Research on 2D Obstacle Avoidance Model Based on Field Rescue

YE ZHOU, Hohai University, Nanjing, Jiangsu, China

NENGXIN MOU, Hohai University, Nanjing, Jiangsu, China

Research on 2D Obstacle Avoidance Model Based on Field Rescue

Ye Zhou
College of Computer Science and Software Engineering
Hohai University
Nanjing, Jiangsu, China
2206010209@hhu.edu.cn

Nengxin Mou
College of Computer Science and Software Engineering
Hohai University
Nanjing, Jiangsu, China
2306010143@hhu.edu.cn

Abstract

This paper presents a novel 2D obstacle avoidance model tailored for field search and rescue robots. Leveraging advanced deep reinforcement learning techniques, the proposed model aims to enhance the robots' navigation capabilities in dynamic and unpredictable environments. We evaluated four distinct reinforcement learning models—Deep Q-Learning (DQL), Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), and Deep Q-Network (DQN)—to identify the most effective approach for real-time obstacle avoidance. Experimental results demon- strata that the DQL model outperforms the others in terms of success rate, average reward, and stability, making it particularly suitable for complex field operations. This research contributes to the development of robust and efficient navigation strategies for autonomous search and rescue missions, with potential implications for broader applications in robotics.

CCS Concepts

• **Computing methodologies** → Artificial intelligence; Planning and scheduling; Robotic planning.

Keywords

2D Obstacle Avoidance, Path Planning, Reinforcement Learning, Robust optimization

ACM Reference Format:
Ye Zhou and Nengxin Mou. 2024. Research on 2D Obstacle Avoidance Model Based on Field Rescue. In *2024 6th International Conference on Mechatronics Systems and Control Engineering (ICMSCE 2024), September 27–29, 2024, Singapore, Singapore*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3719477.3719479

1 Introduction

Field search and rescue operations are critical missions often conducted in dangerous and unpredictable environments. Given the short critical window for such operations, there is a high demand for both precision and efficiency to provide timely assistance to those in distress. Robots capable of conducting search and rescue missions in the field have garnered significant attention because they can traverse hazardous terrains and perform tasks that might be dangerous for rescue personnel. One of the main challenges in deploying robots for search and rescue missions is the development of effective obstacle avoidance algorithms to ensure that robots can navigate safely and efficiently in complex field environments.

Robotics' basic problem is obstacle avoidance. Robots must overcome trash, uneven terrain, and other challenges to reach distressed people in field search and rescue missions. These robots need an efficient two-dimensional obstacle avoidance model in order to autonomously negotiate potential impediments, which will allow them to increase their search and rescue skills.

Within the scope of this study, a novel two-dimensional obstacle avoidance model for field search and rescue robots is presented. A strong set of algorithms is used by the model in order to identify and circumvent barriers real time. The use of our technique makes use of machine learning to enhance the environmental awareness and decision-making capabilities of rescue robots, therefore enabling them to function in more challenging environments.

The findings of this study make three significant contributions. To begin, we first developed a complete framework for obstacle avoidance in two dimensions that is capable of being simply integrated into search and rescue robots. Subsequently, we implemented algorithms that are capable of adjusting the path of the robot in response to the data collected by sensors in order to choose the most expedient path to the desired destination. Third, we conducted a series of experimental scenarios to demonstrate the effectiveness of the model in a variety of field settings. In the following sections, we will outline the strategy and implementation of our model, review the research that has been done on obstacle avoidance for search and rescue robots, and provide the results of our experiments. Future research and the implications of our findings will be discussed in the concluding comments.

2 Literature Review

2.1 Traditional Methods

Classical methods of obstacle avoidance and route planning have been the driving force behind the creation of fundamental concepts that continue to be indispensable to the field of current robotics. The fact that these methods are often straightforward and based on established principles does not change the fact that they have been shown to be effective in a broad variety of circumstances.

2.1.1 Bug Algorithms. To avoid obstacles and accomplish their objectives, robots may use bug algorithms, which are the simplest and most simplistic methods. The robot navigates its way back to the target by following the outlines of the obstacles in Bug1 and Bug2 [1]. Whenever the robot is simply aware of its immediate surroundings, these algorithms are able to function well in situations that are un- predictable or dynamic. If the robot is confronted with a complex environment, such as a labyrinth, it is possible that these tactics will not be effective [2].

2.1.2 Vector Field Histogram (VFH).. Improving real-time obstacle avoidance is the goal of the Vector Field Histogram (VFH) method developed by Borenstein and Ko-ren. Additionally, the VFH algorithm makes use of sensor data to generate a polar histogram of obstacle density, which enables the robot to travel through the locations with the least amount of congestion [3]. Under dynamic conditions in which the robot is required to react quickly, this method is effective [4]. It is possible that the histogram resolution is not good enough for VFH to accurately depict the environment in an environment that is narrow or crowded.

2.1.3 Potential Field Methods. When using potential field techniques, the robot is modeled as a particle that is traveling through a field of forces. The objective is supposed to exert an appealing force, while the obstacles are supposed to exert a repulsive force. After that, the robot travels in the direction of the force vector that was generated, which should take it in the direction of the objective while allowing it to avoid obstacles [5].n spite of the fact that this method is computationally efficient and can be applied in real time, it is prone to local minima, which is a situation in which the robot reaches a point where it is unable to go forward because it is trapped in the middle of barriers [6].

2.2 Deep Learning and Reinforcement Learning Methods

The power of robots to traverse complicated surroundings has been considerably improved as a result of recent advancements in depth learning and reinforcement learning. Large datasets and complex computer models are used by these technologies in order to discover the most effective routes.

2.2.1 Deep Learning. Deep learning, especially through Convolutional Neural Networks (CNNs), has transformed how robots process visual information. CNNs enable robots to interpret their environment from raw sensor inputs and use this data to make navigation decisions. CNNs have been used to create end-to-end learning systems that translate sensory inputs to navigation instructions [7, 8]. This strategy works well in visual-rich contexts where standard approaches may fail.

2.2.2 Reinforcement Learning. Reinforcement Learning (RL) educates agents to make choices by maximizing cumulative rewards and interacting with their environment. Classic RL algorithms like Q-learning can teach robots optimal navigation [9]. Traditional RL algorithms demand plenty of training data and struggle with high-dimensional state spaces [10].

2.2.3 Deep Reinforcement Learning. Deep learning and reinforcement learning allow neural network-based robots to handle high-dimensional input spaces. Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) train robots for complex navigation [11, 12]. DQN's neural network mimics the Q-value function, enabling the robot to learn effective policies from sensory input [9]. However, PPO stabilizes policy updates, improving continuous control [11].

2.3 Current Challenges in 2D Obstacle Avoidance

Despite significant advances, obstacle avoidance still faces several challenges, particularly in dynamic conditions and when dealing with computational limits.

2.3.1 Dynamic Environments. Navigation of robotic systems in dynamic settings, when barriers alter unexpectedly, remains problematic. Traditional methods like potential fields fail in these situations because they make static environmental assumptions [5]. Re- cent reinforcement learning algorithms may adapt to dynamic situations. These technologies involve real-time robot navigation strategy learning and update [13].

2.3.2 Real-Time and Computational Complexity. Making decisions quickly and computing effectively are both necessary components of real-time obstacle avoidance. For the purpose of avoiding collisions, the algorithms need to evaluate the data from the sensors and make navigational decisions in milliseconds. High-dimensional sensory inputs like vision systems make this difficulty worse. Experience replay and transfer learning in deep reinforcement learning enable robots to reuse earlier experiences to improve learning efficiency [14].

2.4 Hybrid Approaches and Advanced Techniques

Hybrid approaches, which mix ancient methods with contemporary techniques, have the potential to deliver the best of both worlds, particularly in complicated contexts where a single strategy is not sufficient.

2.4.1 Hybrid Approaches. The goal of hybrid methods is to achieve a compromise between computing efficiency and flexibility. This is accomplished by combining traditional route planning algorithms with reinforcement learning techniques. As an example, a robot may use a conventional algorithm such as A* for the purpose of global route planning, while depending on reinforcement learning for the purpose of local obstacle avoidance in situations that are dynamic [15]. Because of this combination, the robot is able to design an ideal course while maintaining the flexibility to avoid unanticipated impediments.

2.4.2 Advanced Techniques. The limits of fully model-free systems have been solved by the development of more advanced techniques, such as model-based reinforcement learning. By incorporating a model of the environment into the learning process, these techniques can predict the outcomes of actions and plan accordingly, resulting in more efficient learning and better performance in complex environments [14]. For instance, self-configuring path planning systems use a combination of model- based predictions and real-time adjustments based on deep reinforcement learning to effectively navigate cluttered environments.

2.5 Related Work

Research on 2D obstacle avoidance has explored various approaches, each with its strengths and limitations. This section compares these methods and high- lights the limitations of existing models.

2.5.1 Limitations of Existing Models. A great number of conventional methodologies, such as potential field and vector field histogram approaches, run into challenges when used in contexts that are complicated and include dynamic obstacles. There are a number of issues that might arise as a result of these approaches, including local minima and oscillations in the robot's route [3, 5]. These methods often depend on assumptions about the environment that may not hold true in realistic situations.

2.5.2 Comparison of Related Research. New deep reinforcement learning techniques allow robots to learn from experience and adapt to new settings, overcoming some of the limits of conventional approaches. These methods have drawbacks, such as sampling efficiency and the necessity for huge training data. Classical-modern hybrid methods are promising since they blend their strengths [15].

3 Methodology

3.1 Problem Formulation

This study uses Deep Q-Learning to create a 2D obstacle avoidance model for field search and rescue robots. The robot navigates a grid to find the best route to an objective while avoiding obstacles. The agent must adapt to shifting impediments to attain the objective effectively in a dynamic environment.

3.2 RL Related Definition

3.2.1 RL Agent Definition. State Introduction: The current location of the agent in the two-dimensional grid is represented by the state s_t at time t on the representation. Given the grid's dimensions $n \times n$, the state space consists of n^2 possible states.

Action Introduction: The action a_t available to the agent at time t consists of four discrete movements: up, down, left, and right. These actions move the agent one step in the corresponding direction within the grid.

3.2.2 Q-Learning Definition. Q-Learning is a model-free reinforcement learning algorithm where the agent learns a policy that maximizes the cumulative reward by updating Q-values for each state-action pair. The update rule for Q-learning is defined as:

$$Q\left(st,\ at\right)\ =Q\left(st,\ at\right)+\alpha\left[\ rt\ +\ \gamma\, max\, Q\left(st+1,\ a'\right)\ -\ Q\left(st,\ at\right)\right] \tag{1}$$

where:

- $Q(s_t, a_t)$ is the Q-value for the current state-action pair.
- α is the learning rate.
- r_t is the immediate reward after taking action a_t in state s_t.
- γ is the discount factor.
- s_{t+1} is the resulting state after taking action a_t.
- $\max_{a'} Q(s_{t+1}, a^{'})$ represents the maximum estimated future reward given the next state s_{t+1}.

3.3 Hyperparameter Definition

3.3.1 Network Architecture. The Q-network is a deep neural network used to approximate the Q-value function. The architecture of the network is as follows:

- Input Layer: The input layer consists of 2 neurons representing the agent's position in the grid (x, y).
- Hidden Layers: The network includes two hidden layers, each containing 24 neurons, with ReLU activation functions to introduce non-linearity.
- Output Layer: The output layer consists of 4 neurons, corresponding to the Q-values for the four possible actions (up, down, left, right).

3.3.2 Epsilon. Epsilon ($\in$) in the epsilon-greedy policy controls the trade-off between exploration and exploitation:

- Initial Epsilon: $\in= 1.0$ at the start of training, meaning the agent initially explores the environment by taking random actions.
- Epsilon Decay: Over time, $\in$ decays exponentially by a factor $\in_{decay}$, en couraging the agent to exploit the learned policy.
- Minimum Epsilon: $\in_{min} = 0.01$, ensuring that the agent still explores occasionally even in later stages of training.

The epsilon-greedy policy is mathematically expressed as: If $rand \leq \in$, then:
a_t = random action
Otherwise:
$a_t =_a Q(s_t, a; \theta)$

4 Experiment and Results

4.1 Experiment Setup

4.1.1 Hardware Configuration.

- Processor: Intel Core i7-12700K
- Memory: 16 GB
- GPU: NVIDIA RTX 3050 Ti

4.1.2 Software Environment.

- Python version: 3.8.3
- Libraries:

Gymnasium: Used for creating and managing the reinforcement learning environment.

NumPy: Numerical computing library for matrix operations and random number generation.

PyTorch: Deep learning framework used for implementing and training the Deep Q-Learning (DQL) model.

Stable Baselines3: Provides various reinforcement learning algorithms such as PPO, A2C, and DQN.

Matplotlib: Visualization library used for plotting the experimental results.

Psutil: Used for monitoring system resource usage (CPU, memory).

4.1.3 Experiment Environment.

- Custom Environment: RobustEnv

Map Size: 100x100 2D grid where the agent moves.

Action Space: 4 discrete actions representing movement in the four directions: up, down, left, and right.

Observation Space: The agent's position coordinates represented as a 2D vector.

- Obstacle Configuration:

Hard Obstacles: 1000 obstacles that block the agent's path. Collision results in a failed episode with a significant penalty (-10 points).

Soft Obstacles: 1000 obstacles that the agent can pass through but incur a minor penalty (-1 point) and reduce the maximum steps for the episode.

- Start and Goal: Randomly generated on the map, ensuring that the agent and the target are not in the same initial position, with at least one viable path between them.

4.1.4 *Training and Testing.*

- Models:

DQL (Deep Q-Learning): Value-based reinforcement learning algorithm implemented using PyTorch.

PPO (Proximal Policy Optimization): Policy-based reinforcement learning algorithm implemented using Stable Baselines3, suitable for both continuous and discrete action spaces.

A2C (Advantage Actor-Critic): Distributed reinforcement learning algorithm implemented using Stable Baselines3, with a synchronous update strategy.

DQN (Deep Q-Network): An extended version of Q-learning implemented using Stable Baselines3, combined with deep learning capabilities.

- Training Process:

Training is conducted in the same environment settings for each model to ensure fair comparison.

Training Episodes: Each model is trained for 100,000 episodes, with a maximum of 4,000 steps per episode.

Optimizer:

∗ The DQL model uses the Adam optimizer with a learning rate of 0.001.

∗ PPO, A2C, and DQN models use the default optimizers and parameter settings provided by Stable Baselines3.

Model Saving: The weights of each model are saved after training for future loading and testing.

- Testing Process:

Testing is performed in the same map environment for each model to ensure fair comparison.

Testing Method: Each model is tested on 10,000 different maps. Met- rics such as success rate, average reward, and stability are compared across different models.

4.2 Results Analysis

4.2.1 *Success Rate.*

- Definition: The success rate is the proportion of episodes where the agent successfully reaches the target location.

4.2.2 *Average Reward.*

- Definition: The average reward is the mean cumulative reward per episode, with higher rewards indicating better agent performance in the environment.

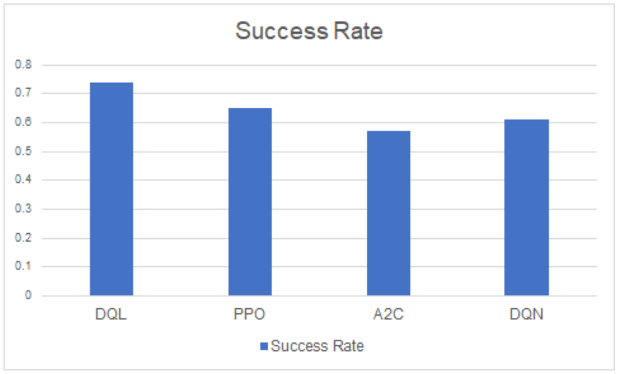

Figure 1: Success rate comparison across different models.

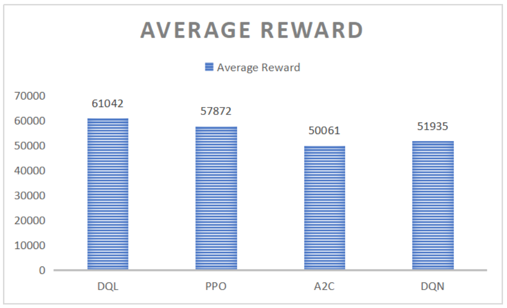

Figure 2: Average reward comparison across different models.

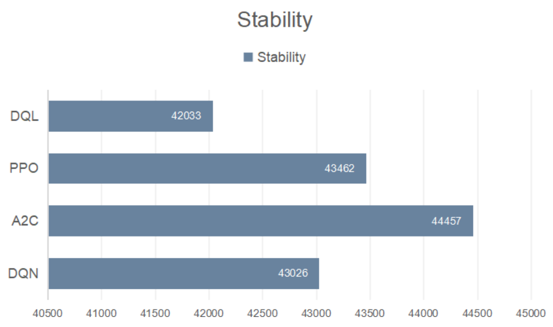

Figure 3: Stability comparison across different models.

4.2.3 *Stability.*

- Definition: Stability is represented by the standard deviation of the reward values, with lower standard deviations indicating more consistent model performance.

4.2.4 *Path Tracing Example.*

- The figure below shows an example of the path traced by the agent during testing. The agent starts at the initial position and navigates through the grid to reach the target while avoiding obstacles.

Table 1: Comparison of DQL, PPO, A2C, and DQN Models

Model	Success Rate	Average Reward	Stability
DQL	0.70	61042	42033
PPO	0.65	57872	43482
A2C	0.60	50061	44457
DQN	0.63	51935	43028

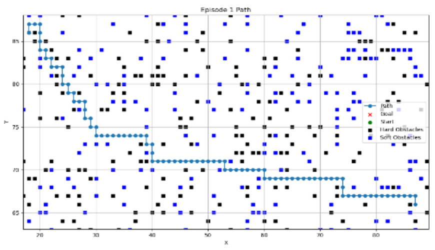

Figure 4: Example of agent's path tracing on a 100x100 grid.

4.3 Conclusion and Future Work

4.3.1 Conclusion. Based on the comparative analysis, the following conclusions can be drawn:

- The DQL model demonstrates superior performance across multiple met rics. It shows the highest success rate, indicating strong navigation capabilities in complex environments.
- The DQL model also achieves the highest average reward, reflecting its effective performance in the environment.
- Additionally, the DQL model exhibits excellent stability, with a low stan- dard deviation in reward values, indicating consistent performance across different test environments.

Through experimental data, it can be demonstrated that the DQN model in this article performs well in different environments, highlighting its navigation ability, stable performance, and high success rate in complex scenarios. This indicates that the DQN model has better robustness in handling real-time obstacle avoidance tasks.

4.3.2 Future Work.

- Hyperparameter Optimization: Further systematic optimization of the DQL model parameters will be attempted to improve success rate and stability.
- Increased Environment Complexity: The environment complexity will be increased, such as by introducing dynamic obstacles and more types of obstacles, to further test the model's robustness and adaptability.

References

[1] V. J. Lumelsky and A. A. Stepanov, "Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape," *Algorithmica*, vol. 2, no. 1, pp. 403-430, 1987.
[2] I. Kamon, E. Rivlin, and E. Rimon, "Path Planning Strategies for a Point Robot Moving Amidst Unknown Obstacles of Arbitrary Shape," *Algorith- mica*, vol. 20, no. 4, pp. 282-312, 1998.
[3] J. Borenstein and Y. Koren, "The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots," *IEEE Transactions on Robotics and Au- tomation*, vol. 7, no. 3, pp. 278-288, 1991.
[4] S. Ban, H. Umetsu, and M. Abe, "VFH+: An Enhanced Vector Field Histogram Method for Mobile Robot Obstacle Avoidance," in *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1572-1577, 1998.
[5] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90- 98, 1986.
[6] P. Arnold and P. Venturino, "Comparison of Obstacle Avoidance Methods for Mobile Robots," in *Proceedings of the 1993 IEEE* International *Con- ference on Robotics and Automation*, vol. 2, pp. 896-901, 1993.
[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.
[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.
[9] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.
[10] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279-292, 1989.
[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Prox- imal Policy Optimization Algorithms," arXiv preprint arXiv:1707.06347, 2017.
[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous Control with Deep Reinforcement Learning," arXiv preprint arXiv:1509.02971, 2015.
[13] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to Navigate in Complex Environments," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
[14] B. Sangiovanni, G. P. Incremona, M. Piastra, and A. Ferrara, "Self- Configuring Robot Path Planning with Obstacle Avoidance via Deep Re- inforcement Learning," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 1024-1031, 2021.
[15] A. Sgorbissa and R. Zaccaria, "Planning and Obstacle Avoidance in Mobile Robotics," *Robotics and Autonomous Systems*, vol. 61, no. 9, pp. 1162-1174, 2013.