

UAV-assisted VR adaptive content prefetching and joint optimization based on user perspective prediction

Zhengyu Zhao¹, Xingyu He^{2*}

¹School of Optical-Electrical & Computer Engineering, University of Shanghai for Science & Technology, Shanghai, China

²College of Publishing, University of Shanghai for Science & Technology, Shanghai, China

*Corresponding Author. Email:sherri_he@163.com

Abstract. As the demand for low latency and high immersion in Virtual Reality (VR) applications continues to grow, traditional passive content update mechanisms based on Age of Information (AoI) struggle to meet real-time requirements in scenarios with rapid user perspective changes. To address this issue, this paper proposes a UAV-assisted adaptive VR content prefetching method based on user perspective prediction. First, under the UAV-assisted edge computing framework, a heterogeneous system model is constructed that integrates user perspective states, scene semantic information, and Age of Information. Second, a lightweight intention prediction module composed of a temporal perspective encoder and cross-modal attention mechanism is designed to achieve accurate prediction of future gaze region probability distributions; a graph attention mechanism is then employed to deeply aggregate the prediction results with multi-source state information, generating a prediction-aware enhanced state representation. On this basis, the enhanced state representation is embedded into a hierarchical reinforcement learning scheduling framework, enabling the transition from passive response to active prefetching. Experimental results demonstrate that the proposed method outperforms comparison methods in terms of weighted end-to-end latency, content timeliness, and system energy efficiency, validating the effectiveness and practicality of the approach.

Keywords: UAV, virtual reality, perspective prediction, hierarchical reinforcement learning, graph attention

1. Introduction

Virtual Reality (VR) technology is evolving continuously, with its high-fidelity, low-latency immersive experiences becoming core requirements for diverse application scenarios including entertainment gaming, remote infrastructure inspection, disaster emergency response, and urban planning [1, 2]. However, in wide-area dynamic environments, traditional VR service modes relying on static modeling or offline pre-rendering face significant challenges regarding frequent scene changes, diverse viewport demands, and content timeliness, exhibiting evident deficiencies in real-time performance and flexibility.

In recent years, Unmanned Aerial Vehicle (UAV) swarms, leveraging their high mobility, three-dimensional deployment flexibility, and payload adaptability, have been regarded as key technological enablers to overcome these bottlenecks [3, 4]. Through aerial dynamic deployment, UAVs can provide flexible

environmental sensing functions while serving as mobile edge nodes for computational offloading and content distribution, opening new architectural pathways for VR services in wide-area dynamic scenarios. Current research on UAV-enabled VR primarily follows two directions: first, utilizing UAVs as aerial mobile sensing terminals to provide users with real-time first-person dynamic perspectives [5, 6]; second, employing UAVs as edge service nodes to enhance VR content rendering offloading efficiency through localized computation and caching [7, 8].

As real-time image acquisition and transmission devices, UAVs must address critical challenges including bandwidth fluctuations, end-to-end latency sensitivity, and real-time encoding and rendering of large-scale visual data. To solve these problems, research has explored multiple directions: Comşa et al. proposed the PriMARL framework, utilizing multi-agent collaborative decision-making for priority ordering and resource allocation of heterogeneous video services, dynamically optimizing wireless resource scheduling for acquired data to improve transmission QoS and user QoE [9]. binti et al. jointly optimized UAV positions, adaptive video resolutions, and transmission power based on DQN and Actor-Critic algorithms, achieving low-latency smooth transmission of real-time acquired data in complex environments [10]. Xiao et al. proposed a safe reinforcement learning-based anti-jamming scheme, intelligently adjusting video compression, channel coding, modulation types, and transmission power to effectively ensure transmission quality of acquired data in malicious jamming environments while reducing energy consumption [11].

As edge caching and computational offloading nodes, UAVs must achieve efficient task allocation and content rendering under conditions of dynamic network topology, limited computational resources, and heterogeneous user demands. Related research primarily covers: Ju et al. addressed the coupling challenge of task offloading and path planning in multi-UAV assisted edge computing, proposing a joint optimization framework based on Multi-Agent Deep Deterministic Policy Gradient (MADDPG), achieving adaptive offloading decisions in dynamic environments through centralized training and decentralized execution mechanisms [12]. Shi et al. addressed task offloading and trajectory scheduling in high-dimensional continuous action spaces for VR scenarios, constructing a multi-agent deep reinforcement learning-based offloading policy learning framework, and introducing prioritized experience replay to improve training efficiency and convergence stability [13]. Zhang et al. proposed a multi-agent deep reinforcement learning algorithm incorporating multi-head self-attention mechanisms to handle heterogeneous task types and hybrid action spaces, enhancing agents' perception of environmental dependencies in VR to optimize joint offloading decisions [14]. Li et al. proposed a collaborative rendering offloading framework that partitions rendering workloads among local devices, UAVs, and base stations according to task characteristics, thereby improving response performance under high-load conditions [15]. Kang et al. modeled the problem based on Partially Observable Markov Decision Processes, proposing a Multi-Agent Proximal Policy Optimization (MAPPO) algorithm to achieve collaborative optimization of VR multi-dimensional resource allocation and task offloading under resource constraints and collision avoidance conditions [16].

Specifically, building upon the aforementioned research, existing approaches for UAV-assisted VR remote rendering and content distribution can be broadly categorized into two schemes: One is the real-time image acquisition and rendering method based on passive response mechanisms. Such methods can promptly obtain the latest scene content after user requests are triggered, thereby ensuring content freshness to a certain extent. However, they typically rely on immediate UAV flight scheduling and high-bandwidth data transmission, resulting in high system end-to-end latency and significant energy consumption costs. The other is the rendering method based on historical image caching and edge computing. By reusing existing cached content and local computing resources, this approach effectively reduces communication load and system energy consumption. However, due to the lack of predictive capability for users' future viewport changes, it suffers

from cache invalidation issues when facing dynamic scenes and rapid viewport switching, making it difficult to meet strict real-time requirements. Consequently, existing methods generally lack modeling of user viewport evolution processes, fail to integrate viewport prediction, content prefetching, and UAV collaborative scheduling into a unified optimization framework, and overlook the intrinsic relationship between prediction-driven active prefetching mechanisms and the trade-off between acquisition costs and content freshness.

To this end, this paper proposes a hierarchical reinforcement learning decision framework enhanced by user viewport prediction and graph attention, achieving unified optimization of perspective-driven active content prefetching and UAV collaborative scheduling. Specifically, the main contributions of this paper are as follows: a) This paper proposes an AoI-aware state representation method incorporating user viewport prediction, jointly encoding users' historical gaze trajectories, grid content AoI, spatial relationships between UAVs and users, and UAV energy and resource states. Combined with graph attention mechanisms and cross-modal attention mechanisms, it adaptively characterizes the relationship between users' potential attention regions and system resource states, thereby constructing prediction-aware enhanced state representations that provide high-quality decision inputs for subsequent hierarchical reinforcement learning; b) This paper constructs a viewport prediction-driven hierarchical reinforcement learning decision framework, where the high-level policy dynamically decides whether to trigger active prefetching and the prefetching priority based on the predicted user future gaze probability distribution, guided jointly by content freshness gains and system cost constraints; the low-level policy, under the constraints of high-level decisions, further completes UAV task allocation and resource scheduling, achieving collaborative optimization of acquisition UAVs and rendering UAVs.

2. System model

This paper considers the UAV-assisted VR system scenario shown in Figure 1. The system consists of M UAVs, N VR scene area grid nodes, and 1 VR user node.

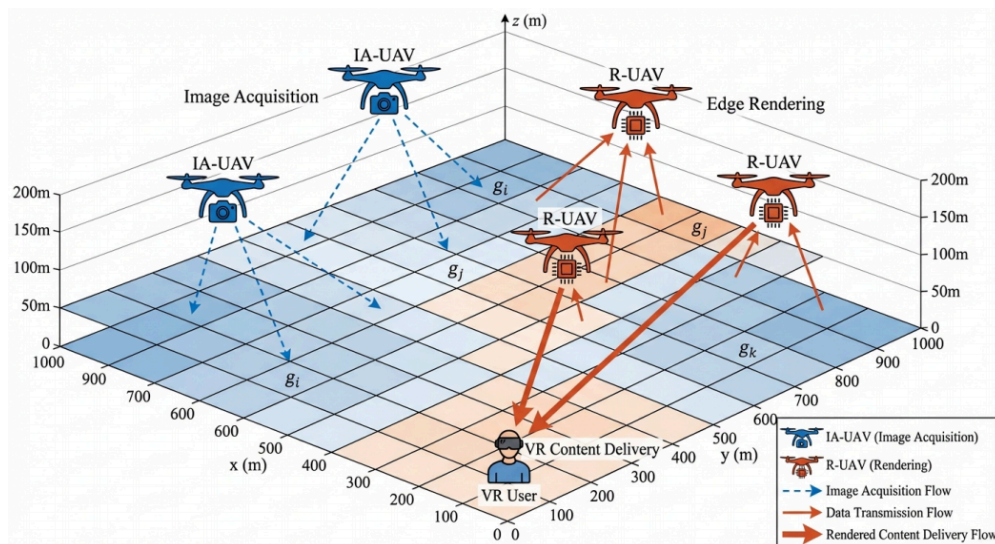


Figure 1. Three-dimensional scenario architecture diagram of UAV-assisted VR system

The UAV cluster is divided into two categories based on functional roles: Image Acquisition UAVs (IA-UAVs) and Rendering UAVs (R-UAVs), denoted as $U_A = \{u_1^A, u_2^A, \dots, u_{M_A}^A\}$ and $U_R = \{u_1^R, u_2^R, \dots, u_{M_R}^R\}$ respectively, with the total number of UAVs being $M = M_A + M_R$. The VR three-dimensional scene

space is divided into N non-overlapping grid sub-regions, with the i -th grid node denoted as g_i . Each grid represents an independent VR perception task domain with independent rendering complexity and content freshness status. The VR user node is denoted as v , with position coordinates $p_v = (x_v, y_v, z_v)$. The user obtains immersive VR experience by receiving rendering results.

The static attributes of the m -th UAV are (v_m^{max}, E_m^{max}) , where v_m^{max} is the maximum flight speed and E_m^{max} is the maximum energy capacity. At each decision moment t , the dynamic state of the UAV includes three-dimensional position $p_m(t) = (x_m(t), y_m(t), z_m(t))$, three-dimensional velocity vector $v_m(t)$, remaining energy $e_m(t)$, available computing resources $f_m(t)$, and available communication resources $b_m(t)$.

At each decision time step t , the system dynamically generates an image rendering task set $T(t) = \{\tau_1, \tau_2, \dots, \tau_K\}$ based on the freshness status of each grid's content, where the k -th task is represented as $\tau_k = (p_{\tau_k}, \theta_k, d_k)$. p_{τ_k} is the grid position corresponding to the task, θ_k is the content freshness indicator (i.e., Age of Information, AoI), and $d_k \in \{0, 1\}$ is the task execution mode decision variable: $d_k = 1$ indicates real-time image acquisition followed by rendering, while $d_k = 0$ indicates rendering using cached historical content.

The AoI of grid g_i is defined as the difference between the current moment and the moment when the grid was last acquired and updated:

$$\theta_i(t) = t - t_i^{last} \quad (1)$$

where t_i^{last} is the timestamp of the last acquisition update of grid g_i , initialized to $t_i^{last} = 0$, implying that all regional content is in a stale state when the system starts and requires priority updating. The wireless channels between UAVs and between UAVs and users are modeled using path loss models. The channel gain from UAV u_m to UAV u_n (or user v) is:

$$h_{m,n}(t) = h_0 \cdot \|p_m(t) - p_n(t)\|^{-\alpha} \quad (2)$$

where h_0 is the channel gain at reference distance, α is the path loss exponent, and $\|\cdot\|$ denotes Euclidean distance. Based on Shannon channel capacity, the instantaneous transmission rate from UAV u_m to node n (UAV or user) is:

$$R_{m,n}(t) = B \log_2 \left(1 + \frac{P_m \cdot h_{m,n}(t)}{\sigma^2 + I_{m,n}(t)} \right) \quad (3)$$

where B is the channel bandwidth, P_m is the transmission power of UAV u_m , σ^2 is the noise power, and $I_{m,n}(t)$ is the co-channel interference strength at the receiver. The total energy consumption of UAVs consists of three components: flight energy, computation energy, and communication energy. Flight energy: The power consumption of UAV u_m flying at velocity $\|v_m\|$ is modeled as:

$$P_m^{fly} = P_0 \left(1 + \frac{3\|v_m\|^2}{U_{tip}^2} \right) + P_i \left(\sqrt{1 + \frac{\|v_m\|^4}{4v_0^4}} - \frac{\|v_m\|^2}{2v_0^2} \right)^{1/2} + \frac{1}{2} d_0 \rho s A \|v_m\|^3 \quad (4)$$

where P_0 and P_i are the blade profile power and induced power in hovering, respectively, U_{tip} is the tip speed of the rotor blade, v_0 is the mean induced velocity in hovering, and d_0, ρ, s, A are the fuselage drag ratio, air density, rotor solidity, and rotor disc area, respectively. Computation energy: When rendering UAV u_m^R executes computational tasks, the computation energy is proportional to the square of CPU operating frequency:

$$E_m^{cmp} = \kappa \cdot C_k \cdot f_m^2 \quad (5)$$

where κ is the effective capacitance coefficient related to chip architecture, C_k is the total CPU cycles required to process task τ_k , and f_m is the computing frequency of the UAV. Communication energy: The communication energy consumed by UAV u_m during duration Δt is:

$$E_m^{com} = P_m \cdot \Delta t \quad (6)$$

3. Uav-assisted VR adaptive content prefetching and joint optimization based on user viewport prediction

3.1. User intent prediction module

3.1.1. Module architecture

The objective of the user intent prediction module is to output the grid gaze probability distribution for the future steps, denoted as $\hat{\pi}(t)$, at each decision time step, based on the user's historical gaze trajectory and current scene state. The module adopts a three-stage serial architecture of temporal encoder scene-aware cross-modal attention probability distribution prediction head, achieving accurate modeling of user viewport intent while maintaining lightweight inference.

The design logic of the three-stage architecture is as follows: The first-stage temporal encoder is responsible for extracting the user's motion inertia and behavioral habits from historical gaze sequences, outputting a compact intent hidden state; The second-stage cross-modal attention mechanism fuses the intent hidden state with scene semantic features, enabling prediction results to depend not only on the user's own motion patterns but also to perceive the attractive effects of scene content on attention; The third-stage prediction head maps the fused features to a normalized probability distribution over the global grid, directly outputting an intent heat map usable for scheduling decisions. The entire module shares scene feature extraction with the scheduling framework, introducing no additional graph computation overhead during inference, with overall latency controlled within the allowable range of a single decision step.

3.1.2. Temporal viewport encoder

To enable the temporal model to simultaneously perceive the user's posture changes, the content state of the current grid, and the user's own movement speed, feature vectors are constructed for each moment $t-l$ ($l = 1, \dots, L$) within the historical window:

$$e_l = [\omega_v(t-l), \varphi_v(t-l), g_{focused}(t-l), \theta_{focused}(t-l), v_v(t-l)] \quad (7)$$

where $\omega_v(t-l)$ and $\varphi_v(t-l)$ are the head yaw and pitch angles at that moment $g_{focused}(t-l) \in \{0, 1\}^N$ is the one-hot encoding of the user-gazed grid, $\theta_{focused}(t-l)$ is the AoI of the corresponding grid at that moment, and $v_v(t-l)$ is the user's movement velocity scalar. The purpose of introducing $\theta_{focused}$ is to allow the model to perceive whether the user historically tends to gaze at fresh content or accepts a certain degree of stale content, providing personalized basis for subsequent prefetching decisions. Arranging the feature vectors of L moments in chronological order yields the input sequence matrix $E(t) = [e_1, e_2, \dots, e_L]^T \in R^{L \times d_e}$, where d_e is the single-step feature dimension.

Gated Recurrent Units (GRU) are employed for temporal encoding of the input sequence $E(t)$. Through the coordination of update gates and reset gates, GRU retains long-term behavioral patterns while assigning higher response weights to recent posture changes. The recurrence process is:

$$h_l = GRU(e_l, h_{l-1}), \quad l = 1, 2, \dots, L \quad (8)$$

Finally, the hidden state of the last step is taken as the compact representation of user intent:

$$h_{intent} = h_L \in R^{d_h} \quad (9)$$

where d_h is the GRU hidden layer dimension. h_{intent} fuses the user's complete trajectory information over the past L steps, serving as the source of query vectors for subsequent scene-aware attention calculations.

GRU is chosen rather than more complex Transformer structures because the historical window length $L \leq 10$ and decision step $\Delta t = 1$ second in this system represent relatively short sequences. GRU offers faster inference speed at this scale, more stable gradient propagation, and easier end-to-end integration with the overall reinforcement learning training flow.

3.1.3. Scene-aware cross-modal attention

User viewport movement is not purely determined by motion inertia; scene content itself has significant guiding effects on attention. Regions with high semantic complexity, grids with recent changes, and content matching user historical preferences all exert attractive effects on the next moment's gaze point. Predictions relying solely on GRU hidden states ignore this scene-driven factor, easily producing large prediction deviations when scene content changes abruptly. Therefore, a cross-modal attention mechanism is introduced to cross-fuse the intent hidden state with scene features of each grid, enabling prediction results to be modulated by both user behavioral inertia and scene semantic attractiveness.

Using the intent hidden state h_{intent} as Query and the enhanced feature representations of each grid node $\{\tilde{x}_i^G\}_{i=1}^N$ as Key-Value, cross-modal attention weights are computed:

$$\alpha_i = \text{softmax}_i\left(\frac{(W_Q \cdot h_{intent})^\top \cdot (W_K \cdot \tilde{x}_i^G)}{\sqrt{d_k}}\right) \quad (10)$$

where $W_Q \in R^{d_k \times d_h}$ and $W_K \in R^{d_k \times d_G}$ are the linear projection matrices for query and key respectively, d_k is the attention head dimension, d_G is the grid enhanced feature dimension, and $\sqrt{d_k}$ is the scaling factor to prevent dot products from entering the saturation zone of softmax. Based on the normalized attention weights $\{\alpha_i\}$, value features of each grid are aggregated to obtain the scene-aware user intent representation:

$$z_{intent} = \sum_{i=1}^N \alpha_i \cdot (W_V \cdot \tilde{x}_i^G) \in R^{d_k} \quad (11)$$

where $W_V \in R^{d_k \times d_G}$ is the linear projection matrix for values. z_{intent} is the weighted perception vector of the overall scene; the higher the attention weight α_i , the stronger the matching degree between the scene features of grid g_i and the current user intent state, and the greater the contribution to the aggregation result.

Here \tilde{x}_i^G directly reuses the output features from the GAT module without additional computation. This parameter sharing design enables the prediction module and scheduling module to jointly rely on the same scene representation, with gradient signals from both propagating back to the GAT during training, prompting the graph representation to optimize simultaneously for scheduling performance and prediction accuracy. After concatenating the temporal encoding hidden state h_{intent} with the scene-aware intent representation z_{intent} , it is mapped to an N -dimensional output space through a two-layer fully connected network and normalized by softmax to obtain the global gaze probability distribution:

$$\hat{\pi}(t) = \text{Softmax}(MLP([h_{intent} || z_{intent}])) \in R^N \quad (12)$$

where $||$ denotes vector concatenation, and MLP consists of two linear transformations and ReLU activation functions, with the output layer dimension being N (total number of grids). The softmax ensures the sum of output probabilities equals 1, satisfying the normalization constraint defined in Equation (4.2). During the inference phase, the top K components of $\hat{\pi}(t)$ with highest probabilities are taken to form the prefetching candidate set for the current decision step:

$$\hat{P}(t) = Top - K(\hat{\pi}(t)), \quad K = \min(M_A, K_{max}) \quad (13)$$

where K_{max} is the preset maximum number of prefetching grids per step, and M_A is the number of currently available IA-UAVs. When some IA-UAVs are low on battery or executing other tasks, the value decreases accordingly, ensuring the prefetching candidate set remains within resource constraints.

The above prediction probability distribution $\hat{\pi}(t)$ will serve as a prior saliency feature input to the graph attention network, guiding attention weight allocation in heterogeneous graph construction.

3.2. Graph attention mechanism

3.2.1. Heterogeneous graph construction

To effectively characterize the multi-dimensional interaction relationships between UAVs, task grids, and users in the system, this section constructs a directed heterogeneous graph $G = (V, E)$, where the node set contains all UAV nodes, grid nodes, and user nodes. As shown in Figure 2, the edge set consists of four types of directed edges corresponding to UAV-UAV, UAV-Grid, Grid-Grid, and UAV-User interaction relationships.

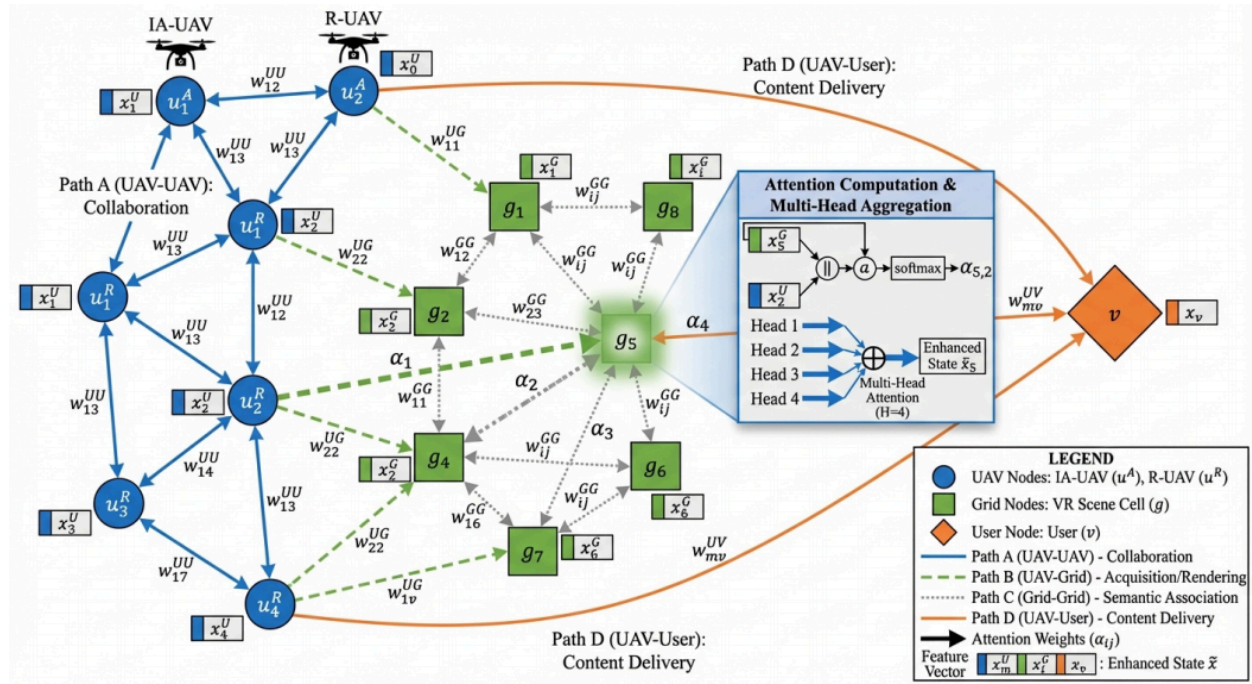


Figure 2. Heterogeneous network enhanced state representation based on graph attention mechanism

(1) Node Feature Definition

UAV node features: The feature vector of the m -th UAV is defined as:

$$x_m^U = [p_m, v_m, e_m, f_m, b_m, r_m] \in R^{10} \quad (14)$$

where $p_m \in R^3$ is the three-dimensional position coordinates, $v_m \in R^3$ is the three-dimensional velocity vector, $e_m \in R$ is the remaining energy, $f_m \in R$ is the available computing frequency, $b_m \in R$ is the available communication bandwidth, and $r_m \in \{0, 1\}$ is the role identifier (0 for acquisition UAV, 1 for rendering UAV).

Grid node features: The feature vector of the i -th grid node is defined as:

$$x_i^G = [c_i, s_i, \rho_i, t_i^{last}, f_i^{vis}, \hat{\pi}(t)] \quad (15)$$

where $c_i \in R^3$ is the geometric center position of the grid, s_i is the one-hot encoding vector of the grid semantic label, $\rho_i \in R$ is the rendering computational load, $t_i^{last} \in R$ is the timestamp of the last acquisition update, $f_i^{vis} \in R^d$ is the visual feature vector extracted from the image corresponding to the grid, and $\hat{\pi}(t)$ is the predicted gaze probability of the grid, incorporated into node features as prior knowledge.

User node features: The feature vector of the user node is defined as its three-dimensional position coordinates:

$$x_v = p_v \in R^3 \quad (16)$$

(2) Edge Weight Definition

The weights of various edges are modeled based on the physical semantics of corresponding interaction relationships, as detailed below.

UAV-UAV edges (collaborative relationship): The edge weight from UAV to comprehensively considers spatial proximity and communication link quality:

$$w_{mn}^{UU} = w_{mn}^{prox} \cdot w_{mn}^{link} \quad (17)$$

where the spatial proximity weight is:

$$w_{mn}^{prox} = \exp\left(-\frac{\|p_m - p_n\|^2}{2\sigma_d^2}\right) \cdot 1[\|p_m - p_n\| \leq R_{max}] \quad (18)$$

where σ_d controls the spatial decay rate and R_{max} is the maximum collaboration radius. the communication link quality weight is:

$$w_{mn}^{link} = \frac{R_{mn}}{R_{max}^{sys}} \quad (19)$$

UAV-Grid edges (acquisition relationship): The edge weight from UAV to grid integrates spatial distance and historical interaction information:

$$w_{mi}^{UG} = w_{mi}^{dist} \cdot w_{mi}^{hist} \quad (20)$$

where the spatial distance weight w_{mi}^{UG} is defined the same as Equation (17) (replacing p_n with grid center c_i); the historical interaction weight w_{mi}^{hist} comprehensively characterizes temporal correlation and content similarity:

$$w_{mi}^{hist} = \exp(-\beta \cdot \Delta t_{mi}) \cdot \frac{f_m^{last} \cdot f_i^{vis} + \epsilon}{\|f_m^{last}\| \|f_i^{vis}\| + \epsilon} \quad (21)$$

where Δt_{mi} is the time interval from when UAV u_m last acquired grid g_i to the current moment, β is the historical forgetting coefficient, f_m^{last} is the image feature vector at that acquisition, and ϵ is a small constant to prevent division by zero.

Grid-Grid edges (semantic correlation): The semantic similarity weight between grid g_i and g_j is:

$$w_{ij}^{GG} = s_i \cdot s_j \quad (22)$$

where s_i and s_j are the semantic label one-hot vectors of the two grids respectively, with the dot product result reflecting their overlap degree in semantic space.

UAV-User edges (offloading relationship): The edge weight from UAV u_m to user v integrates downlink communication capability and content scale:

$$w_{mv}^{UV} = w_{mv}^{dl} \cdot w_k^{size} \quad (23)$$

where the downlink communication capability weight is $w_{mv}^{dl} = R_{mv}/R_{max}^{sys}$, with R_{mv} being the downlink transmission rate from UAV u_m to the user; the content scale weight is:

$w_k^{size} = \exp(-\frac{D_k}{\bar{D}})$ where D_k is the data scale of the current content to be rendered, \bar{D} is the average task data scale, and the exponential decay form ensures smaller content scales correspond to larger weights, avoiding numerical instability issues caused by direct reciprocal operations.

3.2.2. Attention-based node feature aggregation

After constructing the heterogeneous graph and defining various edge weights, this section employs graph attention mechanisms to aggregate node features and generate enhanced representations for each node. Unlike standard GAT, this section explicitly incorporates the heterogeneous edge weights defined in Section 2.1 into the calculation of attention coefficients, making attention allocation depend not only on node feature similarity but also on the joint influence of physical distance, communication quality, and historical interactions.

(1) Attention Coefficient Calculation

For target node i and its neighbor node j , the attention coefficient calculation introduces a linear transformation matrix W_ϕ^e for edge type $\phi(i, j)$ to map edge weight features to a unified dimension:

$$e_{ij} = \text{LeakyReLU}(a^\top [Wx_i \parallel Wx_j \parallel W_\phi^e \cdot w_{ij}]) \quad (25)$$

where W is the node feature transformation matrix, a is the attention vector, \parallel denotes vector concatenation, and w_{ij} is the corresponding edge weight scalar (or vector).

(2) Attention Weight Normalization

The attention coefficients within the neighborhood are normalized using the Softmax function:

$$\alpha_{ij} = \text{Softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N(i)} \exp(e_{ik})} \quad (26)$$

where $N(i)$ represents the set of neighbor nodes of node i in the heterogeneous graph.

(3) Multi-Source Information Aggregation and Enhanced Representation Generation

Based on the normalized attention weights, various types of nodes update their enhanced representations through weighted aggregation of neighbor features.

Grid node enhanced representation: Grid node comprehensively aggregates performance indicator projections from candidate acquisition UAVs (path B) and rendering UAVs, as well as contextual semantic information from neighbor grids (path C):

$$\tilde{x}_i^G = \sigma(\sum_{m \in U_A} \alpha_{mi}^B W_A x_m^U + \sum_{m \in U_R} \alpha_{mi}^B W_R x_m^U + \sum_{j \in N_G(i)} \alpha_{ij}^C W_G x_j^G) \quad (27)$$

where W_A and W_R are linear transformation matrices that map acquisition UAV features and rendering UAV features to acquisition cost space and rendering cost space respectively, making the aggregation result explicitly contain four physical indicators: estimated acquisition delay \tilde{L}_i^A , acquisition energy \tilde{E}_i^A , rendering delay \tilde{L}_i^R , and rendering energy \tilde{E}_i^R ; W_G is used to extract spatial semantic features of neighbor grids, and $\sigma(\cdot)$ is the activation function.

UAV node enhanced representation: UAV node u_m simultaneously aggregates collaborative states from neighbor UAVs (path A), demand features from task grids (path B), and transmission link states from visible users (path D):

$$\tilde{x}_m^U = \sigma(\sum_{n \in N_U(m)} \alpha_{nm}^A W_{UU} x_n^U + \sum_{i \in N_G(m)} \alpha_{im}^B W_{UG} x_i^G + \alpha_{vm}^D W_{UV} x_v) \quad (28)$$

where path D introduces user node information, enabling the enhanced representation to perceive the downlink link quality and transmission distance of the UAV relative to the user, thereby incorporating data backhaul overhead into the evaluation of total end-to-end delay.

User node enhanced representation: User node v only aggregates offloading service capabilities from visible UAVs (path D):

$$\tilde{x}_v = \sigma(\sum_{m \in U} \alpha_{mv}^D W_{VU} x_m^U) \quad (29)$$

This formula indicates that the user node forms a global perception representation of the current aerial-ground network edge offloading capabilities by aggregating the status information of all visible UAVs according to attention weights.

3.3. Hierarchical reinforcement learning

3.3.1. High-level policy network

Responsible for making service mode selection decisions at each decision time slot based on the current system state. Specifically, the high-level agent takes the enhanced state representation output by the graph attention network as input, comprehensively evaluates the content freshness level of each VR task grid and the current system resource load status, and selects the task type to be prioritized in the current time slot from the predefined service mode set, i.e., deciding whether system resources should focus on image acquisition updates or rendering transmission services.

(1) State Space and Action Space

The input state of the high-level agent consists of the global pooled vector of enhanced representations of all nodes. Specifically, the enhanced representations of all UAV nodes and grid nodes are mean-pooled respectively and concatenated with the user node enhanced representation to form the high-level global state vector:

$$s^H(t) = [\frac{1}{M} \sum_{m=1}^M \tilde{x}_m^U(t) \parallel \frac{1}{N} \sum_{i=1}^N \tilde{x}_i^G(t) \parallel \tilde{x}_v(t)] \in R^{d_H} \quad (30)$$

where d_H is the dimension of the high-level state vector, and $\tilde{x}_m^U(t)$, $\tilde{x}_i^G(t)$, $\tilde{x}_v(t)$ are given by Equations (27), (28), and (29) respectively. The high-level action space is defined as a discrete service mode set $A^H = \{0, 1\}$, where $a^H = 1$ indicates prioritizing image acquisition updates, and $a^H = 0$ indicates prioritizing rendering transmission resource scheduling.

(2) Action Selection Policy

The high-level policy's action selection is guided by the dynamic trade-off between content freshness gains and estimated execution costs. To quantify this trade-off, the system freshness pressure index at the current moment is defined as:

$$\Phi(t) = \frac{1}{N} \sum_{i=1}^N \frac{\theta_i(t)}{\theta_{max}} \cdot \mathbb{1}[\theta_i(t) > \theta_{th}] \quad (31)$$

where $\theta_i(t)$ is the AoI of grid g_i at time t , θ_{max} is the maximum tolerance threshold, and θ_{th} is the AoI threshold triggering updates. $\Phi(t) \in [0, 1]$ indicates higher overall content staleness of the system, implying more urgent acquisition requirements.

(3) Reward Function

The high-level policy network adopts the Actor-Critic architecture, optimizing for discounted cumulative reward. The high-level reward function comprehensively measures the contribution of service mode selection to long-term system performance, defined as:

$$r^H(t) = \underbrace{\mu_1 \cdot \Delta \bar{\theta}(t)}_{\text{Freshness Gain}} - \underbrace{\mu_2 \cdot \bar{L}(t)}_{\text{Latency Penalty}} - \underbrace{\mu_3 \cdot \bar{E}(t)}_{\text{Energy Penalty}} - \underbrace{\mu_4 \cdot \mathbb{I}[\exists i : \theta_i(t) > \theta_{max}]}_{\text{Constraint Violation Penalty}} \quad (32)$$

where $\Delta \bar{\theta}(t) = \frac{1}{N} \sum_{i=1}^N [\theta_i(t-1) - \theta_i(t)]^+$ is the decrease in average AoI across all grids within the current time slot, with $[\cdot]^+ = \max(\cdot, 0)$; $\bar{L}(t)$ and $\bar{E}(t)$ are the system-wide weighted average end-to-end latency and total energy consumption respectively; $\mu_1, \mu_2, \mu_3, \mu_4 > 0$ are weight coefficients for each

reward/penalty term. The last term imposes a hard penalty for AoI constraint violations, guiding the policy to spontaneously respect constraints during long-term operation.

3.3.2. Low-level policy network

The low-level policy network is responsible for completing specific UAV scheduling allocation decisions under the constraints of high-level instructions.

(1) State Space and Action Space

The low-level agent's input state further incorporates semantic encoding of high-level actions on top of the high-level enhanced state, enabling conditional perception of service mode constraints:

$$s^L(t) = [\tilde{x}_m^U(t) \parallel \tilde{x}_i^G(t) \parallel \tilde{x}_v(t) \parallel \phi(a^H(t))] \in R^{d_L} \quad (33)$$

where $\phi(a^H(t)) \in R^{d_e}$ is the embedding representation of the high-level action, mapping discrete actions to a continuous semantic space through a learnable embedding layer, enabling the low-level policy to perceive the service mode constraints of the current time slot. Given the service mode $a^H(t)$ determined by the high-level policy, the low-level action space is correspondingly conditioned:

$$R^L(a^H) = \begin{cases} U_A \times N \times [0, 1] \times [P_{min}, P_{max}] & a^H = 1(\text{CaptureMode}), \\ U_R \times N \times [0, 1] \times [P_{min}, P_{max}] & a^H = 0(\text{RenderMode}), \end{cases} \quad (34)$$

where the first dimension is the selection of the executing UAV, the second dimension is the assignment of the target grid, the third dimension is the computational resource allocation ratio $\alpha_m \in [0, 1]$, and the fourth dimension is the transmission power $P_m \in [P_{min}, P_{max}]$. By decomposing the global joint action space $A_{joint} = U \times N \times [0, 1] \times [P_{min}, P_{max}]$ into subspaces $A^L(a^H)$ conditioned by a^H , the search space scale of single-step decision-making is reduced by approximately M/M_A or M/M_R times, effectively alleviating the curse of dimensionality.

(2) Reward Function

The low-level reward function further refines the high-level reward, focusing on measuring the immediate execution effect of specific scheduling schemes at the time slot level, defined as:

$$r^L(t) = \underbrace{v_1 \cdot \eta_{ch}(t)}_{\text{Channel Utilization}} - \underbrace{v_2 \cdot \sum_{m \in \mathcal{U}} E_m^{fly}(t)}_{\text{Flight Energy}} - \underbrace{v_3 \cdot \bar{L}_{exec}(t)}_{\text{Execution Latency}} + \underbrace{v_4 \cdot \mathbb{1}[\mathcal{T}_{done}(t) \neq \emptyset]}_{\text{Task Completion Reward}} \quad (35)$$

where channel utilization $\eta_{ch}(t) = \sum_m R_{m,\cdot}(t) / \sum_m R_m^{max}$ measures the actual utilization of available channel capacity by the current scheduling scheme; flight energy $E_m^{fly}(t) = P_m^{fly} \cdot \Delta t$; $\bar{L}_{exec}(t)$ is the average execution latency of all executed tasks in this time slot, comprehensively considering computational latency $L^{cmp} = C_k / f_m$.

4. Experiments

4.1. Parameters setting

The simulation scenario is set as a $1,000 \times 1,000 \times 200$ meter three-dimensional urban airspace, simulating a typical VR application environment with dense urban building clusters and frequent viewport changes. The scene space is uniformly divided into $10 \times 10 = 100$ non-overlapping grid sub-regions. The initial AoI of each grid is randomly initialized within the range $[0, T_{max}]$ to simulate the heterogeneity of content update states in real scenarios. Detailed parameters are shown in Table 1.

Table 1. Experiment parameters setting

Parameter	value
UAV Number M	6
Channel bandwidth B	20 MHz
Transmission power P_m	200 mW
Noise power N_0	-174 dBm/Hz
Latency weight λ_1	0.6
Energy weight λ_2	0.4
Low-level learning rate η_{low}	1×10^{-3}
High-level learning rate η_{high}	3×10^{-4}
Discount factor γ	0.95
Prediction step $\Delta\tau$	3 steps
Historical window length L	8 steps
Prefetching reward weight λ_3	0.3

4.2. Simulation results

4.2.1. Performance analysis

4.2.1.1. Different numbers of UAVs

Figure 3 presents the variations in latency, energy consumption, and Peak AoI (PAoI) under different UAV quantities. When the number of UAVs increases from 4 to 10, the system's weighted average latency, energy consumption, and PAoI exhibit a trend of first decreasing and then stabilizing. When the number of UAVs increases to 8, latency achieves the minimum reduction and energy efficiency reaches the highest level, indicating that adequate resource supply can effectively improve system performance. When the number of UAVs further increases to 10, the optimization of latency and PAoI approaches saturation, though energy efficiency slightly decreases. Excessive UAVs lead to diminishing marginal returns and may even cause slight performance fluctuations due to increased coordination overhead.

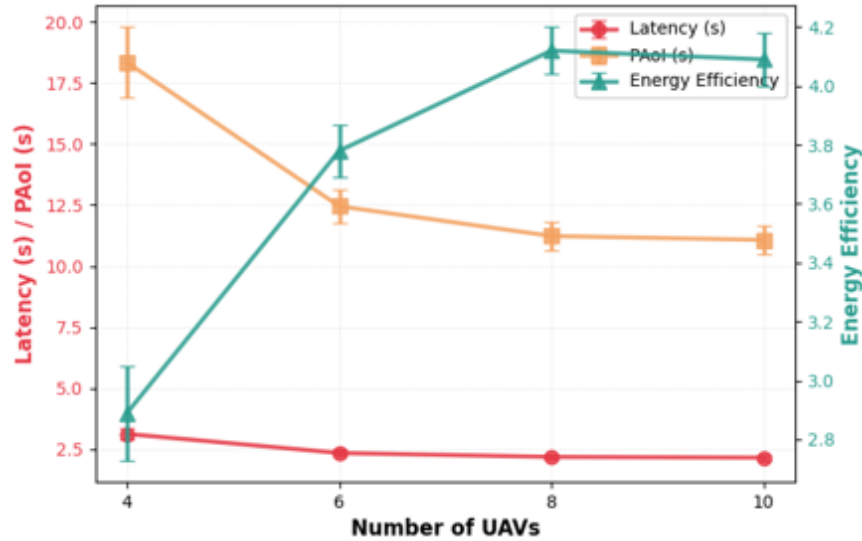


Figure 3. Performance comparison under different numbers of UAVs

4.2.1.2. Joint impact of historical window length and prediction step

Table 2 shows the impact of different combinations of historical window length and prediction step on the Prefetching Hit Rate (PHR), with other parameters maintained at default configurations. As the historical window increases from 2 to 8, PHR significantly improves across all values, indicating that longer historical sequences help capture more complete user behavioral patterns. When further increases to 16, the PHR improvement tends to saturate (the gap from does not exceed 0.4 percentage points), while inference computation increases linearly with sequence length. Considering the trade-off, is the optimal choice for cost-effectiveness. The prediction step has a monotonically negative impact on PHR: larger makes prediction tasks more difficult, resulting in lower hit rates. steps achieves a reasonable balance between hit rate (74.3%) and prefetching lead time (reserving sufficient flight preparation time for IA-UAVs), serving as the default configuration for this system.

Table 2. Prefetching Hit Rate (PHR) under different combinations of L and $\Delta\tau$ (%)

$\frac{\Delta\tau}{L}$	1 Step	2 Steps	3 Steps	5 Steps
2	69.1	65.3	58.2	49.6
4	73.8	70.2	64.7	54.1
8	78.5	76.1	74.3	62.8
16	78.9	76.4	74.6	63.1

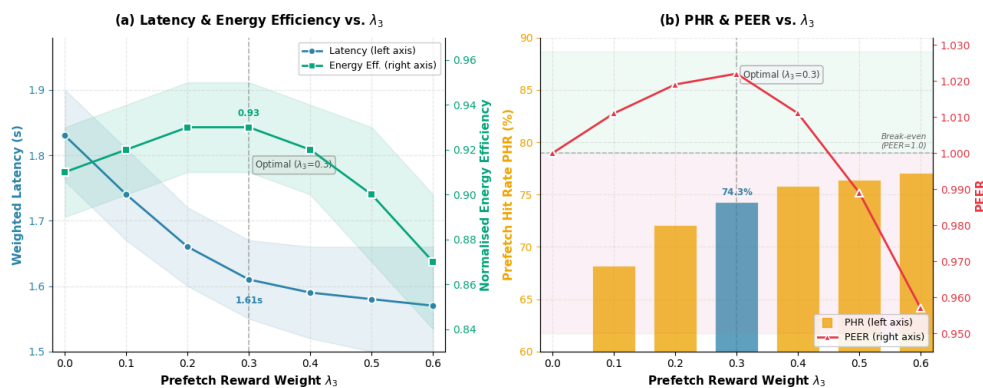
4.2.1.3. Impact of prefetching reward weight

Table 3 further shows the joint impact of prefetching reward weight λ_3 varying in the range on $[0, 0.6]$ weighted latency and normalized energy efficiency ($\lambda_1 = 0.6$, $\lambda_2 = 0.4$ fixed).

Table 3. System performance under different prefetching reward weights λ_3

λ_3	Weighted Latency (s) ↓	Normalized Energy Efficiency ↑	PHR (%) ↑	PEER ↑
0.0	1.83(0.07)	0.91(0.02)	—	1.000
0.1	1.74(0.07)	0.92(0.02)	68.2	1.011
0.2	1.66(0.06)	0.93(0.02)	72.1	1.019
0.3	1.61(0.06)	0.93(0.02)	74.3	1.022
0.4	1.59(0.07)	0.92(0.02)	75.8	1.011
0.5	1.58(0.08)	0.90(0.03)	76.4	0.989
0.6	1.57(0.09)	0.87(0.03)	77.1	0.957

As shown in Figure 4, when $\lambda_3 = 0$, the system degrades to a passive mode without prefetching. As λ_3 increases, the system gradually tends to trigger more prefetching tasks, with weighted latency continuously decreasing and PHR steadily improving. However, after $\lambda_3 > 0.4$, normalized energy efficiency begins to significantly decrease, with PEER falling below 1.0, because excessively aggressive prefetching strategies drive the system to frequently trigger acquisitions for low-hit-rate grids, introducing large amounts of invalid flight maneuvers. At $\lambda_3 = 0.3$, the latency improvement (12.0% reduction compared to passive baseline) and energy efficiency maintenance (PEER=1.022) achieve the optimal comprehensive performance, serving as the default recommended configuration in this paper. The above analysis indicates that system performance is relatively stable when is set in the range, demonstrating certain hyperparameter robustness.

**Figure 4.** Joint sensitivity analysis of system latency, energy efficiency, prefetching hit rate, and energy efficiency ratio under different prefetching reward weights

4.2.2. Ablation study

To quantitatively analyze the independent contributions of each functional component, the following four ablation variants were designed for comparison, with results shown in Table 4.

(1) w/o CrossModal: Removes the cross-modal attention module, directly outputting the prediction distribution via MLP using only the GRU hidden state without fusing scene semantic features.

(2) w/o Pretrain: Skips the supervised pre-training phase, with the prediction module undergoing joint reinforcement learning training directly from random initialization, evaluating the impact of pre-training on convergence quality.

(3) w/o SoftConstraint: Removes the Q-value bias based on prediction probabilities in low-level scheduling; prediction results only implicitly influence policy through grid feature fusion without explicit soft

priority constraints.

(4) Hard Constraint: Replaces the soft constraint with a hard constraint, forcing the low-level policy to prioritize executing prefetching tasks in $\hat{P}(t)$, evaluating the robustness difference between soft and hard constraint designs.

The analysis from Table 4 is as follows. After removing cross-modal attention (w/o CrossModal), PHR drops from 74.3% to 61.4%, a decrease of 17.4%, PAoI rises to 35.8 seconds, and weighted latency increases by 6.8%. This result demonstrates the non-negligible contribution of scene semantic features to prediction accuracy. User viewport fixations are determined not only by motion inertia but also by active attraction effects from high-saliency regions in the scene; relying solely on GRU temporal modeling cannot adequately capture this scene-driven factor. Notably, even without cross-modal attention, w/o CrossModal's latency (1.72s) still outperforms the no-prediction baseline GAT-HRL (1.83s), indicating that the temporal encoder itself can provide effective prediction signals, with cross-modal attention bringing significant further accuracy improvements.

Table 4. Ablation study results

Variant	Weighted Latency (s) ↓	Normalized Energy Efficiency ↑	PAoI (s) ↓	Content Freshness ↑	PHR (%) ↑	PEER ↑
GAT-HRL-Pred(Ours)	1.61 (0.06)	0.93 (0.02)	31.2 (1.4)	0.91 (0.01)	74.3 (2.1)	1.022
w/o CrossModa	1.72 (0.08)	0.90 (0.02)	35.8 (1.9)	0.87 (0.02)	61.4 (2.6)	0.996
w/o Pretrain	1.79 (0.10)	0.88 (0.03)	37.1 (2.2)	0.85 (0.02)	58.7 (3.1)	0.983
w/o SoftConstraint	1.68 (0.07)	0.91 (0.02)	33.4 (1.6)	0.89 (0.01)	74.1 (2.2)	1.008
Hard Constraint	1.69 (0.09)	0.87 (0.03)	32.9 (1.7)	0.90 (0.02)	74.5 (2.0)	0.961

Skipping supervised pre-training (w/o Pretrain) causes PHR to drop to 58.7% and weighted latency to rise to 1.79s, showing the largest gap from the complete model. This reflects the severity of the cold start problem: during early joint training, the prediction module outputs nearly random probability distributions, with low-quality prefetching decisions continuously transmitting misleading signals to the scheduling module, causing the overall training process to converge to suboptimal solutions. Supervised pre-training effectively avoids this problem by providing the prediction module with basic semantic understanding capabilities, serving as a critical step in ensuring joint training quality. After removing soft constraints (w/o SoftConstraint), PHR remains almost unchanged (74.1%), but weighted latency rises from 1.61s to 1.68s, with slight energy efficiency degradation. This indicates that prediction results have implicitly influenced policy decisions through feature fusion, but explicit Q-value bias can more directly and rapidly inject prediction signals into low-level scheduling, giving high-probability grids clear priority in resource competition, thereby further compressing end-to-end latency by approximately 4.1%, demonstrating clear component contribution.

The hard constraint design (Hard Constraint) performs similarly to the complete model in PHR and PAoI, but normalized energy efficiency significantly decreases to 0.87 with PEER dropping to 0.961. Enforced prefetching task hard constraints respond quickly when predictions are accurate, but when predictions deviate or high-priority grids are far away, the system is forced to dispatch IA-UAVs for costly flight maneuvers, unable to learn to choose actually better scheduling solutions, causing unnecessary energy waste. The soft constraint design aims to prioritize following prefetching suggestions when predictions are credible while allowing the scheduling policy to autonomously correct when predictions are unreliable, maintaining latency improvement while preserving energy efficiency robustness.

4.2.3. Comparative analysis

This paper selects the following four baseline algorithms for comparison.

(1) DRL-VR [15]: A VR edge rendering method based on multi-agent deep reinforcement learning that improves edge computing efficiency through joint optimization of UAV hovering positions and rendering task scheduling. However, it does not model real-time image acquisition tasks; in content stale regions, it can only rely on pre-stored historical data, lacking active perception and update mechanisms for content freshness.

(2) Cache-Opt [8]: A UAV-assisted VR service method based on three-dimensional deployment and content hierarchical caching optimization that improves service coverage efficiency through offline optimization of UAV deployment positions and cached content structure. However, its offline pre-caching mechanism struggles to adapt to dynamic scenarios with rapid content changes, with cache hit rates significantly declining under high freshness requirements.

(3) Greedy: A heuristic task scheduling method based on greedy strategy that prioritizes selecting UAVs closest to target grids with highest remaining energy to execute tasks at each decision moment. Mode selection fixedly adopts historical cache rendering, lacking global optimization perspective, easily leading to unbalanced UAV loads and rapid energy depletion.

(4) Random: A random baseline method where UAV selection and task modes (acquisition or rendering) are randomly decided, serving as a lower bound reference for system performance.

Table 5 provides a comprehensive comparison of GAT-HRL-Pred with comparison algorithms across six basic performance indicators. All values are means of 30 independent experiments, with standard deviations in parentheses.

Table 5. Comprehensive comparison of basic performance indicators across Algorithms

Algorithms	Weighted Latency (s) ↓	Normalized Energy Efficiency ↑	PAoI (s) ↓	Load Balance ↑	Task Success Rate (%) ↑	Content Freshness ↑
GAT-HRL-Pred (Ours)	1.61 (0.06)	0.93 (0.02)	31.2 (1.4)	0.91 (0.02)	96.8 (0.7)	0.91 (0.01)
DRL-VR	2.06 (0.09)	0.79 (0.03)	51.3 (2.4)	0.82 (0.03)	88.3 (1.2)	0.74 (0.02)
Cache-Opt	2.14 (0.11)	0.73 (0.04)	57.8 (3.1)	0.78 (0.04)	85.1 (1.5)	0.69 (0.03)
Greedy	2.59 (0.15)	0.61 (0.05)	68.2 (4.0)	0.65 (0.05)	77.4 (2.0)	0.58 (0.03)
Random	3.47 (0.21)	0.43 (0.07)	89.6 (5.8)	0.51 (0.06)	61.2 (2.8)	0.41 (0.04)

As shown in Table 5, regarding latency performance, GAT-HRL-Pred achieves a weighted average end-to-end latency of 1.61 seconds, representing a 21.0% reduction compared to the no-prediction baseline DRL-VR. The fundamental source of this improvement lies in the prefetching mechanism enabling IA-UAVs to complete content updates for high-probability grids before the user viewport arrives; when the user actually gazes at that region, the system no longer needs to experience the complete flight maneuver and image acquisition process, directly entering the rendering and transmission phase, significantly compressing the acquisition waiting component in response latency.

Regarding energy efficiency, GAT-HRL-Pred achieves a normalized system energy efficiency of 0.93, indicating that the prefetching mechanism does not bring net energy costs. The underlying reason is that after a prefetching hit, the subsequent response phase cancels the passive supplementary acquisition flights that would otherwise be mandatory, with saved flight energy exceeding the additional consumption of prefetching flights; furthermore, content pre-updated can be directly reused during the rendering phase, reducing repeated computation overhead.

Regarding freshness, GAT-HRL-Pred achieves a PAoI of 31.2 seconds, with content freshness scores improving from 0.86 to 0.91. The prefetching mechanism fundamentally reduces situations where local regions remain unupdated for long periods by actively maintaining the freshness of high-priority grids, effectively suppressing AoI peaks, and significantly improving the overall freshness of VR rendering content received by users.

Regarding load balance and task success rate, GAT-HRL-Pred achieves 0.91 and 96.8% respectively, both optimal among all deployable methods. Prefetching decisions provide forward-looking information for low-level scheduling, making IA-UAV flight path planning more predictable, reducing sudden flight maneuvers during emergency supplementary acquisitions, and further balancing the task load distribution across UAVs.

5. Conclusion

This paper proposes the GAT-HRL-Pred active prefetching framework, achieving a paradigm shift from passive response to active prefetching in VR content through the collaborative mechanism of user viewport prediction, graph attention-enhanced state representation, and hierarchical reinforcement learning. Experimental results demonstrate that the method reduces weighted end-to-end latency by 12.0%, compresses PAoI by 18.8%, and achieves a Prediction-guided Energy Efficiency Ratio (PEER) of 1.022, confirming that prefetching benefits can cover the additional flight maneuver energy consumption. Ablation studies further validate the irreplaceability of the three core components: cross-modal attention, supervised pre-training, and soft constraint scheduling. These results demonstrate that embedding user intent prediction into UAV collaborative scheduling decisions can effectively resolve the inherent contradiction between content freshness and system energy efficiency in wide-area dynamic scenarios, providing a scalable technical paradigm for delay-sensitive edge computing applications such as space-air-ground integrated architectures and emergency communications.

References

- [1] Alshowair, A., Bail, J., & AlSuwailem, F., (2024). Use of virtual reality exercises in disaster preparedness training: A scoping review. *SAGE Open Medicine*, 12, Article 20503121241241936.
- [2] Tham, J. (2024). Researching with virtual reality: Exploring the methodological affordances of VR for sociotechnical research and implications for technical and professional communication. *IEEE Transactions on Professional Communication*, 67(2), 192–210.
- [3] Xia, X., Fattah, S. M. M., & Babar, M. A. (2023). A survey on UAV-enabled edge computing: Resource management perspective. *ACM Computing Surveys*, 56(3), 1–36.
- [4] Javed, S., Hassan, A., & Ahmad, R., (2024). State-of-the-art and future research challenges in UAV swarms. *IEEE Internet of Things Journal*, 11(11), 19023–19045.
- [5] Yan, L., Wang, Q., & Zhao, J., (2024). Radiance field learners as UAV first-person viewers. In *European Conference on Computer Vision* (pp. 88–107). Cham: Springer Nature Switzerland.
- [6] Wang, Y., Ho, I. W. H., & Wang, Y. (2023). Real-time intersection vehicle turning movement counts from live UAV video stream using multiple object tracking. *Journal of Intelligent and Connected Vehicles*, 6(3), 149–160.
- [7] Dang, T. N., Manzoor, A., & Tun, Y. K., (2023). A contract-theory-based incentive mechanism for UAV-enabled VR-based services in 5G and beyond. *IEEE Internet of Things Journal*, 10(18), 16465–16479.
- [8] Yoo, S., Jeong, S., & Kim, J., (2024). Cache-assisted mobile-edge computing over space-air-ground integrated networks for extended reality applications. *IEEE Internet of Things Journal*, 11(10), 18306–18319.

- [9] Comşa, I. S., Molnar, A., & Tal, I., (2023). Improved quality of online education using prioritized multi-agent reinforcement learning for video traffic scheduling. *IEEE Transactions on Broadcasting*, 69(2), 436–454.
- [10] binti Burhanuddin, L. A., Liu, X., & Deng, Y., (2022). QoE optimization for live video streaming in UAV-to-UAV communications via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 71(5), 5358–5370.
- [11] Xiao, L., Ding, Y., & Huang, J., (2021). UAV anti-jamming video transmissions with QoE guarantee: A reinforcement learning-based approach. *IEEE Transactions on Communications*, 69(9), 5933–5947.
- [12] Ju, T., Li, L., & Liu, S., (2024). A multi-UAV assisted task offloading and path optimization for mobile edge computing via multi-agent deep reinforcement learning. *Journal of Network and Computer Applications*, 229, Article 103919.
- [13] Shi, H., Tian, Y., & Li, H., (2024). Task offloading and trajectory scheduling for UAV-enabled MEC networks: An MADRL algorithm with prioritized experience replay. *Ad Hoc Networks*, 154, Article 103371.
- [14] Zhang, H., Liang, H., & Ale, L., (2025). Attention-based deep reinforcement learning for joint trajectory planning and task offloading in AAV-assisted vehicular edge computing. *IEEE Transactions on Vehicular Technology*. Advance online publication.
- [15] Li, Z., Liang, X., & Liu, J., (2025). Optimizing mobile edge computing for virtual reality rendering via UAVs: A multi-agent deep reinforcement learning approach. *IEEE Internet of Things Journal*. Advance online publication.
- [16] Kang, H., Chang, X., & Mišić, J., (2023). Cooperative UAV resource allocation and task offloading in hierarchical aerial computing systems: A MAPPO-based approach. *IEEE Internet of Things Journal*, 10(12), 10497–10509.