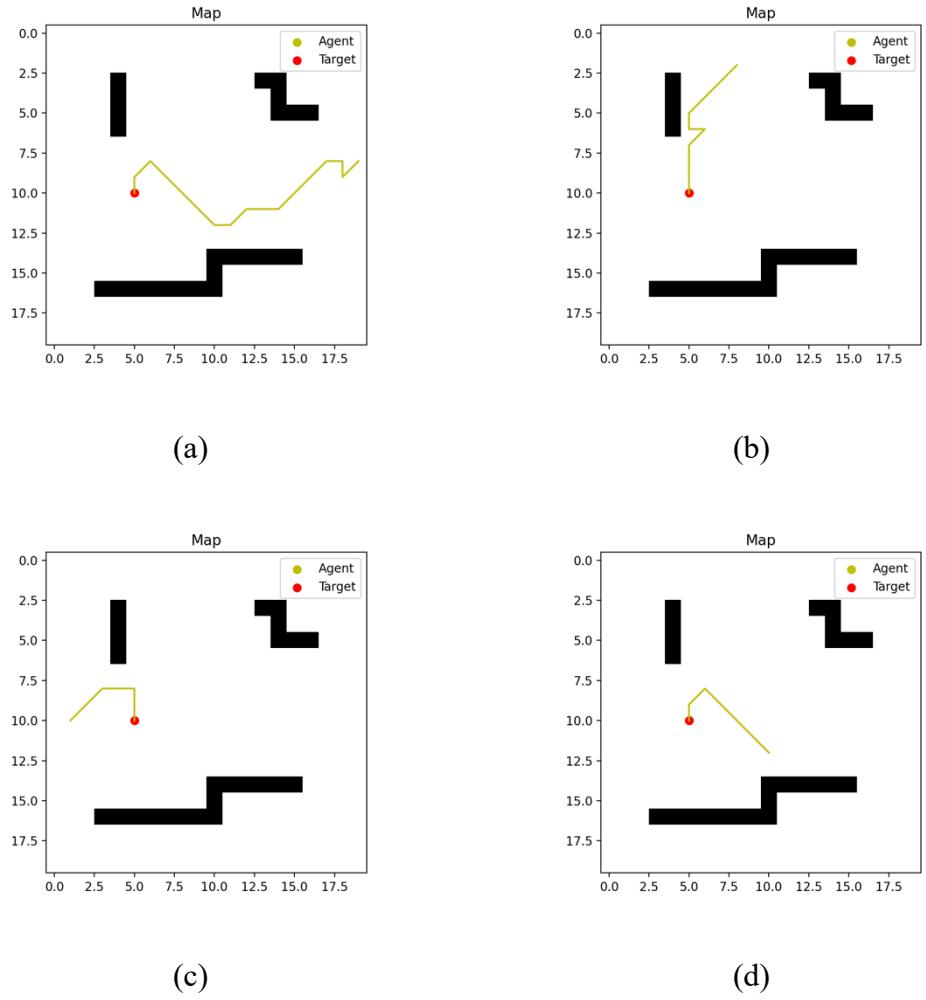


Figure 25: Examples of paths planned in a fixed artificial potential field, without obstacles between the robot and the target.

The robot does not always move directly towards the target (Fig. 24 and Fig. 25) because the deep Q-learning network can make an inaccurate judgment of the total reward $R_{total}(\boldsymbol{P}_{n/anj})$ based on the input of the convolutional neural network (the observation). Although the judgment is inaccurate, the convolutional neural network can identify the general direction of movement in the examples shown in Fig. 24 and Fig. 25. However, the robot can not reach the target every time when it plans its path using the deep Q-learning network. As example when the robot cannot reach the target is shown in Fig. 26. The reason that the robot cannot find the path in this example could be: (1) that this specific start position of robot has not been used to train to convolutional neural network; or (2) during training, the greedy method has not provided a random action that helped the robot escape the position where it gets trapped.

In this project, the training of the neural network has been tested with the robot starting form 40 different positions. Out of the 40 tests, the robot found the target in all but three tests.

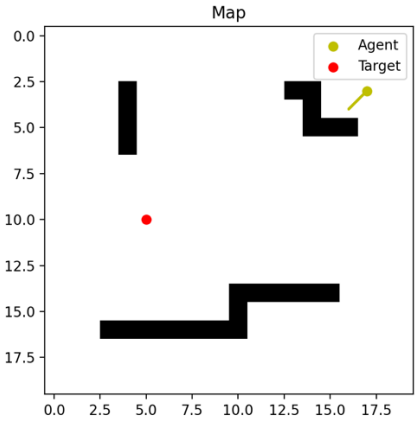


Figure 26: Example of failed path planning.

6.3 DQN Training for General Artificial Potential Field

In this section, eight specific artificial potential fields used randomly during the training of the deep Q-learning network, as well as during path planning using it. The parameters of the artificial potential fields and of the deep Q-learning network are shown in Table 4.

As proposed in [1], the training of deep Q-learning network is trained in two steps. In the first step, the so-called basic learning, the network is trained in an environment with a single point-like obstacle. Then, curriculum learning is implemented. In curriculum learning, the network is trained in progressively more complex environments. The maps and robot start positions are the same as in [1].

Table 4: DQN with general artificial potential field parameters

Parameter		Parameter		Layer		
k_{att}	1	s	150	Layer 1	convolutional	Filters size: 2x2
k_{rep}	20	tt	100	Layer 2	max-pooling	Pool size: 2x2
k_{str}	8	ε_i	0.95	Layer 3	convolutional	Filters size: 2x2
p_0	3	ε_f	0.05	Layer 4	max-pooling	Pool size: 2x2
p_s	8	m	7	Layer 5	flatten	
$\boldsymbol{O}_n$	9x9	η	0.001	Layer 6	fully connected	Neurons: 32
W_1	-10			Layer 7	fully connected	Neurons: 32
W_2	10			Layer 8	fully connected	Neurons: 9
W_3	-0.2					
β	-20					
γ	0.95			Layer 1	Filters number:4	
Map	50x50			Layer 3	Filters number:4	

6.3.1 Basic Learning

During basic learning, the robot should learn how to avoid a point-like obstacle while advancing towards the target. Four specific artificial potential fields are used to train the deep Q-learning network. Each artificial potential field includes one obstacle and one target. The training order of the four specific artificial potential fields is randomized. Fig. 27 shows the training results. The robot learns to avoid the obstacle and arrive near the target. However, the robot does not learn to reach the target accurately. This problem also occurs in [1].

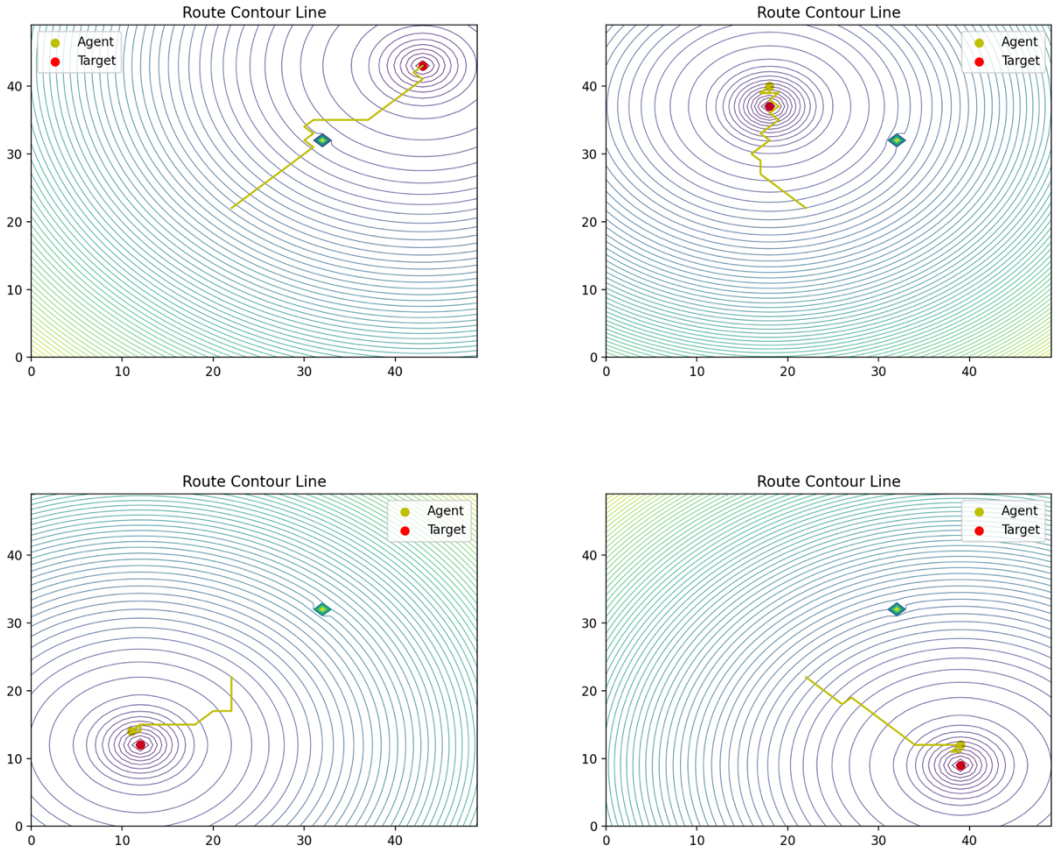


Figure 27: Paths planned by the deep Q network during basic learning.

6.3.2 Curriculum Learning

During curriculum learning, the deep Q-learning network is trained in environment that include more complex obstacles, whose shape and number are designed to allow the robot to deal with arbitrary environment. Specifically, obstacles with four different shapes are used: a single, line-shape obstacle; two point-like obstacles; a concave obstacle; and two line-shaped obstacles, one of which has a bend. To ensure that the deep Q-learning network does not forget what it has learned in the basic learning and in the curriculum learning, when a new artificial potential field needs to be used to train the deep Q-learning network, what the artificial potential fields has been trained will be trained again with the new artificial potential field. For example, when the

artificial potential field in Fig. 28(a) needs to be trained, all artificial potential fields in Fig. 27 will be trained together with it, in random order.

Fig. 28 shows the training results of curriculum learning. Fig. 28(a) and (b) show that the robot can avoid the specific obstacles and reach the position close to the target directly. In Fig. 28(c) and (d), the robot fails to find the target. The convolutional neural network can identify the obstacles and the robot does not hit the obstacles. However, it cannot bypass them. Fig. 29 (a)-(d) show that the robot can still find the target in the environments used for basic learning. However, the paths planed in the four specific artificial potential fields have changed because the network parameters $\boldsymbol{\theta}$ has been changed during the curriculum learning.

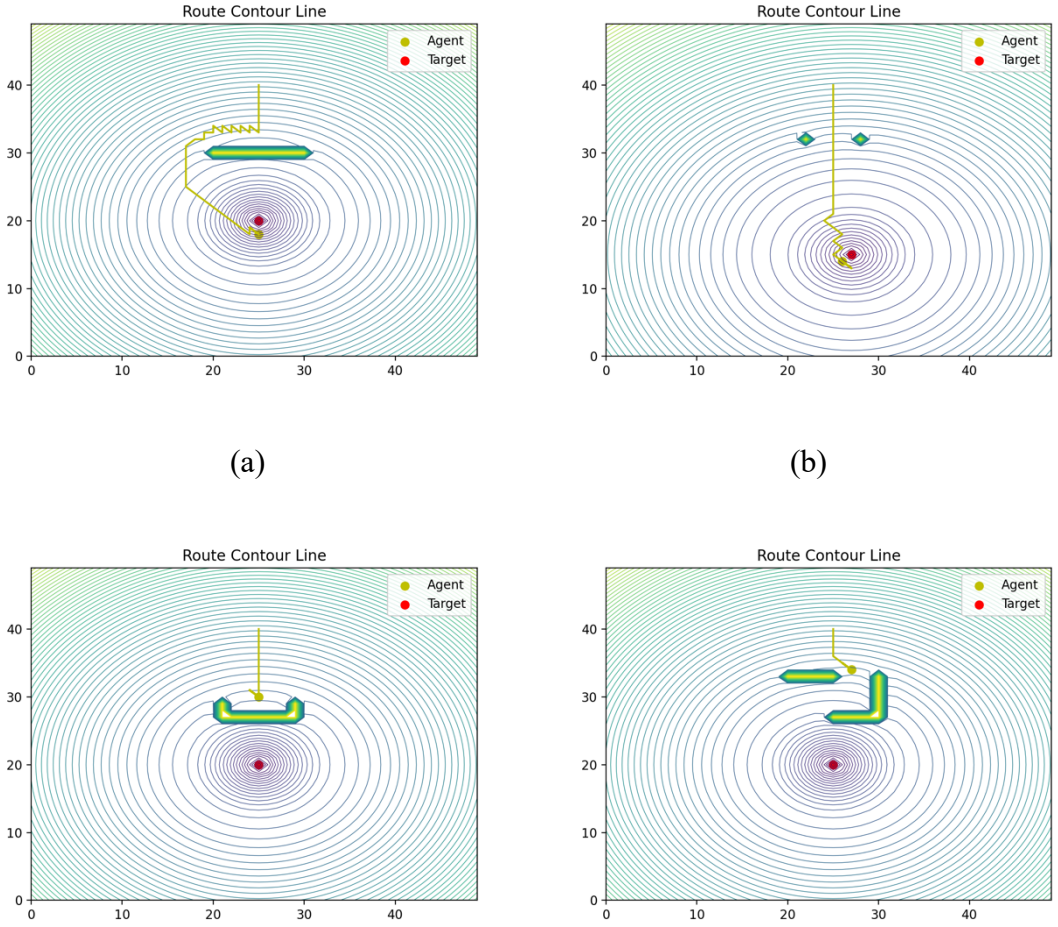


Figure 28: Paths planning by the deep Q network during curriculum learning: the robot reaches the target in the environments i(a) and (b), and does not reach the target in the environment in (c) and (d).

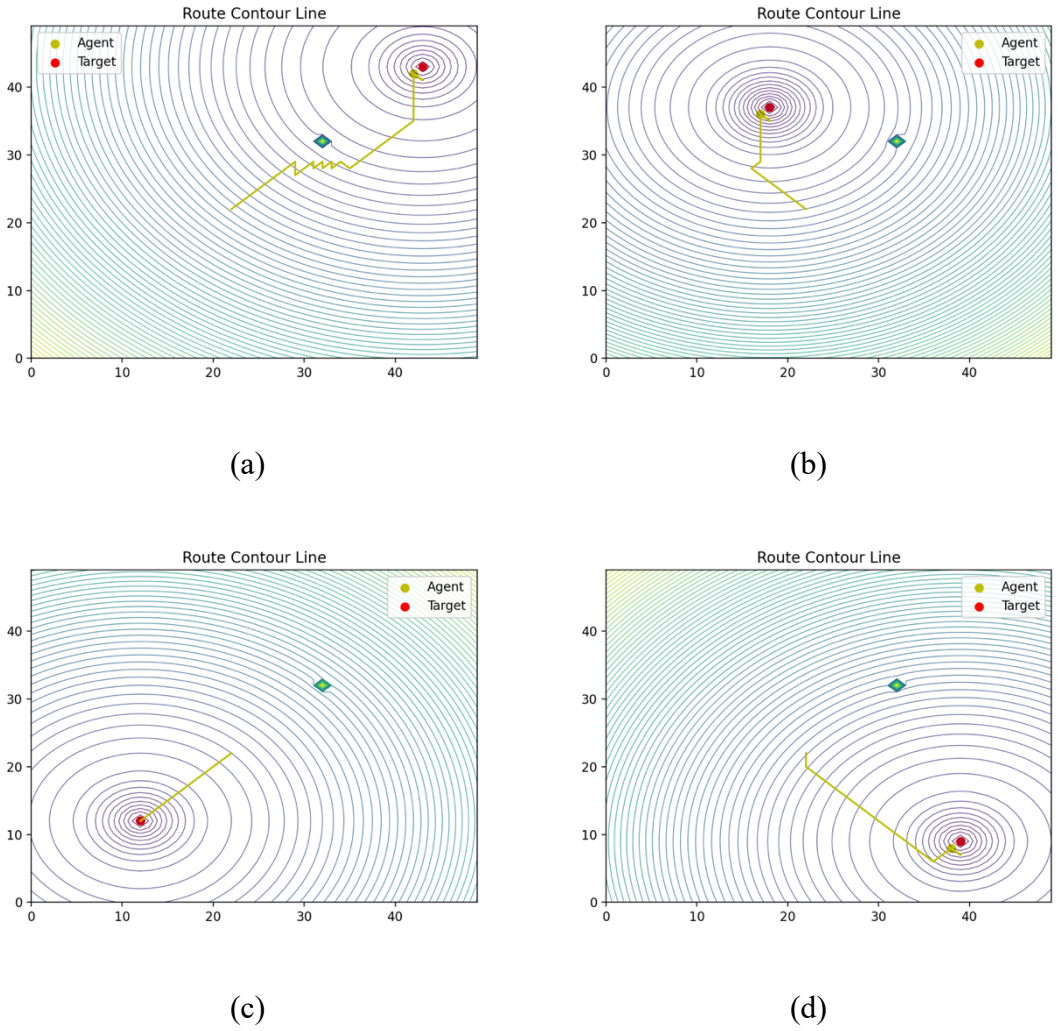


Figure 29: Path planning in specific artificial potential fields: the robot reaches the target in all four cases.

Fig. 30 shows four tests in four general artificial potential fields where the robot uses the deep Q-learning network to plan its path to the target. The robot finds its way to the target in three of the four tests.

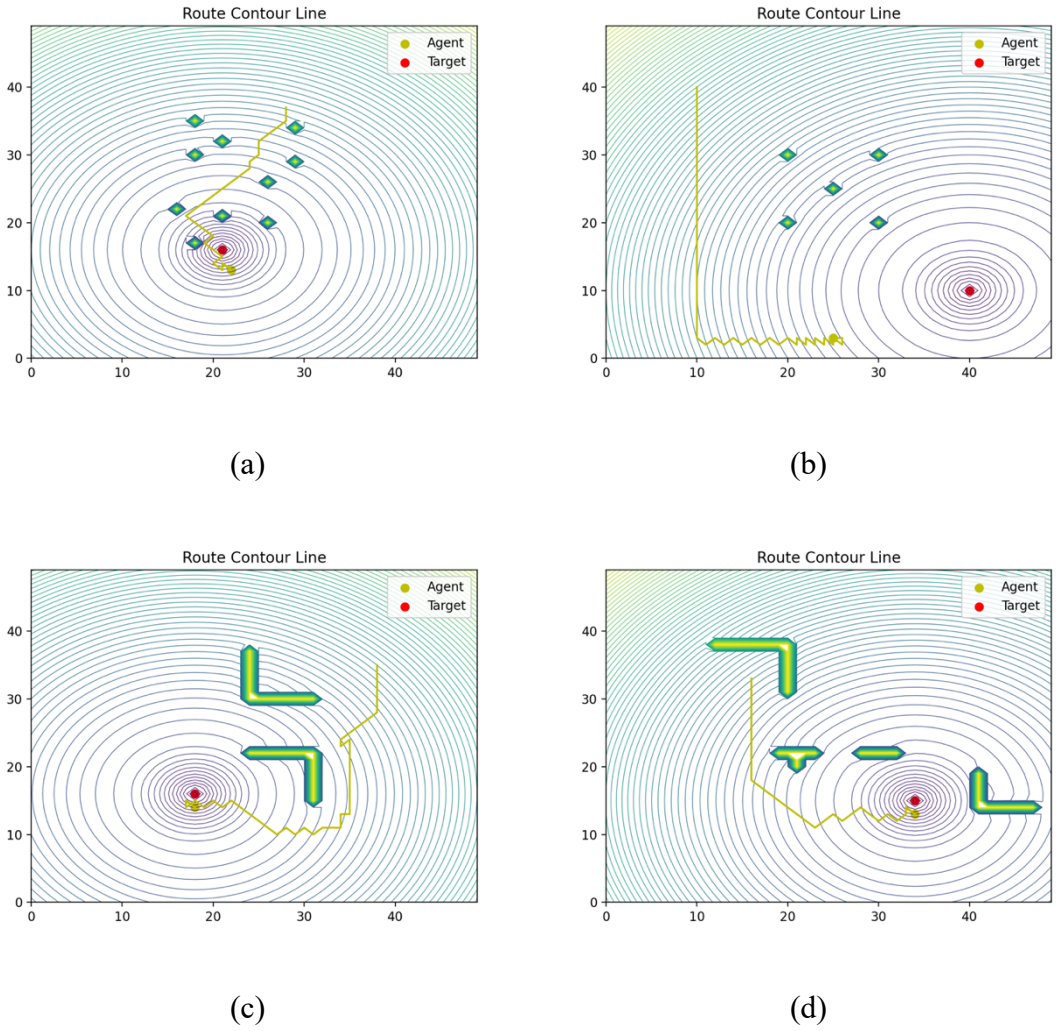


Figure 30: Paths planes in arbitrary potential fields after curriculum learning: the robot reaches the target in the environment in (a), (c), and (d), but not in the environment in (b).

The robot fails to plan a path to the target in Fig. 30(b). A possible reason could be that the distance between robot start position and the target is longer in this test than in training. However, the tests in Fig. 31 (a), (c) and (d) indicate that the robot can plan a path to the target even when it starts further away from the target than during training in these artificial potential fields. In turn, this means that the distance between the initial robot position and the target is not the cause of the path planning failure in Fig. 30 (b). Another possible reason for the failure in Fig. 30 (b) could be that the network has not learned to predict the total rewards appropriately in the region where this failure happens. Fig. 32 shows that path planning tests in which the robot starts in the region in which the path planning in Fig. 30 (b) fails. Note that all tests in Fig. 32 fail. Thus, Fig. 32 shows that the deep Q-learning network cannot help the robot select good actions based on observations of the potential field in the vicinity of the end robot position in Fig. 30(b).

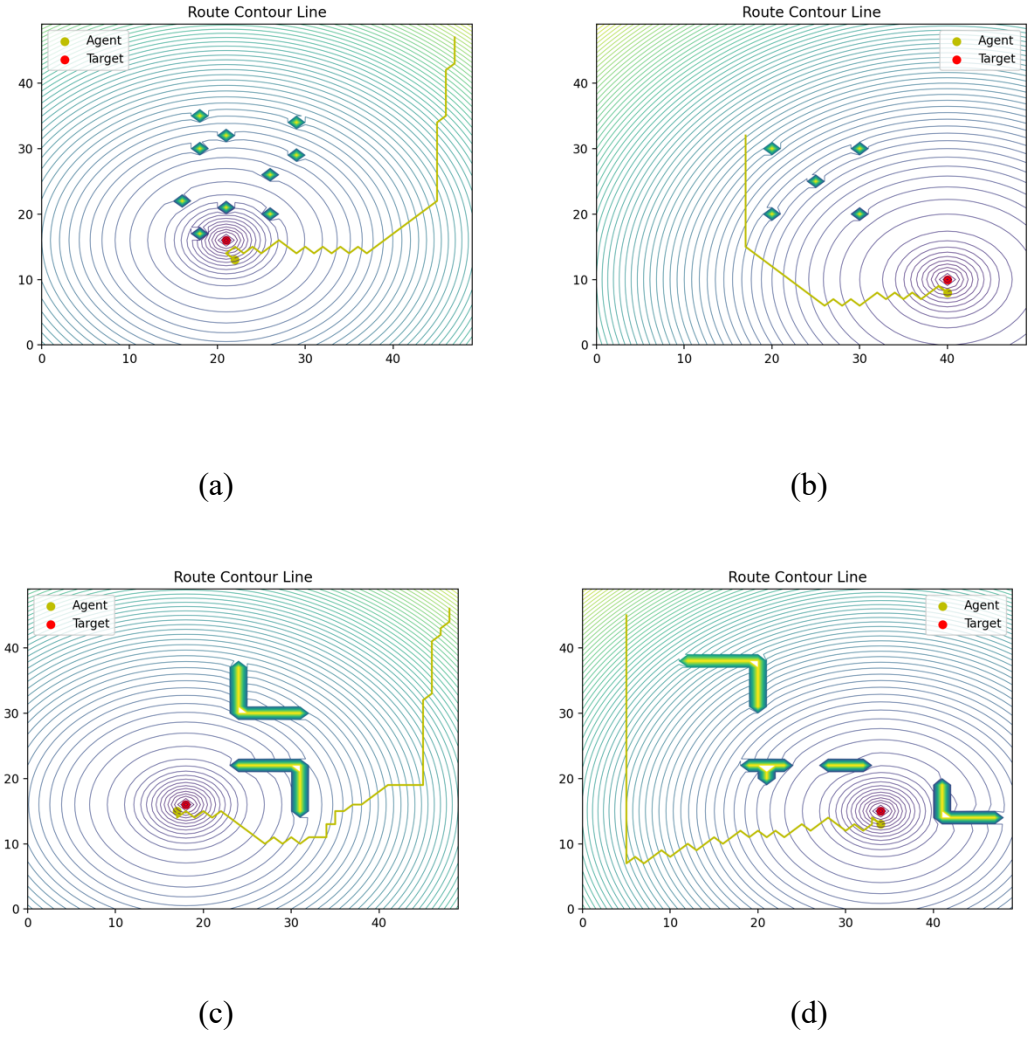


Figure 31: Test of failed path planning in Fig. 30(b) with shorter distance between the target and the robot start position, and test paths planning in Fig. 30(a), (c), and (d) using longer distance between the target and the robot start position

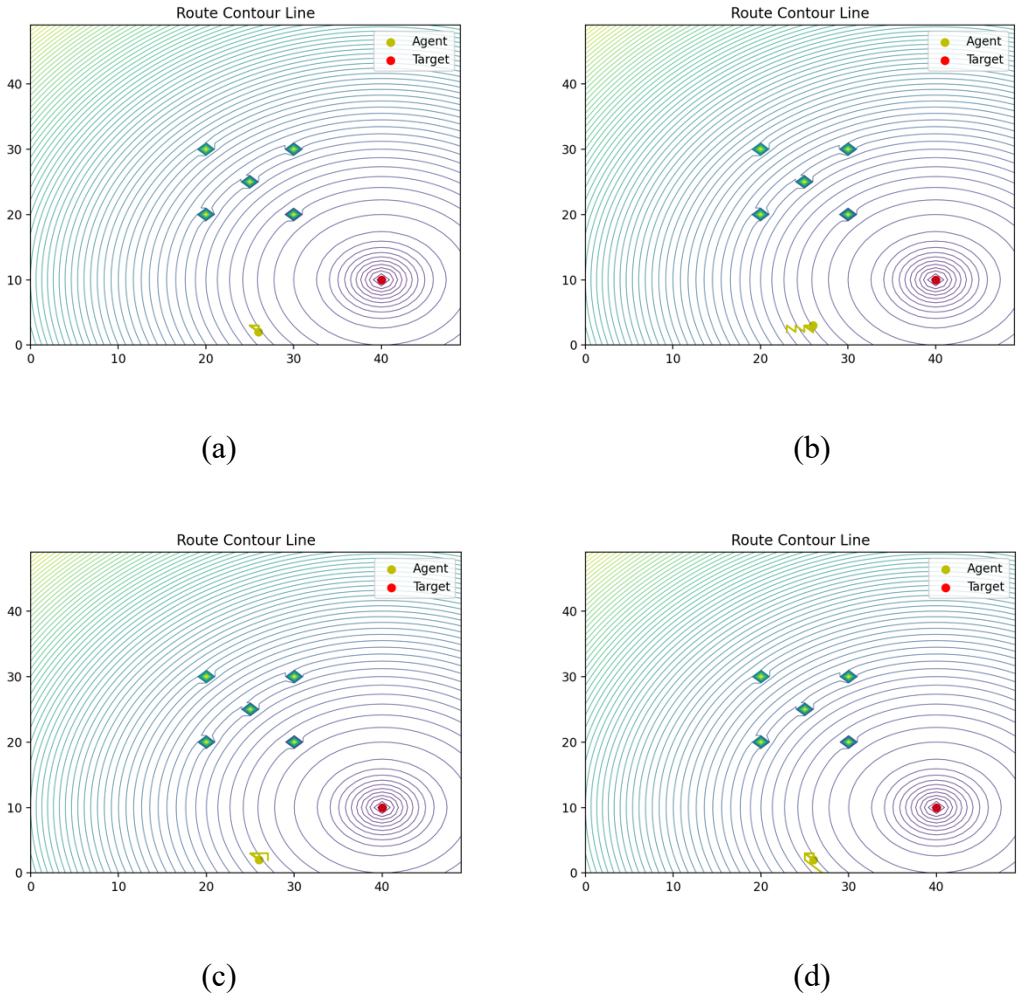


Figure 32: Tests of robot start positions around the robot end position in Fig. 30(b)

Path planning using the deep Q-learning network can be compared to path planning using gradient descent by contrasting the results in Fig. 30 to the results in Fig. 33. The robot moves in the same artificial potential fields in these figures. Note that the robot cannot find its way to the target in any of the fields when it plans the path using gradient descent but always becomes trapped at a local minimum. Thus, the deep Q-learning network improves robot path planning compared to steepest descent although it is not guaranteed to be successful in every potential field and regardless of the initial robot position.

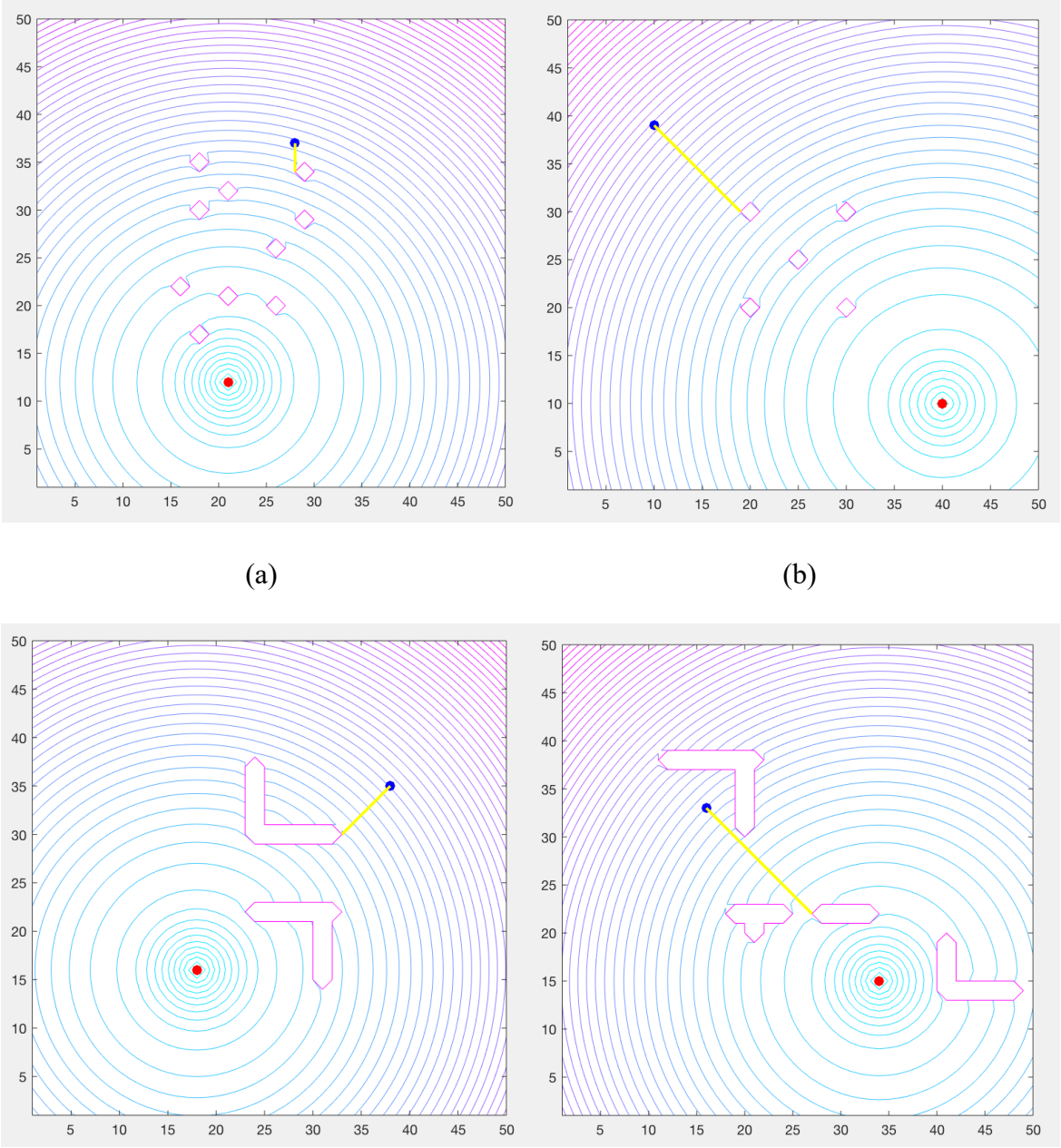


Figure 33: Paths planned by the gradient descent method in the same environment and for the same initial robot position as in Fig. 30: the robot cannot find the target in any of the four tested environment.

6.4 DQN Robot Formation

Path planning in a fixed artificial potential field and four general artificial potential fields are implemented by letting the leader robot use the deep Q-learning network to find the target. The two followers use only gradient descent in an artificial potential field to plan their motions. Fig. 34 shows the paths of the three-robot formation in the four general artificial potential fields from Fig. 30. Fig. 35 and Fig. 36 show the paths of the three-robot formation in the fixed artificial potential fields in Fig. 24 and Fig. 25, respectively. In Fig. 34, Fig. 35, and Fig. 36, the agent (yellow dot) is the leader. The agent2 (blue dot) is the follower1. The agent3 (black dot) is the follower2. The target is red dot. The followers can follow the leader in all tests.

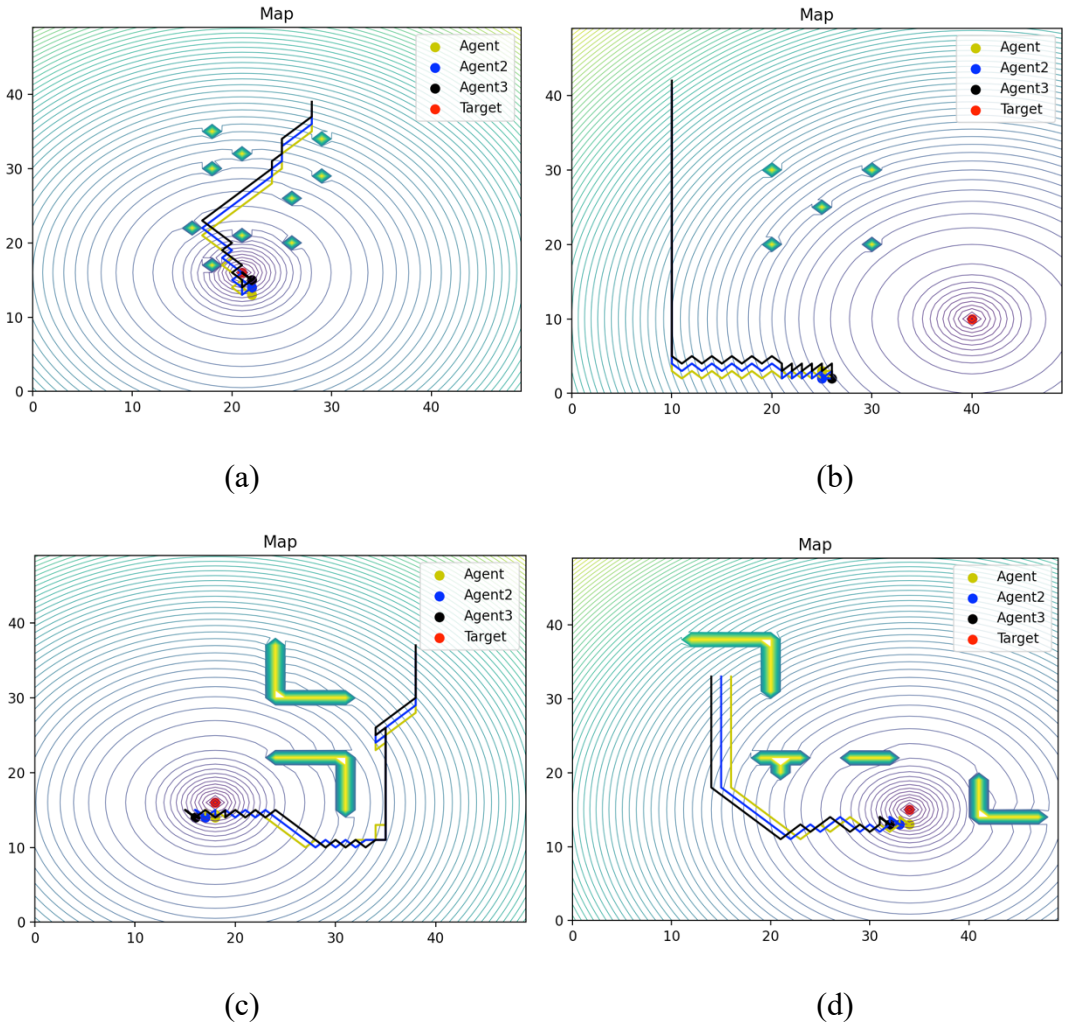


Figure 34: Paths planned by the robot formation in general artificial potential fields: as in the case of single robot path planning, the formation finds the target in the environment in (a), (c), and (d), but fails to reach the target in the environment in (b).

As anticipated, Fig. 35 and Fig. 36 illustrate that the planning for the robot formation is successful when the leader can successfully plan its path to the target, and unsuccessful otherwise. In other word, the path planning for the formation succeeds or fails when the path planning for a single robot succeeds or fails.

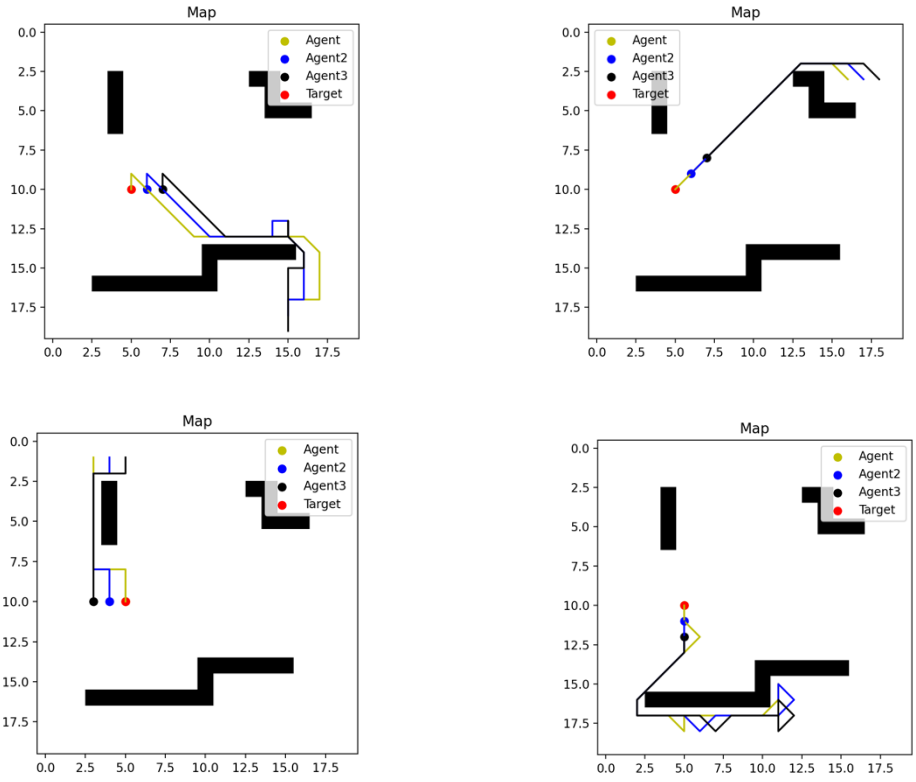


Figure 35: Path planning for a three-robot formation a in fixed artificial potential field, when the obstacles obstruct the direct path between the formation and the target.

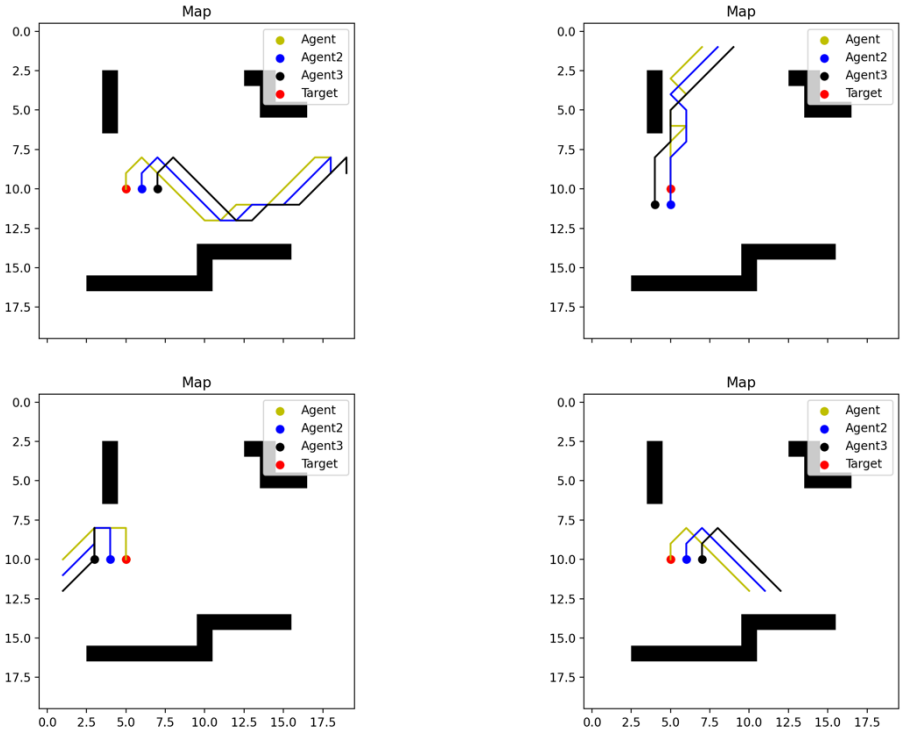


Figure 36: Path planning for a three-robot formation a in fixed artificial potential field, when no obstacles obstruct the direct path between the formation and the target.

7. Conclusion and Recommendations

As proposed in [1], this project uses deep Q-learning network and artificial potential fields to achieve the path planning for a single robot and for a three-robot formation.

The experimental results indicate that a deep Q-learning network can get good results in the fixed artificial potential field created by an environment in which a target and the obstacles are fixed. Out of the 40 tests carried out in this project, the robot finds the target in all but three tests.

However, the deep Q-learning network is less effective at helping the robot to find the target in a general artificial potential field, generated by a target and obstacles at arbitrary position in the environment. When the artificial potential field is not fixed, the deep Q-learning network first learns how to avoid one obstacle and find target. Then, the network learns how to avoid complex obstacles and find target. However, it is hard for the network to remember what it has learned first. During curriculum learning, the robot learns to avoid two specific obstacles and find the targets. However, the robot fails to learn how to avoid other two specific obstacles and cannot find the targets. In the end, three out of four tests in general artificial potential fields are successful. During the training process of the general artificial potential field, the robot start positions are chosen to be the same as in [1]. If the robot start positions are changed to other positions, the training result will be different. The robot start positions in test maps are also the same as in [1]. If the robot start positions are different from [1], it is probable that the robot may not be able to avoid obstacles to reach the targets. Thus, the proposed deep Q network path planning method proposed in [1] is dependent on the initial conditions. A more robots' technique is needed to lessen this dependency.

Similar to the single robot, the three-robot formation successfully finds the target in the fixed artificial potential field. All eight tests are successful. The two followers can follow the leader successfully in the four tested general artificial potential fields. Two potential directions of future work are: (1) path planning using the deep Q-learning network and artificial potential fields in an environment with moving obstacles; and (2) path planning without artificial potential fields by changing the observation to an image map. In terms of path planning for robot formations, collision-free path planning of an arbitrary robot formation, as well as of multiple robot formations simultaneously, will be considered. The goal will be to drive each formation to is target while avoiding the obstacles and the other formations.

Bibliography

1. Q. Yao *et al*., "Path Planning Method With Improved Artificial Potential Field—A Reinforcement Learning Perspective," in *IEEE Access*, vol. 8, pp. 135513-135523, 2020, doi: 10.1109/ACCESS.2020.3011211.
2. Roderick, M., MacGlashan, J., & Tellex, S. (2017). Implementing the deep q-network. *arXiv preprint arXiv:1711.07478*.
3. Zhang, H. Y., Lin, W. M., & Chen, A. X. (2018). Path planning for the mobile robot: A review. *Symmetry, 10(10), 450.*
4. Buniyamin, N., Ngah, W. W., Sariff, N., & Mohamad, Z. (2011). A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development, 5*(2), 151-159.
5. Tang, B., Zhu, Z., & Luo, J. (2016). Hybridizing particle swarm optimization and differential evolution for the mobile robot global path planning. *International Journal of Advanced Robotic Systems, 13*(3), 86.
6. Ningning Qi, Bojun Ma, Xian'en Liu, Zhenxin Zhang and Dongchun Ren, "A modified artificial potential field algorithm for mobile robot path planning," *2008 7th World Congress on Intelligent Control and Automation*, 2008, pp. 2603-2607, doi : 10.1109/WCICA.2008.4593333.
7. Kovács, B., Szayer, G., Tajti, F., Burdelis, M., & Korondi, P. (2016). A novel potential field method for path planning of mobile robots by adapting animal motion attributes. *Robotics and Autonomous Systems*, 82, 24-34.
8. Zhu, H., Wang, J., & Li, J. (2013, May). A novel potential field method for path planning of mobile robot. In *2013 25th Chinese Control and Decision Conference (CCDC)* (pp. 2811-2814). IEEE.
9. Qidan, Z., Xunyu, Z., Jingqiao, Z., & Jian, W. (2008, July). Improved path planning and tracking methods for mobile robot. In *2008 Chinese Control and Decision Conference* (pp. 1290-1295). IEEE.
10. Sabudin, E. N., Omar, R., CKAN, H., & Melor, C. K. (2016). Potential field methods and their inherent approaches for path planning. *ARPN J. Eng. Appl. Sci*, *11*(18), 10801-10805.
11. Chen, G., & Liu, J. (2019). Mobile robot path planning using ant colony algorithm and improved potential field method. *Computational intelligence and neuroscience, 2019.*
12. Orozco-Rosas, U., Montiel, O., & Sepúlveda, R. (2019). Mobile robot path planning using membrane evolutionary artificial potential field. *Applied Soft Computing*, 77, 236-251.
13. Bayat, F., Najafinia, S., & Aliyari, M. (2018). Mobile robots path planning: Electrostatic potential field approach. *Expert Systems with Applications*, 100, 68-78.

14. Chuan-ling, L., Jing, T., & Jing-yu, Y. (2011, July). Path planning of mobile robot using new potential field method in dynamic environments. In *2011 Seventh International Conference on Natural Computation* (Vol. 2, pp. 1011-1014). IEEE.
15. Hong, Z., Liu, Y., Zhongguo, G., & Yi, C. (2011, April). The dynamic path planning research for mobile robot based on artificial potential field. In *2011 International Conference on Consumer Electronics, Communications and Networks (CECNet)* (pp. 2736-2739). IEEE.
16. Lv, L., Zhang, S., Ding, D., & Wang, Y. (2019). Path planning via an improved DQN-based learning policy. *IEEE Access, 7*, 67319-67330.
17. Bae, H., Kim, G., Kim, J., Qian, D., & Lee, S. (2019). Multi-robot path planning method using reinforcement learning. *Applied sciences*, *9*(15), 3057.
18. J. Li *et al*, "An improved DQN path planning algorithm," *The Journal of Supercomputing, vol. 78, (1)*, pp. 616-639, 2021;2022;.
19. S. Zhou *et al*, "A deep Q-network (DQN) based path planning method for mobile robots," in 2018, . DOI: 10.1109/ICInfA.2018.8812452.
20. Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
21. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
22. Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., ... & Saurous, R. A. (2017). Tensorflow distributions. *arXiv preprint arXiv:1711.10604*.