

Article

Spacecraft Safe Proximity Policy Based on Graph Neural Network Safe Reinforcement Learning

Heng Zhou ^{1,2}, Jingxian Wang ^{1,2}, Monan Dong ^{1,2}, Yong Zhao ^{1,2}, Yuzhu Bai ^{1,2,*} and Rong Chen ^{1,2,*}

¹ College of Aerospace Science and Engineering, National University of Defense Technology, Changsha 410073, China; zhouheng18@nudt.edu.cn (H.Z.); wangjingxian19@nudt.edu.cn (J.W.); dong_mn@nudt.edu.cn (M.D.); zhaoyong@nudt.edu.cn (Y.Z.)

² State Key Laboratory of Space System Operation and Control, Changsha 410073, China

* Correspondence: baiyuzhu06@nudt.edu.cn (Y.B.); chenrong14@nudt.edu.cn (R.C.)

Abstract

Spacecraft safe proximity, as a critical component of on-orbit servicing missions, primarily encounters the following two challenges: the partial observability of the environment surrounding the service spacecraft and the necessity to evade uncertain obstacles. A safe reinforcement learning algorithm based on a graph neural network is proposed to address the constrained Markov decision problem in partially observable scenarios for spacecraft safe proximity missions. A graph neural network mechanism is introduced to solve the problem of dynamic variations in the quantity and location of obstacles in the observation area of the service spacecraft. The graph attention network is used to facilitate the extraction of feature information from the graph structure, which is then utilized as input for the subsequent reinforcement learning algorithm. The Soft Actor–Critic–Lagrangian algorithm is adopted to deal with the problems of tuning reward function parameters and balancing safety and optimality. By introducing Lagrange multipliers, the constrained optimization problem is transformed into an unconstrained optimization problem. In order to verify the effectiveness of the algorithm proposed in this paper, a spacecraft safe proximity environment model with dynamic obstacles is constructed, and the GAT-SACL algorithm proposed in this paper is validated by the Monte Carlo shooting method. The results show that the GAT-SACL algorithm possess excellent exploratory characteristics and delivers significant advantages in balancing optimality and safety.

Keywords: safe proximity mission; safe reinforcement learning; graph neural network; soft actor critic–Lagrangian



Academic Editor: George Z. H. Zhu

Received: 24 January 2026

Revised: 19 February 2026

Accepted: 23 February 2026

Published: 26 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

On-orbit services [1], an efficient method for extending satellite lifespan, have experienced rapid development in recent years, exemplified by the success of the MEV1&2 commercial project. Spacecraft safe proximity is a prerequisite for performing on-orbit services, which has garnered significant attention from researchers. The “safe proximity” encompasses two concepts [2]: the first, “proximity,” requires the service spacecraft to maneuver close to the target and maintain relative stability; the second, “safety,” requires the service spacecraft to avoid colliding with the target spacecraft and obstacles, such as space debris, during the approach process to prevent spacecraft from suffering damage.

Nowadays, it is difficult for ground detection equipment to accurately perceive space debris with a radius of less than 10 cm in space. There may exist several small pieces

of space debris along the pre-planned trajectories according to ground observation data, which seriously disrupt spacecraft operations. Consequently, it is essential to employ onboard sensors for the real-time detection of minor debris surrounding the spacecraft and to adapt orbital maneuvering strategies accordingly. In traditional trajectory planning methods based on optimization theory [3,4], firstly a secure path is obtained by employing optimization algorithms; subsequently, a closed-loop controller is devised to guide the spacecraft along the optimized trajectory, and ultimately, a compensation controller is implemented to quickly correct the deviation between the actual trajectory and the preliminary planned trajectory. However, there are problems with the trajectory planning approach for spacecraft obstacle avoidance based on optimization techniques, such as high reliance on initial guesses and poor adaptability to dynamic obstacles. Moreover, in traditional analysis methods, represented by artificial potential functions [5], the estimated trajectories sometimes may converge on local minima, hence increasing the likelihood of obstacle avoidance failures and threatening the security of spacecraft. Therefore, research on how to design safe and real-time trajectory planning and control strategies to adapt to the complex and rapidly changing space environment is critical for ensuring safe operation of spacecraft, especially for safe proximity.

In recent years, reinforcement learning has achieved significant results in the field of intelligent decision-making [6,7]. By utilizing the advantages of neural network fitting function features and parameterizing agent strategies through neural networks, it can effectively handle various complex task scenarios. Many researchers have applied reinforcement learning algorithms to solve spacecraft trajectory planning and control problems [8], including orbital rendezvous [9–11], space debris warning and automatic obstacle avoidance [12] and orbital pursuit–evasion [13,14]. Typically, among existing papers of single-spacecraft scenarios, the controlled spacecraft is regarded as an agent in reinforcement learning theory, and the dynamic models of the controlled spacecraft and other surrounding spacecraft or obstacles are constructed as the environment. The current state and action of the agent are taken as input to the environment, and the state and reward of the next moment can be obtained through interaction with the environment. The agent possesses two neural networks: an actor network that offers policy and a critic network that offers estimation. To optimize the cumulative reward function, many methods, including gradient descent, are employed. In complex task scenarios, to prevent the agent from falling into unsafe situations, the agent is subject to various state and action constraints. The evaluation of the optimality of policy objectives and the satisfaction of constraints is mainly reflected in the reward function. Therefore, the design and tuning of the reward function are the most core and difficult problems to solve trajectory planning and control problems using reinforcement learning. Qu et al. [15] constructed a spacecraft rendezvous control architecture based on Deep Deterministic Policy Gradient (DDPG) [16] for the problem of spacecraft proximity maneuvering and rendezvous, where a reward function consisting of five parts is designed, including a rendezvous and docking reward, collision avoidance reward, docking direction reward, velocity reward, and fuel consumption reward. Compared with traditional methods, the proposed strategy effectively reduced energy consumption by 16.44%. Sharma et al. [17] proposed a spacecraft rendezvous guidance method based on reinforcement learning considering spacecraft safety for the continuous-finite-thrust spacecraft rendezvous and docking problem. The reward function was designed by combining error-based rewards and rewards based on obstacle warning and collision avoidance constraints, and the model was trained by using DDPG and Proximal Policy Optimization (PPO) [18] methods, respectively. However, in traditional reinforcement learning algorithms, security constraint penalties are usually included in the reward function. Although the reward function constructed according to this method

is more intuitive, it cannot balance optimization performance and security. If the safety penalty is too small, it is likely that risks cannot be avoided; however, if the security penalty is too high, the agent tends to adopt conservative decisions and may not even be able to fully explore the environment [19]. Therefore, security reinforcement learning that focuses on state constraints has emerged [20].

In 2015, García et al. [21] initially defined safe reinforcement learning as a learning strategy that maximizes expected reward during the learning and/or deployment process to assure system performance and/or satisfy safety constraints. In contrast to the Markov decision process (MDP) in conventional reinforcement learning, a cost function is incorporated into the constrained Markov decision process (CMDP), corresponding to safe reinforcement learning, which is employed to define the safety constraints distinctly. Ha et al. [22] introduced the Soft Actor–Critic–Lagrangian (SAC–Lagrangian) architecture for training quadrupedal robots in walking, ensuring that the agent adheres to the maximum limits of pitch and roll angles at every time step. The Lagrange multiplier is introduced into the classical SAC algorithm [23] to convert the optimal problem with cumulative cost constraints into an unconstrained optimal problem, thereby transforming it into a gradient ascent of the strategy and a gradient descent of the Lagrange multiplier. Subsequently, the Lagrange multiplier method for solving safe reinforcement learning problems has been utilized across several reinforcement learning architectures. The OpenAI team [24] provides architectures such as Proximal Policy Optimization–Lagrangian (PPO–Lagrangian) and Trust Region Policy Optimization–Lagrangian (TRPO–Lagrangian) in Safety Gym, which is typically utilized as a benchmark. Currently, certain scholars have applied safe reinforcement learning algorithms to deal with spacecraft trajectory planning tasks. Mu et al. [25] proposed an orbital maneuver strategy based on Penalized Proximal Policy Optimization (P3O) [26] for the task of spacecraft pulse maneuvers aimed at avoiding multiple pieces of space debris while maintaining regular on-orbit operations. This strategy significantly diminished the frequency of spacecraft collision violations and incorporated Long Short-Term Memory (LSTM) networks to extract the states of varying quantities of space debris, thereby enhancing the algorithm's scalability and feature extraction capabilities. Consequently, safe reinforcement learning algorithms can effectively handle collision avoidance constraints, greatly improving the safety of spacecraft conducting on-orbit missions.

In the scenario of spacecraft safe proximity missions, a large amount of space debris floats around the target spacecraft, interfering with the trajectory planning of the service spacecraft. As the service spacecraft approaches the target spacecraft, it can observe the space debris within a specific range in real time using sensors and obtain the real-time on-orbit situation of the target spacecraft. During a certain duration, the service spacecraft only needs to consider avoiding those obstacles with potential collision risks, while other obstacles that are located at a safe distance or even outside the observation area will not influence the decision-making of the current service spacecraft. Therefore, the primary issue addressed in this paper is to design a safe proximity strategy that can adapt to obstacles in the observation range with real-time variations in state and quantity and extract effective information among obstacles with varying positions and collision risk levels. In spacecraft swarms, graph theory is typically employed to delineate the connectivity relationships among swarm members [27], where nodes represent data elements and contain their intrinsic information, while edges represent the interrelationships between data and reflect the distinctive information of the graph structure, mirroring the relationship between spacecraft. Consequently, it is deemed appropriate to incorporate graph theory concepts to illustrate the interaction between service spacecraft and obstacles. A graph neural network (GNN) is a type of neural network that can be directly trained on graph structure, whose

primary function is to process node or structural features in graph-structured data via information transfer, transformation, and aggregation among vertices. GNNs have been extensively used for representing and learning configurations such as graphs, point clouds, and manifolds and have yielded substantial results in structured scenarios such as social networks, recommendation systems, physical systems, chemical molecule prediction, and knowledge graphs [28,29]. In the domain of unmanned aerial vehicles (UAVs), numerous researchers have integrated graph neural networks with reinforcement learning to address the challenge of drone swarm control [30]. Zhao et al. [31] proposed a Multi-Agent Potential Field function Learning algorithm utilizing position-based graph attention (PGAT-MAPFL) to solve the control problem of large-scale UAV swarms. The algorithm took UAVs, obstacles, and reference points as nodes and the relationships between UAVs, between UAVs and obstacles, and between UAVs and reference points as edges. PGAT was utilized to extract graph structural information and enhance the adaptability of UAV swarms to dynamic environments. Zhao et al. [32] subsequently proposed a multi-agent reinforcement learning algorithm based on graph attention networks to address the collaborative search and tracking problem of UAV swarms. Yang et al. [33] proposed a multi-agent reinforcement learning algorithm based on graph attention networks and partially observable mean fields for the coordination and adversarial tasks of UAV swarms under communication environment constraints. The graph attention module delineated the influence of adjacent agents on the central agent, while the mean field module approximated this influence as the average effect of pertinent adjacent agents. In the domain of aerospace, numerous researchers have applied graph neural networks for spacecraft fault detection [34] and mission planning [35] problems, yet there is a paucity of studies integrating graph neural networks with reinforcement learning techniques to address challenges in spacecraft orbit design.

This paper proposes a graph-based safe reinforcement learning method to tackle the challenges of safe proximity trajectory planning and control for spacecraft, aiming to guide the service spacecraft to the expected point while avoiding collision with high-value target spacecraft and dynamic space debris. The contributions of this paper are as follows:

(1) To address the issue of the varying dynamic obstacles in the observation area, the GAT mechanism is introduced, wherein the service spacecraft, expected point, target spacecraft, and space debris are viewed as nodes, while the relationships between the service spacecraft and other nodes are viewed as edges. Only the weight coefficients of obstacles such as the target spacecraft and space debris within the observation range are taken into account, enhancing the algorithm's adaptability to dynamic environments.

(2) To address the safety constraints of service spacecraft, the SAC–Lagrangian algorithm is adopted, and reasonable reward and cost functions are formulated to guarantee that the service spacecraft adheres to collision avoidance constraints during the safe proximity to the expected point.

(3) The GAT-SACL control algorithm is proposed, and a safe proximity environment with multiple dynamic obstacles is constructed in this paper. The Monte Carlo shooting method is used to evaluate and validate the safety and reliability of the algorithm in dealing with constrained problems.

The structure of the article is arranged as follows. Section 2 presents the problem description and modeling. Section 3 introduces the GAT-SACL method. Section 4 clarifies the training process and the performance of the proposed method. Section 5 provides a summary.

2. Preliminaries and Modeling

2.1. Problem Formulation

In the scenario of spacecraft safe proximity missions, the service spacecraft begins from the initial location hundreds of meters away from the target spacecraft and moves towards the expected point near the target spacecraft. Considering the shape of the target spacecraft and service spacecraft, the terminal expected point is established at a position about tens of meters away from the target spacecraft, which is equivalent to the dimensions of the spacecraft. The expected point serves as both the ending of the safe proximity mission and the beginning of the final approach phase for the subsequent rendezvous and docking mission. During the approach, the service spacecraft can only partially observe the surrounding environment. It can detect the presence and condition of space debris within a specific observation range via its sensor in real time, while the space debris outside the observation area cannot be observed. It is assumed that the sensor can perceive the state of space debris without deviation, disregarding the observation noise. The service spacecraft conducts continuous thrust maneuvers to avoid obstacles, including valuable target spacecraft and floating space debris, and then safely arrives at the expected point without collision. The task scenario is shown in Figure 1.

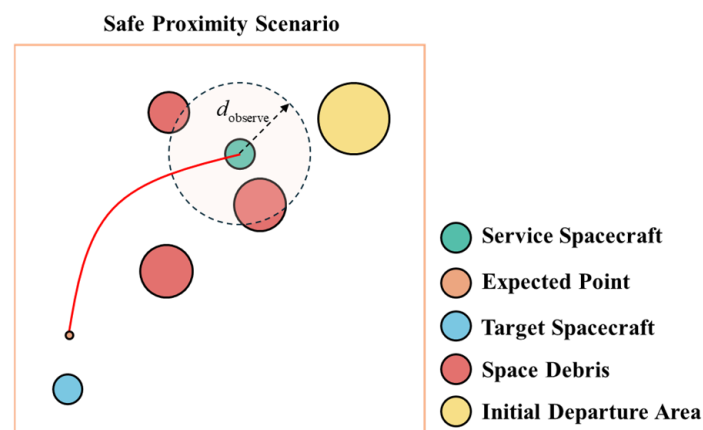


Figure 1. 2D safe proximity scenario.

The mathematical model of the spacecraft safe proximity scenario can be expressed as:

$$\begin{aligned} \min \quad & \lim_{t \rightarrow \infty} E = E_u + E_c \\ \text{s.t.} \quad & \begin{cases} \lim_{t \rightarrow \infty} \mathbf{x}_s = \mathbf{x}_{\text{target}} \\ \lim_{t \rightarrow \infty} \mathbf{v}_s = \mathbf{v}_{\text{target}} \\ \|\mathbf{x}_s - \mathbf{x}_{\text{target}}\| \geq d_{\text{safe}} \\ \|\mathbf{x}_s - \mathbf{x}_{\text{debris}}^i\| \geq d_{\text{safe}}, i \in \mathcal{N} \end{cases} \end{aligned} \quad (1)$$

In Equation (1), the overall goal of the mission is to minimize the fuel consumption E , where E_u denotes the fuel required to reach the expected point and E_c denotes the fuel required to avoid space debris. In constraint conditions, the first and second terms represent the terminal state constraints, indicating that the service spacecraft's terminal position and velocity converge to the expected state over time. The third and fourth terms represent the safety process constraints, which mean that the distance between the service spacecraft and the target spacecraft, as well as that between the service spacecraft and neighboring space debris, must not fall below the minimum safety distance during the maneuver process.

2.2. Dynamics of Spacecraft Safe Proximity Scenarios

In the scenario of safe proximity, the designed state is composed of the position and velocity of the service spacecraft, as well as the position of obstacles, including the target spacecraft and space debris. Since the relative distance between the service spacecraft and target spacecraft is limited to 100 m, the service spacecraft is modeled in the LVLH coordinate system with the target spacecraft as the origin. In this system, the X axis aligns with the geocentric vector, the Y axis is orthogonal to the X axis in the orbital plane and directs towards the motion direction of the target spacecraft, and the Z axis is orthogonal to the orbital plane, as illustrated in Figure 2. Both the service spacecraft and space debris follow the CW dynamic model [36], and the dynamic equation of the service spacecraft is described as follows:

$$\begin{cases} \dot{x}_s = v_{sx} \\ \dot{y}_s = v_{sy} \\ \ddot{x}_s = 2n\dot{y}_s + 3n^2x_s + a_x \\ \ddot{y}_s = -2n\dot{x}_s + a_y \end{cases} \quad (2)$$

where $\mathbf{r}_s = [x_s, y_s]^T$, $\mathbf{v}_s = [v_{sx}, v_{sy}]^T$ denote the position and velocity of the service spacecraft respectively, $\mathbf{a} = [a_x, a_y]^T$ represents the acceleration of the service spacecraft, and n signifies the orbital angular velocity, which is solely dependent on the orbit's semi-major axis.

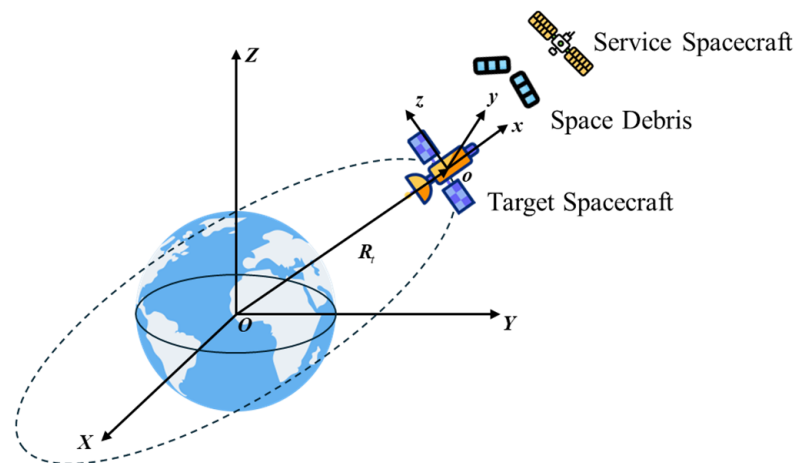


Figure 2. Schematic diagram of safe proximity scenario in the LVLH coordinate system.

Similarly, the dynamics equation of the i th space debris is described as:

$$\begin{cases} \dot{x}_o^i = v_{ox}^i \\ \dot{y}_o^i = v_{oy}^i \\ \ddot{x}_o^i = 2n\dot{y}_o^i + 3n^2x_o^i \\ \ddot{y}_o^i = -2n\dot{x}_o^i \end{cases} \quad (3)$$

where $\mathbf{r}_o^i = [x_o^i, y_o^i]^T$, $\mathbf{v}_o^i = [v_{ox}^i, v_{oy}^i]^T$ denote the position and velocity of the i th space debris.

2.3. Graph Structure of Spacecraft Safe Proximity Missions

In the scenario of spacecraft safe proximity missions, the service spacecraft is required to maneuver towards the expected point, while it must simultaneously avoid obstacles such as the target spacecraft and space debris within the observation range. The entire process encompasses multiple objects such as the service spacecraft itself, the terminal expected point of the service spacecraft, the target spacecraft and space debris. Diverse

items exert varying effects on the service spacecraft. The expected point attracts the service spacecraft and reaches it, while the obstacles repel the service spacecraft and push it away. This section focuses on describing the relationship between various objects and the service spacecraft by graph structure.

Graph theory is employed to delineate the structural graph of the service spacecraft in the safe proximity mission. In graph theory [37], the set of vertices is denoted as V , the set of edges is denoted as E , and the graph is denoted as $G = \{V, E\}$. In this paper, the vertex encompasses three types of nodes, including the service spacecraft, the expected point, and other neighboring obstacles. The set of neighboring obstacles for the service spacecraft is denoted as $O = \{o \in V_o : ||r_s - r_o|| \leq d_{\text{observe}}\}$, where V_o represents the set of obstacles that need to be avoided, such as the target spacecraft and space debris. Hence, the set of vertices is denoted as $V = \{0, 1, \dots, N - 1\}$, where $N = 2 + n$, denoting that the nodes include one service spacecraft, one expected point, and n pieces of obstacles within the observation range. The service spacecraft is coded as 0, and the expected point is coded as 1, while the neighboring obstacles are coded sequentially from 2 to $(N - 1)$. Among the three types of vertices, the characteristic information of the service spacecraft and the expected points is essential. The quantity and state of obstacle features vary during the movement of the service spacecraft. If there exist no obstacles, then $V = \{0, 1\}$. When new obstacles emerge in the observation range, corresponding new obstacle nodes are added. When previous obstacles quit the observation range and cease to affect the trajectory of the service spacecraft, departing obstacle nodes are diminished. As shown in Figure 3, the graph structure of the service spacecraft at different moments changes with time.

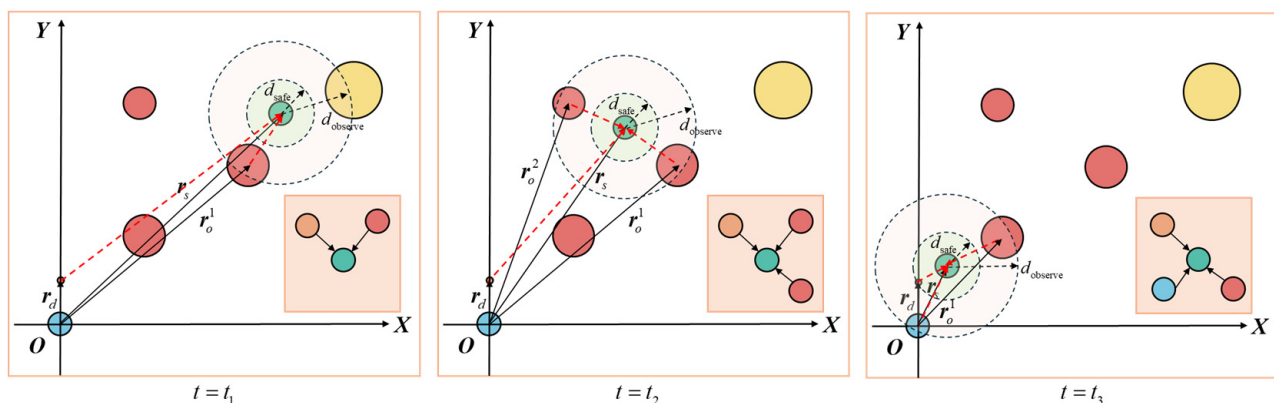


Figure 3. Schematic diagram of service spacecraft graph structure at different moments.

Furthermore, the interaction between vertices is characterized by the mapping relationship inherent in the edge. The edge is usually described as $e_{ij}(i \in N, j \in N)$, which represents the connectivity from node j to node i . Among the three types of vertices, there are two categories of edges: one type maps the position information of the service spacecraft and the expected point, indicating the maneuverability of the service spacecraft to reach the expected point. The other type of edge maps the position information of service spacecraft and neighboring obstacles, indicating the safety of the service spacecraft to avoid the obstacles. This paper focuses solely on the impact of the expected point and obstacles on the service spacecraft, given that the expected point is determined by the mission scenario and remains constant throughout the episode, while the obstacles exist objectively in the space environment and their positions are unaffected by the state of the service spacecraft. Specifically, the directed edges are established where the expected point and neighboring obstacles direct towards the service spacecraft. In general, graph structure offers substantial benefits in depicting the dynamic environment. It can adapt to varying quantities of obstacles and extract significant information with fixed dimensions

by processing data with different dimensions so as to facilitate the subsequent training of the network [38].

2.4. Constrained Markov Decision Process

The problem of spacecraft safe proximity trajectory planning and control can be regarded as a constrained Markov decision process. The service spacecraft, as the subject of reinforcement learning, is referred to as the agent. The service spacecraft parameters and other relevant parameters are referred to as the state, while the control torque for the service spacecraft is referred to as the action. The dynamic models of the service spacecraft, target spacecraft, floating space debris, and other impediments, along with the terminal and process constraints, are all considered as the environment. The service spacecraft takes action a at the current state s to interact with the environment according to the policy π . The service spacecraft updates its state s_{t+1} via the dynamic model in the environment and evaluates the reward function r and cost function c respectively by calculating the difference between the current state and the expected state of the service spacecraft, as well as the distance to obstacles. The data acquired via environmental interactions is stored in a buffer, and the policy π is optimized by the learning algorithm to maximize the expected reward function while ensuring that the expected cost function meets the constraints. The schematic diagram of safe reinforcement learning is shown in Figure 4.

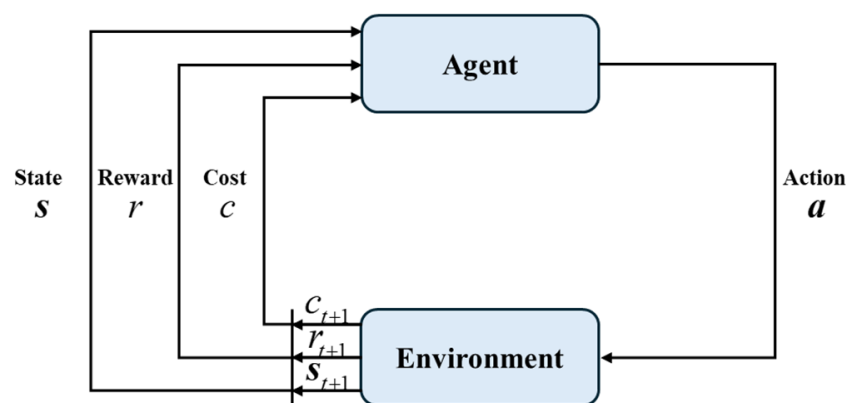


Figure 4. Schematic diagram of safe reinforcement learning.

The constrained Markov decision process is typically characterized by a seven-tuple $(S, A, p, r, c, d, \gamma)$, comprising the state space S , the action space A , the transition probability $p : S \times A \times S \rightarrow \mathbb{R}$, denoting the probability of taking action a at the current state s_t to achieve the subsequent state s_{t+1} , the reward function $r : S \times A \rightarrow \mathbb{R}$, denoting the instantaneous reward obtained by taking action a at the state s_t , the cost function $c : S \times A \rightarrow \mathbb{R}$, denoting the cost corresponding to taking action a at the state s_t to assess constraint satisfaction, the cost constraint upper limit d , and the discount factor $\gamma \in (0, 1)$. The goal of safe reinforcement learning algorithms is to seek the policy π to maximize the performance index $J_R = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T r_t \right]$ while adhering to the constraint index $J_C = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T c_t \right] \leq d$. The mathematical model for constrained optimization problems can be described as follows:

$$\begin{aligned}
 & \max_{\pi_\theta} \quad \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T r_t \right] \\
 & \text{s.t.} \quad \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T c_t \right] \leq d
 \end{aligned} \tag{4}$$

2.4.1. State and Action

The service spacecraft updates its own state and receives the command of the expected terminal state in real time and gathers position information of the target spacecraft and other obstacles within the observation range. It is assumed that the service spacecraft can make decisions only based on local knowledge. In the actor network, the state of the service spacecraft consists of two parts, including intrinsic state parameters and graph structural features, and it is defined as

$$s_t = (s_t^a, s_t^G) \quad (5)$$

where $s_t^a = (r_s, v_s, r_d, v_d)$ represents the position and velocity of the service spacecraft at the current moment as well as the expected position and velocity at the terminal moment, referred to as the expected point's state. $s_t^G = \{s_t^V, s_t^E\}$ represents the characteristics and structural information for spacecraft safe proximity missions. The set of node features is defined as $s_t^V = \{s_t^s, s_t^d, s_{t,1}^o, s_{t,2}^o, \dots, s_{t,n}^o\}$, $s_{t,i}^o \in O$, where the input features of each node $s_t^j = (r_j, e_j)$, $j = s, d, o$ consist of the position parameters and one-hot encodings for the serving spacecraft, the expected point, and neighboring obstacles, respectively. The implementation of one-hot encoding here is primarily intended to distinguish the influence of various nodes on the spacecraft's position. The service spacecraft, as the core of the graph structure, maintains close associations with both the expected point and obstacles, encoded as $[1, 0, 0]$. The expected point acts as an attractor to guide the spacecraft to approach it, encoded as $[0, 1, 0]$, while obstacles function as repulsive points to deter the service spacecraft, encoded as $[0, 0, 1]$, as illustrated in Figure 5. In addition, the edge relationship is described by a two-dimensional tensor edge index.

$$s_t^E = \begin{bmatrix} j_1 & j_2 & \cdots & j_k & \cdots & j_M \\ i_1 & i_2 & \cdots & i_k & \cdots & i_M \end{bmatrix} \in \mathbb{R}^{2 \times M} \quad (6)$$

where the k th column represents the edge from node j_k to node i_k , and M denotes the number of edges.

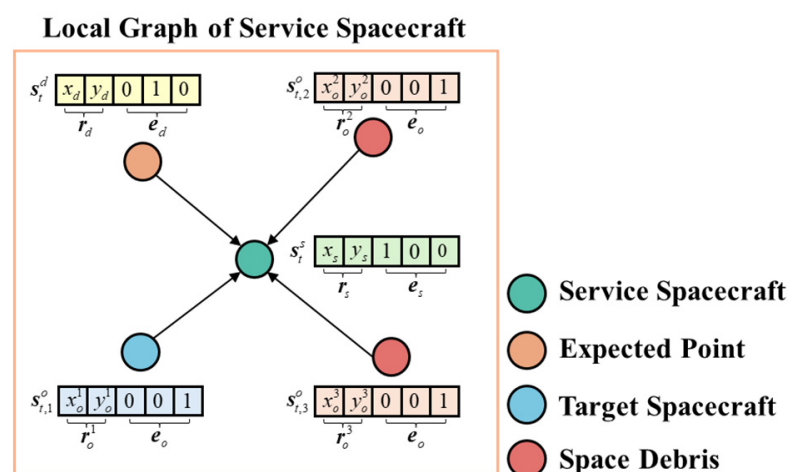


Figure 5. State representation of the service spacecraft in the actor network.

In the critic network, it is assumed that the evaluator possesses a comprehensive view of the whole situation and adopts a fully informed perspective, scoring the current state of the service spacecraft and the actions taken based on global state information. Hence, the input state of the critic network is the global parameter

$$s_t^{\text{all}} = (r_s, v_s, r_d, v_d, r_o^1, r_o^2, \dots, r_o^M) \quad (7)$$

which consists of the position and velocity of the service spacecraft and the expected point, as well as the position of the target spacecraft and all space obstacles. It is emphasized that all state parameters are normalized by dividing the state quantity by its upper limit.

According to the dynamic model presented in Section 2.2, a continuous thrust control mechanism is proposed for the safe proximity orbit design of the service spacecraft. The action in this scenario is acceleration $\mathbf{a} = (a_x, a_y)$ along the X and Y axes. The acceleration magnitude should satisfy the requirements of the upper limit of control torque $a_i \in [-1, 1], i = x, y$.

2.4.2. Cost

In the scenario of spacecraft safe proximity missions, on the one hand, the service spacecraft is required to ensure its own safety while approaching the expected point and not collide with floating space debris; on the other hand, it is necessary to ensure the safety of the high-value target spacecraft, and the service spacecraft should not collide with the target spacecraft. In the LVLH coordinate system fixed at the centroid of the target spacecraft, the position and velocity of space debris follow the CW dynamic equation and change gradually over time. In this paper, the target spacecraft and space debris are collectively referred to as obstacles. If the distance between the service spacecraft and the obstacle exceeds observation range d_{observe} , it is deemed that they are at a safe distance, and the obstacle avoidance maneuver does not need to be incorporated into the service spacecraft's trajectory planning. If the distance is less than observation range d_{observe} , it is considered that the service spacecraft faces a high risk of collision, and the obstacle avoidance mission has failed. If the distance is between the safe distance d_{safe} and observation range d_{observe} , the obstacle has entered the service spacecraft's field of vision. In trajectory planning, the service spacecraft needs to assess the potential collision risk with the obstacle while striving to strike a balance between avoiding it and reaching the expected point.

At the current time step, if the distance between the service spacecraft and the obstacle is less than the safe distance d_{safe} , the number of collisions is noted as 1; otherwise, it is noted as 0. The cost function resulting from the i th obstacle is expressed as

$$c_i = \begin{cases} +1 & \text{if } \|\mathbf{r}_s - \mathbf{r}_o^i\| \leq d_{\text{safe}} \\ 0 & \text{else} \end{cases} \quad (8)$$

According to the definition, the essence of the cost function is the number of collisions between the service spacecraft and obstacles.

In the safe proximity scenario, the total cost function caused by neighboring obstacles is designed as

$$c_{\text{total}} = \sum_{i \in O} c_i \quad (9)$$

Ideally, the goal of safe reinforcement learning is to ensure that the expected cumulative cost function is less than or equal to 0; that means $\mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T c_t \right] \leq 0$. In practice, the cumulative cost function is intended to be less than a tiny positive value.

2.4.3. Reward

Reward functions serve as the bridge between artificial intelligence algorithms and specific task scenarios. Various reward functions have been designed to address the requirements of different spacecraft task scenarios. The reward function for the spacecraft safe proximity mission scenario proposed in this paper consists of four parts, including a terminal reward, process reward, fuel penalty, and obstacle collision penalty. The terminal reward and process reward assess the arrival of service spacecraft. The fuel penalty

evaluates the fuel consumption of service spacecraft, and the obstacle collision penalty evaluates the safety of service spacecraft.

(1) Terminal reward

The core goal of the spacecraft safe proximity mission is to drive the service spacecraft to the expected terminal point. Consequently, the most intuitive technique for designing a reward function is to assign a substantial positive reward to the service spacecraft when it arrives at the expected point and its position and velocity converge to the target values. Once the service spacecraft reaches its destination, it successfully completes the task and concludes the current episode. Otherwise, the service spacecraft does not meet the terminal status requirements at the current step, and the episode continues. The terminal reward can be defined as follows:

$$r_{\text{done}} = \begin{cases} 100 & \text{if } \|r_s - r_d\| \leq d_{\text{reach}} \text{ and } \|v_s - v_d\| \leq v_{\text{reach}} \\ 0 & \text{else} \end{cases} \quad (10)$$

where $r_{s,d} = \|r_s - r_d\|$ represents the distance between the current position and the expected point position, and $v_{s,d} = \|v_s - v_d\|$ represents the difference between the current velocity and the expected terminal velocity.

(2) Process reward

Although the terminal reward depicts the arrival of the service spacecraft at the terminal moment, depending solely on the terminal reward, the service spacecraft receives feedback after hundreds of exploratory steps, making it difficult to achieve the goal and to obtain meaningful rewards even until the end of the maximum episode step. Therefore, the process reward is further introduced to encourage the service spacecraft to navigate towards the expected point to overcome the problems of low exploration efficiency and the challenges in achieving terminal objectives resulting from sparse rewards. If the service spacecraft is nearer to the expected point and its velocity is more aligned, a reward is granted. If the service spacecraft deviates from the expected point or exceeds the designated velocity, a penalty is imposed. The process reward can be defined as follows:

$$r_{\text{process}} = - \left(\frac{\|r_s - r_d\|^2}{L_{\text{max}}^2} + \frac{\|v_s - v_d\|^2}{V_{\text{max}}^2} \right) \quad (11)$$

where L_{max} and V_{max} are the maximum distance and maximum velocity of the state space, respectively, which are used to normalize the relative position and velocity errors. If the reward function is positive, the spacecraft is likely to oscillate around the target value without reaching the expected point in an effort to maximize the positive reward. Therefore, to prevent reward hacking, the process reward is designated as a negative value. If the service spacecraft approaches the target point, a minor negative reward will be assigned; conversely, if it leaves away from the expected point, a substantial negative penalty will be enforced.

(3) Fuel penalty

Fuel consumption is an important factor restricting the maneuverability of spacecraft in orbit. Therefore, during the process of safe proximity, the less fuel consumption, the higher the benefit. If the fuel consumption is higher, a greater penalty will be imposed. If the fuel consumption is reduced, a lower penalty will be imposed. The fuel penalty can be defined as follows:

$$r_{\text{fuel}} = -k_{\text{fuel}}(a_x^2 + a_y^2) \quad (12)$$

where the constant k_{fuel} is the penalty coefficient.

(4) Obstacle collision penalty

Safe collision avoidance is the primary premise for ensuring the safety of service spacecraft. If the service spacecraft detects that the distance to the obstacle exceeds the safe distance d_{safe} , it is deemed to be in a safe position without punishment. If the distance between the service spacecraft and obstacle is less than the safe distance d_{safe} , it is considered that the service spacecraft tends to collide with this obstacle, and the safety of the service spacecraft is significantly threatened, so a larger penalty is imposed. The obstacle collision penalty resulted from the i th obstacle can be defined as follows:

$$r_{\text{obstacle}}^i = \begin{cases} -0.5 & \text{if } \|r_s - r_o^i\| \leq d_{\text{safe}} \\ 0 & \text{else} \end{cases} \quad (13)$$

where $r_{s,o}^i = \|r_s - r_o^i\|$ denotes the distance between the service spacecraft and the i th obstacle in the observation range, and d_{safe} denotes the minimum safe distance between the service spacecraft and the obstacle.

In conclusion, the reward function proposed in this paper is formulated as

$$r_{\text{total}} = r_{\text{done}} + r_{\text{process}} + r_{\text{fuel}} + \sum_{i \in O} r_{\text{obstacle}}^i \quad (14)$$

3. Spacecraft Safe Proximity Policy Based on Safe Reinforcement Learning

A Soft Actor Critic algorithm based on the Lagrange method, incorporating the graph attention network, is proposed to address the issue of real-time variations in the quantity and position of obstacles within the observation range, referred to as GAT-SACL. In Section 3.1, the principle of the GAT network is delineated, which is used to extract the characteristic information regarding the service spacecraft, the expected point, and dynamically altering obstacle nodes. In Section 3.2, the SACL algorithm is illustrated in detail, wherein the constrained optimization problem is reformulated as an unconstrained optimization problem by introducing a Lagrange multiplier. In Section 3.3, the GAT-SACL algorithm architecture is delineated.

3.1. Principle of Graph Neural Network

The graph structure is mainly composed of nodes and edges. In this paper, the service spacecraft, the expected point, and obstacles within the observation range, including the target spacecraft and space debris, are regarded as nodes. The relationship between the service spacecraft and the expected point, as well as that between the service spacecraft and obstacles in the observation range, is regarded as an edge. The position information corresponding to the node is regarded as the input feature of the node, and the hidden feature information derived from the graph neural network is considered as the output feature of the node, which is then utilized as the input to train the actor network for reinforcement learning. The service spacecraft can only perceive obstacles within the observation range, and the quantity and position of obstacles continually fluctuate as the spacecraft explores the environment. Consequently, acquiring knowledge about the position features of obstacles and their effects on the service spacecraft is a crucial function of graph neural networks. If the obstacle is outside the observation range, it will not affect the service spacecraft's decision-making in the subsequent timeframe. If the obstacles are within the observation range, the service spacecraft extracts information according to the position of the neighboring obstacles. The closer the obstacles are to the service spacecraft, the greater the impact on the service spacecraft's maneuver strategy, and the farther the obstacles are from the service spacecraft, the smaller the impact on the service spacecraft's

maneuver strategy. The conventional graph neural network is incapable of accommodating variations in the number of nodes. The graph attention network has a significant effect in solving the dynamic environment of increasing or decreasing nodes by dynamically learning the weight coefficients. Therefore, this paper introduces graph attention networks to extract information from dynamic graph structures.

Furthermore, the service spacecraft must equilibrate the cumulative impact of the attractive force from the expectation point and the repulsive force from the obstacle on the spacecraft as a whole. Another function of graph attention networks is to adaptively learn and adjust the influence weight of the expected point and neighboring obstacles on the service spacecraft according to the characteristic parameters. If there are no obstacles in the observation range, the safe proximity of the service spacecraft is simplified as an arrival problem, focusing solely on the trajectory planning towards the expected point. If a neighboring obstacle is identified and it obstructs the path to the expected point, the service spacecraft must formulate an appropriate maneuver strategy to actively avoid the obstacle while simultaneously striving to approach the expected point as closely as possible, thereby reducing fuel consumption necessitated by trajectory correction due to obstacle avoidance. If the neighboring obstacle is in the opposite direction of the path or away from the expected point, its impact on the service spacecraft's decision is negligible, and the intention to approach the expected point remains dominant. Therefore, this paper employs graph attention networks to capture the relationships within the graph structure regarding the effects of different types of objects on spacecraft.

The graph attention network [39] structure is illustrated in Figure 6, the fundamental component of which is the graph attention layer. The input to the graph attention layer is features of N nodes $s_t^V = \{s_t^s, s_t^d, s_{t,1}^o, s_{t,2}^o, \dots, s_{t,n}^o\}$, $s_i \in \mathbb{R}^5$, and the output is new features of these nodes $s_t^{V'} = \{s_t^{s'}, s_t^{d'}, s_{t,1}^{o'}, s_{t,2}^{o'}, \dots, s_{t,n}^{o'}\}$, $s_i' \in \mathbb{R}^{F'}$, where F' represents the number of GAT output network layers.

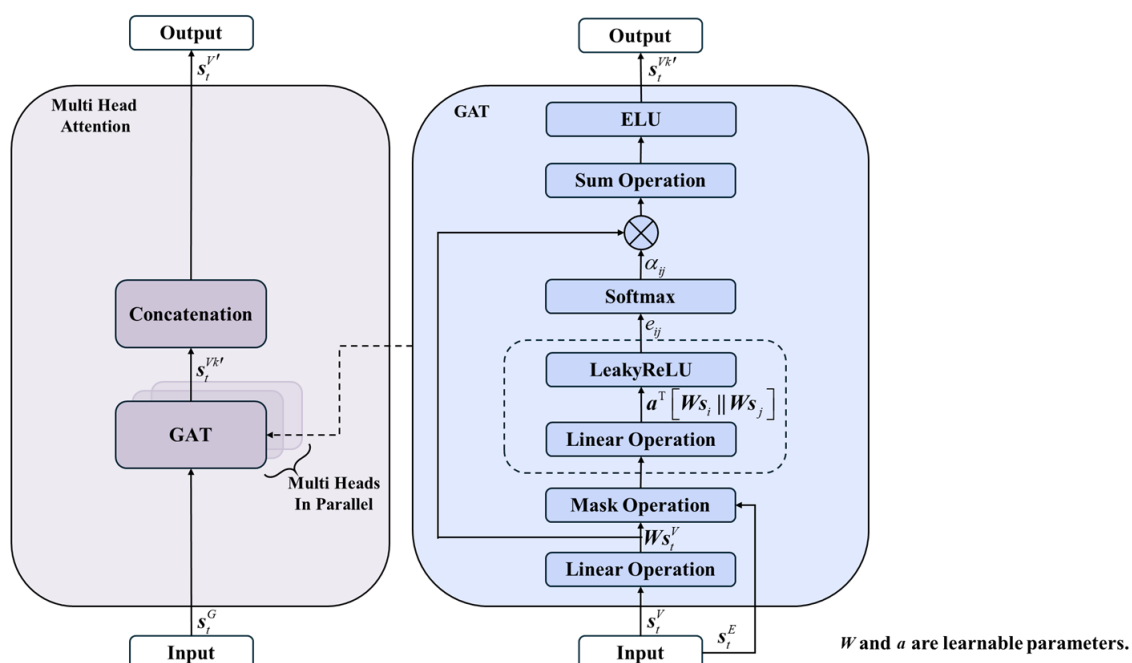


Figure 6. The structure of GAT.

First, the input features are linearly transformed into $Ws_i \in \mathbb{R}^{F'}$ through a weight matrix $W \in \mathbb{R}^{F' \times 5}$. To avoid the issue of structural information loss caused by traversal computation among nodes $e_{ij}(i \in N, j \in N)$, masked attention is adopted to introduce

graph structural information into the attention mechanism. In masked attention, the attention coefficient of node i only takes into account the neighboring nodes of node i while ignoring the influence of non-neighboring nodes.

Then, the self-attention of the node is calculated through a shared attention mechanism $a: \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$. A single-layer feedforward neural network is used as the attention mechanism, adopting LeakyReLU as the activation function. The attention coefficient $e_{ij} \in \mathbb{R}$ of node j relative to node i can be calculated by:

$$e_{ij} = a(\mathbf{W}\mathbf{s}_i, \mathbf{W}\mathbf{s}_j) = \text{LeakyReLU}\left(\mathbf{a}^T [\mathbf{W}\mathbf{s}_i || \mathbf{W}\mathbf{s}_j]\right) \quad (15)$$

where $\mathbf{a} \in \mathbb{R}^{2F'}$ denotes the weight coefficient of the attention mechanism, and $[||\cdot||\cdot] \in \mathbb{R}^{2F'}$ represents the concatenation of two elements. The attention coefficient e_{ij} reflects the importance node j to node i .

To facilitate the comparison of attention coefficients for different nodes, the softmax function is utilized to normalize the attention coefficients. The normalized attention coefficient of node j to node i is described as

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \quad (16)$$

The new features of node i by the graph attention layer can be represented as a linear combination of neighboring node features, considering different levels of influence to node i :

$$\mathbf{s}_i' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\mathbf{s}_j\right) \quad (17)$$

where $\sigma(\cdot)$ represents the ELU activation function.

To stabilize the self-attention learning process, the multi-head attention mechanism is utilized to compute attention coefficients in parallel using multiple threads. By concatenation, the output features obtained through multi-head attention can be represented as

$$\mathbf{s}_i' = ||_{k=1}^K \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{s}_j\right) \quad (18)$$

where K represents the total number of threads. α_{ij}^k and $\mathbf{W}^k$ denote the attention coefficient and weight matrix computed by the k th thread, respectively. Since the concatenated parameters comprises KF' features, the concatenation method is typically deployed at the intermediate layer of the network, and its output can further serve as the input for subsequent networks.

3.2. Soft Actor Critic–Lagrangian Algorithm

The Soft Actor Critic algorithm [23], as a classic algorithm in the field of reinforcement learning, introduces an entropy term into the objective function of policy optimization based on the traditional Actor Critic architecture. It maximizes both the reward function and entropy, encouraging the agent to explore more stochastic strategies. The SAC algorithm possesses the advantages of policy randomization, strong exploration capability, and robustness, and it has beneficial applications in complex dynamics fields such as robotics. However, like all traditional reinforcement learning algorithms, when solving constrained Markov decision processes, the SAC algorithm usually embeds the constraint conditions directly into the reward function. The training results of the algorithm are highly dependent on reward design and parameter tuning, resulting in the algorithm's optimality and safety

being determined by designers' experience. Therefore, this paper adopts the optimization idea in safe reinforcement learning and transforms the hard-constraint problem into an unconstrained problem by introducing the Lagrange multiplier.

The safe reinforcement learning algorithm based on SAC aims to solve the maximum entropy reinforcement learning problem considering safety constraints:

$$\begin{aligned} & \max_{\pi} \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi}} \left[\sum_t \gamma^t r(s_t, a_t) \right] \\ \text{s.t.} & \begin{cases} \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi}} \left[\sum_t \gamma^t c(s_t, a_t) \right] \leq d \\ \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi}} [-\log(\pi_t(a_t|s_t))] \geq \mathcal{H}_0 \end{cases} \end{aligned} \quad (19)$$

where the first constraint condition is the cost function constraint, and the second is the entropy constraint.

By introducing the Lagrange multiplier κ and temperature parameters α , the constrained problem is transformed into an unconstrained problem [40]

$$\begin{aligned} & \max_{\pi} \min_{\kappa \geq 0} \min_{\alpha \geq 0} L(\pi, \kappa) \doteq f(\pi) - \kappa g(\pi) - \alpha h(\pi) \\ \text{where} & \begin{cases} f(\pi) = \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi}} \left[\sum_t \gamma^t r(s_t, a_t) \right] \\ g(\pi) = \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi}} \left[\sum_t \gamma^t c(s_t, a_t) \right] \\ h(\pi) = \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi}} [\log(\pi_t(a_t|s_t))] + \mathcal{H}_0 \end{cases} \end{aligned} \quad (20)$$

For the reward critic network, the reward soft Q -function is optimized by minimizing the Bellman residual

$$J_{Q_r}(\theta^r) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[\frac{1}{2} (Q_{\theta^r}(s_t, a_t) - (r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p} [V_{\theta^r}(s_{t+1})]))^2 \right] \quad (21)$$

For the cost critic network, similar to the reward critic network, the cost soft Q -function is optimized by minimizing the Bellman residual

$$J_{Q_c}(\theta^c) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[\frac{1}{2} (Q_{\theta^c}(s_t, a_t) - (c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p} [V_{\theta^c}(s_{t+1})]))^2 \right] \quad (22)$$

For the actor network, the actor loss function is

$$J_{\pi}(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi_{\phi}} [\alpha \log \pi_{\phi}(a_t|s_t) - Q_{\theta^r}(s_t, a_t) + \kappa Q_{\theta^c}(s_t, a_t)] \quad (23)$$

where the Lagrange multiplier $\kappa \geq 0$ is updated by minimizing the safety constraint loss function

$$J_s(\kappa) = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi_{\phi}} [\kappa (d - Q_{\theta^c}(s_t, a_t))] \quad (24)$$

The temperature parameter α is updated by minimizing the entropy loss function

$$J_e(\alpha) = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi_{\phi}} [-\alpha (\log \pi_{\phi}(a_t|s_t) + \mathcal{H}_0)] \quad (25)$$

The pseudo-code of SACL is shown in Algorithm 1. Two types of Q -networks are implemented, including the reward Q -network and the cost Q -network. To enhance the stability and robustness of the algorithm, two reward Q -networks are established. The minimum value of the two Q -values is selected as the Q -estimate, reducing the overestimation bias induced by neural network approximation. In addition, each Q -network corresponds

to a target Q-network, which is updated based on the Q-network using an exponential moving average method.

Algorithm 1. SACL Algorithm

Input: $\theta_1^r, \theta_2^r, \theta^c, \phi$	► Initialize parameters
$\bar{\theta}_1^r \leftarrow \theta_1^r, \bar{\theta}_2^r \leftarrow \theta_2^r, \bar{\theta}^c \leftarrow \theta^c$	► Initialize target network weights
$\mathcal{D} \leftarrow \emptyset$	► Initialize an empty replay pool
for each iteration do	
for each environment step do	
$a_t \sim \pi_\phi(a_t s_t)$	► Sample action from the policy
$s_{t+1} \sim p(s_{t+1} s_t, a_t)$	► Sample transition from the environment
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), c(s_t, a_t), s_{t+1})\}$	► Store the transition in the replay pool
end for	
for each gradient step do	
$\theta_i^r \leftarrow \theta_i^r - \lambda_{Q_r} \hat{\nabla}_{\theta_i^r} J_{Q_r}(\theta_i^r)$ for $i \in \{1, 2\}$	► Update the reward Q-function parameters
$\theta^c \leftarrow \theta^c - \lambda_{Q_c} \hat{\nabla}_{\theta^c} J_{Q_c}(\theta^c)$	► Update the cost Q-function parameters
$\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$	► Update policy weights
$\kappa \leftarrow \kappa - \lambda_\kappa \hat{\nabla}_\kappa J_\kappa(\kappa)$	► Adjust Lagrange-multiplier
$\alpha \leftarrow \alpha - \lambda_\alpha \hat{\nabla}_\alpha J_e(\alpha)$	► Adjust temperature parameter
$\bar{\theta}_i^r \leftarrow \tau \theta_i^r + (1 - \tau) \bar{\theta}_i^r$ for $i \in \{1, 2\}$	► Update reward target network weights
$\bar{\theta}^c \leftarrow \tau \theta^c + (1 - \tau) \bar{\theta}^c$	► Update cost target network weights
end for	
end for	
Output: $\theta_1^r, \theta_2^r, \theta^c, \phi$	► Optimized parameters

3.3. GAT-SACL Algorithm

The framework of the GAT-SACL algorithm is shown in Figure 7. At the beginning of each episode, the environment is initialized, and the action is generated by the actor network. Subsequently, the agent interacts with the environment to obtain the state for the next step, as well as the reward function and cost function, until the episode finishes. A new episode is then initiated, and the above process repeats. The experience data $(s_t^{\text{all}}, s_t, a_t, r_t, c_t, s_{t+1}^{\text{all}}, s_{t+1})$ is stored in the replay buffer at each step. When the data in the replay buffer reaches 2000, the agent begins to train 50 times every 50 steps, sampling a batch of data to train the network parameters during each iteration. Subsequently, the networks are evaluated three times every 2500 steps; at the same time, the average reward, average cost, and episode success rate are recorded. When data in the replay buffer exceeds the maximum capacity, the new data will replace the earliest stored data.

The episode end conditions for the spacecraft safe proximity mission encompass the following three situations: (1) If the service spacecraft reaches the expected point and the position error and velocity error satisfy the terminal constraints, it is considered that the service spacecraft successfully accomplishes the proximity mission. (2) If the service spacecraft moves out of the observation space, the episode is truncated prematurely. This issue is due to the fact that when the service spacecraft crosses the boundary, it is far away from the terminal expected point, even if it has another opportunity to return to the observation space, and the fuel consumption in European space exceeds that of the

spacecraft consistently operating within the observation space. (3) If the step of the current episode reaches the maximum step, the episode is over, regardless of whether the expected point is achieved. Among these end conditions, the task is successfully completed solely when the first end condition is realized. If the last two conditions are met, the episode is truncated with the task failing, and then a new episode begins. It is worth noting that the collision truncation condition is not established in this paper. If the episode is truncated when the service spacecraft collides with an obstacle, it is not conducive to fully exploring the environment behind the obstacle barrier, and it is difficult to design the reward function. Hence, the collision truncation condition is disregarded, and the collision penalty is incorporated into the reward function and cost function as a constraint. Once the service spacecraft collides with obstacles, a large penalty is imposed on the reward function, and the number of collisions is added to the cost function. By optimizing the objective function, a safe policy with the highest reward score and lower cost constraint is obtained.

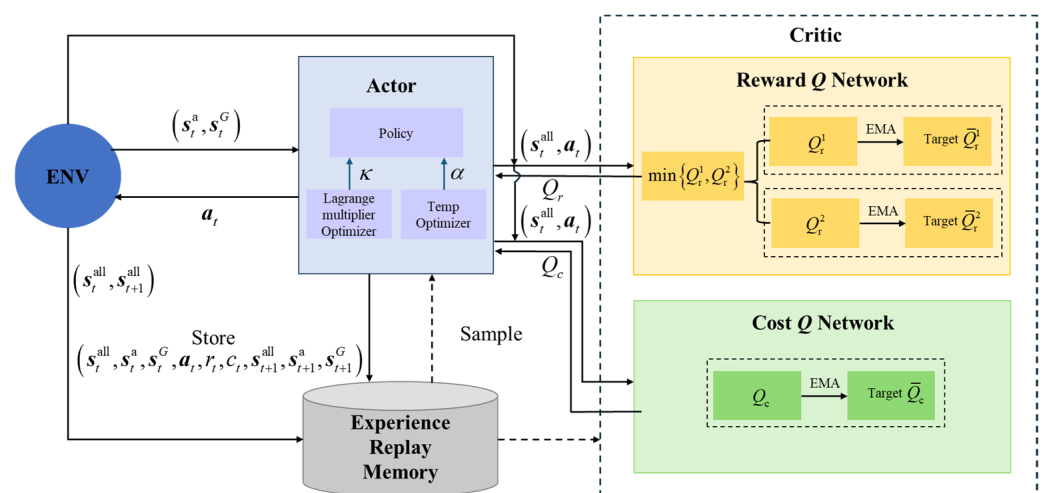


Figure 7. The framework of the GAT-SACL algorithm.

The structure of actor and critic networks is shown in Figure 8. The actor network consists of fully connected networks and graph neural networks. The service spacecraft's state parameters s_t^a are transformed via two fully connected layers. The graph structure features s_t^G are transformed via two graph attention network layers, with the first layer employing multi-head attention mechanisms and the latter using one head. Then, the outputs of the fully connected networks and graph attention networks are concatenated together, and the combined data is subsequently fed to two hidden layers and an output layer, utilizing the tanh function to constrain continuous action within the range of $[-1, 1]$. It should be noted that the uncertain dimension of the graph structure features s_t^G may lead to dimension conflicts with the fixed dimension of the following networks after directly using GATConv to process graph data. Therefore, firstly, the sampled batch of graph structures are combined into a comprehensive graph through batch processing. The comprehensive graph is then fed into the graph attention network layers to learn the feature information. Subsequently, the information undergoes global average pooling, according to initial subgraphs, to obtain the fixed-dimension data as input for the following networks. The reward critic network and the cost critic network are composed of fully connected networks with the same structure. The global state parameters s_t^all and action a_t are concatenated together and then fed into two fully connected network hidden layers, followed by one output layer.

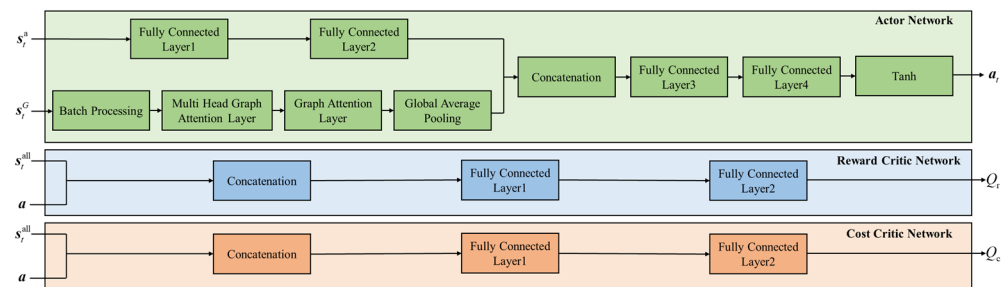


Figure 8. The structure of actor and critic networks of the GAT-SACL algorithm.

4. Simulation and Analysis

This section presents the training performance and test results of GAT-SACL. In Section 4.1, the simulation platform and environment for the safe proximity are introduced, including scenario parameters and algorithm parameters. In Section 4.2, test results are presented through extensive tests, and the simulation results are analyzed and compared with the pseudo-spectrum method.

4.1. Scenario Settings

In this section, a 2D environment for the spacecraft safe proximity scenario is established using Python 3.9, including a service spacecraft, a target spacecraft, and three pieces of space debris. The visual presentation of the spacecraft safe proximity scenario based on Pygame is illustrated in Figure 9. The parameter settings for this scenario are presented in Table 1. In the beginning of each episode, the service spacecraft is located within the initial departure area with a radius of 10 m and moves at a random initial speed. Then, it passes through several pieces of space debris floating in orbit and ultimately arrives at the expected point, which is 20 m away from the target spacecraft. The tolerance for the terminal position error to the expected point is 1 m, and that for the terminal velocity error is 0.1 m/s. The target spacecraft is located at the origin of the LVLH coordinate system, while space debris is located in an initial range with a random velocity and then floats freely according to the orbital dynamics model. The observation range of the service spacecraft is 25 m, and the minimum safe distance between the service spacecraft and the obstacle is 5 m. The parameter settings for GAT-SACL are presented in Table 2, and the network architecture is shown in Figure 10.

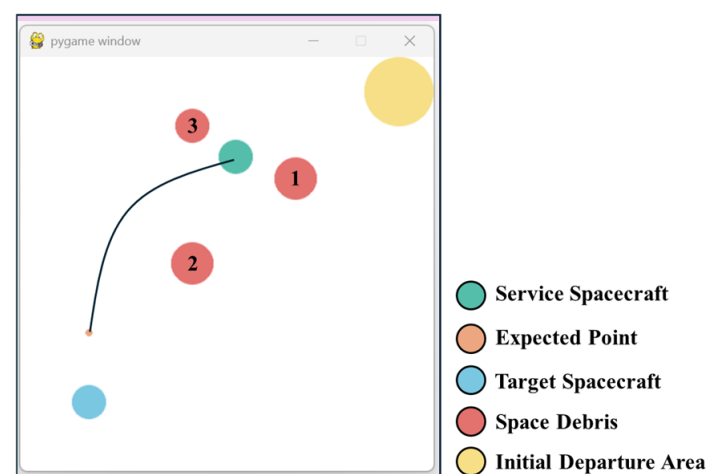


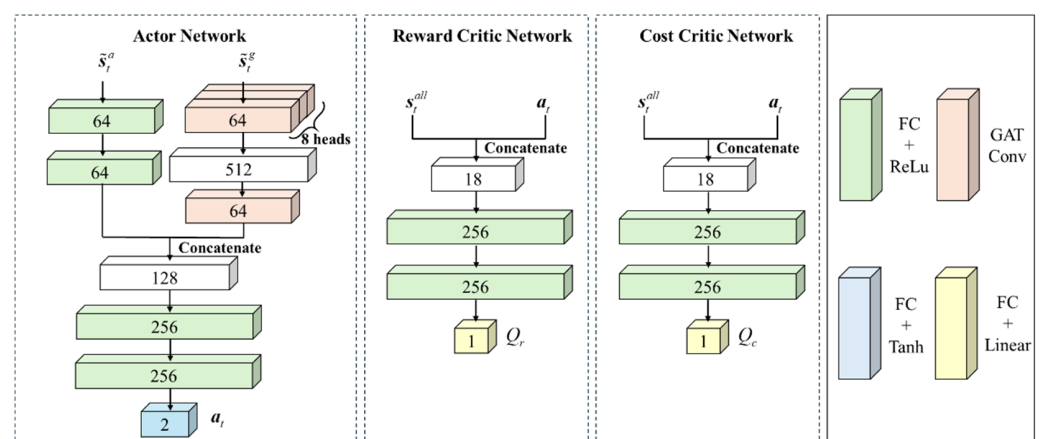
Figure 9. The safe proximity scenario based on Pygame.

Table 1. The parameter settings for the safe proximity scenario.

Type	Parameters	Value
Scenario Setting	Orbit Semi-Major Axis of Target Spacecraft (km)	6889.577
	Observation Space X (m)	[−20, 100]
	Observation Space Y (m)	[−20, 100]
	The Maximum Distance (m)	100
	The Maximum Velocity (m/s)	10
	Integral Step Size (s)	0.1
	Observation Range of Service Spacecraft (m)	25
Initial Setting of Service Spacecraft	Center of Initial Position Area (m)	(90, 90)
	Radius of Initial Position Area (m)	10
	Initial Velocity Range	[−0.001, 0.001]
Initial Setting of Obstacle	Radius of Target Spacecraft (m)	5
	Position of Target Spacecraft (m)	(0, 0)
	Radius of Space Debris 1 (m)	6
	Initial Position of Space Debris 1 (m)	(60, 65)
	Radius of Space Debris 2 (m)	6
	Initial Position of Space Debris 2 (m)	(30, 40)
	Radius of Space Debris 3 (m)	5
	Initial Position of Space Debris 3 (m)	(30, 80)
	Initial Velocity Range	[−0.01, 0.02]
Constraint	Expected Point Position (m)	(0, 20)
	Tolerant Distance (m)	1
	Tolerant Velocity (m/s)	0.1
	Lower Limit of Safety Distance (m)	5

Table 2. The parameter settings for the GAT-SACL algorithm.

Parameters	Value
Max Train Step	2×10^6
Max Step in Episode	1000
γ	0.99
Actor Learning Rate	0.0003
Critic Learning Rate	0.0003
Lagrange Multiplier Learning Rate	0.0003
Soft Update Factor Initial Value	0.12
Reply Buffer Size	1×10^6
Batch Size	256

**Figure 10.** The network structure of the GAT-SACL algorithm.

4.2. Test Result and Analysis

During the training process, evaluation is conducted in several episodes every several steps, and the reward of these episodes is recorded. In order to reflect the completion of the safe proximity task, the average reward is defined as:

$$\bar{R}_\tau = \frac{1}{N_{\text{eva}}} \sum_{t=1}^{N_{\text{step}}} R_{\text{eva}}^t \quad (26)$$

where N_{eva} represents the number of evaluation episodes per test time, N_{step} represents the total steps for each evaluation episode and R_{eva}^t represents the instantaneous reward for each step t . A higher average reward indicator signifies more success of arrival, less fuel consumption, and a lower risk of collision with obstacles.

Figure 11 shows the variation in the average reward with training steps. It can be found that the average reward converges to around -100 . The network evaluated with the highest score is selected as the test model.

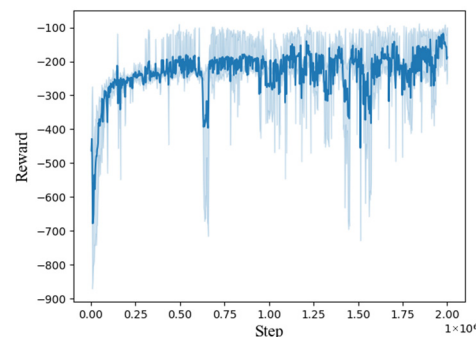


Figure 11. Average reward during training.

After training the service spacecraft several times with different random seeds, two distinct intelligent obstacle avoidance strategies emerged. The first strategy encourages the service spacecraft to bypass the obstacle group, resulting in a relatively safe path, which is referred to as the conservative strategy. The second strategy drives the service spacecraft passing through the obstacle group across a narrower safe passage, which is referred to as the radical strategy. The Monte Carlo shooting method is employed to test the experiment performance of these two strategies 100 times. Figure 12 illustrates the trajectory diagrams of 100 episodes for both strategies. All of the 100 episodes' service spacecraft start from the initial departure area (the yellow dotted circle) in the upper right corner and reach the expected area (the orange dotted circle) in the lower left of the figure. The left diagram represents the trajectory of the conservative strategy, while the right diagram depicts the trajectory of the radical strategy. In both strategies, the service spacecraft successfully avoids obstacles and reaches the expected point. It is evident from the figure that the trajectories of the service spacecraft, which depart from different positions in the initial departure area at varying speeds, ultimately converge to a safe proximity trajectory. Moreover, the curvature of this trajectory demonstrates the avoidance trends away from obstacles. In the conservative strategy, due to the obstacle group clustering in the upper part of the observation space, the service spacecraft flies along the boundary of the safe area beneath obstacles 1 and 2, thereby circumventing the obstacle avoidance issue with obstacle 3. In the radical strategy, the service spacecraft first passes through the safe area between obstacles 1 and 3 and then passes through the safe area between obstacles 2 and 3, ultimately successfully reaching the expected point. The conservative path is at a lower collision risk than the radical path.

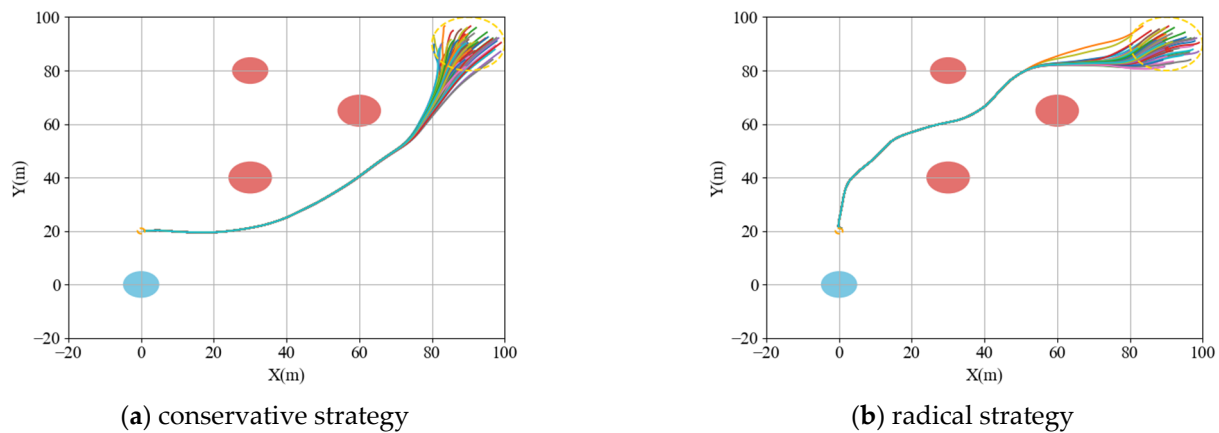


Figure 12. Trajectory chart for 100 test episodes.

Figure 13 shows the statistical graph of the minimum distance between the service spacecraft and obstacles for 100 episodes. It can be observed that the minimum distance between the service spacecraft and the outer contour of each obstacle exceeds the required safe distance of 5 m (the red dotted line). In the conservative strategy, the minimum distance between the service spacecraft and obstacle 3 reaches 40 m, which exceeds the radius of the observation range, indicating that obstacle 3 has not entered the view field of the service spacecraft during the whole flight, which reduces the number of obstacles that need to be avoided and lowers the complexity of the task. The minimum distance between the service spacecraft and obstacles 1 and 2 is approximately 6 m, slightly greater than the safe distance of 5 m. This indicates that the service spacecraft is flying along the boundary of the safe area beneath these two obstacles. In the radical strategy, the minimum distance between the service spacecraft and obstacles 1 and 3 fluctuates closely around 6 m, while the minimum distance between the service spacecraft and obstacle 2 is around 8 m, which is due to the service spacecraft passing through the narrow passage between the three obstacles, and the service spacecraft is relatively close to all of them.

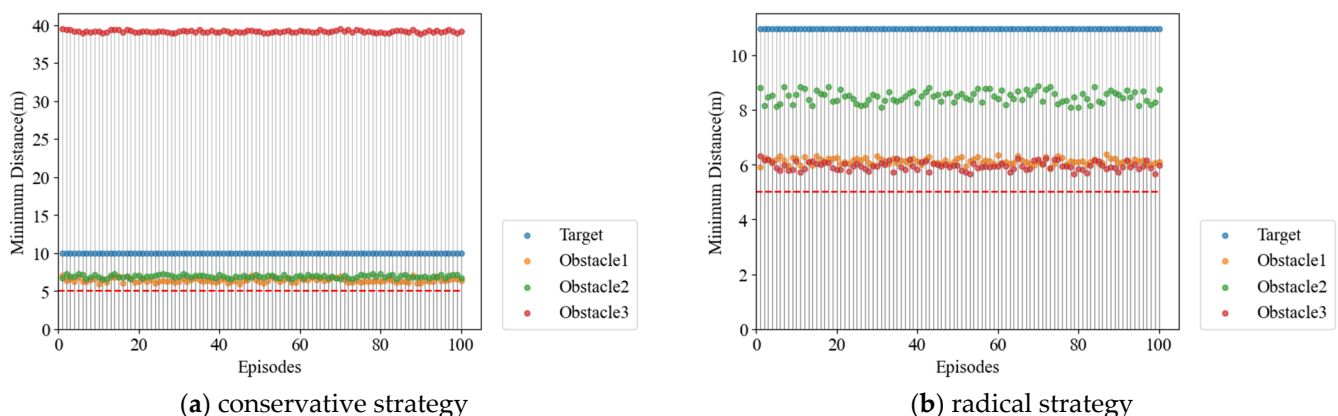


Figure 13. Minimum distance between the service spacecraft and obstacles for 100 test episodes.

Figure 14 shows the distribution of reward, cost, and constraint violation for 100 episodes. The left panel of Figure 14 illustrates the distribution of constraint violation and reward for 100 episodes. It can be observed that neither strategy violates the safety constraints. The right panel of Figure 14 displays the distribution of fuel consumption and reward for 100 episodes. Compared to the conservative strategy, the radical strategy yields lower rewards and higher fuel consumption. This is because the radical strategy requires real-time control of the service spacecraft to navigate along the safe boundaries of

three obstacles when passing through the obstacle cluster, ensuring its state remains within a narrow safe passage that always satisfies the collision avoidance constraints.

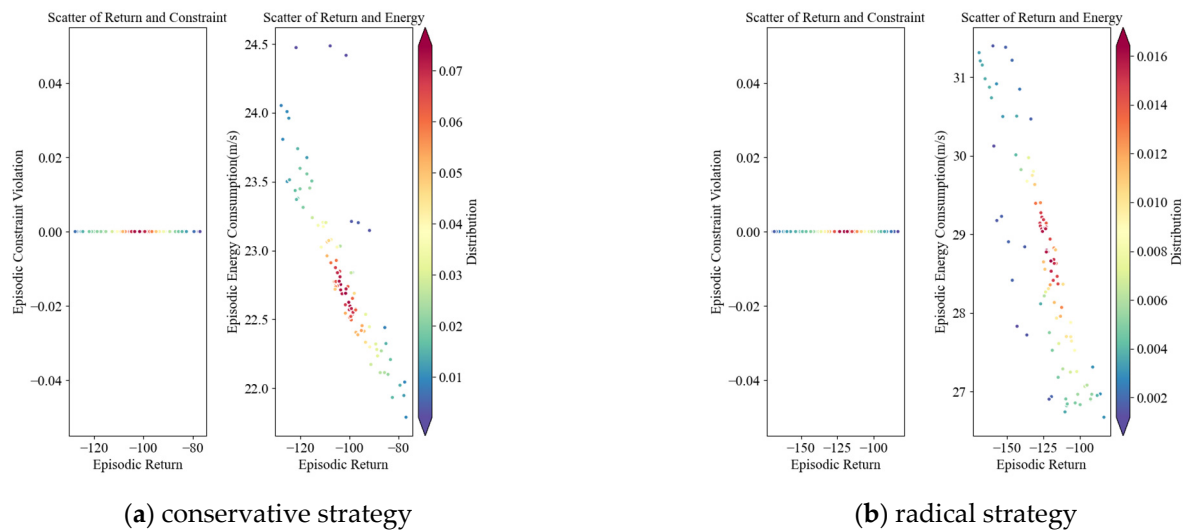


Figure 14. Distribution diagram for 100 test episodes. The left is the distribution diagram of constraint violation and reward, and the right is the distribution diagram of fuel consumption and reward.

Then, to intuitively demonstrate the flight situation of the service spacecraft, a test episode is selected to analyze its position, velocity, acceleration, reward function, and cost function. The trajectory for a typical episode is shown in Figure 15. Figures 16 and 17 illustrate the variations in the position error and velocity error of the service spacecraft. In both strategies, the position can converge to 1 m around the expected point, and the velocity can converge to within 0.1 m/s. This indicates that the GAT-SACL control strategy can effectively achieve the safe proximity task. Figure 18 shows the variations in the acceleration of the service spacecraft. The magnitudes of the accelerations for both strategies satisfy the upper limit requirements.

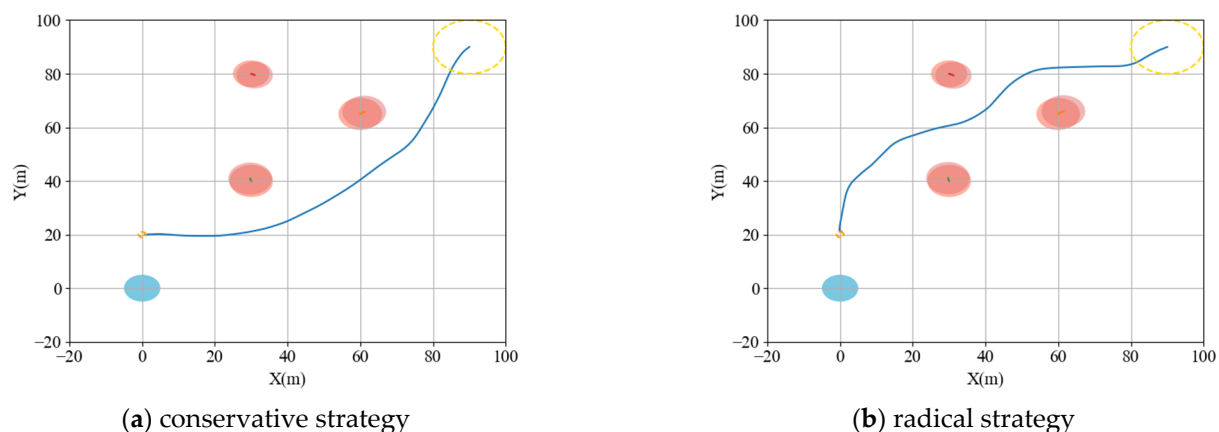
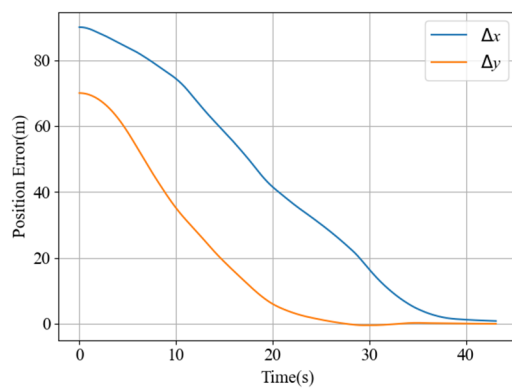
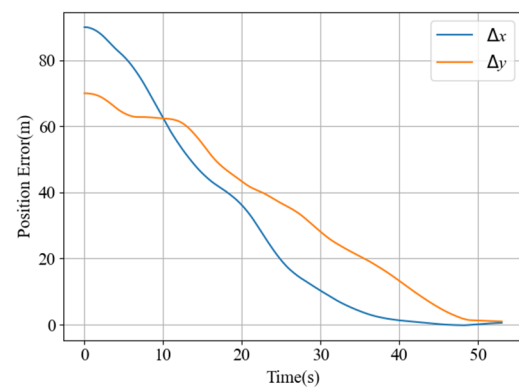


Figure 15. Trajectory chart in a typical episode.

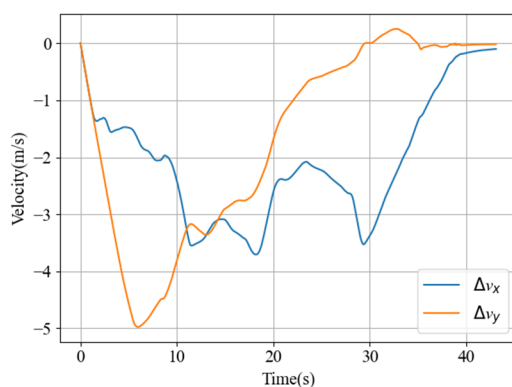
Figure 19 illustrates the variation in the minimum distance between the service spacecraft and obstacles. It can be observed that as the service spacecraft approaches the expected point, the distance to the target spacecraft gradually decreases, while the distance to various space debris generally exhibits a trend of decreasing first and then increasing. This phenomenon is consistent with intuitive analysis. Moreover, the distance to all obstacles remains greater than the minimum safe distance (the red dotted line), indicating that the service spacecraft always meets the safety constraints of obstacle avoidance.



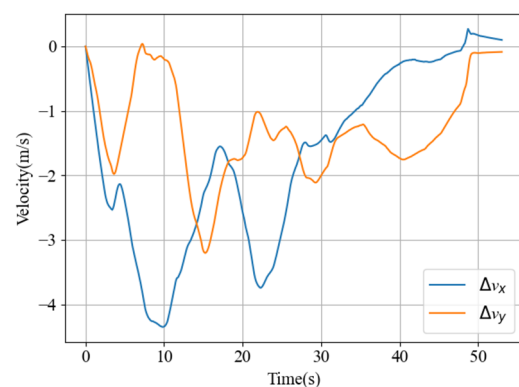
(a) conservative strategy



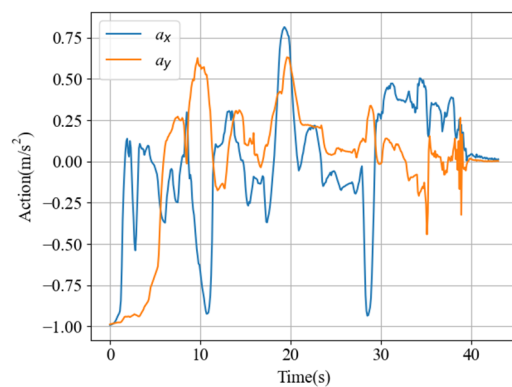
(b) radical strategy

Figure 16. Curve of the position error of the service spacecraft in a typical episode.

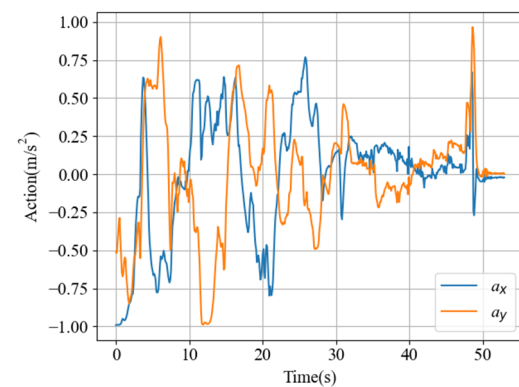
(a) conservative strategy



(b) radical strategy

Figure 17. Curve of the velocity error of the service spacecraft in a typical episode.

(a) conservative strategy



(b) radical strategy

Figure 18. Curve of the acceleration of the service spacecraft in a typical episode.

Figures 20 and 21 show the variation in the reward function and cost function, respectively. It can be observed that the reward function remains negative and gradually increases before the spacecraft approaches the expected point. When the service spacecraft satisfies the terminal constraints, a very large positive reward is obtained, indicating the completion of the task. The cost function remains zero because it always adheres to the obstacle avoidance constraints during the whole process.

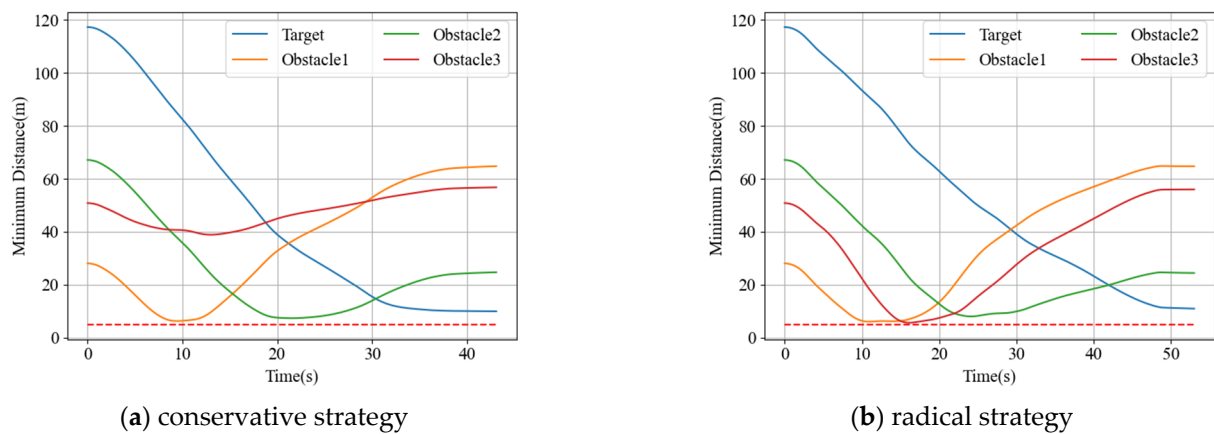


Figure 19. Curve of the distance between the service spacecraft and obstacles in a typical episode.

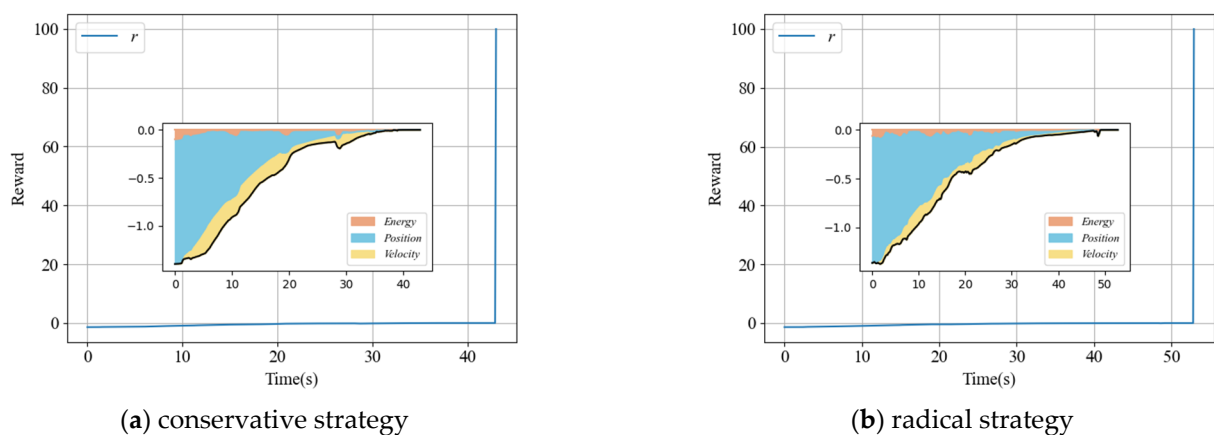


Figure 20. Curve of the reward of the service spacecraft in a typical episode.

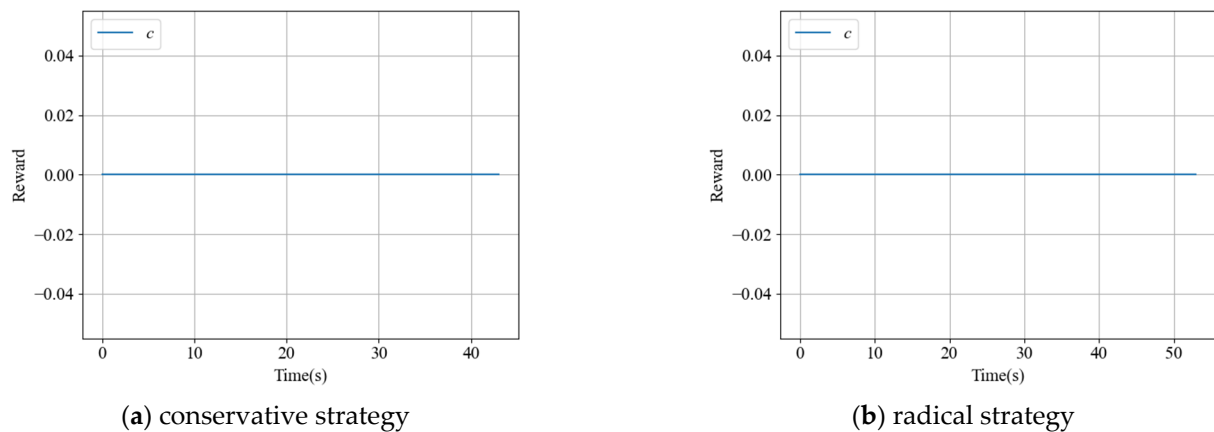


Figure 21. Curve of the cost of the service spacecraft in a typical episode.

Figure 22 shows the results of the pseudo-spectral method by the GPOPS toolbox to solve the safe proximity task. From the trajectory diagram, it can be found that the optimization result is very similar to the radical strategy of reinforcement learning, and the service spacecraft passes through the obstacle group to reach the expected point. Compared with GAT-SACL, the trajectory optimized by the pseudo-spectral method has small fluctuations, but the position and velocity change curve is relatively smooth, and the terminal accuracy is very high. This distinction is because the pseudo-spectral method is only optimized for certain events, while the reinforcement learning method is optimized for random events, and its strategy has stronger adaptability.

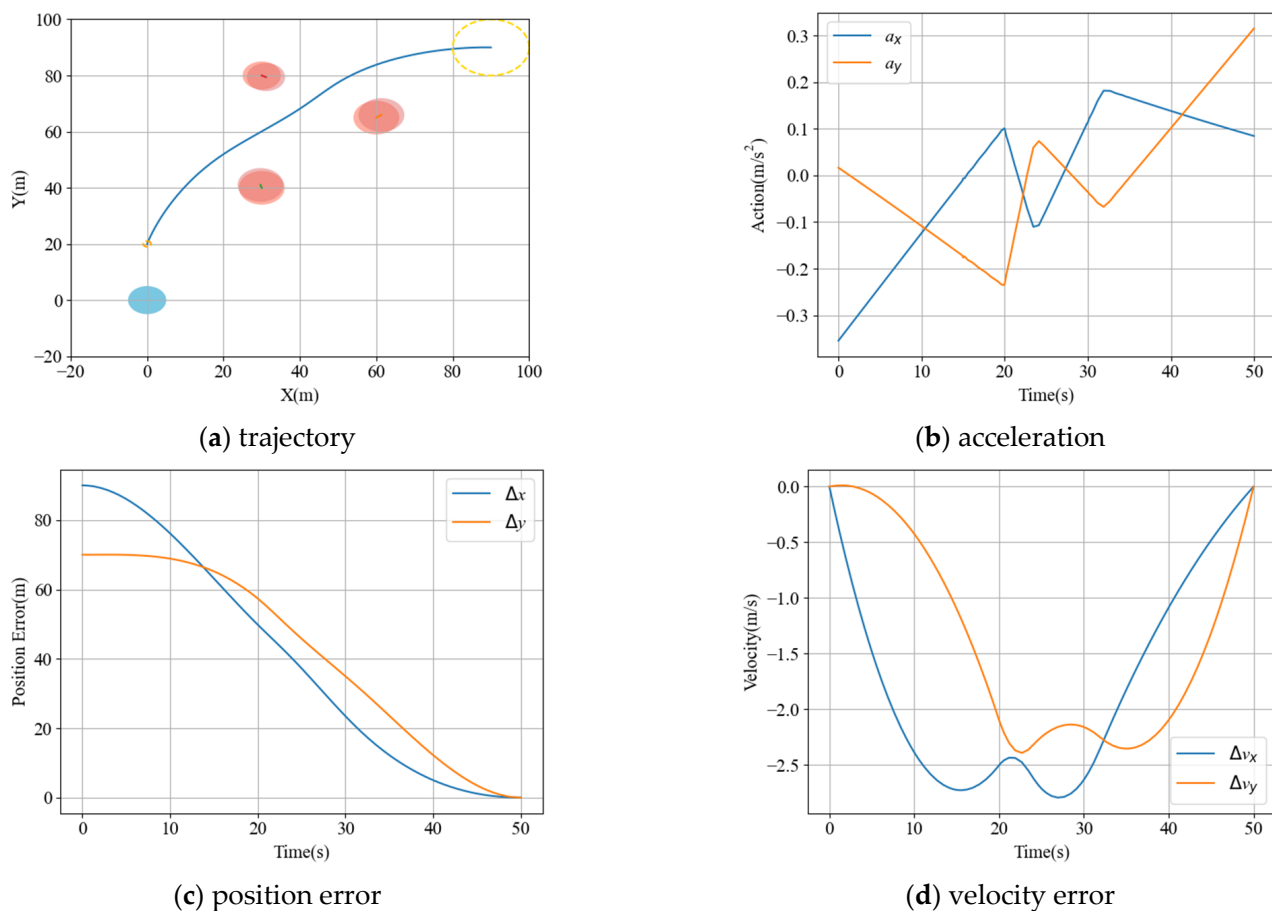


Figure 22. Results of GPOPS.

Table 3 displays the comparison of test results. The 95% confidence intervals of the mean values of episode reward, episode cost, success rate, collision rate, fuel consumption, and episode time are calculated by using the bias correction accelerated bootstrap method (BCA method, resampling $b = 10,000$ times). During training, both the GAT-SACL and GAT-SAC algorithms produced conservative and radical strategies. According to the table, the reward function of the conservative strategy is greater than that of the radical strategy, and the fuel consumption of the conservative strategy is less than that of the radical strategy. The reward of the GAT-SACL conservative strategy is ahead of that of GAT-SAC, and the reward of the GAT-SACL radical strategy is slightly inferior to that of GAT-SAC. In comparison to the GAT-SAC algorithm, the advantage of the GAT-SACL algorithm is that it can learn the strategies that meet the security constraints as soon as possible. Compared with reinforcement learning algorithms, the fuel consumption of the GPOPS algorithm is the lowest, only half of that of GAT-SACL. However, when facing the dynamic environment, GAT-SACL learned a variety of different obstacle avoidance strategies through exploration, which reflects the robustness and strong exploration ability of the reinforcement learning algorithm.

In order to further verify the generalization of the algorithm, five task scenarios are designed, as shown in Table 4. Table 5 illustrates the results of the generalization experiment. In Case 1, the randomness of the initial position of obstacles is considered, and the success rates for both strategies are reduced to varied extents. The success rate of the radical strategy is much lower than that of the conservative strategy. This is because the conservative strategy avoids the outer edge of the obstacle group, while the radical strategy shuttles within the obstacle group and is more susceptible to the positional variations of individual debris. Case 2 and Case 3 focus on the increase in the number of obstacles. In

Case 2, obstacle 4 is added between obstacles 1 and 2. The initial position of obstacle 4 is randomly generated in the area with the initial position as the center and a radius of 3 m. Because the conservative strategy chooses to move along the lower edge of obstacles 1 and 2, its success rate is less affected, while the aggressive strategy seeks a trajectory among the obstacle group, which is greatly affected by obstacle 4. Case 3 adds obstacle 5 below the conservative trajectory and obstacle 6 above the radical trajectory to observe the impact of obstacle position on strategy selection. It is found that the conservative strategy and the radical strategy continue to operate along their respective trajectories, with no interchange between the two strategies. For the obstacle avoidance of new fragments outside the initial scene obstacle group, the radical strategy performs better, and for the obstacle avoidance of new fragments within the obstacle group in the initial scene, the conservative strategy performs better. Case 4 and Case 5 examine the impact of observation area range on the results. The sensing radius is narrowed in Case 4 and is expanded in Case 5. From the table, it can be found that increasing the observation area can slightly improve the expected reward. On the one hand, increasing the observation area helps the service spacecraft perceive obstacles in advance to avoid them. On the other hand, because the global obstacle parameters are utilized in the critic network to measure policy, the gain effect is not significant. Through the above series of generalization experiments, it is verified that the algorithm has certain robustness for space debris with different positions and higher density, and the service spacecraft can still reach the terminal expected point even if the risk of collision with space debris increases.

Table 3. Comparison of test results.

Policy	Episode Reward	Episode Cost	Success	Collision	Fuel Consumption (m/s)	Episode Time (s)
GAT-SACL (conservative)	−102.796 [−105.150, −100.489]	0	100%	0%	22.855 [22.753, 22.972]	43.068 [42.942, 43.194]
GAT-SACL (radical)	−124.074 [−128.096, −120.363]	0	100%	0%	28.640 [28.396, 28.909]	52.778 [52.519, 53.047]
GAT-SAC (conservative)	−119.164 [−122.224, −116.348]	0	100%	0%	28.910 [28.795, 29.013]	64.353 [64.173, 64.542]
GAT-SAC (radical)	−123.202 [−126.138, −120.461]	0	100%	0%	27.077 [26.996, 27.164]	56.591 [56.430, 56.757]
GPOPS	-	-	100%	0%	13.279 [12.566, 14.096]	50

Table 4. Design of generalized experimental scenes.

Number	Name	Parameters	Value
Case 1	Random Initial Position of Obstacles	Radius of Initial Position Area (m)	3
Case 2	4 Pieces of Space Debris	Initial Position of Space Debris 4 (m) Radius of Space Debris 4 (m)	[45, 50] 5
Case 3	6 Pieces of Space Debris	Initial Position of Space Debris 5 (m) Radius of Space Debris 5 (m) Initial Position of Space Debris 6 (m) Radius of Space Debris 6 (m)	[20, 60] 5 [50, 100] 5
Case 4	Reduced Sensing Radius	Observation Range of Service Spacecraft (m)	20
Case 5	Extended sensing Radius	Observation Range of Service Spacecraft (m)	30

Table 5. Generalization results for different cases.

Case	Policy	Episode Reward	Episode Cost	Success	Collision	Fuel Consumption (m/s)	Episode Time (s)
Case 1	GAT-SACL (conservative)	−105.406 [−107.851, −103.098]	5.22 [3.14, 8.09]	82% [72%, 88%]	18% [10%, 25%]	22.855 [22.753, 22.972]	43.068 [42.942, 43.194]
Case 1	GAT-SACL (radical)	−132.162 [−136.994, −127.7485]	16.19 [11.8, 21.856]	62% [51%, 70%]	38% [28%, 47%]	28.642 [28.397, 28.909]	52.766 [52.507, 53.034]
Case 2	GAT-SACL (conservative)	−103.166 [−105.646, −100.765]	0.74 [0.20, 1.84]	95% [88%, 98%]	5% [1%, 9%]	22.863 [22.761, 22.979]	43.063 [42.938, 43.188]
Case 2	GAT-SACL (radical)	−126.693 [−130.737, −122.823]	5.24 [3.40, 7.67]	77% [67%, 84%]	23% [15%, 31%]	28.640 [28.395, 28.908]	52.778 [52.519, 53.047]
Case 3	GAT-SACL (conservative)	−106.133 [−108.835, −103.563]	6.68 [4.664, 9.03]	68% [58%, 76%]	32% [22%, 40%]	22.859 [22.757, 22.975]	43.068 [42.942, 43.194]
Case 3	GAT-SACL (radical)	−126.704 [−130.757, −122.834]	5.25 [3.41, 7.67]	77% [67%, 84%]	23% [15%, 31%]	28.643 [28.399, 28.911]	52.777 [52.518, 53.046]
Case 4	GAT-SACL (conservative)	−102.800 [−105.152, −100.493]	0	100%	0%	22.864 [22.761, 22.979]	43.068 [42.943, 43.193]
Case 4	GAT-SACL (radical)	−124.078 [−128.083, −120.364]	0	100%	0%	28.652 [28.407, 28.920]	52.757 [52.496, 53.025]
Case 5	GAT-SACL (conservative)	−102.797 [−105.150, −100.489]	0	100%	0%	22.863 [22.762, 22.979]	43.063 [42.937, 43.189]
Case 5	GAT-SACL (radical)	−124.067 [−128.083, −120.354]	0	100%	0%	28.594 [28.349, 28.861]	52.779 [52.521, 53.049]

5. Conclusions

In this paper, an SACL algorithm based on a graph attention network (GAT-SACL) is proposed to address two challenges, including the partially observable environment and obstacle avoidance safety constraints. On the one hand, the graph neural network is introduced to depict the dynamic observation environment and is utilized to extract the hidden graph structure information to assist the reinforcement learning algorithm to train the actor network. On the other hand, a Lagrange multiplier is introduced into the reinforcement learning algorithm to ensure that the safety constraints of obstacle avoidance are met. The simulation results show that the proposed algorithm has significant advantages in balancing optimality and security and excels in exploring multiple strategies.

In the future, the single-agent safe reinforcement learning algorithm will be further extended to a multi-agent environment to achieve spacecraft cluster safe proximity in complex and changing environments.

Author Contributions: Conceptualization, H.Z. and Y.B.; methodology, H.Z. and R.C.; software, H.Z. and M.D.; validation, J.W. and H.Z.; formal analysis, J.W.; investigation, M.D. and R.C.; resources, R.C.; data curation, J.W. and M.D.; writing—original draft preparation, H.Z.; writing—review and editing, Y.B. and R.C.; visualization, M.D.; supervision, Y.Z.; project administration, Y.Z.; funding acquisition, Y.B. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under grant 12502410, 12472047, and 62401597, the Foundation of the National Key Laboratory of Science and Technology on Space-Born Intelligent Information Processing under grant TJ-03-25-01, and the Natural Science Foundation of Hunan Province, China, under grant 2025JJ50016.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Li, W.-J.; Cheng, D.-Y.; Liu, X.-G.; Wang, Y.-B.; Shi, W.-H.; Tang, Z.-X.; Gao, F.; Zeng, F.-M.; Chai, H.-Y.; Luo, W.-B.; et al. On-Orbit Service (OOS) of Spacecraft: A Review of Engineering Developments. *Prog. Aerosp. Sci.* **2019**, *108*, 32–120. [\[CrossRef\]](#)
- Chen, R.; Chen, Z.; Bai, Y.; Zhao, Y.; Yao, W.; Wang, Y. Ground Experiment of Safe Proximity Control for Complex-Shaped Spacecraft. *IEEE Trans. Ind. Electron.* **2023**, *70*, 11535–11543. [\[CrossRef\]](#)
- Zhang, J.; Chu, X.; Zhang, Y.; Hu, Q.; Zhai, G.; Li, Y. Safe-Trajectory Optimization and Tracking Control in Ultra-Close Proximity to a Failed Satellite. *Acta Astronaut.* **2018**, *144*, 339–352. [\[CrossRef\]](#)
- Morgan, D.; Chung, S.-J.; Hadaegh, F.Y. Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming. *J. Guid. Control Dyn.* **2014**, *37*, 1725–1740. [\[CrossRef\]](#)
- Wang, Y.; Chen, X.; Ran, D.; Zhao, Y.; Chen, Y.; Bai, Y. Spacecraft Formation Reconfiguration with Multi-Obstacle Avoidance under Navigation and Control Uncertainties Using Adaptive Artificial Potential Function Method. *Astrodynamics* **2020**, *4*, 41–56. [\[CrossRef\]](#)
- Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [\[CrossRef\]](#)
- Zhou, M.; Luo, J.; Vilella, J.; Yang, Y.; Rusu, D.; Miao, J.; Zhang, W.; Alban, M.; Fadakar, I.; Chen, Z.; et al. SMARTS: Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. *arXiv* **2010**, arXiv:2010.09776.
- Tipaldi, M.; Iervolino, R.; Massenio, P.R. Reinforcement Learning in Spacecraft Control Applications: Advances, Prospects, and Challenges. *Annu. Rev. Control* **2022**, *54*, 1–23. [\[CrossRef\]](#)
- Scorsoglio, A.; Furfaro, R.; Linares, R.; Massari, M. Relative Motion Guidance for Near-Rectilinear Lunar Orbits with Path Constraints via Actor-Critic Reinforcement Learning. *Adv. Space Res.* **2023**, *71*, 316–335. [\[CrossRef\]](#)
- Federici, L.; Scorsoglio, A.; Zavoli, A.; Furfaro, R. Meta-Reinforcement Learning for Adaptive Spacecraft Guidance during Finite-Thrust Rendezvous Missions. *Acta Astronaut.* **2022**, *201*, 129–141. [\[CrossRef\]](#)
- Hovell, K.; Ulrich, S. Deep Reinforcement Learning for Spacecraft Proximity Operations Guidance. *J. Spacecr. Rocket.* **2021**, *58*, 254–264. [\[CrossRef\]](#)
- Yang, L.; Wang, J.; Jiang, J.; Bai, X.; Xu, M.J. Low-orbit Space Debris Warning and Autonomous Collision Avoidance for Space Environment Governance. *Phys. Conf. Ser.* **2025**, *3015*, 012005. [\[CrossRef\]](#)
- Zhang, J.; Zhang, K.; Zhang, Y.; Shi, H.; Tang, L.; Li, M. Near-Optimal Interception Strategy for Orbital Pursuit-Evasion Using Deep Reinforcement Learning. *Acta Astronaut.* **2022**, *198*, 9–25. [\[CrossRef\]](#)
- Li, X.; Wang, X. Online Solution for Orbital Pursuit-Evasion Game via Heterogeneous Proximal Policy Optimization. *IEEE Trans. Aerosp. Electron. Syst.* **2025**, *61*, 12044–12058. [\[CrossRef\]](#)
- Qu, Q.; Liu, K.; Wang, W.; Lu, J. Spacecraft Proximity Maneuvering and Rendezvous with Collision Avoidance Based on Reinforcement Learning. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 5823–5834. [\[CrossRef\]](#)
- Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D. Deterministic Policy Gradient Algorithms. In *Proceedings of the International Conference on Machine Learning, ICML, Beijing, China, 21–26 June 2014*; JMLR: Cambridge, MA, USA, 2014; pp. 387–395.
- Sharma, K.P.; Kumar, I.; Singh, P.P.; Anbazhagan, K.; Albarakati, H.M.; Bhatt, M.W.; Ziyadullayevich, A.A.; Rana, A.; A, S.S. Advancing Spacecraft Rendezvous and Docking through Safety Reinforcement Learning and Ubiquitous Learning Principles. *Comput. Hum. Behav.* **2024**, *153*, 108110. [\[CrossRef\]](#)
- Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347. [\[CrossRef\]](#)
- Zhang, L.; Zhang, Q.; Shen, L.; Yuan, B.; Wang, X.; Tao, D. Evaluating Model-Free Reinforcement Learning toward Safety-Critical Tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023*; AAAI: Washington, DC, USA, 2023; Volume 37, pp. 15313–15321.
- Gu, S.; Yang, L.; Du, Y.; Chen, G.; Walter, F.; Wang, J.; Knoll, A. A Review of Safe Reinforcement Learning: Methods, Theories, and Applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 11216–11235. [\[CrossRef\]](#) [\[PubMed\]](#)
- Garcia, J.; Fernandez, F. A Comprehensive Survey on Safe Reinforcement Learning. *J. Mach. Learn. Res.* **2015**, *16*, 1437–1480.
- Ha, S.; Xu, P.; Tan, Z.; Levine, S.; Tan, J. Learning to walk in the real world with minimal human effort. *arXiv* **2020**, arXiv:2002.08550. [\[CrossRef\]](#)
- Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A. Soft Actor-Critic Algorithms and Applications. *arXiv* **2018**, arXiv:1812.05905.
- Ray, A.; Achiam, J.; Amodei, D. Benchmarking Safe Exploration in Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1910.01708.
- Mu, C.; Liu, S.; Lu, M.; Liu, Z.; Cui, L.; Wang, K. Autonomous Spacecraft Collision Avoidance with a Variable Number of Space Debris Based on Safe Reinforcement Learning. *Aerosp. Sci. Technol.* **2024**, *149*, 109131. [\[CrossRef\]](#)
- Zhang, L.; Shen, L.; Yang, L.; Chen, S.; Yuan, B.; Wang, X.; Tao, D. Penalized Proximal Policy Optimization for Safe Reinforcement Learning. *arXiv* **2022**, arXiv:2205.11814. [\[CrossRef\]](#)

27. Xue, X.; Yue, X.; Yuan, J. Connectivity Preservation and Collision Avoidance Control for Spacecraft Formation Flying in the Presence of Multiple Obstacles. *Adv. Space Res.* **2021**, *67*, 3504–3514. [\[CrossRef\]](#)
28. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph Neural Networks: A Review of Methods and Applications. *AI Open* **2020**, *1*, 57–81. [\[CrossRef\]](#)
29. Khemani, B.; Patil, S.; Kotecha, K.; Tanwar, S. A Review of Graph Neural Networks: Concepts, Architectures, Techniques, Challenges, Datasets, Applications, and Future Directions. *J. Big Data* **2024**, *11*, 18. [\[CrossRef\]](#)
30. Munikoti, S.; Agarwal, D.; Das, L.; Halappanavar, M.; Natarajan, B. Challenges and Opportunities in Deep Reinforcement Learning with Graph Neural Networks: A Comprehensive Review of Algorithms and Applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 15051–15071. [\[CrossRef\]](#)
31. Zhao, B.; Huo, M.; Li, Z.; Yu, Z.; Qi, N. Graph-Based Multi-Agent Reinforcement Learning for Large-Scale UAVs Swarm System Control. *Aerosp. Sci. Technol.* **2024**, *150*, 109166. [\[CrossRef\]](#)
32. Zhao, B.; Huo, M.; Li, Z.; Feng, W.; Yu, Z.; Qi, N.; Wang, S. Graph-Based Multi-Agent Reinforcement Learning for Collaborative Search and Tracking of Multiple UAVs. *Chin. J. Aeronaut.* **2025**, *38*, 103214. [\[CrossRef\]](#)
33. Yang, M.; Liu, G.; Zhou, Z.; Wang, J. Partially Observable Mean Field Multi-Agent Reinforcement Learning Based on Graph Attention Network for UAV Swarms. *Drones* **2023**, *7*, 476. [\[CrossRef\]](#)
34. Lai, Y.; Zhu, Y.; Li, L.; Lan, Q.; Zuo, Y. STGLR: A Spacecraft Anomaly Detection Method Based on Spatio-Temporal Graph Learning. *Sensors* **2025**, *25*, 310. [\[CrossRef\]](#)
35. Jacquet, A.; Infantes, G.; Meuleau, N.; Benazera, E.; Roussel, S.; Baudoui, V.; Guerra, J. Earth Observation Satellite Scheduling with Graph Neural Networks. *arXiv* **2024**, arXiv:2408.15041. [\[CrossRef\]](#)
36. Clohessy, W.H.; Wiltshire, R.S. Terminal Guidance System for Satellite Rendezvous. *J. Aerosp. Sci.* **1960**, *27*, 653–658. [\[CrossRef\]](#)
37. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2016**, arXiv:1609.02907.
38. Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R.; Smola, A. Deep sets. *arXiv* **2017**, arXiv:1703.06114. [\[PubMed\]](#)
39. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
40. Yang, Q.; Simão, T.D.; Tindemans, S.H.; Spaan, M.T.J. WCSAC: Worst-case soft actor critic for safety-constrained reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021*; AAAI: Washington, DC, USA, 2023; Volume 35, pp. 10639–10646.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.