Enhanced PTZ Camera Dispatch Scheme for 3D Environments Based on Deep Reinforcement Learning

Jia-Ming Liang, *Member, IEEE*, Shashank Mishra, and Chih-Hong Lin

***Abstract*—In recent years, pan-tilt-zoom (PTZ) cameras with horizontal, vertical rotation, and zoom capabilities have enhanced the flexibility of fixed cameras. However, existing research on PTZ cameras has not adequately addressed key monitoring requirements such as viewing angle and resolution (pixels per foot), often resulting in skewed perspectives or insufficient imaging quality. This study focuses on 3D environments and aims to optimize PTZ camera dispatch for maximum target coverage while minimizing the number of cameras used. We formulate the problem as a mixed-integer linear programming problem, which is known to be NP-hard, and propose an intelligent solution using Fast-Prioritized Experience Deep Q-Network (FPE-DDQN). This method leverages self-awareness and iterative learning to improve decision-making by incorporating past dispatch experiences and an innovative reward function design. FPE-DDQN prioritizes cameras based on their coverage potential and dispatch cost, significantly reducing the number of cameras needed while accelerating the adjustment of rotation angles and focal lengths. Experimental simulations demonstrate that FPE-DDQN achieves near-optimal coverage and dispatch efficiency, with the performance gap to the optimal solution being less than 0.52% to 4.7%, while requiring only 1/10 to 1/100 of the computational time compared to the optimal solution.**

***Index Terms*—monitor resolution, PTZ camera, reinforcement learning, video surveillance, view angle.**

I. Introduction

In various environments such as factories, offices, residential areas, and department stores, unexpected incidents such as fires, casualties, explosions, and security events may occur at any time. These events require immediate viewing and monitoring to ensure the safety of lives and property. However, traditional surveillance cameras can only capture fixed directions, the scenes, and monitor fixed ranges. In the event of incidents occurring outside the surveillance range, such as theft, malicious damage, or dangerous accidents, it becomes challenging to promptly and accurately view the area of the incident. In addition, if the surveillance target is beyond the camera's zoom range (e.g., too close or too far) or the viewing angle is too skewed, it can result in ineffective imaging issues. Moreover, achieving comprehensive surveillance would require deploying a large number of fixed cameras, incurring substantial costs. Therefore, *pan-tilt-zoom (PTZ)* cameras with horizontal, vertical rotation, and zoom capabilities have been developed [1]-[3]. Compared to traditional fixed cameras, they possess dynamic and flexible control capabilities, allowing real-time rotation and zoom to reduce skewed perspectives and monitor blind spots, while enhancing imaging resolution.

However, current literature on relevant PTZ cameras [4]-[6] has not adequately addressed surveillance requirements in a 3D environment, including imaging resolution and viewing angles. Imaging resolution represents the number of pixels per foot in an image, measured in *pixels-per-foot (PPF)*. The viewing angle indicates the angle formed between the vector from a point on the covered object to the camera and the facing vector of the covered target. Without sufficient consideration of these monitoring requirements, precise surveillance cannot be achieved [7],[8]. Therefore, this paper comprehensively considers the above monitoring conditions and addresses a high-coverage, low-cost, and efficient PTZ camera dispatch problem. Initially, we model this problem as a mixed-integer linear programming (MILP), which is known as an NP-hard problem. Subsequently, we propose an intelligent method that utilizes *deep reinforcement learning (DRL)* to calculate and determine the best camera dispatch configuration.

Specifically, this method is based on the self-learning and iterative computation of reinforcement learning. It leverages past successful learning experiences and ε-greedy selection strategies, combined with a clever reward function emphasizing high coverage and low cost. Simultaneously, it reduces the potential number of cameras to minimize dispatch sets, expediting the calculation of optimal camera rotation angles and focal length settings, thereby achieving higher surveillance coverage with fewer cameras. Experimental results validate that this method can approach the optimal solution in terms of surveillance coverage and number of required cameras, but takes only $1/10 \sim 1/100$ computational time of the optimal solution.

The major contributions of this paper are four-fold:

- First, to the best of our knowledge, this is the first paper to address the PTZ camera dispatch problem in 3D environments while fully considering surveillance quality in terms of viewing angle and resolution for 3D objects
- Second, we propose a novel *Fast-Prioritized Experience*

The authors are with the Department of Electrical Engineering, National University of Tainan, Tainan 701027, Taiwan. (e-mail: jmliang@ieee.org, d10982003@stumail.nutn.edu.tw, m10982012@@stumail.nutn.edu.tw)

Jia-Ming Liang is also a visiting scholar in the Department of Computer Science and Engineering at the University of Minnesota, Minneapolis, 55455, USA. (e-mail: liangj@umn.edu)

Corresponding authors: Jia-Ming Liang and Shashank Mishra.

Deep Q-Network (FPE-DDQN) for optimizing PTZ camera dispatch by integrating self-awareness, imitation augmented, and iterative learning, while elaborating an innovative reward function design that prioritizes cameras based on their coverage potential and dispatch cost.

- Third, we conduct extensive experiments to validate the performance of FPE-DDQN, showing that it approximates the optimal solution in terms of coverage ratio and required cameras. The proposed method achieves a remarkable reduction in computational time, requiring only 1/10 to 1/100 of the time compared to traditional optimization methods, demonstrating its practical feasibility for real-world applications.
- Fourth, we have implemented a prototype system to demonstrate its practical applicability in real-world scenarios and showcase its effectiveness in 3D environments.

The organization of this paper is as follows. Section II discusses the related literature; Section III describes the relevant background knowledge, including monitoring requirements, and then defines the problem of this paper; Section IV introduces the proposed reinforcement learning method; Section V evaluates the performance of the experimental methods. Finally, Section VI concludes the paper.

Table I. Comparison of Existing Works

State-of-the-art	Novel Issues					Performance Features		
	3D Coverage	View Angle	Resolution (PPF)	PTZ Camper (Pan, Tilt, Zoom in/out)	Real-Time	Dispatch Effectiveness	Deployment Cost	Computational Complexity
Work [9][10]		Δ				Δ		
Work [11][13]		Δ			Δ	✓		Δ
Work [12]	✓				✓	✓		
Work [14]	✓	✓	✓			✓		
Work [15]		✓		✓	✓	✓	✓	
Work [16][17]		✓		Δ	Δ	✓		
Work [18][19][20][21]	✓	✓		Δ	Δ	✓		Δ
Work [23][24][25]	Δ	✓	Δ	Δ				
Work [26][27][28]		✓			Δ	✓		Δ
Work [22]		✓	✓			✓	✓	
Our work	✓	✓	✓	✓	✓	✓	✓	✓

Meaning of Symbols: Available (✓); Partial available (Δ)

II. Related Work

In the literature, the works [9]-[12] have addressed coverage issues in the context of Internet of Things (IoT) sensing and monitoring applications. Among them, literature [9] proposes a smart optimization algorithm, resampled particle swarm optimization, to enhance coverage and node efficiency. The study [10] introduces an adaptive coverage and connectivity method to address coverage and connectivity issues in sensor networks while ensuring coverage and connectivity. The reference [11] presents a meta-heuristic algorithm based on genetic algorithms to maximize coverage in sensor networks and overcome limitations in existing meta-heuristic methods. The work [12] introduces a dynamic programming framework to optimize the overlapping coverage of visual sensors, employing a global greedy search algorithm to reduce redundant cameras. However, these studies [9]-[12] focus primarily on circular sensing areas and may not be suitable for applications requiring directional and sectoral coverage in video surveillance systems.

Therefore, the references [13]-[22] focus on coverage issues in visual sensing and image monitoring applications in 2D environments. The work [13] addresses uncovered angles by proposing a coverage enhancement algorithm with nodes featuring adjustable angles to repair coverage holes and achieve full-angle coverage. The study [14] introduces coverage using deformable contour shapes, utilizing multiple cameras to cover object boundaries, ensuring coverage performance. The work [15] presents two algorithms to address over-provisioned and under-provisioned Q-Coverage problems in camera networks. The literature [16] focuses on monitoring airport grounds and converts the area coverage problem into a boundary-deployed camera network coverage maximization problem, enhancing coverage through a heterogeneous camera network. The work [17] proposes an alternate global greedy algorithm to significantly improve traditional surveillance system strategies, achieving maximized coverage. The study [18] addresses coverage targets in closed annular belts, such as in ecological reserves or pollution monitoring, using distributed particle swarm optimization to enhance coverage efficiency. Subsequently, the references [19]-[21] introduce full-view coverage to capture objects from any direction, where the angle formed between the facing vector and the vector from the object to the camera must be below a specified threshold. Specifically, the work [19] studies deploying a minimal camera sensor network and proposes a greedy heuristic with the differential evolution algorithm to solve the problem. The study [20] investigates maximizing the number of full-view coverage targets by adjusting camera directions and introduces a pipage rounding-based approximation algorithm to enhance performance. The work [21] presents a centralized algorithm based on the largest demand first-serve strategy to reduce the cost and improve the efficiency of full-view coverage. The study [22] introduces an iterative screening algorithm to achieve full-view coverage while effectively reducing the deployment of numerous cameras. However, these studies [12]-[22] are applicable primarily to 2D plane coverage and may face challenges in monitoring targets with different heights and depths, such as those in 3D space, regions, and objects.

Therefore, the references [23]-[28] focus on coverage research in 3D environments. Specifically, the work [23] investigates the optimization problem of deploying 3D directional wireless sensor networks and proposes an improved differential evolution algorithm to increase coverage opportunities. The study [24] investigates maximizing coverage in 3D industrial scenes and extending network lifetime, introducing object detection techniques to address camera

coverage issues. The literature [25] explores maximizing camera coverage and utilizes convolutional long- and short-time memory algorithms to produce maximum coverage. The work [26] addresses the optimization problem of multi-target tracking, employing intelligent high frame rate (HFR) target-tracking algorithm to solve visual tracking issues. The study [27] proposes the establishment of a coverage intensity model and a novel scene partition, developing a parallel gradient optimization approach to achieve optimal coverage performance. The literature [28] uses multi-grid polygons to describe a 3D object, combining space partition, greedy, and local search to propose a network deployment algorithm for visual sensors. However, the above studies [23]-[28] insufficiently consider monitoring requirements for the target objects in 3D environments, including viewing angles and image resolution. Therefore, they may not fully ensure the quality of surveillance in a 3D environment. Table I summarizes the comparison of state-of-the-art.

III. Preliminary

In this section, we introduce the functionalities and control parameters of PTZ cameras. Then, we discuss the representation of monitored objects in 3D space and define two critical surveillance requirements, including the view angle and image resolution, and then introduce the coverage metric in 3D space. Finally, we define the problem of this paper.

A. PTZ Camera Architecture

The PTZ (pan-tilt-zoom) camera [1],[2],[3] is a novel type of camera equipped with dynamic control capabilities such as pan (left-right rotation), tilt (up-down inclination), and zoom (focal length adjustment), as illustrated in Fig. 1. This camera features remote control capabilities, allowing immediate adjustment of horizontal rotation angles, vertical tilt angles, and focal lengths through control settings, enabling real-time monitoring of different field views.

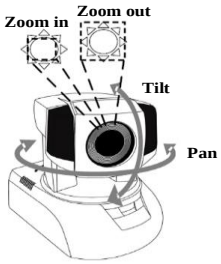


Fig. 1. PTZ camera [1],[2],[3]

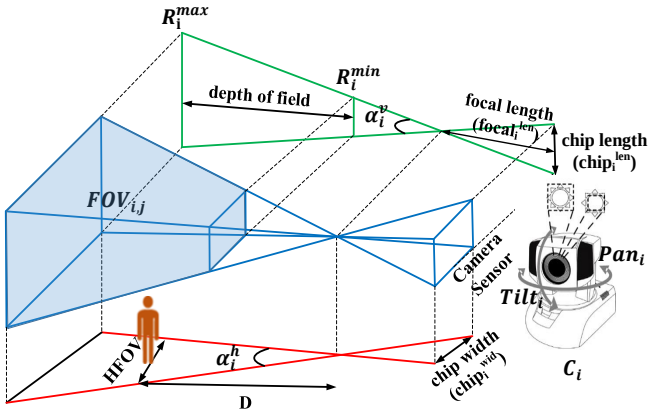


Fig. 2. The field of view (FoV) for a PTZ camera

In this paper, we use $C_i, i = 1..N$ to represent the i-th PTZ camera. Since PTZ cameras can form different *fields of view (FoV)* depending on different rotation parameter settings, we use $FoV_{i,j}$ as the j-th field of view generated by the camera C_i, which will be set according to different horizontal angles (Pan_i), different vertical inclination angles ($Tilt_i$), and different focal lengths ($focal_i^{len}$), i.e., $FoV_{i,j} \in FoV_i = \{(Pan_i, Tilt_i, focal_i^{len})\}$. This can be illustrated in Fig. 2.

Specifically, each field of view $FoV_{i,j}$ includes an angle of view and a depth of field. The angle of view can be divided into horizontal view (α_i^h) and vertical view (α_i^v), where the horizontal angle of view (α_i^h) is related to the focal length ($focal_i^{len}$) setting of the camera and the width of the image sensor ($chip_i^{wid}$), which is defined as follows:

$$\alpha_i^h = 2tan^{-1}\left(\frac{chip_i^{wid}}{2focal_i^{len}}\right). \quad (1)$$

The vertical angle of view (α_i^v) is related to the length of the focal length ($focal_i^{len}$) and the length of the image sensor ($chip_i^{len}$), which is defined as follows:

$$\alpha_i^v = 2tan^{-1}\left(\frac{chip_i^{len}}{2focal_i^{len}}\right). \quad (2)$$

In addition, the depth of field refers to the relative imaging distance before and after the focal point, and the minimum and maximum lengths of this distance range are R_i^{min} and R_i^{max}, where R_i^{min} is half of the pan focus distance [29], which is calculated by

$$R_i^{min} = \frac{focal_i^{len}}{2}.$$

The distance R_i^{max} is between the pan focal distance and the image sensor defined at infinity is called the hyperfocal distance, which is defined by

$$R_i^{max} = \frac{focal_i^{len} \times number_{pixels}{}^h}{chip_i^{wid} \times Resolution^{req}}, \quad (3)$$

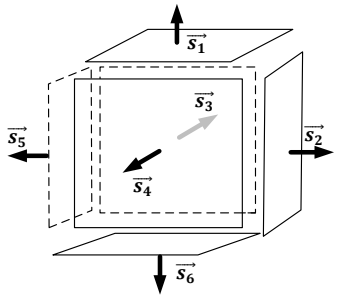
where $Resolution^{req}$ represents the resolution requirement (this will be clear later on), while $number_{pixels}{}^h$ denotes the number of horizontal pixels in the image. Consequently, if the distance between the target object and the camera exceeds R_i^{max} or falls below R_i^{min}, concrete imaging becomes unfeasible [21]. As illustrated in Fig. 2, the two triangles in the horizontal direction of the camera's FoV are similar triangles. Therefore, the ratio of R_i^{max} to HFOV is equivalent to the ratio of focal length $focal_i^{len}$ to photosensitive component width $chip_i^{wid}$. By multiplying both sides by the number of horizontal pixels $number_{pixels}{}^h$ yields Eq. (3).

B. Object Description in 3D Environment

In this paper, we consider an environment situated in 3D space. Within this space, each object $O_k, k = 1..M$ is composed of multiple faces, and each face possesses a directional orientation. We represent this orientation using a facing vector $\vec{s}$. Taking object O_k in Fig. 3 as an example, it comprises six faces, corresponding to six facing vectors, i.e., $O_k = \{\vec{s_1}, \vec{s_2}, \ldots, \vec{s_6}\}$.

To facilitate the calculation of the proportion of each face covered by the camera, we divide each face into multiple grids, with the center of each grid denoted as P_a (also referred to as an *object node* or *target point*), as shown in Fig. 4. Each node $P_a = [x_a, y_a, z_a]$ possesses three-dimensional coordinates and its facing vector (orientation) $\vec{F_a}$. Therefore, nodes on the

same face share the same facing vector, indicating they face in the same direction.



$$o_k = \{\vec{s_1}, \vec{s_2}, \dots, \vec{s_6}\}$$

Fig. 3. Schematic diagram of a 3D object

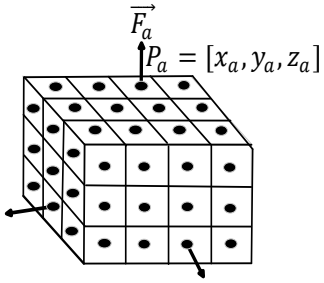


Fig. 4. The 3D object is divided into grid points, each with an associated facing vector.

C. View Angle

The view angle $\emptyset_{i,a}$, also known as the viewing angle difference, is defined as the angle between the orientation of an object's face $\vec{F_a}$ and the direction of the camera C_i, i.e.,

$$\emptyset_{i,a} = cos^{-1}\left(\frac{\overrightarrow{P_aC_i} \cdot \vec{F_a}}{|\overrightarrow{P_aC_i}||\vec{F_a}|}\right). \quad (4)$$

Here, C_i is the camera, P_a is the surveillance target, and $\vec{F_a}$ is its facing vector, which can be illustrated in Fig. 5.

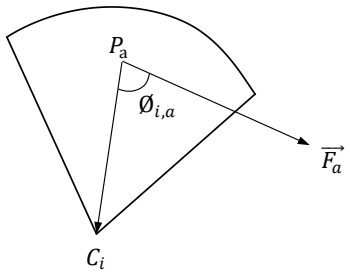


Fig. 5. Schematic Diagram of View Angle $\emptyset_{i,a}$

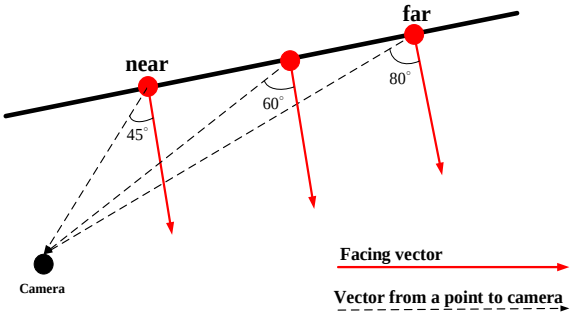


Fig. 6. Relationship between view angle and extension of object faces

When this angle $\emptyset_{i,a}$ is larger, it indicates that the object's face is more inclined in the captured image. Taking Fig. 6 as an example, the angles are 45°, 60°, and 80°, respectively. If the view angle $\emptyset_{i,a}$ in the surveillance image is too inclined (e.g., greater than 60°), the surface information of the object cannot be clearly identified in image recognition [8].

D. Imaging Resolution (Pixel per foot)

Imaging resolution is one of the crucial factors reflecting surveillance quality, measured in PPF (*pixels-per-foot*). PPF signifies the number of pixels corresponding to each unit foot of an object in the camera image, i.e.,

$$\text{Resolution(PPF)} = \frac{number_{pixels}{}^{h}}{HFOV}. \quad (5)$$

Here, $number_{pixels}{}^{h}$ represents the number of horizontal pixels of the camera image, and HFOV denotes the camera's horizontal field of view width (in feet) when the object is at a distance D (in Fig. 2) from the focal point of the camera. The relationship is given by

$$\text{HFOV} = \text{D} \times \frac{chip_{\text{i}}^{wid}}{focal_{\text{i}}^{len}}. \quad (6)$$

Therefore, low resolution will affect the monitoring quality and also degrade imaging performance [8].

E. Surveillance Constraints

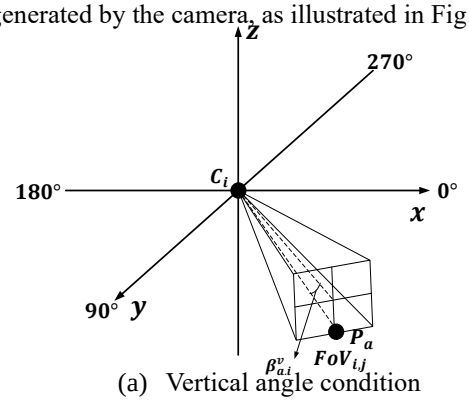
Combining the above, we use $\gamma_{i,j}^{a} \in \{0,1\}$ to indicate whether the node P_a can be covered by the camera's field of view $FoV_{i,j}$. It is defined as

$$\gamma_{i,j}^{a} = \rho_{i,j}^{a} \times \sigma_{i,j}^{a} \times \tau_{i,j}^{a}, \qquad \gamma_{i,j}^{a} \in \{0,1\}. \quad (7)$$

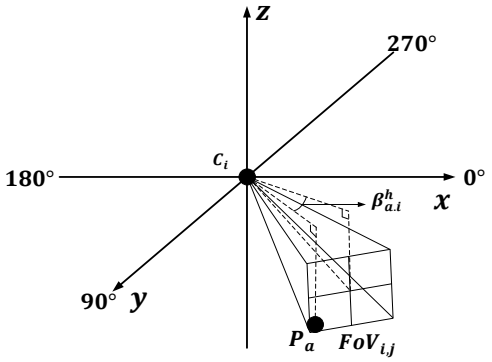
Specifically, $\rho_{i,j}^{a} \in \{0,1\}$ indicates whether node P_a is within the field of view $FoV_{i,j}$. Its value is 1 if satisfied, otherwise 0. Similarly, $\sigma_{i,j}^{a} \in \{0,1\}$ indicates whether node P_a is within the depth distance of $FoV_{i,j}$, with a value of 1 indicating satisfaction and 0 indicating otherwise. Additionally, $\tau_{i,j}^{a} \in \{0,1\}$ indicates whether the angular difference of the camera orientation is less than the required view angle. A value of 1 signifies satisfaction, while 0 indicates otherwise. Consequently, all three monitoring requirements are considered met when $\gamma_{i,j}^{a} = 1$; otherwise, it is deemed unsatisfied with $\gamma_{i,j}^{a} = 0$.

1) Coverage Condition 1: $\rho_{i,j}^{a}$

The first coverage condition $\rho_{i,j}^{a} \in \{0,1\}$ is used to determine whether node P_a is located within the field of view $FoV_{i,j}$ generated by the camera, as illustrated in Fig. 7.



(a) Vertical angle condition



(b) Horizontal angle condition

Fig. 7. Coverage Condition of $\rho^a_{i,j}$

Here, $\beta^h_{a,i}$ represents the horizontal angular difference of node P_a relative to the camera C_i's $FoV_{i,j}$, and $\beta^v_{a,i}$ is the vertical angular difference relative to the camera C_i's $FoV_{i,j}$. Therefore, for node P_a to be within the view angle of $FoV_{i,j}$, it must simultaneously satisfy the horizontal angle condition, i.e.,

$$-\frac{\alpha^h_i}{2} \le \beta^h_{a,i} \le \frac{\alpha^h_i}{2}, \tag{8}$$

and also satisfy the vertical angle condition, i.e.,

$$-\frac{\alpha^v_i}{2} \le \beta^v_{a,i} \le \frac{\alpha^v_i}{2}. \tag{9}$$

Thus, when node P_a simultaneously satisfies Eqs. (8) and (9), it indicates that node P_a meets the coverage condition within $FoV_{i,j}$'s view angle. In other words, $\rho^a_{i,j} = 1$; otherwise, $\rho^a_{i,j} = 0$, i.e.,

$$\rho^a_{i,j} = \begin{cases} 1, \text{if satisfying Eqs. (8) and (9)} \\ 0, \text{otherwise} \end{cases}. \tag{10}$$

2) Coverage Condition 2: $\sigma^a_{i,j}$

Coverage Condition of $\sigma^a_{i,j} \in \{0,1\}$ is used to determine whether node P_a is within the depth distance of $FoV_{i,j}$[1], indicating that the object should be able to achieve clear imaging on the photosensitive element and meet the requirements of imaging resolution. Therefore, the distance between node P_a and camera C_i must be greater than R^{min}_i and less than R^{max}_i, i.e.,

$$R^{min}_i \le \left|\overrightarrow{P_a C_i}\right| \le R^{max}_i. \tag{11}$$

When the distance between node P_a and camera C_i satisfies Eq. (11), it indicates that node P_a is within the depth distance of $FoV_{i,j}$, meaning $\sigma^a_{i,j}$ has a value of 1. Otherwise, $\sigma^a_{i,j}$ has a value of 0, as follows:

$$\sigma^a_{i,j} = \begin{cases} 1, & if\ R^{min}_i \le \left|\overrightarrow{P_a C_i}\right| \le R^{max}_i \\ 0, & otherwise \end{cases}. \tag{12}$$

3) Coverage Condition 3: $\tau^a_{i,j}$

Coverage Condition of $\tau^a_{i,j}$ is used to determine whether the camera orientation angle is less than the required view angle. The requirement of view angle (θ^{req}) ensures that the orientation of object facing $\overrightarrow{F_a}$ and the vector between the camera and the object ($\overrightarrow{P_a C_i}$) are not too large. Therefore, we use θ^{req} to represent the minimum angle requirement. If the view angle $\emptyset_{i,a}$ is less than θ^{req}, it indicates that the surface information of the surveillance target can be clearly observed, i.e.,

$$\emptyset_{i,a} \le \theta^{req}. \tag{13}$$

When Eq. (13) is satisfied, it means that the view angle is less than the required view angle, i.e., $\tau^a_{i,j}$ has a value of 1. Otherwise, $\tau^a_{i,j}$ has a value of 0, as follows:

$$\tau^a_{i,j} = \begin{cases} 1, & if\ \emptyset_{i,a} \le \theta^{req} \\ 0, & otherwise \end{cases}. \tag{14}$$

E. Coverage Ratio of Surveillance Targets

In this paper, we measure the coverage ratio of all surveillance targets being captured by cameras, which is defined as:

$$Coverage\ Ratio = \frac{M_{covered}}{M_{total}}. \tag{15}$$

Here, $M_{total} = \sum_{k=1..M} \sum_{P_a \in O_k} P_a$ represents the total number of nodes P_a contained in all surveillance targets O_k, where $k = 1..M$ and $M_{covered} = \sum_{k=1..M} \sum_{\{P_a | \gamma^a_{i,j}=1, P_a \in O_k\}} P_a$ represents the total number of nodes that truly satisfy all the monitoring conditions.

F. Problem Definition

In this paper, the objective is to explore the dispatch of PTZ cameras in a 3D environment. Assuming there are N PTZ cameras available for selection, each camera $C_i, i = 1..N$, can adjust its field of view (FoV) through its horizontal angle setting (Pan_i), vertical tilt angle setting ($Tilt_i$), and focal length ($focal^{len}_i$), forming the corresponding FoV set: $FoV_i = \{(Pan_i, Tilt_i, focal^{len}_i)\}$. It is known that there are M surveillance targets, O_k, $k = 1..M$. Each object can be viewed as a polyhedron, with each face divided into several grids forming nodes $P_a \in O_k$ with orientations $\overrightarrow{F_a}$. The problem is to determine how to select the best camera subset from the N available cameras (denoted as $X_i \in \{0,1\}$) and determine their appropriate FoVs ($FoV_{i,j} \in FoV_i$) to fully meet the surveillance condition restrictions, including view angle, resolution, and viewing depth distance, expressed as $\gamma^a_{i,j} \in \{0,1\}$ to indicate coverage satisfaction. The objective is to minimize the selected camera set, i.e.,

$$\min_{X_i, Pan_i, Tilt_i, focal^{len}_i} \sum_{i=1..N} X_i, \tag{16}$$

and maximize the coverage ratio on surveillance targets O_k, $k = 1..M$, i.e.,

[1] It is important to note that if physical obstacles are present in the environment, we can verify whether camera C_i has a direct line of sight to object P_a by checking for an intersection between the vector $\overrightarrow{P_a C_i}$ and the surfaces of the physical obstacles. Here, several existing technologies can convert these physical obstacles in images into coordinate descriptions and their surface representations [43], [44], [45].

$$\max_{X_i, Pan_i, Tilt_i, focal_i^{len}} \frac{M_{covered}}{M_{total}}, \quad (17)$$

while satisfying the surveillance conditions of Eqs. (7) ~ (14).

Since this problem is a mixed-integer linear programming problem (MILP), i.e., $X_i \in \{0,1\}$ and $Pan_i \in \boldsymbol{R}, Tilt_i \in \boldsymbol{R}, focal_i^{len} \in \boldsymbol{R}$, it is classified as NP-hard [30]. Therefore, obtaining an optimal solution within a reasonable time becomes challenging when the number of cameras and surveillance targets increases significantly.

IV. The Proposed Schemes

Since this problem is NP-Hard, finding the optimal solution within a feasible timeframe is impractical. To address this challenge, we propose an innovative and efficient heuristic approach that leverages the power of Deep Reinforcement Learning (DRL) to calculate and evaluate the optimal dispatch settings for PTZ cameras. The rationale behind this approach lies in its ability to utilize *self-learning* and *iterative learning* mechanisms intrinsic to reinforcement learning, enabling the system to dynamically adapt and optimize camera dispatch. Specifically, this method draws upon past successful learning experiences and employs an ε-greedy dispatch strategy, allowing for the calculation of a customized reward function that prioritizes both high coverage ratios and low dispatch costs. By incorporating a weighting mechanism, the system effectively reduces the number of surrounding cameras in the dispatch set, focusing on the most impactful ones, which in turn accelerates the adjustment of optimal rotation angles and focal lengths. The innovation of this approach lies in its capacity to achieve significantly higher surveillance coverage with fewer cameras, making it both computationally efficient and practical for large-scale deployments. Fig. 8 illustrates the structural framework of our proposed method. It begins with Q-Learning as the foundational base, presenting the key component designs: *state representation*, *action space construction*, and *reward function design*. It then progressively evolves into DQN, and ultimately transitions into our advanced method, FPE-DDQN, further elaborating on the *Self-Imitation Mechanism* and the integration of *Fast Prioritized Experience (FPE)*. In the following sections, we will expand from the base to the enhanced one.

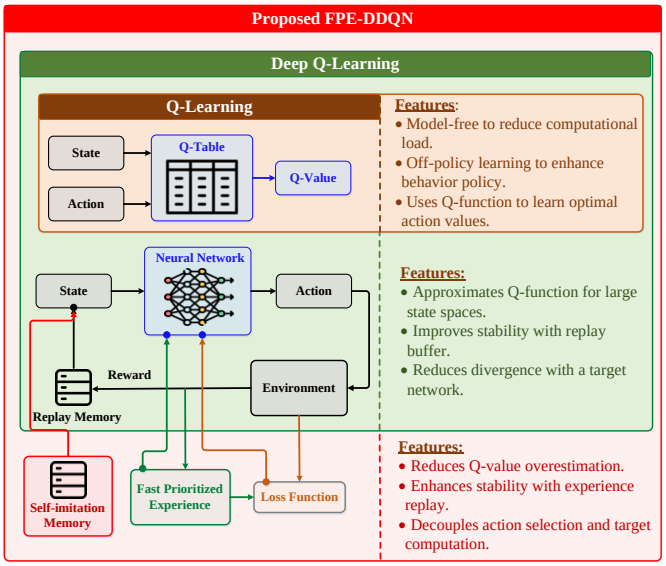


Fig. 8 The structural framework of our proposed method.

A. Reinforcement Learning Approach

Reinforcement learning can be described as a *Markov Decision Process (MDP)*, which comprises three key elements: (1) State Space $\mathbb{S}$, (2) Action Space $\mathbb{A}$, and (3) Reward Function $\mathbb{R}$:

(1) **State Space** $\mathbb{S}$ is used to describe the complete environmental state and we define it as

$$\pi^* = \underset{a_t \in A}{arg\,max}\, Q(s_t, a_t). \quad (18)$$

where s_t represents the environmental state at time t, which depends on the current number of covered targets $M_{covered}$ and the current number of selected cameras N_t. For example, if one camera is selected at time t, then N_t is recorded as $N_{t+1} = N_t + 1$. This information is used to control the selection of cameras.

(2) **Action Space** $\mathbb{A}$ includes different actions taken in different environments s_t. In this problem, it signifies selecting a subset of cameras for dispatch. Before choosing an action, we calculate how many grid nodes that each camera can cover and exclude sets of cameras from the action space that cannot cover any grid nodes to reduce the size of the action space.

(3) **Reward Function** $\mathbb{R}$ is used to reflect rewarding behaviors conducive to task completion. In this method, it is defined as

$$r(s_t, a_t) = \begin{cases} \frac{M_{covered}}{M_{total}} \times (1 - \frac{N_{selected}}{N_{total}}) + \sum_{P_a \in O_k, k=1..M} \frac{1}{N_a}, & if\ N_a \geq 1 \\ 0, & otherwise \end{cases}, \quad (19)$$

where $M_{covered}$ is the number of object nodes that have been covered and M_{total} is the total number of target nodes, thus $M_{covered}/M_{total}$ representing the target coverage ratio, incentivizing the agent to cover as many surveillance targets as possible. $N_{selected}$ denotes the number of selected cameras, thus $(1 - N_{selected}/N_{total})$ penalizes selecting too many cameras to prevent the agent from using too many cameras. In addition, $1/N_a$ represents the chance for an uncovered grid node P_a to be covered by N_a cameras. A smaller N_a indicates that grid node P_a is more difficult to cover, thus increasing the reward. In this way, the reward function can determine the best camera selection to potentially increase the *coverage ratio* (i.e., $M_{covered}/M_{total}$), reduce *dispatch cost* (i.e., $1 - N_{selected}/N_{total}$), and enhance coverage opportunities (i.e., $\sum_{P_a \in O_k, k=1..M} 1/N_a$). This means that if an object is less likely to be covered by cameras or is located at the far end of the field of view, its associated weight increases. As a result, cameras capable of covering this object are given higher priority, improving their chances of being selected.

B. Q-Learning Algorithm

The single agent *Q-Learning (SQ)* algorithm [31] is one of the most well-known reinforcement learning algorithms. It aims to obtain an action selection policy π^* by using the highest reward $Q(s_t, a_t)$ achievable in the current state s_t, as follows:

$$\pi^* = \underset{a_t \in A}{arg\,max}\, Q(s_t, a_t). \quad (20)$$

Here, $Q(s_t, a_t)$ represents the Q-value table established by Q-Learning and is updated using the following equation:

$$Q(s_t, a_t) \leftarrow (1 - \delta) Q(s_t, a_t) + \delta \left[r_t + \gamma \max_{a_t \in A} Q(s_{t+1}, a_t) \right],$$

where δ is the learning rate, and $\gamma \in [0,1]$ is the discount rate for future rewards. Additionally, to balance exploration of unknown environments and exploitation of the current best Q-values, it employs an ε-greedy strategy to select actions. Specifically, in each decision-making phase, the agent selects the action with the highest Q-value with a probability of $(1-\varepsilon)$, and it randomly chooses another action $a_t \in \mathbb{A}$ with a probability of ε.

In general, if there are not many state-action pairs, the single agent Q-Learning algorithm can converge the Q-value table to the optimum. However, for our camera dispatch problem, the state space grows exponentially with the environment's size, limiting the efficiency of tabular methods.

C. *Deep Q-Learning Algorithm (DQN)*

Next, we will introduce *Deep Q-Learning (DQL)* [32], which employs deep neural networks, also known as Deep Q-Learning (DQN), to estimate Q-values.

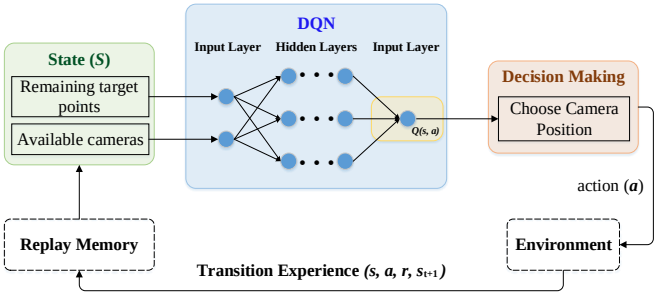


Fig. 9. Concept of DQN

Specifically, the main distinction between DQL and Q-Learning lies in its use of two neural networks: the *evaluation network* and the *target network*. Memory data and random sampling are employed during training to avoid data correlation, as shown in Fig. 9. The evaluation network updates iteratively, while the target network, initialized with the evaluation network's weights, remains fixed for several iterations. This separation stabilizes the learning process by reducing correlation between target and estimated Q-values.

We denote θ and θ^- as the weights of the evaluation and target networks, with outputs $Q(s_t, a_t; \theta)$ and $\hat{Q}(s_{t+1}, a_t; \theta^-)$, respectively. The evaluation network updates θ to minimize the loss function, defined as the mean squared error between target and predicted Q-values:

$$L(\theta) = \left[r_t + \max_{a \in \mathbb{A}} \hat{Q}(s_{t+1}, a_t; \theta^-) - Q(s_t, a_t; \theta)\right]^2.$$

Stochastic gradient descent updates θ as:

$$\theta \leftarrow \theta - \delta \cdot \nabla_\theta L(\theta).$$

In standard DQN, selecting actions based on maximum Q-values $\hat{Q}(s_{t+1}, a_t; \theta^-)$ can cause overestimation, where $\hat{Q}$ exceeds Q in early learning stages, slowing convergence. Note that DQN would select the action with the maximum Q-value as the target output. However, this leads to overestimation, meaning that $\hat{Q}(s, a; \theta^-)$ would be greater than $Q(s, a; \theta)$ in the early learning stages, resulting in slower convergence.

D. *Double DQL Algorithm*

In this paper, we adopt the *Double Deep Q-Network (DDQN)* [33] to address the slow convergence issue of DQN. Specifically, the DDQN mechanism separates the evaluation of action values from action selection, allowing the loss function to be represented as follows:

$$L^{DDQN}(\theta) = \left[r_t + \gamma \hat{Q}(s_{t+1}, a^*; \theta^-) - Q(s_t, a_t; \theta)\right]^2. \quad (21)$$

The selected action a^* can be obtained from

$$a^* = arg \max_{a \in \mathbb{A}} Q(s_{t+1}, a_t; \theta). \quad (22)$$

Finally, the formula for updating the Q-function can be written as

$$Q(s_t, a_t; \theta) \leftarrow (1-\delta) \times Q(s_t, a_t; \theta) + \delta\left[r_t + \gamma \hat{Q}(s_{t+1}, a^*; \theta^-)\right]. \quad (23)$$

E. *Imitation Augmented Deep Reinforcement Learning*

Traditional DQN uses transition experiences from interacting with the environment stored in a buffer, sampled randomly during training to utilize past experiences and break the temporal correlation of these samples to enhance training stability. In the training of the evaluation network, the algorithm randomly selects a mini-batch $B_{m,b}$ of samples from the replay buffer M_m to update the parameters of the two neural networks. In our problem, the positions of nodes and cameras are known, thus the traditional experience replay mechanism would make the algorithm learn the entire state space inefficiently. The significant innovation of our work is based on the elaboration of the *Self-Imitation Augmented* approach [34], which incorporates the concept of self-imitation into deep reinforcement learning. This method utilizes successful experiences to expedite the learning process, referred to as *Fast Prioritized Experience (FPE)*. Specifically, we enhance DDQN framework by integrating the self-imitation mechanism, which prioritizes storing successful experiences and those with high coverage generated during exploration in a specialized buffer. This prioritized buffer is crucial for training the DDQN algorithm efficiently. Moreover, we further streamline the learning process by removing actions from the action space that cannot cover any nodes. Fig. 10 illustrates the framework of our proposed Fast Prioritized Experience-based Deep Reinforcement Learning approach.

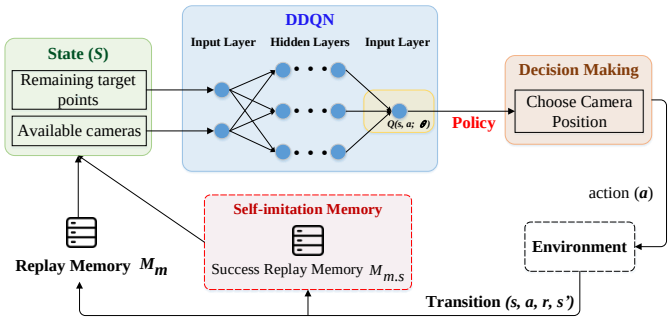


Fig. 10. Fast prioritized experience (FPE) based DDQN

F. *Learning Algorithm*

The details of the algorithm are described as follows. The inputs are the number and position of cameras and the objects to be covered, as well as resolution and visibility angle constraints. Firstly, initialize the prediction network and target network with

random weights θ and θ^- (**Algorithm 1**: lines 1-2). The training loop executes N_{ep} episodes. When the environment generates the current state, the algorithm selects an action $a_t \in \mathbb{A}$ based on the ϵ-greedy policy and the estimated Q-values output by the prediction network (**Algorithm 1**: line 8). Transition experiences (s_t, a_t, r_t, s_{t+1}) are temporarily stored in the buffer D. If the algorithm achieves the predetermined coverage ratio for cameras and objects in an episode, these transition experiences are stored in the replay buffer M_m and the buffer for successful experiences M_s; otherwise, only transition experiences are stored in M_m. In the training of the prediction network, the algorithm randomly selects a mini-batch $B_{m,b}$ of samples from the buffers M_m and M_s, and updates the weights θ using Eqs. (21) ~ (23), with the parameters of the target network copied from the prediction network every T_u steps. For example, at the beginning of the algorithm, the environment is reset, setting the positions of nodes and cameras and assigning them three-dimensional positions. Nodes also require an orientation vector. Then, a random number is generated. If this random number is less than or equal to the exploration rate ϵ, cameras are randomly selected and assigned random directions. If the random number is greater than ϵ, the algorithm calculates all Q-values generated by the current action set using the neural network and selects the action with the maximum Q-value. After executing the action, the next state, reward, coverage ratio of this action, cumulative coverage ratio, and number of selected cameras are returned. The current state, next state, reward, and selected action are then stored in the buffer D. If the total coverage ratio at the end of the episode reaches the predetermined value, the transition experiences $(s_t, a_t, r_t, s_{t+1}), t = 0, \dots, N_s$ of this episode are stored in M_m and $M_{m,s}$; otherwise, they are only stored in M_m, where N_s is the last step time point of the episode. Next, a mini-batch of size $B_{m,b}$ is randomly selected from M_m and $M_{m,s}$ for training, to update the weights of the prediction network and target network, until all episodes are completed.

Algorithm 1. **Enhanced Camera Dispatch Algorithm**

Input: Set of cameras (N), set of objects to be monitored (M), resolution requirement ($Resolution^{req}$), and visibility angle requirement (θ^{req})

Output: Coverage ratio, selected set of cameras, computation time

1: Initialize the prediction network $Q(s_t, a_t; \theta)$ with random weights θ.

2: Initialize the target network $\hat{Q}(s_{t+1}, a_t; \theta^-)$ with weights $\theta^- = \theta$.

3: Clear buffer D, replay memory M_m, and self-imitation memory $M_{m,s}$

4: Initialize state $s_0 = (M_{covered}, N_t)$, where M_{covered} represents the number of covered objects, and N_t is the number of selected cameras.

5: **for** episode $e = 1$ *to* N_{ep} **do**

6: Set selectable cameras S and node positions.

7: **for** each step t in the episode **do**

8: Select an action $a_t \in \mathbb{A}$ based on the ϵ-greedy policy from the current state

$$a_t = \begin{cases} \arg\max_a Q(s_t, a_t; \theta), & \text{with probability } 1-\epsilon, \\ \text{random action}, & \text{with probability } \epsilon. \end{cases}$$

9: Apply action a_t to update coverage, and then remove covered nodes

$M_{\text{covered}} \leftarrow M_{\text{covered}} \cup \{\text{new covered nodes}\}$.

10: Calculate the current coverage ratio $M_{\text{covered}}/M_{total}$ and number of selected cameras N_t, updating the next state $s_{t+1} = (M_{\text{covered}}, N_{t+1})$, where N_{t+1} is the updated number of cameras dispatched.

11: //Reward Calculation

Compute the reward value by

$$r(s_t, a_t) = \frac{M_{covered}}{M_{total}} \times \left(1 - \frac{N_{selected}}{N_{total}}\right) + \sum_{P_a \in O_k, k=1..M} \frac{1}{N_a},$$

where $1/N_a$ represents the chance for an uncovered grid node P_a to be covered by N_a cameras.

12: //Store Experience

Obtain the next state s_{t+1} and Store (s_t, a_t, r_t, s_{t+1}) in buffer D

13: **if** coverage ratio $M_{\text{covered}}/M_{total}$ reaches the predetermined value **then**

14: Store $(s_t, a_t, r_t, s_{t+1})_{t=0,\dots,N_s}$ in M_m and $M_{m,s}$

15: **else if** coverage ratio does not reach the predetermined value **then**

16: Store $s = (s_t, a_t, r_t, s_{t+1})$ in M_m

17: **end if**

18: //Mini-batch Update

Sample a mini-batch of size $B_{m,b}$ from M_m and $M_{m,s}$

19: Compute the target y_t for each experience (s_t, a_t, r_t, s_{t+1}) by

$y_t = r_t + \gamma \cdot \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_t; \theta^-)$,

where γ is the discount factor.

20: Update the prediction network $Q(s_t, a_t; \theta)$ by minimizing the loss:

$$L(\theta) = \frac{1}{|B_{m,b}|} \sum_{i=1}^{B_{m,b}} \left(y_t^{(i)} - Q(s_t^{(i)}, a_t^{(i)}; \theta)\right)^2.$$

21: //Update Target Network:

Every T_u steps, synchronize the target network:

$$\theta^- \leftarrow \theta.$$

22: Update the state $s_t \leftarrow s_{t+1}$

23: **end for**

24: **end for**

G. *Complexity Analysis of Algorithms*

Table II compares the algorithm based on Fast prioritized experience DDQN (FPE-DDQN) with other deep reinforcement learning algorithms: DQN and DDQN. The metrics of interest are the time complexity of iterations and the number of learning episodes, which refers to the rounds required to achieve a higher coverage ratio after exploring the environment.

Table II. Comparison of Time Complexity of Different Reinforcement Learning Algorithms

Algorithm	Time Complexity	Learning Episodes
SQ	$O(\lvert\mathbb{A}\rvert)$	2313
DQN	$O(B_{m,b} \cdot \alpha\beta)$	277
DDQN	$O(B_{m,b}(1+\alpha\beta))$	218
FPE-DDQN (ours)	$O(B_{m,b}(1+\alpha\beta))$	213

Here, α and β respectively represent the number of hidden layers and neurons per layer in the prediction network. The traditional DQN algorithm has a time complexity of $O(B_{m,b} \cdot \alpha\beta)$ during the training of the prediction network. In comparison to DQN, DDQN requires an additional step to select the optimal action a^* from the mini-batch $B_{m,b}$, which does not involve any training, resulting in a time complexity of $O(B_{m,b})$. Therefore, the total time complexity of DDQN is $O(B_{m,b}(1+\alpha\beta))$. Our proposed algorithm shares the same mini-batch size and parameter updating process as DDQN, thus also having a time complexity of $O(B_{m,b}(1+\alpha\beta))$. Compared to other reinforcement learning algorithms, our proposed algorithm demonstrates better performance in terms of the number of learning episodes.

V. Performance Evaluation

In this section, we will compare different methods to evaluate the performance of the proposed approach. We first introduce the experimental environment and parameter settings. Then, we explore the impact of number of available cameras on performance. Finally, we examine the influence of covered target numbers on performance.

The simulated space for this experiment has dimensions of $1000 \times 1000 \times 10$ cubic meters, with monitored targets sized at $2 \times 2 \times 2$ cubic meters, ranging from 100 to 500 in number, randomly generated within the simulated environment. Additionally, the parameters for PTZ cameras are based on the Compro IP570 [1], with a horizontal rotation range of -170° to 170°, a vertical range of -20° to 90°, and a focal length adjustable between 3.8 to 45.6 mm. Regarding monitoring conditions, the default visibility angle requirement is set at 60° and the resolution requirement is 40 PPF [8]. Detailed experimental parameters are shown in Table III.

Table III. Experimental Environment Parameters [1],[8]

Environment parameters	Value	PTZ Camera parameters	Value
Environment size (in m)	1000 × 1000 × 10	Horizontal angle (Pan_i)	-170°~ 170°
The total number of objects (M)	100 ~ 500	Vertical tilt angle($Tilt_i$)	-20°~ 90°
Object size (in m)	2 × 2 × 2	Focal length ($focal_i^{len}$)	3.8 ~ 45.6 mm
Total number of PTZ cameras (N)	100 ~ 500	**Surveillance Constraints**	**Value**
Object Deployment	random	Viewing angle (θ^{req})	60°
Camera Deployment Distance	5 ~ 10 m	$Resolution^{req}$	40 PPF

In the following experiments, we compared the Centralized Force-Directed Algorithm (**CFA**) [38], Centralized Greedy Algorithm (**CGA**) [39], Iterative Screening Algorithm (**ISA**) [22], and Integer-Linear Programming (**ILP**). Specifically, **CGA** uses G-EFA at the start of each iteration to find the view with the maximum coverage set, then selects the view covering the most nodes, with the corresponding camera not selected in the next iteration. **CFA** utilizes G-EFA at the beginning of each iteration to find all cameras covering the most nodes and calculates the *force* for each view in the maximum coverage set and prioritizes selecting the view with the highest Force. **ISA** considers the view range of each camera and selects cameras based on a greedy approach, finally determining the direction for each camera. **ILP** obtains the optimal solution using integer programming functions.

Table IV lists the parameters used in our method, achieving a balance between performance and model complexity. Specifically, this scheme relies on a carefully designed neural network architecture and a set of optimized hyperparameters. The architecture consists of two hidden layers, with the first containing 64 neurons and the second containing 32 neurons, both activated using the ReLU function. The optimizer employed is Adam, known for its adaptive learning rate benefits. A discount factor of 0.9 is chosen to balance short-term and long-term rewards, while the learning rate is set at 0.01. The exploration rate is kept low at 0.1 to encourage the exploitation of learned strategies after the initial exploration phase. The replay buffer, which stores past experiences for training, is configured to hold 2,000 transitions, with prioritized experience replay enabled to allow the model to focus on significant transitions.

Table IV. Relevant Parameters of Our Method [47], [48]

Parameter	Value
Number of Episodes (N_{ep})	500
Batch Size ($B_{m,b}$)	32
Target Network Update Frequency (T_u)	100
Discount Factor (γ)	0.9
Exploration Rate (ε)	0.1
Learning Rate (δ)	0.01
Hidden Layers and Neurons	2, (64, 32)
Replay Buffer Size (M_m)	2000
Prioritized Experience Replay Buffer Size ($M_{m,s}$)	2,000
Optimizer	Adam
Activation Function	ReLU

A. Impact of Available Cameras

1) Coverage Ratio

First, we compare the influence of different numbers of available cameras on coverage ratios, with the number of monitored objects set to $M = 300$. As shown in Fig. 11, the coverage ratios of all schemes increase with the number of available cameras. **ISA** exhibits the lowest coverage ratio because its camera selection strategy, based solely on the number of nodes, and the strategy for selecting directions, tends to ignore better combinations. **CGA** and **CFA** have relatively poor coverage ratios because **CGA** selects cameras solely based on the number of covered nodes, resulting in a lower coverage ratio due to its overly greedy mechanism (selecting only the maximum value regions), while **CFA** improves on this by selecting views with maximum force to mitigate the adverse effects of the greedy algorithm on coverage ratios. For the scheme **SQ**, it utilizes a tabular approach to compute Q-values and lacking experience replay mechanisms, exhibits limited performance. Reinforcement learning algorithms learn from mistakes to balance node coverage and covering opportunities, resulting in higher coverage ratios. **DDQN** addresses overestimation issues present in **DQN**, and **FPE-DDQN** further improves upon **DDQN** by incorporating self-imitation

mechanisms, thus achieving performance close to the optimal solution, especially when the number of dispatched cameras exceeds 300. It is worth noting that our method achieves performance extremely close to the optimal solution (**ILP**), with a coverage ratio difference of less than 0.71%.

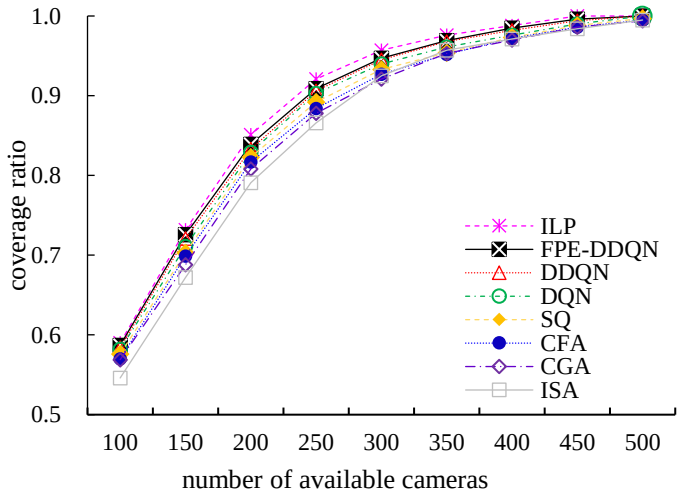


Fig. 11. Comparisons of coverage ratios

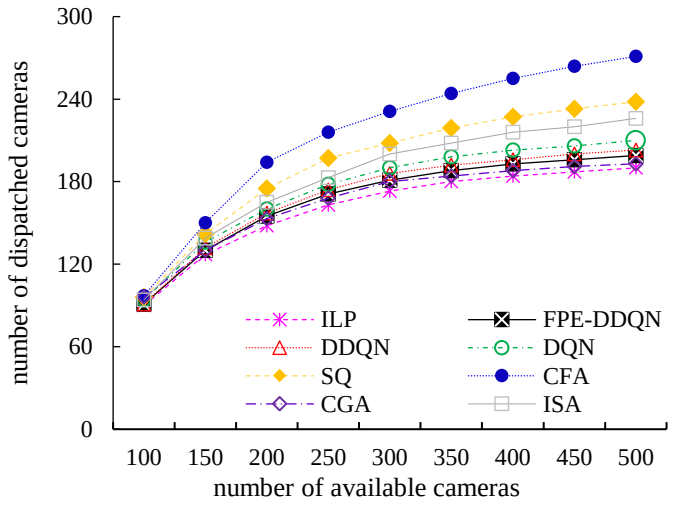


Fig. 12. Comparison of the number of required cameras

2) *Number of Required Cameras*

Next, we compare the results concerning the number of required cameras, with the number of monitored objects set to $M = 300$. As depicted in Fig. 12, the number of required cameras for all methods increases with the number of available cameras. Among them, **CFA** performs the worst, as its strategy of selecting the maximum force value results in the redundancy of cameras due to initially selected cameras covering nodes that could be subsequently covered by cameras selected later. **SQ** performs slightly better due to limitations in tabular construction and the absence of an experience replay mechanism, resulting in poorer performance compared to other deep reinforcement learning methods. Following is **ISA**, which aims to reduce the number of dispatched cameras but sacrifices some coverage ratio. For the Deep reinforcement learning methods, they can find strategies superior to greedy methods through interaction with the environment. In terms of the performance of the number of dispatched cameras, **FPE-DDQN** performs the best, followed by **DDQN** and **DQN**. Since **CGA** minimizes the number of dispatched cameras by selecting the maximum node coverage, it sacrifices the coverage ratio in favor of fewer camera dispatches. Here, it is worth noting that our method achieves performance close to the optimal **ILP** solution, with a gap of less than 4.7% in terms of the number of cameras required.

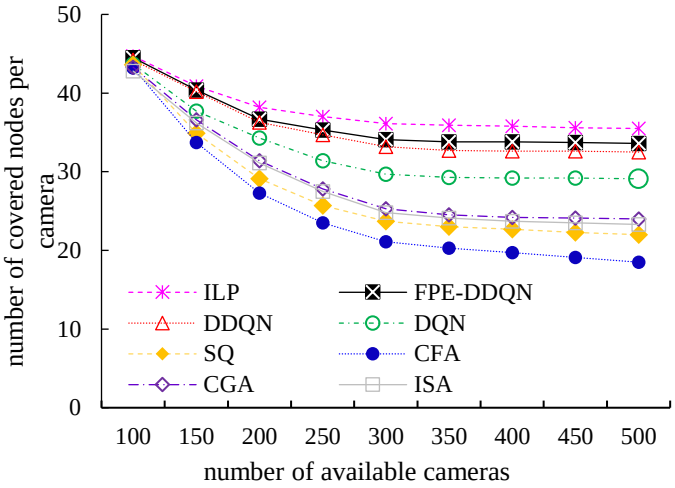


Fig. 13. Comparison of the average number of object nodes covered per camera

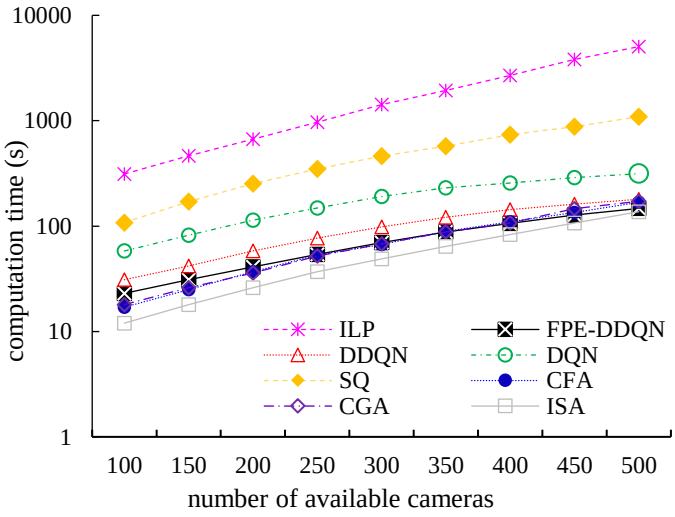


Fig. 14. Comparison of Computation Time

3) *Number of objects covered per Camera*

Furthermore, we compare the performance concerning the average number of object nodes covered per camera, with the number of monitored objects also set to $M = 300$. As illustrated in Fig. 13, all schemes exhibit a steady increase in the average number of object nodes covered per camera with an increasing number of available cameras, indicating that the number of required cameras is sufficient. Strategies employed by **CGA** and **CFA** lead to poorer coverage ratios and a higher number of dispatched cameras, resulting in inferior performance in this metric. Although **ISA** performs well in terms of the number of cameras used, it covers fewer nodes on average, resulting in only moderate coverage per camera. For the deep reinforcement learning methods, they exhibit better performance. Note that **FPE-DDQN** achieves the closest average number of nodes covered per camera to the optimal **ILP** solution, with a gap of less than 0.52% in terms of the number of objects covered per camera.

4) *Computation Time*

Finally, we compare the computation time of different methods, with the number of monitored objects again set to $M = 300$. The hardware specifications for this experiment platform include an Intel(R) Core(TM) i7-12700, 32 GB RAM, and NVIDIA GeForce RTX3070 Graphics Card, operated under the PyTorch 1.10.1 software environment. As shown in Fig. 14, all schemes experience an increase in computation time with an increasing number of available cameras, attributed to the larger computation set resulting from a greater number of cameras. Reinforcement learning methods exhibit longer computation times due to the initial training period requiring exploration of the environment. Our **FPE-DDQN** improves upon **DDQN**'s experience replay mechanism to avoid excessive exploration in areas covering only a few objects, thereby reducing computation time. Both **CFA** and **CGA** require searching for all views of cameras and iterative selection of cameras and views. Their complete algorithmic time complexity is $O(\sum_{i\in N}|E_i| \times |N||M|^3)$, indicating that time is primarily spent searching for camera views, which can even exceed the time spent by **FPE-DDQN** as the number of nodes or dispatched cameras increases. **ISA**, a greedy-based iterative search algorithm, exhibits lower computation times than **FPE-DDQN** when the number of dispatched cameras is fewer than 400. However, as the number of dispatched cameras increases, the computation time rises rapidly. Note that **ILP** requires considering all combinations of camera selections, which results in the longest computation time. In contract, our method achieves similar performance while requiring only 0.01% to 0.08% of the computation time, making it much more practical for real-world applications.

B. *Impact of Number of Monitored Objects*

1) *Coverage ratio*

Fig. 15 illustrates the coverage ratio of different schemes, with the number of available cameras $N = 300$ and the monitored objects $M = 100 \sim 500$. As depicted, when the number of objects increases, it becomes increasingly challenging to cover all objects, leading to a declining trend in all curves. **CGA** and **CFA** exhibit the poorest coverage ratios due to their greedy mechanisms that only select views covering the maximum number of nodes. As the number of covered objects increases, they are more prone to obtaining local optima, resulting in lower coverage ratios. **CFA**, compared to **CGA**, utilizes a strategy of selecting views with the maximum force, which helps alleviate the impact of the greedy method on coverage ratios. **ISA**, designed to reduce the number of dispatched cameras, experiences a significant decrease in coverage ratios as the number of monitored objects increases. **SQ** lacks an experience replay mechanism results in inferior stability compared to other deep reinforcement learning methods. For the deep reinforcement learning algorithms, they learn from mistakes to strike a balance between the number of covered nodes and coverage opportunities, thus achieving higher coverage ratios. Among them, **DDQN** addresses the overestimation issue of **DQN**, while **FPE-DDQN** further enhances **DDQN** by incorporating a self-imitation mechanism, resulting in performance second only to the optimal solution. Here, we also note that our method achieves performance extremely close to the optimal solution (**ILP**), with a coverage ratio difference of less than 0.01%.

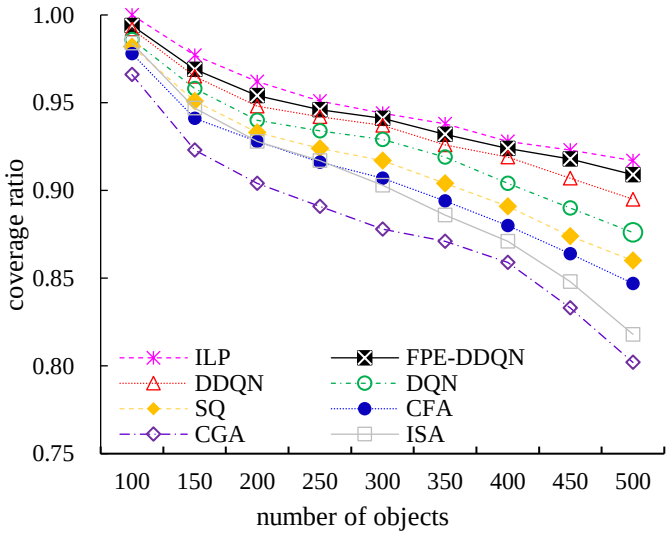


Fig. 15. Comparison of Coverage ratios

2) *Number of Required Cameras*

Fig. 16 shows the number of cameras required by different schemes. All schemes utilize more cameras to cover the increasing number of objects, resulting in an upward trend in the curves until all cameras are used. Among them, **CFA** performs the worst due to its strategy of selecting views with the maximum force value, leading to redundant camera dispatching by repeatedly covering already covered nodes. **SQ**'s inferior stability compared to other deep reinforcement learning methods is attributed to the absence of an experience replay mechanism. The inferior stability of **SQ** is attributed to the absence of an experience replay mechanism. For the deep reinforcement learning methods, they outperform others. Note that **FPE-DDQN** achieves the closest number of nodes covered per camera to the optimal solution.

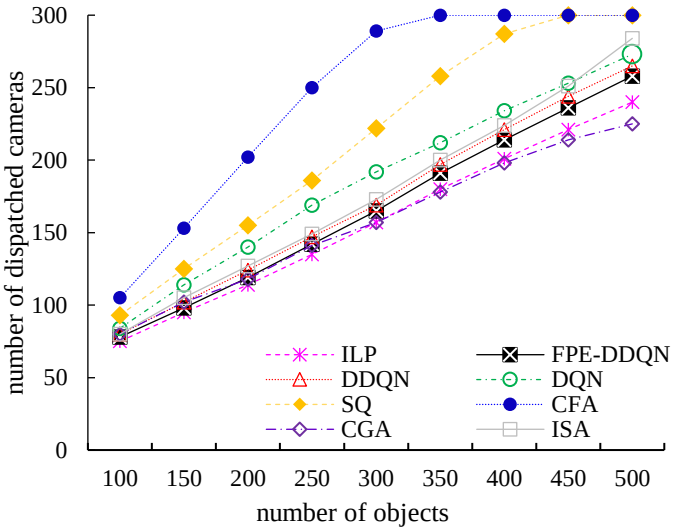


Fig. 16. Comparison of the number of required cameras

3) *Average Number of Nodes Covered per Camera*

Next, we compare the performance concerning the average number of nodes covered per camera, where the number of available cameras is $N = 300$. As shown in Fig. 17, all schemes exhibit a stable increase with an increasing number of dispatched cameras. **CGA** performs the worst in coverage ratio, while **CFA** requires more cameras, resulting in poorer

performance for both methods. **ISA** strikes a better balance between coverage ratio and selected cameras, resulting in a higher average number of nodes covered per camera compared to **CGA** and **CFA**. Deep reinforcement learning methods exhibit better performance in both coverage ratio and the number of required cameras. Among them, **FPE-DDQN** achieves the closest performance to the optimal **ILP** solution, with a gap of less than 0.88% in terms of the number of objects covered per camera.

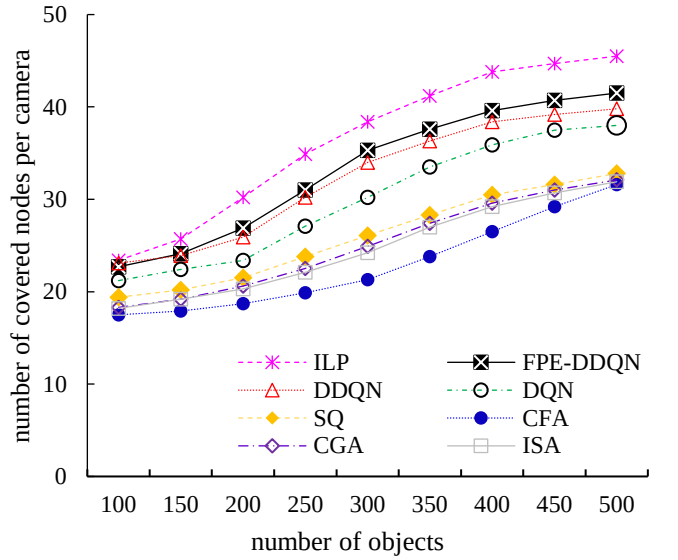


Fig. 17. Comparison of the average number of nodes covered per camera

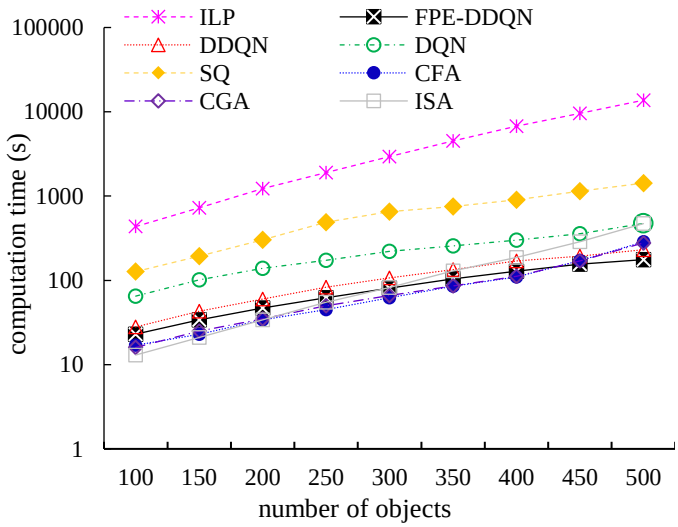


Fig. 18. Comparison of computation time

4) *Computation Time*

Fig. 18 depicts the computation time for each scheme, which increases as the number of monitored objects rises. Reinforcement learning methods generally have longer computation times due to the initial exploration of the environment during training. However, our **FPE-DDQN** improves upon **DDQN**'s experience replay mechanism to reduce computation time by avoiding excessive exploration and trimming action space. The complete algorithmic time complexity of **CFA** and **CGA** primarily lies in searching for camera views, even exceeding **FPE-DDQN**'s time as the number of nodes or cameras increases. Since **ILP** considers all combinations of object selections, it requires the longest computation time. In contrast, our method not only achieves the optimal solution of the **ILP** but also requires only 0.01% to 0.08% of the computation time, making it more practical for real-world applications.

C. *Observations of Relevant Parameters*

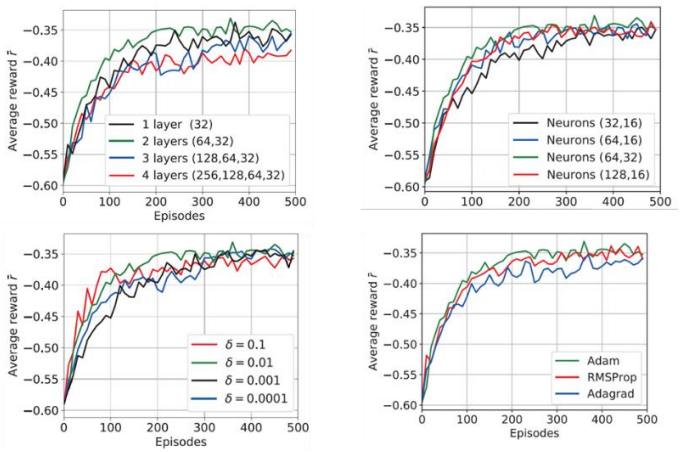


Fig. 19. Observations of relevant parameters

Fig. 19 illustrates the impact of different learning rates on **FPE-DDQN**, showing that $\delta = 0.01$ achieves the highest average reward and relatively faster convergence. Hence, we set the learning rate δ to 0.01. Different optimizers' effects on **FPE-DDQN** reveal that using Adam achieves the fastest convergence, thus chosen as our method's optimizer. The influence of different hidden layer numbers on learning performance indicates that increasing the number of hidden layers enhances learning performance but reduces convergence speed. Additionally, learning performance gradually increases with an increasing number of neurons in each layer until overfitting occurs with too many neurons, resulting in decreased performance. We find that having 2 hidden layers and (64, 32) neurons respectively yields optimal learning performance, thus setting these parameters as hyper-parameters for our **FPE-DDQN** architecture.

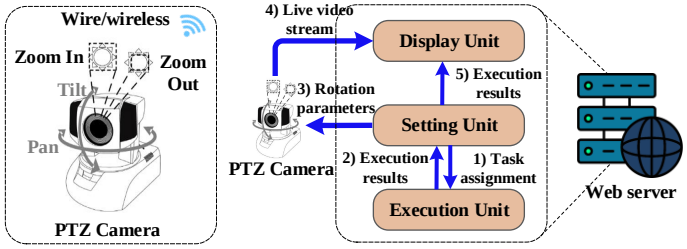
C. *Discussion of Limitations*

The proposed PTZ camera dispatch scheme is based on deep reinforcement learning, which offers significant improvements in coverage efficiency and cost reduction. In the following, we will discuss some limitations and potential failure cases to consider. First, the scalability may be a concern when extending the solution to larger-scale surveillance environments. However, this issue could be mitigated by increasing the spacing between object points or utilizing object edge endpoints as reference points to reduce computational demands. Second, the environmental lighting conditions may affect the visibility of monitored targets. Fortunately, most modern cameras support night vision functionality [40]-[42], greatly enhancing performance in nighttime surveillance scenarios. Third, some irregular shapes of monitored targets may impose considerable challenges for defining and describing these targets. Fortunately, several existing technologies can convert physical obstacles in images into coordinate descriptions [43]-[45]. Note that using simple geometric shapes to describe monitored targets, such as triangular pyramids, square pyramids, pentagonal pyramids, and cylinders, that can also provide a convenient method for users and enables them to easily obtain information about physical

obstacles for the system.

VI. System Prototype Implementation

We have implemented the system prototype and the architecture of the camera dispatch system is illustrated in Fig. 20, which consists of multiple PTZ cameras and a web server. The PTZ cameras are used to capture images, with the images being streamed as standard video feeds for preview purposes. The positions of the PTZ cameras are fixed, and they can be remotely controlled via wired/wireless communication through the web server, enabling horizontal or vertical rotation and zooming of the images, as shown in Fig. 20(a). We develop a website on the web server to provide real-time video surveillance services. Users can define multiple urgent monitoring tasks on the webpage and then observe the results of image preview events. This webpage contains three main units: 1) *setting*, 2) *execution*, and 3) *display*. The setting unit primarily allows users to define detailed monitoring requirements, including the monitoring areas' size, number, required PPF (pixels per foot), and view angle. The execution unit is used to perform the camera dispatching algorithm, which calculates a set of cameras that can achieve higher coverage rates while meeting the monitoring requirements of the target objects. The display unit shows the results of the algorithm, including listing the coverage ratio, computation time, parameters configured for PTZ cameras, and previewing real-time monitoring videos. The system operation process is depicted in Fig. 20(b).



(a) (b)

Fig. 20. Illustration of (a) PTZ Cameras and (b) system workflow.

A. System Design & Hardware Components

The prototype of this system has been developed as part of this work. Two algorithms are utilized in this system, including the inter-linear programming (ILP) strategy and the method designed in this study. Users can define multiple urgent monitoring tasks and select either the ILP scheme or the proposed scheme to dispatch a group of cameras that satisfy the monitoring requirements. The hardware components in the system includes four network PTZ cameras and a web server. The web server operates on a Windows 10 64-bit operating system and utilizes MATLAB R2018, Python, VLC Plug-in version 2.1.3, HTML 5, and jQuery version 1.11.0. The hardware configuration includes a DELL Optiplex 990 with an Intel Core i7-12700 CPU, an NVIDIA GeForce RTX 3070 graphics card, and 32 GB of RAM. The system features four Compro IP570 PTZ cameras, each with a maximum resolution of 1280 × 1024. These cameras possess impressive rotation capabilities, allowing for a pan range of 340 degrees and a tilt range of 100 degrees. In addition, they are equipped with a 12x optical zoom for enhanced image capture.

B. System Workflow

We developed the video surveillance system using HTML, JavaScript, CSS, and AJAX, as shown in Fig. 21. The workflow of the web service operation is described as follows. First, users need to define the monitoring area's size, PPF, view angle, and the algorithm used in the advanced monitoring settings panel. This panel provides multiple monitoring parameters for configuration, including 1) the number of monitoring events, 2) the size of the monitoring objects, 3) PPF resolution requirements, 4) view angle restrictions, and 5) the choice of algorithms. Next, users click on the monitoring area position on the monitoring environment floorplan displayed on the webpage. The monitoring environment floorplan allows users to specify monitoring positions and shapes directly on this floorplan. Once the monitoring parameters and positions are determined, events are triggered. We use jQuery to capture the positions clicked by users, use JavaScript to send the event positions and parameters for camera dispatching computation. The computed results, including the dispatched cameras and rotation parameters, are sent back and displayed on the monitoring environment floorplan using jQuery combined with CSS syntax. Additionally, the rotation parameters are sent to the cameras via AJAX, and the cameras will adjust their direction towards the event area. After completing the above process, event monitoring can begin.

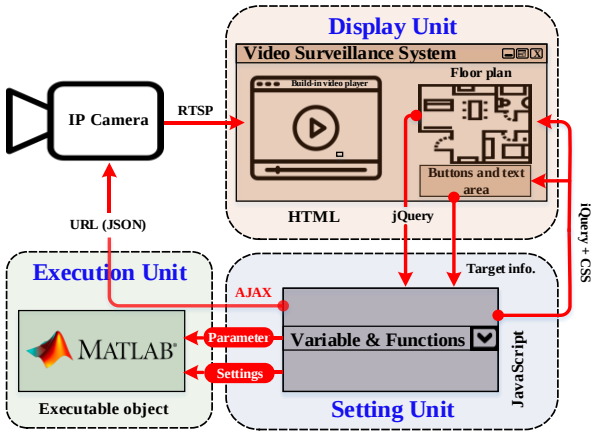


Fig. 21. Workflow of the web service

C. Real-World Application

Finally, we introduced a real-world application for our system. This scenario simulates two sudden events (e.g., fires) in a real-world environment. Considering there are sudden events occur in the environment and their corresponding coordinates are identified, the PTZ cameras will be dynamically dispatched to monitor and meet the resolution and view angle requirements. The real-time results can be displayed in the camera view and the camera dispatching results will be immediately visualized on the display unit. This functionality can enhance situational awareness, facilitates emergency response coordination, and supports future rescue operations.

VII. Conclusion

In this paper, we have addressed the surveillance camera dispatch issues in a 3D environment considering comprehensive monitoring conditions including viewing angles and surveillance resolution, using novel PTZ cameras. Our objective is to maximize coverage ratios with fewer cameras. We modeled this problem as a mixed-integer linear programming problem

and proposed a deep reinforcement learning-based approach to solve it effectively. Specifically, by incorporating a self-imitation mechanism into the conventional deep reinforcement learning framework, we accelerated the learning process by leveraging successful transitional experiences. Moreover, by customizing a high-coverage, low-cost reward function, we potentially reduced the required set of cameras around monitored objects, speeding up adjustments to optimal camera rotation angles and focal lengths with fewer cameras. Simulation results validate our scheme's superiority over existing methods in coverage ratios and the number of required cameras. Additionally, our scheme demonstrates better performance in computation time as the number of objects and cameras increases. Compared to traditional reinforcement learning methods, our scheme significantly improves the coverage ratio, decreases the number of required cameras, increases the number of objects covered per camera, and reduces time complexity. This substantial improvement underscores the effectiveness and efficiency of the self-imitation augmented approach.

ACKNOWLEDGEMENTS

This research is sponsored by NSTC 113-2221-E-024-011. Thanks to the students for their participation in the experiment and system development, and special thanks to Prof. Tseng and Prof. Du for their technical consultations and valuable insights.

REFERENCES

[1] Network PTZ Camera, Accessed: April 6, 2024. [Online]. Available: http://www.knshk.com/index.php?route=product/product&product_id=2284

[2] Gun-type PTZ Camera, Accessed: April 6, 2024. [Online]. Available: https://www.king-net.com/123574/index.php?action=product_detail&prod_no=P0030100006664

[3] Dome PTZ Camera, Accessed: April 6, 2024. [Online]. Available: http://www.ai-sc.com.tw/products/info.php?id=9809&title_id=2076

[4] C. Muñoz, J. Huircan, F. Huenupan, P. Cachaña, "PTZ Camera Tuning for Real Time Monitoring of Cows in Grazing Fields," *IEEE Latin American Symposium on Circuits & Systems (LASCAS)*, pp. 1-4, 2020.

[5] P. Kumari, N. Nandyala, A. K. S. Teja, N. Goel and M. Saini, "Dynamic Scheduling of an Autonomous PTZ Camera for Effective Surveillance," *IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 437-445, 2020.

[6] H. U. Unlu, P. S. Niehaus, D. Chirita, N. Evangeliou and A. Tzes, "Deep Learning-Based Visual Tracking of UAVs Using a PTZ Camera System," *Annual Conference of the IEEE Industrial Electronics Society,* pp. 638-644, 2019.

[7] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, "A review of video surveillance systems," *Journal of Visual Communication and Image Representation*, vol. 77, pp. 103116, 2021.

[8] J. Wang et al., "Deep High-Resolution Representation Learning for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349-3364, 2021.

[9] X. Wang, H. Zhang, S. Fan H. Gu, "Coverage Control of Sensor Networks in IoT Based on RPSO," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3521-3532, 2018.

[10] A. Prasanth, S. Jayachitra, "A Novel Multi-Objective Optimization Strategy for Enhancing Quality of Service in IoT-enabled WSN Applications," *Peer-to-Peer Network Application,* vol. 13, pp. 1905-1920, 2020.

[11] N. T. Hanh, H. T. T. Binh, N. X. Hoai, M. S. Palaniswami, "An Efficient Genetic Algorithm for Maximizing Area Coverage in Wireless Sensor Networks", *Information Sciences*, vol. 488, pp. 58-75, 2019.

[12] R. Funada et al., "Distributed Coverage Hole Prevention for Visual Environmental Monitoring With Quadcopters Via Nonsmooth Control Barrier Functions," in *IEEE Transactions on Robotics*, vol. 40, pp. 1546-1565, 2024.

[13] S. Peng and Y. Xiong, "A New Angle Coverage Scheduling Optimization Method for Heterogeneous Nodes in Directional Sensor Networks," *Annual Conference of the IEEE Industrial Electronics Society*, pp. 4549-4554, 2020.

[14] C. Li, X. Chen and L. Chai, "Coverage Optimization of Multi-Camera for Deformable Contour Shapes," *China Automation Congress (CAC)*, pp. 4655-4660, 2021.

[15] N. Mottaki, H. Motameni, H. Mohamadi, "A Genetic Algorithm-Based Approach for Solving The Target Q-Coverage Problem in Over and Under Provisioned Directional Sensor Networks," *Physical Communication*, vol. 54, pp. 1-15, 2022.

[16] W. Li, X. Wang and S. Han, "Coverage Enhance in Boundary Deployed Camera Sensor Networks for Airport Surface Surveillance," *IEEE Access*, vol. 9, pp. 145728-145738, 2021.

[17] M. S. S. Suresh, A. Narayanan and V. Menon, "Maximizing Camera Coverage in Multicamera Surveillance Networks," *IEEE Sensors Journal*, vol. 20, no. 17, pp. 10170-10178, 2020.

[18] G. Chen, Y. Xiong, J. She, M. Wu and K. Galkowski, "Optimization of the Directional Sensor Networks With Rotatable Sensors for Target-Barrier Coverage," *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8276-8288, 2021.

[19] X. Zhu, M. C. Zhou and A. Abusorrah, "Optimizing Node Deployment in Rechargeable Camera Sensor Networks for Full-view Coverage," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11396-11407, 2021.

[20] J. Jia, C. Dong, Y. Hong, L. Guo, Y. Yu, "Maximizing Full-View Target Coverage in Camera Sensor Networks," *Ad Hoc Networks*theft. 94, pp. 1-10, 2019.

[21] J. Chen, H. Liu, Q. Zhang and S. He, "Orientation Optimization for Full-View Coverage Using Rotatable Camera Sensors," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10508-10518, 2019.

[22] K. Shi et al., "Towards Optimal Deployment for Full-View Point Coverage in Camera Sensor Networks," *IEEE Internet of Things Journal*, vol. 1, pp. 1-1, 2022.

[23] B. Cao, X. Kang, J. Zhao, P. Yang, Z. Lv and X. Liu, "Differential Evolution-Based 3-D Directional Wireless Sensor Network Deployment Optimization," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3594-3605, 2018.

[24] Z. -W. Sun, Z. -X. Hua, H. -C. Li and Y. Li, "A Flying Bird Object Detection Method for Surveillance Video," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1-14, 2024.

[25] Z. -W. Sun, Z. -X. Hua, H. -C. Li and H. -Y. Zhong, "Flying Bird Object Detection Algorithm in Surveillance Video Based on Motion Information," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1-15, 2024.

[26] M. Jiang, R. Sogabe, K. Shimasaki, S. Hu, T. Senoo and I. Ishii, "500-Fps Omnidirectional Visual Tracking Using Three-Axis Active Vision System," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1-11, 2021.

[27] X. Zhang, X. Chen, F. Farzadpour and Y. Fang, "A Visual Distance Approach for Multicamera Deployment with Coverage

Optimization," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 3, pp. 1007-1018, 2018.
[28] F. Jiang, X. Zhang, X. Chen and Y. Fang, "Distributed Optimization of Visual Sensor Networks for Coverage of a Large-Scale 3-D Scene," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 6, pp. 2777-2788, 2020.
[29] Depth of field, Accessed: Aug. 17, 2022. [Online]. Available: https://zh.m.wikipedia.org/zh-tw/%E6%99%AF%E6%B7%B1
[30] A. A. Zishan, I. Karim, S. S. Shubha, A. Rahman, "Maximizing Heterogeneous Coverage in over and Under Provisioned Visual Sensor Networks," *Journal of Network and Computer Applications*, Vol. 124, pp. 44-62, 2018.
[31] C. Watkins, P. Dayan, "Q-Learning," Kluwer Academic Publishers, Boston, pp. 1-14, 1992.
[32] E. Camci, M. Gupta, M. Wu and J. Lin, "QLP: Deep Q-Learning for Pruning Deep Neural Networks," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 10, pp. 6488-6501, 2022.
[33] B. Ning, F. Lin, S. Jaimungal, "Double Deep Q-Learning for Optimal Execution," *Applied Mathematical Finance*, Vol. 28, no. 4, pp. 361-380, 2021.
[34] Y. -J. Chen and D. -Y. Huang, "Joint Trajectory Design and BS Association for Cellular-Connected UAV: An Imitation-Augmented Deep Reinforcement Learning Approach," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2843-2858, 2022.
[35] R. Zhu, G. Jianxin, W. Feng, B. Lin, and Y. Huang, "Energy Efficient Transmission Power Control Policy of the Delay Tolerable Communication Service," *IEEE Access*, vol. 8, pp. 175 815–175 826, 2020.
[36] R. Li, Y. Zhao, C. Wang, X. Wang, V. C. M. Leung, X. Li, and T. Taleb, "Edge Caching Replacement Optimization for D2D Wireless Networks Via Weighted Distributed DQN," *IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6.
[38] V. P. Munishwar, and N. B. Abu-Ghazaleh. "Coverage Algorithms for Visual Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 4, pp. 1-36, 2013.
[39] J. Ai, A. A. Abouzeid. "Coverage by Directional Sensors in Randomly Deployed Wireless Sensor Networks," *Journal of Combinatorial Optimization*, vol. 11, pp. 21-41, 2006.
[40] "Tapo C520WS," Accessed: October 28, 2024. [Online]. Available: https://www.tp-link.com/tw/home-networking/cloud-camera/tapo-c520ws/
[41] "TP-Link VIGI 4MP," Accessed: October 28, 2024. [Online]. Available: https://www.tp-link.com/tw/business-networking/vigi-network-camera/vigi-c340s/
[42] "D-link DCS-7513," Accessed: October 28, 2024. [Online]. Available: https://www.dlinktw.com.tw/home/product?id=113
[43] Y. Li, J. He, C. Chen and X. Guan, "Intelligent Physical Attack Against Mobile Robots with Obstacle-Avoidance," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 253-272, 2023.
[44] M. A. Bühler and A. Lamontagne, "Coordinating Clearance and Postural Reorientation When Avoiding Physical and Virtual Pedestrians," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 30, pp. 1612-1620, 2022.
[45] S. Delmas, F. Morbidi, G. Caron, M. Babel and F. Pasteau, "SpheriCol: A Driving Assistant for Power Wheelchairs Based on Spherical Vision," *IEEE Transactions on Medical Robotics and Bionics*, vol. 5, no. 2, pp. 387-400, May 2023.
[46] Qu, Q., Liu, K., Li, X., Zhou, Y., & Lü, J., "Satellite observation and data-transmission scheduling using imitation learning based on mixed integer linear programming," *IEEE Transactions on Aerospace and Electronic Systems*, vol 2, no 59, pp. 1989-2001, 2022.
[47] X. Wang et al., "Deep Reinforcement Learning: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 5064-5078, April 2024.[48] C. Pan, Z. Peng, Y. Li, B. Han and D. Wang, "Flocking of Under-Actuated Unmanned Surface Vehicles via Deep Reinforcement Learning and Model Predictive Path Integral Control," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1-11, 2024.