QoE-Oriented Cooperative VR Rendering and Dynamic Resource Leasing in Metaverse

Nan Liu†, Tom H. Luan†, Yuntao Wang†, Yiliang Liu†, and Zhou Su†*
†School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, China
*Corresponding Author: zhousu@ieee.org

***Abstract*—The rise of the Metaverse has ushered in a new era of social networking, offering users deeply engaging spaces to connect and participate in social activities. However, rendering these virtual environments is resource-intensive. With many users accessing simultaneously and requiring diverse Metaverse services, optimizing Metaverse resources to deliver the best quality-of-experience (QoE) for users is a significant challenge. In this paper, we propose a cooperative virtual reality (VR) rendering and dynamic resource leasing mechanism to address this issue. Specifically, we first introduce a cooperative VR scene pre-rendering framework between users and Planets (*i.e.*, edge servers hosting users), and establish a new user QoE metric which considers both rendering delay and visual quality. We formulate the multidimensional rendering resources (*e.g.*, GPU, CPU, and outbound bandwidth) leasing problem between Planets and users as a double-layer decision problem, and devise a hybrid action multi-agent reinforcement learning-based dynamic resource auction mechanism to efficiently allocate limited resources of Planets in a distributed and adaptive manner. Extensive simulations demonstrate that our proposed scheme outperforms the representatives in user QoE and resource utilization efficiency. Particularly, the proposed scheme shows at least an 18-fold improvement in QoE over other schemes, demonstrating its capability in providing immersive Metaverse experiences.**

***Index Terms*—Metaverse, cooperative VR rendering, QoE, MARL, dynamic resource leasing.**

I. Introduction

THE Metaverse constructs a three-dimensional virtual shared space that can enable users to engage in immersive social activities, offering a revolutionary model for future social networks [2]–[4]. Existing Metaverse applications, such as virtual reality (VR) games, immersive workplaces, and virtual music concerts, demonstrate the immense value and potential, and have garnered considerable interest from both academia and industry. In 2024, OpenAI unveiled Sora, an artificial intelligence-generated content (AIGC) model capable of creating immersive Metaverse scenes from text instructions [5]. The virtual concert featuring Travis Scott in the social Metaverse game Fortnite attracted more than 12 million players [6], an unprecedented number for a live concert.

In Metaverse activities, users receive various sensory stimuli, making the human-centric quality-of-experience (QoE) an important indicator to measure the performance of Metaverse applications. The QoE of users is mainly influenced by the quality of VR scenes, which includes the richness of visual details and the smoothness of scene transitions [7]. AIGC technology offers a promising rendering method, which can improve the visual quality by creating realistic and imaginative VR scenes. However, existing wearable VR devices typically lack sufficient multidimensional rendering resources (*e.g.*, GPU and CPU) to support high visual quality and low-latency generation of intensive VR scenes [8]–[10], making the utilization of resources from cloud and edge servers crucial. Moreover, the Metaverse system may involve a massive number of users simultaneously, causing competition for servers' resources among them. Therefore, how to efficiently utilize multidimensional rendering resources to enhance user QoE while ensuring the scalability of the Metaverse system is a challenging issue.

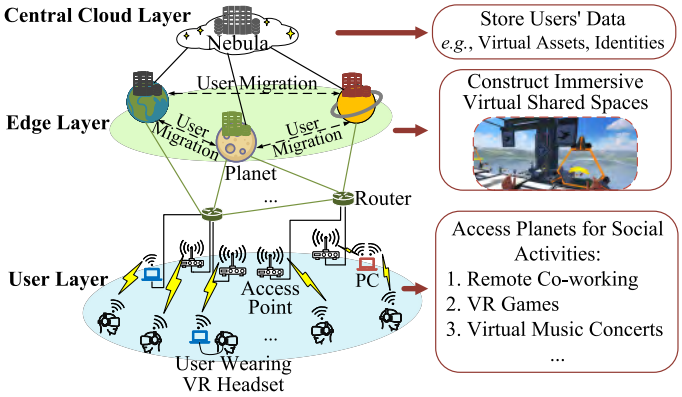

Fig. 1: Three-layer network architecture of Metaverse.

The cloud-edge-end collaborative architecture represents a promising QoE enhancement solution, where Metaverse users can render high-quality VR scenes in cooperation with cloud and edge servers. For example, in Unity apps such as Viking and Nature, the foreground interaction of a VR scene is rendered locally on the user's device, while the background environment can be rendered by the edge server [11]. Fig. 1 shows a typical cloud-edge-end collaborative architecture of Metaverse system consisting of three layers. The top layer is a central cloud server, referred to as *Nebula*, storing users' data, such as virtual assets and identities. The middle layer comprises edge servers, referred to as *Planets*, which are located in different regions of Internet and can use their resources to assist users in rendering virtual spaces. The bottom layer comprises Metaverse users, referred to as *avatars*, who connect to Metaverse through terminals such as VR headsets and head-mounted displays. Users from different locations can participate in virtual social activities, such as multi-player VR

A preliminary version of this paper appears in IEEE MSN 2023 [1].

gaming and remote co-working, by logging into Planets [12].

Under the collaborative architecture of Metaverse, users can lease rendering resources from Planets in an on-demand manner to improve the rendering quality of VR scenes. However, the rendering resources of each Planet are limited, and a Planet may lease its resources to multiple users to host parallel VR social activities. Therefore, it necessities a dynamic and efficient edge rendering resource leasing mechanism for users' QoE enhancement in the Metaverse. In the literature, various research efforts have been made in this field using fractional programming [13], matching game and hedonic coalition game [14], Stackelberg game [15], auction theory [9], contract theory [16], reinforcement learning (RL) [17], *etc*. However, to practically deploy the cooperative rendering and resource leasing mechanism in the Metaverse, it still faces a series of critical challenges:

- *QoE dilemma in visual quality and rendering delay.* Achieving higher visual quality with equal resources implies longer rendering delay, which in turn decreases users' QoE. The two key factors affecting QoE (*i.e.*, visual quality and rendering delay) are in conflict with each other, thus necessitating a well-designed QoE model that comprehensively considers the both aspects.
- *Distributed discrete-continuous hybrid decision-making.* For the user layer, each user needs to make decisions regarding both discrete action (*i.e.*, which Planet to access) and continuous actions (*i.e.*, the rendering quality of VR scenes and the amount of resources required). For the Planet layer, each Planet needs to determine the prices of rendering resources and then lease them to the users it hosts. The double-layer distributed optimization of discrete-continuous hybrid actions remain unexplored in existing works on cooperative VR rendering.
- *High adaptability to dynamic Metaverse environment.* As Metaverse social activities in Planets evolve and users can dynamically join and depart, the users' demands for visual quality and rendering resources is time-varying. The cooperative rendering and resource leasing mechanism should adapt to the dynamic Metaverse environment and maximize the long-term benefits for involved entities.

To address the aforementioned problems, we first introduce a cooperative pre-rendering framework between Planets and users. Under this framework, a new QoE model which serves as the foundation for resolving the QoE dilemma is proposed. Then, we devise a multi-agent HyAR TD3 (MA-HyAR-TD3) algorithm for users to distributively and adaptively attain optimal discrete-continuous hybrid actions regarding Planet selection, rendering quality, and resource leasing quantity. In MA-HyAR-TD3, the hybrid action representation (HyAR) component is to construct a representation space to allow RL under discrete-continuous hybrid action spaces, while the twin-delayed deep deterministic policy gradient (TD3) component is to adapt to the Metaverse dynamics and it trains policies in the representation space. Finally, we devise a rendering resource auction mechanism, where Planets use the resource pricing scheme to lease their limited resources to users. The main contributions of this paper are summarized as follows:

- *Novel QoE model.* We introduce a Planet-user cooperative pre-rendering framework and establish a novel delay-sensitive and interest-aware user QoE model under this framework. The pre-rendering delay cannot exceed user's mobility time, which serves as the basis for users to experience smoothly. Also, the QoE is influenced by the visual quality of VR scenes, especially when the user has varying degrees of interests in distinct virtual objects. Our QoE model comprehensively incorporates both pre-rendering delay and visual quality of VR scenes.
- *Distributed and adaptive algorithm.* We formulate the dynamic rendering resource leasing problem with discrete-continuous hybrid decision variables between Planets and users as a double-layer distributed auction problem, and propose a multi-agent RL (MARL)-based distributed and adaptive approach to solve it.
- *Extensive experimental evaluations.* We evaluate the performance of the proposed scheme through extensive experiments. The results validate that the proposed scheme is superior to the representatives in enhancing user QoE and resource utilization efficiency.

The remainder of this paper is organized as follows. Sect. II reviews the related works. Sect. III characterizes the logic of design and the system model, which includes the network model, the cooperative pre-rendering model, and the VR QoE model. Sect. IV formulates the dynamic rendering resource leasing problem between Planets and users as an auctioneer-bidder double-layer decision problem. Sect. V presents the MA-HyAR-TD3 algorithm for dynamic rendering resource auction. The numerical simulations are shown in Sect. VI. The paper is concluded in Sect. VII.

II. Related Work

In this section, we review the related works on QoE-driven edge-empowered VR scene rendering mechanisms (in Sect. II-A) and resource allocation methods in Metaverse systems (in Sect. II-B).

A. QoE-Driven Edge-Empowered VR Scene Rendering

Recently, there are increasing studies focusing on users' QoE enhancement in VR scene rendering, by harnessing the multidimensional rendering resources from edge servers (*i.e.*, Planets). Lai *et al.* [11] use the edge server to pre-render background panoramas, which are then pre-fetched by the Metaverse user, in order to improve the visual quality of VR scenes. Chen *et al.* [18] utilize the pixel similarity of the same VR scene observed from different angles to reduce the rendering workload of the background. Zhang *et al.* [19] only render the VR scene within the user's field of view (FoV) on the edge server. Wang *et al.* [20] propose a saliency-driven scheme to enhance the QoE under the limited network bandwidth. Xu *et al.* [21] propose an edge-device collaborative real-time VR rendering framework to minimize motion-to-photon (MTP) latency. Meng *et al.* [22] enhance vehicular Metaverse users' QoE by optimizing the video buffer under the constraints of rendering and transmission delay. Chen *et al.* [23] optimize the edge server's resource management to

reduce the average delay between players in VR games. Du *et al.* [16], [24] propose a VR scene rendering approach based on the user-object attention values to achieve higher QoE using equal rendering resources.

Among these studies, [11], [18]–[20] mainly focus on the single user scenario. Distinguished from them, this paper investigates the VR scene rendering in the multi-user Metaverse scenario. [21]–[23] study the impact of delay on the user QoE, while [24] and [16] study the user's subjective evaluation on the visual quality. However, these works do not establish a VR QoE model that comprehensively considers both delay and visual quality, as well as the conflicting nature between these two aspects.

B. Resource Allocation in Metaverse System

Various recent research efforts have been reported on optimizing the resource allocation in Metaverse systems. Zhao *et al.* [13] propose a new fractional programming method for optimizing the utility-cost ratio, aiming to achieve the optimal communication and computation resource allocation. Zhang *et al.* [25] study the reconfigurable intelligent surface-assisted unmanned aerial vehicle (UAV) networks for Metaverse, which aims to optimize the UAV transmit power for maximized energy efficiency. Jiang *et al.* [14] develop a joint resource management and service assignment mechanism in Metaverse based on matching game and hedonic coalition formation game to maximize users' QoE. Jiang *et al.* [15] investigate a hierarchical three-stage Stackelberg game approach to determine optimal pricing strategies for computing resources on edge servers. Chu *et al.* [17] develop a deep RL (DRL) based self-learning algorithm to manage multi-tier computing resources. Long *et al.* [26] model the multi-user Metaverse environment as a graph and uses the multi-agent soft actor–critic method to determine the resource usage strategy for each user. Hieu *et al.* [27] design a DRL-based hybrid training method to optimally manage radio and computing resources for Metaverse users.

Among these studies, [13]–[15], [25] mainly focus on one-shot resource allocation and do not consider system dynamics, which is studied in this paper. [17], [26], [27] propose the DRL based resource allocation schemes to adapt to the system dynamics. However, because they are not specifically designed for the discrete-continuous hybrid action spaces of agents, these schemes are not applicable to our problem.

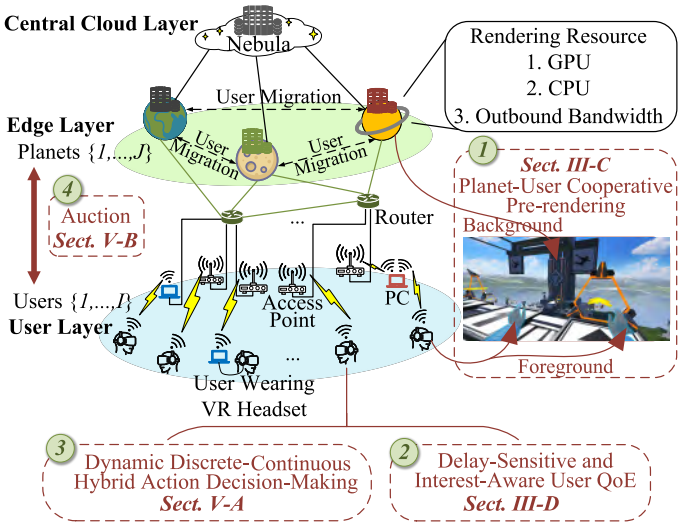

Fig. 2: Network model and logic of design.

III. System Model and Logic of Design

This section introduces the system model and the logic of design. System model comprises the network model (in Sect. III-A), the cooperative pre-rendering model (in Sect. III-C), and the VR QoE model (in Sect. III-D). The design logic of cooperative VR rendering and dynamic resource leasing mechanism is presented in Sect. III-B. The key notations are summarized in Table I.

A. Network Model

Fig. 2 illustrates the investigated cloud-edge-end collaborative Metaverse network architecture, consisting of a central Nebula server, a set of $\mathcal{J} = \{1, \ldots, J\}$ Planets, and a set of $\mathcal{I} = \{1, \ldots, I\}$ users. The central Nebula server stores users' basic information such as avatars and virtual assets. Users can

TABLE I: A Summary of Key Notations

Notation	Description
$\mathcal{J}$, $\mathcal{I}$, $\mathcal{H}$	Sets of Planets, users, and rendering resources
$d_{i,j}$	Network distance between user i and Planet j
V_j^h	Amount of rendering resource h owned by Planet j
$I_{i,n}$	User i's interest in virtual object n
$r_{i,n}$	Rendering resolution density requested by user i for object n
k_i	Compression ratio of the background panorama requested by user i
r_{th}, k_{th}	Minimum acceptable resolution density and compression ratio
$u_{i,n}$	Total surface area of object n in all neighbor panoramas of user i
$g_{i,j}$, $f_{i,j}$, $w_{i,j}$	Amount of GPU, CPU, and outbound bandwidth leased by user i from Planet j
$v_{i,t}$	The Planet that user i chooses to access in time round t
$p_{j,t}^h$	Price of one unit of resource h announced by Planet j in time round t
$b_{i,t}^h$	User i's bid for resource h in time round t
q_j^h	Operational expenditures incurred by Planet j for utilizing one unit of resource h

be located in different geographical locations. The network distance between each user i and each Planet j is denoted by $d_{i,j}$, indicating the number of routers along the shortest data transmission path between them. Users can attend various types of Metaverse activities in the system, such as remote coworking, VR gaming, and VR concerts. Due to limitations of local computational resources, users need to lease resources from Planets to complete rendering tasks in Metaverse activities. Planets are deployed in the edge of the Metaverse network in different regions and are closer to the users than the Nebula server. Each Planet obtains basic information of users that access it from the central Nebula server and renders three-dimensional shared immersive virtual spaces for users. Each Planet has a set $\mathcal{H}$ of H types of rendering resources such as GPU, CPU and outbound bandwidth. We denote the capacity vector $\mathbf{V}_j = [V_j^1, \ldots, V_j^H]$ to represent the amount of rendering resources owned by Planet j.

B. Logic of Design

Under the Metaverse network architecture, to efficiently and dynamically utilize servers' resources for user QoE enhancement, we design the cooperative VR rendering and dynamic resource leasing mechanism. The logic of design is illustrated in Fig. 2 and described in the following four successive phases:

1) *Planet-user cooperative pre-rendering.* A complete VR scene can be divided into *foreground interaction* and *background environment*, as described in Sect. III-C. The foreground interaction is rendered locally on the user's device, while the background environment is pre-rendered by the Planet and is transmitted to the user's device in time to combine with the foreground to form a complete scene frame.
2) *User QoE characterization.* In the cooperative pre-rendering framework, the pre-rendering delay of the background cannot exceed the user's mobility time; otherwise, it will cause visual stuttering. Moreover, the user has varying degrees of interest in different virtual objects, which influences the subjective evaluation of visual quality. The QoE model designed in Sect. III-D comprehensively incorporates both aspects, thus making it both delay-sensitive and interest-aware.
3) *Dynamic discrete-continuous hybrid action decision-making.* User dynamically makes decisions regarding both discrete action (*i.e.*, which Planet to access) and continuous actions (*i.e.*, the rendering quality of VR scenes and the amount of resources required). In Sect. V-A, we design a discrete-continuous hybrid action MARL algorithm for dynamic decision-making.
4) *Rendering resource leasing mechanism.* A Planet may lease its limited resources to multiple users simultaneously, leading to competition for resources among users. In Sect. V-B, we design the resource leasing mechanism as an auction, where Planets lease their limited resources to users according to the received bids.

C. Cooperative Pre-Rendering Model

To construct high-quality VR scenes, we present a Planet-user cooperative pre-rendering model, by harnessing the computational resources of distributed Planets for alleviating rendering burden on users' end devices. According to [11], a complete frame of a VR scene can be divided into *foreground interaction* and *background environment*, as shown in Fig. 3, where the blue semi-transparent area signifies foreground interaction, while the rest represents the background environment.

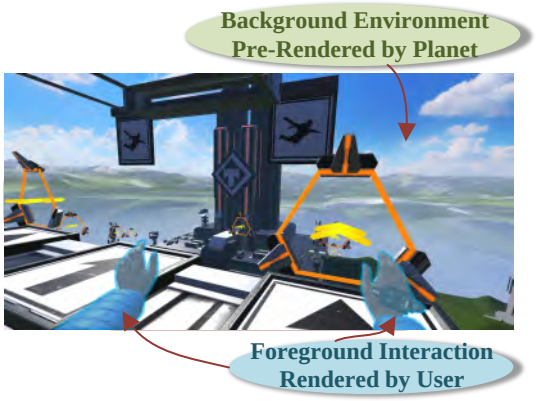

Fig. 3: An example of the division of foreground interaction and background environment in a scene from the VR game Population One.

- The foreground interaction refers to the actions of the Metaverse user, as well as her interactions with virtual objects and other avatars. It demands less rendering load and has higher latency requirements. Therefore, the foreground interaction can be rendered locally on the user's device.
- The background environment can be regarded as the "map" where users immersively engage in Metaverse social activities. Other users' activities can also impact the status of background environment. The background environment requires a larger rendering area, richer visual details, and consequently, a heavier rendering load than the foreground interactions. Therefore, it can be rendered on the Planet with powerful computational capabilities.

The background panorama frame rendered by Planet is then transmitted to the user's device. The Metaverse user, based on her current head pose, cut out the background environment within the FoV from this panorama and combine it with locally rendered foreground interaction to form a complete VR scene frame. The local foreground interaction rendering can be synchronized with the background environment rendering on the Planet, so the time required for rendering a complete scene frame is given by:

$$T_{\text{co\_rendering}} = \max(T_{\text{Planet\_env}}, T_{\text{user\_intr}}) + T_{\text{integrate}}, \quad (1)$$

where the process of background environment rendering can be divided into five processes: 1) the user sends a request to the Planet to render the background of her current location (including requirements for rendering quality), 2) the Planet renders the background panorama, 3) the Planet encodes the background panorama, 4) transmission, and 5) the user de-

codes the background panorama. Therefore, the time required for rendering the background panorama is given by:

$$T_{\text{Planet\_env}} = T_{\text{request}} + T_{\text{render}} + T_{\text{encode}} + T_{\text{transmit}} + T_{\text{decode}}. \quad (2)$$

Leverage mobility to pre-render panoramas. Rendering a background environment panorama involves the above five sequential processes, which requires considerable time and results in lower frame rate [11]. Moreover, it is possible that by the time users receive the panoramas, they may have already moved to a different location, significantly deteriorating their QoE. To address the aforementioned issues, we employ the pre-rendering approach and leverage user's mobility in VR activities to mask the first four terms in Eq. (2), thereby reducing the Planet-user cooperative rendering time $T_{\text{co\_rendering}}$.

Specifically, we discretize the virtual world into dense grid points, as illustrated in Fig. 4, where the scene observed by the user at a particular location aligns with the scene at the nearest grid point. Due to the unpredictable nature of user movement in VR activities, it is uncertain which grid point the user will visit next. Therefore, upon reaching a certain grid point, the user immediately requests Planet to render the background panoramas of all neighbor grid points (neighbor grid points are defined as those situated at a distance of 1 density from the current grid point, and we assume that during any time step, the user can move at most 1 density). Notably, these requesting, rendering, encoding, and transmission tasks should be completed before the next time step. Therefore, when moving to a neighbor grid point, the user only needs to decode, clip, and combine it with the locally rendered foreground interaction to complete the rendering of an entire scene frame. Then the time of Planet-user cooperative pre-rendering is given by:

$$\begin{aligned} &T_{\text{co\_pre\_rendering}} = \max(T_{\text{decode}}, T_{\text{user\_intr}}) + T_{\text{integrate}}, \\ &s.t. \quad T_{\text{request}} + T_{\text{render}} + T_{\text{encode}} + T_{\text{transmit}} \leq T_{\text{unit}}, \end{aligned} \quad (3)$$

where T_{unit} denotes the time for a user to move from a grid point to a neighbor point.

Illustrating example of pre-rendering. An example of pre-rendering is shown in Fig. 4. At initialization time t_0, the user is positioned at grid point 0, where all background panoramas of its neighbor grid points 1-6 have been well rendered, encoded, and transmitted to the user. At time t_1, the user moves to point 3 and promptly requests Planet to render background panoramas for points 7-9. Meanwhile, the

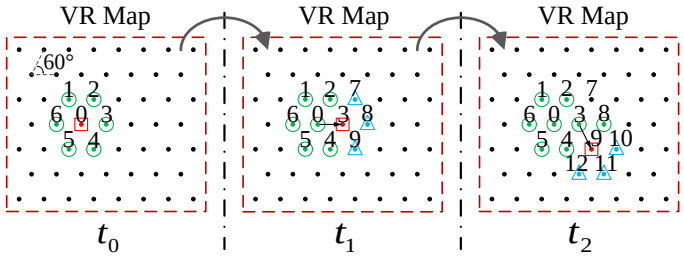

Fig. 4: Leverage mobility to pre-render panoramas.

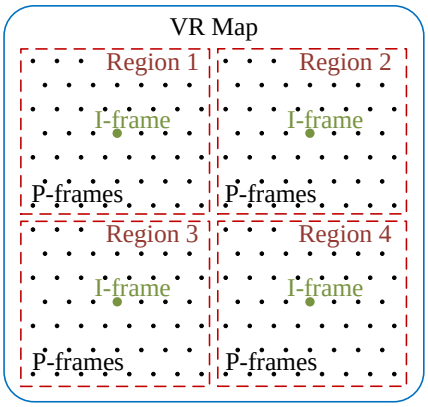

Fig. 5: An example of regions within a VR map, as well as I-frame and P-frames within each region.

user decodes the panorama of point 3, renders foreground interaction, and then combines the clipped panorama with the foreground interaction to generate a complete scene frame. At time t_2, the user moves to point 9, then the user will filter the pre-rendered background panoramas from the previous time step based on a quality threshold. Panorama in point 8 with quality score exceeding the threshold will be stored locally on the user's device for future use, while that in point 7 below the threshold will be discarded. Consequently, at time t_2, the user will only need to request Planet to render background panoramas for neighboring grid points 10-12.

VR map segmentation and compression. In the VR map, grid points are densely packed, and the panoramas of nearby points exhibit high similarity. We can utilize this spatial similarity by employing video coding methods to achieve high-quality compression of panoramas, thus reducing their transmission time over the network. Specifically, we divide the VR map into multiple regions, as illustrated in Fig. 5. The panorama of the central point in each region is encoded as an I-frame, which can be independently decoded. The panoramas of other grid points within the region are encoded as P-frames, requiring the information from the I-frame for decoding. We assume that users store all I-frames and basic information of P-frames (*i.e.*, the surface area of each virtual object) locally before engaging in VR activities. Additionally, the value of density needs to be properly designed: too large a density may cause discontinuities when users move, while too small a density may result in excessive rendering workload. According to [28], an appropriate density is equivalent to 2 cm of the real world. For example, if the user's movement speed in the VR world is 1 m/s, then $T_{\text{unit}} = 20$ ms.

D. Delay-Sensitive and Interest-Aware VR QoE Model of User

Based on the Planet-user cooperative pre-rendering model, the QoE of users in VR activities is primarily influenced by the following two aspects:

- *Objective pre-rendering delay.* We use pre-rendering delay $T_{\text{pre-render}}$ to represent $T_{\text{request}} + T_{\text{render}} + T_{\text{encode}} + T_{\text{transmit}}$, and it should satisfy $T_{\text{pre-render}} \leq T_{\text{unit}}$, which serves as the basis of the Planet-user cooperative pre-rendering. Otherwise, users may experience visual stut-

tering, leading to a reduction in their QoE. As such, the VR QoE model should be *delay-sensitive*.

- *Users' subjective evaluation of visual quality.* Due to user's varying levels of interests in different objects within VR scenes, objects with lower levels of interests are rendered at a reduced quality level, albeit above a defined threshold, while objects with higher levels of interests receive higher quality rendering. This not only ensures users immerse themselves, but also enhances their subjective evaluation. As such, the VR QoE model should be *interest-aware*.

To comprehensively consider pre-rendering delay and user's subjective evaluation of visual quality, we use the Weber-Fechner Law to construct the VR QoE model, which can be expressed as the *connection coefficient* multiplied by the *logarithm of stimulus intensity* [29], [30]. In the VR QoE model, the *connection coefficient* corresponds to pre-rendering delay, named as objective key performance indicator (KPI). Meanwhile, *stimulus intensity* corresponds to visual quality, with its logarithmic form representing user's subjective perception of visual quality, named as subjective evaluation. In the following, we formulate the subjective evaluation and objective KPI to model user's VR QoE.

1) ***Subjective evaluation decided by visual quality.*** In the pre-rendering process, two main factors influence the visual quality of background panoramas: *rendering resolution* and *compression ratio*. The resolution determines the level of detail in the panoramas, while the compression ratio determines how much original visual information is retained during encoding. Additionally, the user has varying levels of interests in different objects within a VR scene, guiding her gaze and thus impacting her subjective evaluation of visual quality. Rendering objects of higher interests at higher resolutions enables better quality rendering in areas where users are more attentive, consequently enhancing their subjective evaluation of background panoramas. The subjective evaluation of user i can be modeled as follows,

$$QoE_i^{\text{sub}} = \sum_{n=1}^{N_i} I_{i,n}[\kappa \ln(\frac{r_{i,n}}{r_{\text{th}}}) + \ln(\frac{k_i}{k_{\text{th}}})], \tag{4}$$

where N_i denotes the number of objects in the background, $I_{i,n}$ denotes user i's interest in object n, which can be obtained by machine learning-based methods [16]. $r_{i,n}$ denotes the rendering resolution density requested by user i for object n, and k_i denotes the compression ratio of the background panorama requested by user i. r_{th} and k_{th} represent the minimum acceptable resolution density and compression ratio, respectively. κ is an adjustment coefficient.

2) ***Objective KPI decided by pre-rendering delay.*** We design the objective KPI QoE^{ob} as a monotonically decreasing function of $T_{\text{pre-render}}$. When $T_{\text{pre-render}} < T_{\text{unit}}$, $QoE^{\text{ob}} > 0$ and $T_{\text{pre-render}} > T_{\text{unit}}$, we have $QoE^{\text{ob}} < 0$. If user i requests Planet j to pre-render the panoramas, then the objective KPI can be modeled as follows,

$$QoE_i^{\text{ob}} = \ln(\frac{T_{\text{unit}} + 1}{T_{i,j}^{\text{pre-render}} + 1}). \tag{5}$$

Next, we analyze the four components of $T_{i,j}^{\text{pre-render}}$ in Eq. (3) separately.

Request time $T_{i,j}^{\text{request}}$. Request time denotes the time it takes for a rendering request to travel from the user's device to the Planet. It is directly proportional to the network distance between the user and the Planet (*i.e.*, d_{ij}) and can be expressed as follows,

$$T_{i,j}^{\text{request}} = \beta' d_{ij}, \tag{6}$$

where β' is the time taken for the request to pass through a single router.

Rendering time $T_{i,j}^{\text{render}}$. Planet uses its GPU resource to render background panoramas. For each user, the total number of pixels to be rendered equals the sum of the pixels of all objects in the panoramas of neighbor grid points. The number of pixels of object n, for example, is the product of its surface area $u_{i,n}$ and the pixel density $r_{i,n}$ assigned to it by user i. Rendering time can be expressed as follows,

$$T_{i,j}^{\text{render}} = \frac{\sum_{n=1}^{N_i} u_{i,n} r_{i,n}}{g_{i,j}}, \tag{7}$$

where $g_{i,j}$ denotes the amount of GPU resource leased by user i from Planet j, measured in pixels/ms.

Encoding time $T_{i,j}^{\text{encode}}$. After rendering, Planet uses its CPU resource to encode the panoramas. Given the amount of CPU resource, encoding time is directly proportional to the size of rendering task [7] (*i.e.*, total number of pixels). Therefore, encoding time can be expressed as follows,

$$T_{i,j}^{\text{encode}} = \frac{c_j^{\text{CPU}} \sum_{n=1}^{N_i} u_{i,n} r_{i,n}}{f_{i,j}}, \tag{8}$$

where c_j^{CPU} denotes the number of CPU cycles needed to encode a single pixel in Planet j, and $f_{i,j}$ denotes the amount of CPU resource leased by user i from Planet j, measured in cycles/ms.

Transmission time $T_{i,j}^{\text{transmit}}$. The encoded panoramas need to be transmitted from the Planet to the user. The transmission time is divided into *propagation time* and *sending time*. Propagation time refers to the time taken for the first byte of the encoded panoramas to propagate through the network, which is consistent with the request time $T_{i,j}^{\text{request}}$. Sending time depends on the bandwidth of the transmission path. In this paper, it is assumed that the outbound bandwidth of Planet is the bottleneck bandwidth in the transmission path because a Planet may simultaneously handle multiple users' rendering tasks, causing competition for bandwidth resources. Therefore, transmission time can be expressed as follows,

$$T_{i,j}^{\text{transmit}} = \beta' d_{ij} + \frac{k_i \cdot h \sum_{n=1}^{N_i} u_{i,n} r_{i,n}}{w_{i,j}}, \tag{9}$$

where h denotes the size of a single pixel (in bits), and $w_{i,j}$ denotes the amount of outbound bandwidth resource leased by user i from Planet j.

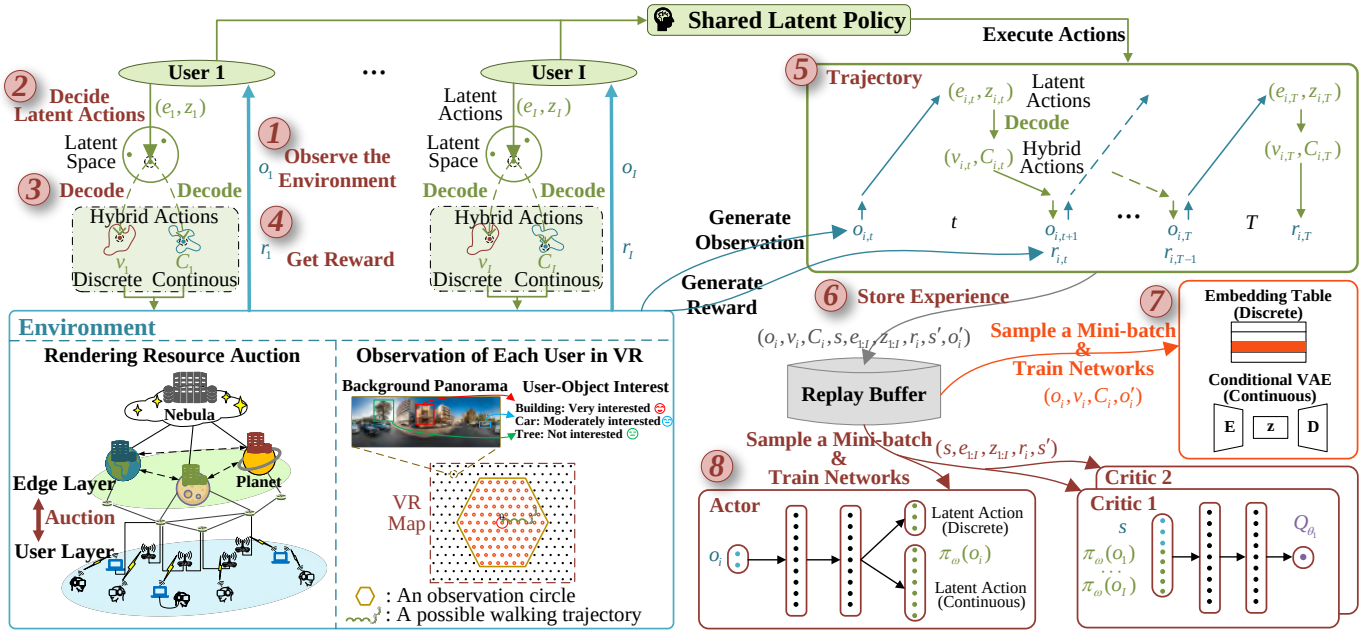

Fig. 6: Overall structure of the auctioneer-bidder double layer decision problem and the proposed multi-agent HyAR TD3 for rendering resource auction mechanism.

Pre-rendering delay $T_{i,j}^{\text{pre-render}}$. By adding up the time of requesting, rendering, encoding and transmission, we obtain the expression for pre-rendering delay:

$$T_{i,j}^{\text{pre-render}} = \beta d_{ij} + \frac{\sum_{n=1}^{N_i} u_{i,n} r_{i,n}}{g_{i,j}} + \frac{c_j^{\text{CPU}} \sum_{n=1}^{N_i} u_{i,n} r_{i,n}}{f_{i,j}} + \frac{k_i \cdot h \sum_{n=1}^{N_i} u_{i,n} r_{i,n}}{w_{i,j}}, \tag{10}$$

where $\beta = 2\beta'$.

Finally, we define the comprehensive VR QoE model that takes into account both the pre-rendering delay and subjective evaluation of visual quality, when user i accesses Planet j:

$$QoE_i = \ln\left(\frac{T_{\text{unit}} + 1}{T_{i,j}^{\text{pre-render}} + 1}\right) \times \sum_{n=1}^{N_i} I_{i,n}\left[\kappa \ln\left(\frac{r_{i,n}}{r_{\text{th}}}\right) + \ln\left(\frac{k_i}{k_{\text{th}}}\right)\right]. \tag{11}$$

IV. Problem Formulation of Dynamic Rendering Resource Leasing

In this section, we formulate the dynamic rendering resource leasing problem between Planets and users as an auctioneer-bidder double-layer decision problem, with Metaverse users acting as bidders and Planets as auctioneers.

1) For each user (*i.e.*, bidder), she needs to lease rendering resources from the Planet to construct immersive virtual background environments for VR activities. Based on the current prices of all Planets' resources and the surrounding background environment, each user should determine the following actions: i) which Planet to connect to, ii) the resolution density of each object in the background, iii) the compression ratio for encoding, iv) the amount of GPU, CPU and outbound bandwidth resources leased from the Planet. Then, each user sends bids for resources to the corresponding Planet. Since each user makes decisions in a decentralized manner and cannot observe the rendering demands of other users, we formulate the problem of each user as a *Decentralized Partially Observable Markov Decision Process (Dec-POMDP) with discrete time rounds* (in Sect. IV-A).

2) For each Planet (*i.e.*, auctioneer), after receiving the bids from users, it needs to design a pricing mechanism to effectively trade its limited rendering resources by updating the prices. We formulate the problem of each Planet as a *resource pricing problem* (in Sect. IV-B).

In the rest part of this section, the problem formulations of users and Planets are provided respectively. Overall, the general structure of this section and Sect. V is shown in Fig. 6.

A. Bidder layer: formulation as a Dec-POMDP

The time required for each user to make decisions increases the difficulty of meeting the constraint $T_{\text{pre-render}} \leq T_{\text{unit}}$. To decrease the frequency of decision-making and allow ample time for each decision-making process, we implement the following two update rules in the cooperative pre-rendering model in Sect. III-C:

- *Rule 1:* each user makes a pre-rendering decision at the beginning of every round (where 1 round = $X \cdot T_{\text{unit}}$), then proceeds with each step (where 1 step = 1 T_{unit}) based on the decision made in that round;
- *Rule 2:* during each step, the user requests the Planet to render panoramas of grid points within a distance of D densities (*e.g.*, when $D = 2$, the user requests panoramas not only of neighbor points but also of the neighbors' neighbor points).

Next, we give the definition of the Dec-POMDP as a tuple $< \mathcal{I}, \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma >$, and its detailed components are outlined below.

Observation and State. User i's observation at round t can be expressed as follows,

$$o_{i,t} = [U_i^t, I_i^t, P_t^{\text{GPU}}, P_t^{\text{CPU}}, P_t^{\text{Bandwidth}}, D_i^t] \in \mathcal{O}, \tag{12}$$

which comprises the following two parts:

1) Internal observation. The internal observation includes the surface areas $U_i^t = \{u_{i,1}^t, ..., u_{i,N}^t\}$ of all objects within the observation circle, as well as the user's interests $I_i^t = \{I_{i,1}^t, ..., I_{i,N}^t\}$ among the objects. The observation circle means the range of all points the user could potentially visit in this round, with a radius of X. The environment segment in Fig. 6 illustrates an observation circle with a radius $X = 5$.

2) External observation. The external observation includes the prices of all Planets' GPU resources $P_t^{\text{GPU}} = \{p_{1,t}^{\text{GPU}}, ..., p_{J,t}^{\text{GPU}}\}$, CPU resources $P_t^{\text{CPU}} = \{p_{1,t}^{\text{CPU}}, ..., p_{J,t}^{\text{CPU}}\}$ and bandwidth resources $P_t^{\text{Bandwidth}} = \{p_{1,t}^{\text{Bandwidth}}, ..., p_{J,t}^{\text{Bandwidth}}\}$, as well as the network distances $D_i^t = \{d_{i,1}^t, ..., d_{i,J}^t\}$ between the user and all Planets.

The environment state describes the observations of all users, which can be expressed as follows,

$$s_t = \{o_{1,t}, ..., o_{I,t}\} \in \mathcal{S}. \tag{13}$$

Action. User i's action at round t can be expressed as follows,

$$a_{i,t} = [v_{i,t}, R_i^t, k_{i,t}, g_{i,t}, f_{i,t}, w_{i,t}] \in \mathcal{A}, \tag{14}$$

which comprises the following two parts:

1) Discrete action. The discrete action $v_{i,t}$ represents the Planet that user i chooses to access in round t.

2) Continuous action. The continuous action $C_{i,t}$ includes the resolution densities $R_i^t = \{r_{i,1}^t, ..., r_{i,N}^t\}$ of all objects for rendering, compression ratio $k_{i,t}$ for encoding, as well as the amount of GPU resource $g_{i,t}$, CPU resource $f_{i,t}$ and outbound bandwidth resource $w_{i,t}$ leased from the accessed Planet.

State Transition. At round t, each user observes $o_{i,t}$, and executes action $a_{i,t}$ based on the policy $\mu(o_{i,t})$. Then, each user calculates the bids according to the following formulas:

$$b_{i,t}^{\text{GPU}} = p_{j,t}^{\text{GPU}} \cdot g_{i,t}, \tag{15}$$

$$b_{i,t}^{\text{CPU}} = p_{j,t}^{\text{CPU}} \cdot f_{i,t}, \tag{16}$$

$$b_{i,t}^{\text{Bandwidth}} = p_{j,t}^{\text{Bandwidth}} \cdot w_{i,t}, \tag{17}$$

and sends them to the corresponding Planet. After receiving the bids, each Planet updates the resources' prices to $p_j'^{\text{GPU}}$, $p_j'^{\text{CPU}}$, $p_j'^{\text{Bandwidth}}$ (updated prices will be the observations of users in the next round $t+1$), and calculates the actual amount of resources that can be allocated to each user as $g_{i,t}'$, $f_{i,t}'$ and $w_{i,t}'$. Then, each user uses these resources to engage in VR activities until round $t+1$. The environment state in round $t+1$ can be calculated by state transition function $\mathcal{T}$ as follows,

$$s_{t+1} = \mathcal{T}(s_t, a_{1:I,t}, m_{1:I,t}), \tag{18}$$

which is influenced by the current state s_t, all users' actions $a_{1:I,t}$ and environment stochasticity $m_{1:I,t}$. $m_{1:I,t}$ represents the mobility of users in round t. Specifically, round t consists of X steps, during each of which the user randomly chooses to move to one of the neighbor grid points or remain stationary. Therefore, at the beginning of the next round $t+1$, the user's position is random, and the surface area of each object observed by the user is also random.

Reward. The reward each user receives in each round is the difference between her QoE and the total payment. The QoE of each user in one round is the average QoE over X steps. The total payment includes the leasing fee of rendering resources and the bandwidth fee paid to the routers. In general, the reward of user i at round t is expressed as follows,

$$\begin{aligned} r_{i,t} = &\frac{\sum_{x=1}^{X} QoE_i^x}{X} - \rho(p_{j,t+1}^{\text{GPU}} \cdot g_{i,t}' + p_{j,t+1}^{\text{CPU}} \cdot f_{i,t}' \\ &+ p_{j,t+1}^{\text{Bandwidth}} \cdot w_{i,t}') - x \cdot d_{i,j}, \end{aligned} \tag{19}$$

where ρ is an adjustment coefficient, and x denotes the fee paid to a single router.

Objective. The observation-action-state transition process continues for each round until the episode (T rounds) ends. Each user aims to discover an optimal policy $\mu(o_{i,t})$ which maximizes the cumulative discounted reward:

$$G_i = \sum_{t=0}^{T} \gamma^t r_{i,t}, \tag{20}$$

where $\gamma \in [0, 1]$ denotes the discount factor, which determines the importance of the future rewards.

B. Auctioneer layer: formulation as a pricing problem

Each Planet utilizes its rendering resources to construct immersive virtual shared spaces for multiple users to engage in various social Metaverse activities. Users should pay for leasing the rendering resources of Planets, and each Planet's revenue is the received payments from users. In addition, while leasing resources to facilitate Metaverse services for users, each Planet needs to undertake the corresponding operating expenditures such as electricity costs. Therefore, the payoff of each Planet j in each time round t can be denoted as the revenue minus its expenditures, *i.e.*,

$$\begin{aligned} \Pi_{j,t} = \sum_{i \in U_{j,t}} \Big[&(p_j'^{\text{GPU}} - q_j^{\text{GPU}}) \cdot g_{i,t}' + (p_j'^{\text{CPU}} - q_j^{\text{CPU}}) \cdot f_{i,t}' \\ &+ (p_j'^{\text{Bandwidth}} - q_j^{\text{Bandwidth}}) \cdot w_{i,t}' \Big], \end{aligned} \tag{21}$$

where $U_{j,t}$ denotes the set of all users accessing Planet j in round t. q_j^{GPU}, q_j^{CPU} and $q_j^{\text{Bandwidth}}$ represent the operational expenditures incurred by Planet j for using one unit of GPU, CPU and outbound bandwidth resource, respectively.

For each Planet, it aims to design a distributed pricing-based incentive mechanism to motivate users' participation and promote long-term Planet-user rendering collaboration. Specifically, in each time round t, Planet j updates the price vector $\mathbf{p}_j' = \{p_j'^h | \forall h \in \mathcal{H}\}$ by simultaneously satisfying the following three constraints:

- *Individual rationality constraint:* $\Pi_{j,t} \geq 0$, indicating that each Planet will receive non-negative payoff during cooperative rendering. Namely, each Planet should ensure that $p_j'^h \geq q_j^h, \forall h \in \mathcal{H}$;
- *Resource capacity constraint:* In each round, the amount of resource allocated by each Planet cannot exceed the

Planet's capacity of that kind of resource. For example, for GPU of Planet j, $\sum_{i\in U_{j,t}} g'_{i,t} \leq V_j^{\text{GPU}}$ should be satisfied, where V_j^{GPU} denotes the GPU capacity;

- *Participation incentive constraint:* As demonstrated in [2], the global Metaverse market is still in its infancy. In order to attract more users to engage in Metaverse social activities, we introduce the participation incentive constraint [2], which stipulates that, when satisfying the above two constraints, the resource prices should be kept as low as possible to motivate user participation.

V. Multi-Agent HyAR TD3 for Dynamic Rendering Resource Auction

In this section, to address the auctioneer-bidder double-layer decision problem, we propose a hybrid action MARL algorithm for dynamic and efficient rendering resource auction. Specifically, for the bidder layer (in Sect. V-A), we design the MA-HyAR-TD3 algorithm to distributively address Dec-POMDP problems for all users regarding Planet access, rendering quality, and resource leasing quantity. Then, each user calculates the bids and sends them to the corresponding Planet. For the auctioneer layer (in Sect. V-B), after receiving the bids, we design the rendering resource pricing mechanism to address each Planet' pricing problem. In the following, we present the detailed strategy-making process for the bidder layer and the auctioneer layer, respectively.

A. Bidder Layer: Multi-Agent HyAR TD3 Algorithm

To cooperate with Planets in rendering VR scenes, each user should execute both the discrete action (*i.e.*, v) and the continuous action C (*i.e.*, $\{R, k, g, f, w\}$) in every round. To effectively tackle the RL problem with a discrete-continuous hybrid action space, we employ HyAR [31] to construct a compact, decodable and semantically smooth latent representation space for the original hybrid action space. Then, users can utilize conventional MARL algorithms (*e.g.*, MATD3) to learn latent policies in the representation space and decode the latent actions into the original action space for interacting with the environment. In the following, we present the construction of the latent representation space and the learning of the latent policy, respectively. Finally, we present the overall algorithm of the MA-HyAR-TD3.

Stage 1: Construction of the latent representation space. Firstly, we establish a learnable embedding table with J rows and d_1 columns ($E_\zeta \in \mathbb{R}^{J\times d_1}$). In E_ζ, each row $e_{\zeta,v}$ (where v is the row index) is a continuous vector of dimension d_1 that represents the discrete action v. Then, we map the continuous action C into the latent action $z \in \mathbb{R}^{d_2}$ in the latent representation space conditioned on the user's observation o and the discrete action's representation $e_{\zeta,v}$, using a conditional Variational Auto-Encoder (VAE) q_ϕ with learnable parameter ϕ. Specifically, conditional VAE q_ϕ takes o, $e_{\zeta,v}$ and C as input and outputs the mean μ and standard deviation σ of a Gaussian distribution $\mathcal{N}(\mu,\sigma)$, where the latent action z is sampled from it. For any latent actions $e \in \mathbb{R}^{d_1}$ and $z \in \mathbb{R}^{d_2}$ in the representation space, they should be able to be decoded into original hybrid action space. For continuous action, we can employ a conditional VAE decoder p_ψ with learnable parameter ψ to decode. p_ψ takes o and $e_{\zeta,v}$ as conditions, and deterministically decode z into the continuous action $\widetilde{C}$ in the original action space. For discrete action, the user decodes it into $\widetilde{v}$ through nearest-neighbor lookup in the embedding table.

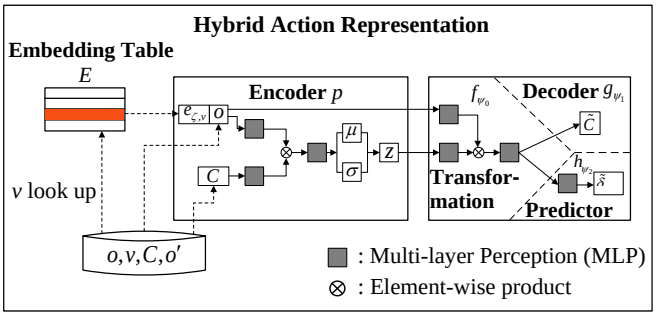

Fig. 7: The overall workflow of encoding and decoding in hybrid action representation model.

The overall workflow of encoding and decoding is shown in Fig. 7, which can be expressed as below:

$$\textbf{Encoding: } e_{\zeta,v} = E_\zeta(v), z \sim q_\phi(\cdot|C, o, e_{\zeta,v}), \tag{22}$$

$$\begin{aligned}\textbf{Decoding: } \widetilde{v} &= g_E(e) = \text{argmin}_{v'\in\mathcal{V}}||e_{\zeta,v'} - e||_2,\\ \widetilde{C} &= p_\psi(z, o, e_{\zeta,v}),\end{aligned} \tag{23}$$

where $p_\psi = g_{\psi_1} \circ f_{\psi_0}$, *i.e.*, the cascade of decoder network and transformation network.

Next, we present the learning process of the parameters of the embedding table E_ζ, as well as the encoder q_ϕ and decoder p_ψ of the conditional VAE. Although E_ζ, q_ϕ, and p_ψ can form a concise and decodable latent representation space, this space may prove insufficient for training policies and Q-functions due to the tasks' strong reliance on comprehending the dynamics of the environment. Therefore, we should also train the action representation through unsupervised environmental dynamics prediction, making it semantically smooth, *i.e.*, actions which are proximal in the latent space can exhibit analogous effects on environmental dynamics.

In general, with a batch of transitions (o_i, v_i, C_i, o'_i) from replay buffer $\mathcal{D}$, we jointly train E_ζ, q_ϕ and p_ψ by minimizing the loss function L_{HyAR} as below:

$$\begin{aligned}L_{\text{HyAR}}(\phi,\psi,\zeta) = \mathbb{E}\Big[&\alpha\cdot||C_i - \widetilde{C}_i||_2^2 + \lambda\cdot||\delta_{P_i,P'_i} - \widetilde{\delta}_{P_i,P'_i}||_2^2\\ &+ \eta\cdot D_{\text{KL}}(q_\phi(\cdot|C_i, o_i, e_{\zeta,v_i})||\mathcal{N}(0,I))\Big],\end{aligned} \tag{24}$$

where the first term is the L_2-norm square reconstruction error between the original continuous action C_i and the regenerated action $\widetilde{C}_i$; the third term is the Kullback-Leibler divergence D_{KL} between the latent variable z_i and the standard Gaussian distribution; the second term is the L_2-norm square prediction error between the actual observation (the prices of resources here) residual δ_{P_i,P'_i} and the prediction $\widetilde{\delta}_{P_i,P'_i}$; α, η and λ are hyper-parameters to weight the three terms. $\delta_{P_i,P'_i} = P'_i - P_i$, where P is a part of the observation and represents the prices

of all planets' resources. $\widetilde{\delta}_{P_i,P_i'}$ is predicted by the network p_ψ as follows,

$$\textbf{Prediction: } \widetilde{\delta}_{P_i,P_i'} = p_\psi(z_i, o_i, e_{\zeta,v_i}), \tag{25}$$

where $p_\psi = h_{\psi_2} \circ f_{\psi_0}$, *i.e.*, the cascade of predictor network and transformation network.

Stage 2: Learning of the latent policy by MATD3 in the representation space. MATD3 extends TD3 [32] in the multi-agent scenario. In MATD3 algorithm, each user possesses one policy (actor) network and two value (critic) networks. Since the users we consider have the same observations, actions, and reward functions, employing the parameter sharing (PS) framework can enhance the efficiency of training their policies, *i.e.*, all users share the same actor network π_ω with learnable parameter ω and critic networks $Q_{\theta_{l=1,2}}$ with learnable parameters $\theta_{l=1,2}$.

The main idea of MATD3 is to take the environment state s, all users' action $a_{1:I}$ and rewards $r_{1:I}$ for centralized training, while only requiring local observation of each user for decentralized execution. Specifically, each user i takes observation o_i as the input of the actor network π_ω, which outputs the latent action vectors as follows,

$$e_i \in \mathbb{R}^{d_1} \text{ and } z_i = \pi_\omega(o_i) \in \mathbb{R}^{d_2}. \tag{26}$$

The double critic networks Q_{θ_1}, Q_{θ_2} take the state s, along with latent actions $e_{1:I}$, $z_{1:I}$ from all users as inputs. They then produce estimates of the double Q-values $Q_{\theta_1}(s, e_{1:I}, z_{1:I})$, $Q_{\theta_2}(s, e_{1:I}, z_{1:I})$, which assess the effectiveness of actions given the current state. Then, latent actions can be decoded to actions in the original hybrid action space by Eq. (23) for interacting with the environment. Consequently, the environment transitions to state s', and the user receives reward r_i. Next, with a batch of transition samples $(s, e_{1:I}, z_{1:I}, r_i, s')$ from buffer $\mathcal{D}$, the shared critics are trained by Clipped Double Q-Learning, employing the following loss function for each critic $l = 1, 2$:

$$L_{\text{CDQ}}(\theta_l) = \mathbb{E}\Big[(y_i - Q_{\theta_l}(s, e_{1:I}, z_{1:I}))^2\Big], \tag{27}$$

where y_i is the Temporal Difference (TD) target, which can be represented as follows,

$$y_i = r_i + \gamma \min_{l=1,2} Q_{\overline{\theta}_l}(s', \pi_{\overline{\omega}}(o_1'), ..., \pi_{\overline{\omega}}(o_I')). \tag{28}$$

Here, $\overline{\theta}_{l=1,2}$ are parameters of the critic target networks, and $\overline{\omega}$ is the parameter of the actor target network. The actor (latent policy) is trained by the deterministic policy gradient method [33], which can be formulated as:

$$\nabla_\omega J(\omega) = \mathbb{E}\Big[\nabla_{\pi_\omega(o_i)} Q_{\theta_1}(s, e_{1:I}, z_{1:I}) \cdot \nabla_\omega \pi_\omega(o_i)\Big]. \tag{29}$$

Then the online and target network parameters of the critics and actor are updated respectively as follows for $l = 1, 2$:

$$\theta_l \leftarrow \theta_l - \epsilon^{\theta_l} \nabla_{\theta_l} L_{\text{CDQ}}(\theta_l) \text{ and } \overline{\theta}_l \leftarrow \tau\theta_l + (1-\tau)\overline{\theta}_l, \tag{30}$$

$$\omega \leftarrow \omega + \epsilon^{\omega} \nabla_\omega J(\omega) \text{ and } \overline{\omega} \leftarrow \tau\omega + (1-\tau)\overline{\omega}, \tag{31}$$

where $\epsilon^{\theta_{l=1,2}}$ and ϵ^{ω} are learning rates of gradient descent for the online critic networks and gradient ascent for the online actor network, respectively.

Algorithm 1: Multi-Agent HyAR TD3 for I Users

Initialize parameters ω, $\theta_{l=1,2}$ randomly for online shared actor network π_ω and critic networks Q_{θ_1}, Q_{θ_2}
Copy them to target networks $\overline{\theta}_{l=1,2} \leftarrow \theta_{l=1,2}$, $\overline{\omega} \leftarrow \omega$
Initialize parameters ζ, ϕ and ψ randomly for embedding table E_ζ and conditional VAE q_ϕ, p_ψ
Prepare replay buffer $\mathcal{D}$
// Stage 1: Construction of the representation space
repeat
  Sample a mini-batch of (o_i, v_i, C_i, o_i') from $\mathcal{D}$
  Update ζ, ϕ and ψ using Eq. (24)
until *reaching max representation training times*;
// Stage 2: Learning of the latent policy by MATD3
repeat
  Initialize the environment state s and each user's local observation o_i
  for *time round* $t = 1$ *to* T **do**
    for *user* $i = 1$ *to* I **do**
      // select latent representation actions
      $e_i, z_i = \pi_\omega(o_i) + \xi$, with $\xi \sim \mathcal{N}(0, \sigma)$
      // decode into original hybrid action space
      $v_i = g_E(e_i), C_i = p_\psi(z_i, o_i, e_{\zeta,v_i})$
    Execute actions $a_{1:I}$, observe reward r_i and next local observation o_i', global state transitions to s'
    Store $(v_i, C_i, s, e_{1:I}, z_{1:I}, r_i, s')$ to $\mathcal{D}$
    Sample a mini-batch of $(o_i, s, e_{1:I}, z_{1:I}, r_i, s', o_i')$
    Update $Q_{\theta_{l=1,2}}$ using Eq. (27)
    Update π_ω using Eq. (29)
    Update $Q_{\overline{\theta}_{l=1,2}}$, $\pi_{\overline{\omega}}$ using Eq. (30) and Eq. (31)
until *reaching max total rounds*;

The pseudo-code of multi-agent HyAR TD3 for each Metaverse user is summarized in Algorithm 1.

B. Auctioneer Layer: Rendering Resource Pricing Mechanism

Based on the bids sent from users, each Planet updates the prices of all rendering resources in each time round t in a distributed manner, as shown in Lines 6–8 in Algorithm 2. Theorem 1 demonstrates the rendering resource pricing mechanism of each Planet.

Theorem 1. *The rendering resource pricing scheme, which meets the three constraints in Sect. IV-B, is shown as follows with $h \in \mathcal{H}$:*

$$p_{j,t+1}^h = \max\{q_j^h, \frac{\sum_{i \in U_{j,t}} b_{i,t}^h}{V_j^h}\}, \forall j, t, h. \tag{32}$$

Proof of Theorem 1. In the designed resource pricing scheme, we can conclude that the updated resource price $p_{j,t+1}^h$ is no less than q_j^h, satisfying the individual rationality constraint. We now focus on how it satisfies the capacity constraint.

After updating the resource prices, the actual amount of resource $g_{i,t}'$ (taking GPU resource as an example) that can be leased satisfies the following inequality:

$$g_{i,t}' \cdot p_{j,t+1}^{\text{GPU}} \le g_{i,t} \cdot p_{j,t}^{\text{GPU}}, \forall j, t. \tag{33}$$

Algorithm 2: Resource Auction in Each Time Round

Each user gets local observation o_i, and the information of resources' prices $\mathbf{p}$ of all Planets
for *user* $i = 1$ *to* I **do**
  $e_i, z_i = \pi_\omega(o_i) + \xi$, with $\xi \sim \mathcal{N}(0, \sigma)$
  $v_i = g_E(e_i), C_i = p_\psi(z_i, o_i, e_{\zeta, v_i})$
  Calculate the bids using Eqs. (15)-(17) and send them to the corresponding Planet
for *Planet* $j = 1$ *to* J **do**
  Receive bids from users and update prices of resources using Eq. (32)
  Broadcast updated prices to all users
for *user* $i = 1$ *to* I **do**
  Observe the updated prices, and calculate the actual amount of resources that can be leased using Eq. (34)
  Use the leased resources to render VR scenes

The above inequality can be further processed as follows,

$$g'_{i,t} \leq \frac{g_{i,t} \cdot p^{\text{GPU}}_{j,t}}{p^{\text{GPU}}_{j,t+1}}, \forall j, t, \tag{34}$$

$$c^{\text{allocated\_GPU}}_{j,t} \leq \sum_{i \in U_{j,t}} \frac{g_{i,t} \cdot p^{\text{GPU}}_{j,t}}{p^{\text{GPU}}_{j,t+1}}, \forall j, t, \tag{35}$$

where $c^{\text{allocated\_GPU}}_{j,t}$ represents the amount of GPU resource allocated by Planet j at time round t, satisfying

$$c^{\text{allocated\_GPU}}_{j,t} = \sum_{i \in U_{j,t}} g'_{i,t}, \forall j, t. \tag{36}$$

According to the resource pricing mechanism, we have

$$p^{\text{GPU}}_{j,t+1} \geq \frac{\sum_{i \in U_{j,t}} b^h_{i,t}}{V^h_j}, \forall j, t. \tag{37}$$

As such, we can obtain

$$c^{\text{allocated\_GPU}}_{j,t} \leq \frac{\sum_{i \in U_{j,t}} g_{i,t} \cdot p^{\text{GPU}}_{j,t}}{\sum_{i \in U_{j,t}} b^h_{i,t}} \cdot V^h_j = V^h_j, \tag{38}$$

which satisfies the resource capacity constraint.

Additionally, the proposed resource pricing mechanism does not have any additional conditions that would cause price growth, thus satisfying the participation incentive constraint as well. □

Algorithm 2 describes the proposed rendering resource auction mechanism, which includes the dynamic decision-making processes of both the user layer and the Planet layer, as well as the rendering resource auction process between them.

VI. Performance Evaluation

In this section, the experimental setup and implementation details are first introduced (in Sect. VI-A), followed by the experimental results and discussions from both the training phase (in Sect. VI-B) and the test phase (in Sect. VI-C).

A. Experimental Setup

The experimental scenario consists of 1 central Nebula server, 3 Planets, and 5 users. Each user can access any Planet, and network distances between users and Planets are randomly selected from the range of $[10, 100]$.

1) User Parameter Setting. Planets assist users in rendering VR scenes, where there are up to 30 types of virtual objects in the background environment. The interest levels of each user in these virtual objects are taken from a uniform distribution in the range of $[1, 100]$. In this paper, the total area of a background panorama is set to 100, with the highest and lowest resolutions per unit area being 5×10^6 and 100, respectively. In each episode, users and Planets engage in 10 rounds of rendering resource auctions, during which the quantities of resources requested by single user are selected in the range of $[1 \times 10^5, 5 \times 10^7]$ pixels/ms for GPU, $[1 \times 10^4, 5 \times 10^6]$ cycles/ms for CPU, and $[1 \times 10^4, 6 \times 10^6]$ bits/ms for outbound bandwidth, respectively. For the parameters in the user's reward expression, we set the adjustment coefficients κ and ρ as 50 and 100, the moving time between two neighbor points $T_{\text{unit}} = 20$ ms, the radius of the observation circle $X = 1$, and the transmission fee paid to single router $x = 1$ ¢.

2) Planet Parameter Setting. Planets utilizes the GPU, CPU, and outbound bandwidth resources to render VR background environments for users. The operational expenditures are taken according to uniform distributions in the range of $[1, 3] \times 10^{-5}$ ¢ for unit GPU (1 pixel/ms), unit CPU (1 cycle/ms), and unit outbound bandwidth (1 bit/ms). The setting of resource capacities of Planets is summarized in Table II.

3) Implementations. We use Adam optimizer to train the representation, actor, and critic networks with learning rates equals to 10^{-4}, 3×10^{-4}, and 3×10^{-4}, respectively. We first train the embedding table and conditional VAE. We employ a random policy for 50,000 time rounds, accumulating all experiences into the representation replay buffer, followed by 40,000 training iterations, each with a batch size of 64. Then, for RL in the representation space and other MARL methods, we conduct 400,000 time rounds and store experiences in the RL replay buffer. After the initial 64 time rounds, training is conducted every two rounds, with a batch size of 64 for each training iteration. We set target networks' update rate $\tau = 5 \times 10^{-3}$, and discount factor $\gamma = 0.99$. For a fair comparison, both the architecture of networks and the random seeds are the same among different MARL methods.

4) Baselines. We compare the proposed MA-HyAR-TD3 algorithm with three MARL methods and the random scheme in the rendering resource auction process.

- *MA-PA-DDPG*. PA-DDPG [34] is a scheme that applies the deep deterministic policy gradients (DDPG) algorithm

TABLE II: Resource Capacities of 3 Planets

Planets	GPU 10^8 pixels/ms	CPU 10^7 cycles/ms	Bandwidth 10^7 bits/ms
Planet 1	1.22	2.34	2.61
Planet 2	1.10	2.65	3.51
Planet 3	1.24	2.72	2.25

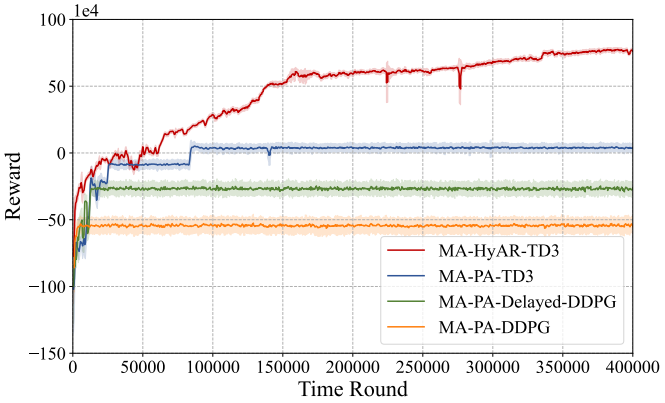

Fig. 8: Average reward for different MARL methods during training process.

[35] to parameterized action space. As the original PA-DDPG does not support multi-agent scenarios, to be fair and objective, we extend the PA-DDPG algorithm to the multi-agent scenario and rename it as MA-PA-DDPG.

- *MA-PA-Delayed-DDPG.* Based on MA-PA-DDPG, we further reduce the update frequency of the actors to half of that of the critics, and rename this scheme as MA-PA-Delayed-DDPG.
- *MA-PA-TD3.* In this scheme, the DDPG component in MA-PA-DDPG is replaced with the TD3 variant. As such, the only difference between this approach and the one we propose lies in the handling of the discrete-continuous hybrid action space.
- *Random scheme.* In this scheme, users randomly choose their actions including which Planet to access, the rendering quality of VR scenes, and the amount of required resources.

B. Training Performance Evaluation

In this section, we evaluate the training performance of the four MARL methods, as shown in Fig. 8. The solid lines depict the moving average reward of all users over the past 100 episodes, while the shaded areas represent the corresponding standard deviations. We notice that initially, the average rewards of the four MARL methods are similar and relatively low. This is because, at this stage, all users are exploring the environment and have not yet learned the optimal strategy. However, as the training process continues, the policies of the four MARL methods improve, and the average rewards show an upward trend. For MA-PA-DDPG, its converged average reward is the lowest, around -54×10^4. In the MA-PA-Delayed-DDPG method, since the critic's update frequency is set to twice that of the actor, this allows the actor to update based on more stable and accurate Q-values [36], [37]. As a result, the converged average reward is higher than that of MA-PA-DDPG, around -26×10^4. In MA-PA-TD3, it builds upon the MA-PA-Delayed-DDPG by introducing twin networks for both the critic and the actor. This alleviates the adverse effects of bootstrapping [38], thereby raising the converged average reward to around 4×10^4. Due to embedding the dependence of continuous action on corresponding discrete action in the latent space, our proposed MA-HyAR-TD3 approach converges to the highest average reward, around 77×10^4.

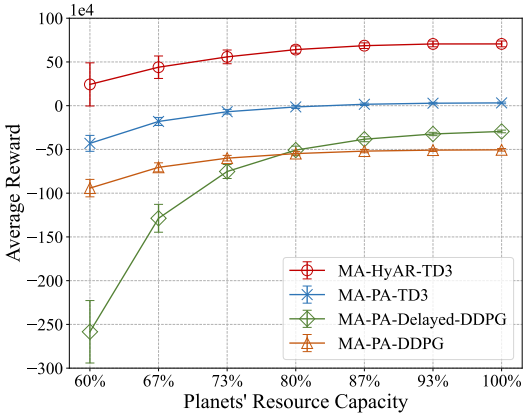

Fig. 9: Impact of resource capacities on average reward.

C. Test Performance Evaluation

In this section, we conduct tests on the trained models to validate the superiority of the proposed MA-HyAR-TD3 approach in terms of the user reward, user QoE, and resource utilization of Planets.

1) Impact of Planets' resource capacities on average reward. We reduce the resource capacities of each Planet from the original capacities to 60% of them, to observe the impact of Planets' resource capacities on the reward of users, as shown in Fig. 9. The points represent the average reward of all users over 100 episodes, while the error bars indicate the corresponding standard deviations. We can observe a decrease in the average reward for all MARL methods as the resource capacities of Planet decline. This is because competition among multiple users for limited resources becomes more intense, resulting in a decrease in the actual amount of resources that each user can obtain. Our proposed MA-HyAR-TD3 approach consistently outperforms other approaches, whether in situations of abundant or limited resources.

2) User QoE comparison. We compare the user QoE across four MARL methods and the random scheme, as depicted in Fig. 10. The bars represent the average QoE of all users over 100 episodes, while the error bars indicate the corresponding standard deviations. It is evident that only MA-HyAR-TD3 and MA-PA-TD3 exhibit an average QoE greater than 0, and our proposed approach significantly outperforms other schemes. According to Eq. (11), user QoE is the product of objective KPI and subjective evaluation of visual quality. To further understand the experimental results of average QoE in Fig. 10, we conduct separate experiments for users' objective KPI and subjective evaluation, as shown in Fig. 11 and Fig. 12.

Pre-rendering success percentage. Fig. 11 illustrates the percentage of pre-rendering delays meeting condition $T_{\text{pre-render}} \leq T_{\text{unit}}$ across all users over 1000 time rounds for five methods. As a necessary constraint for Planet-user cooperative pre-rendering, the higher this percentage, the higher the success rate of cooperative pre-rendering process, resulting in reduced

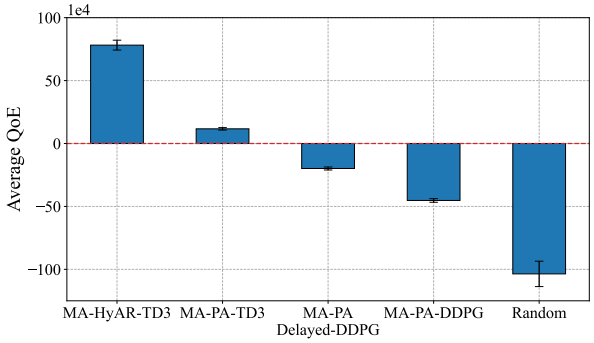

Fig. 10: User QoE comparison across four MARL methods and the random scheme.

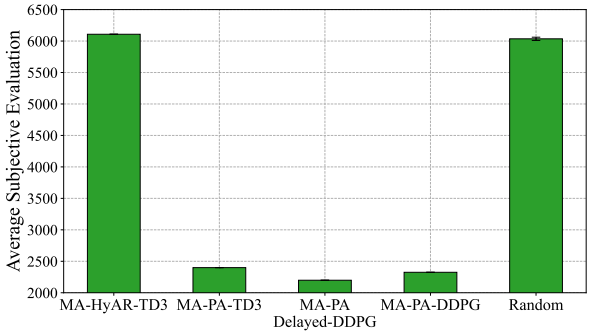

Fig. 12: Subjective evaluation of visual quality comparison across four MARL methods and the random scheme.

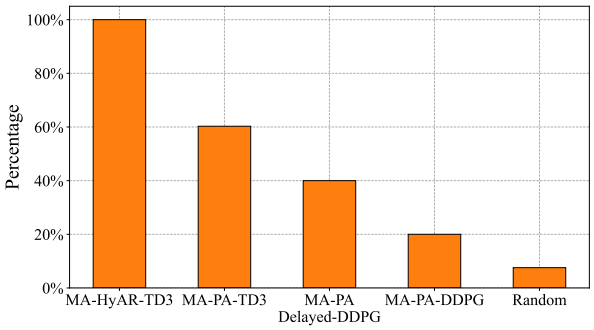

Fig. 11: Pre-rendering success percentage for four MARL methods and the random scheme.

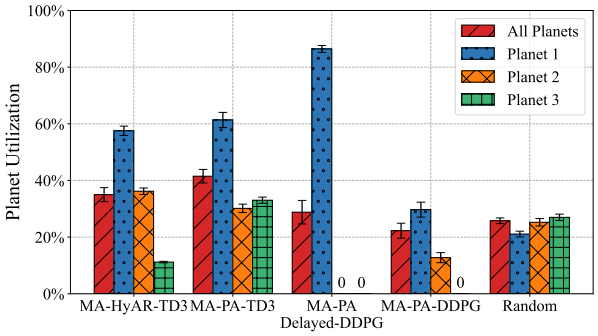

Fig. 13: Planet utilization comparison across four MARL methods and the random scheme.

stuttering for users. It can be observed that the percentage for MA-HyAR-TD3 is 100%, indicating that all users receive Planet-rendered backgrounds promptly in all time rounds. If the condition is not met, the objective KPI will be less than 0, resulting in the average QoE of MA-PA-Delayed-DDPG, MA-PA-DDPG, and the random scheme being negative.

Subjective evaluation of visual quality. Fig. 12 depicts the average subjective evaluation of visual quality by users for the five methods. The bars represent the average subjective evaluation of all users over 100 episodes, while the error bars indicate the corresponding standard deviations. The subjective evaluation of visual quality is influenced by two aspects: resolution and compression ratio. We can infer that under the MA-HyAR-TD3 and random scheme, users' requested resolution and compression ratio are significantly higher than the other three methods, thereby resulting in higher visual quality. However, the background visuals need to be transmitted to users' VR devices promptly; otherwise, even with high visual quality, users cannot experience. Thereby, the random scheme has high average subjective evaluation but achieves the lowest average QoE.

3) Planet resource utilization comparison. We compare the resource utilization of Planets across four MARL methods and the random scheme, as depicted in Fig. 13. The bars represent the average utilization of resources over 100 episodes, while the error bars indicate the corresponding standard errors. It is observed that in the MA-PA-Delayed-DDPG and MA-PA-DDPG methods, some Planets remain completely idle, suggesting that users are not fully utilizing the resources of Planets. Fig. 9 also illustrates this finding: in the range of [60%, 73%], the average reward of MA-PA-Delayed-DDPG drops sharply. This is because all users only choose the first Planet to access, making this method particularly sensitive to resource depletion.

We also observe that MA-HyAR-TD3 has a lower overall utilization rate compared to MA-PA-TD3. However, it attains a higher pre-rendering success percentage while also achieving higher subjective evaluations of visual quality. In Fig. 14, we further investigate this counterintuitive finding. The scatter plots in Fig. 14 depict the requested quantities of three types of resources and the actual leased quantities after auction *vs* their corresponding workloads. It is observed that for GPU and outbound bandwidth, the resources requested in MA-PA-TD3 exhibit a polarized phenomenon: either requesting the maximum resources (5×10^7 for GPU, 6×10^6 for outbound bandwidth), or requesting the minimum resources (10^5 for GPU, 10^4 for outbound bandwidth). Requesting the maximum resources increases Planets' utilization, while requesting the minimum resources increases the time to complete the tasks. When the proportion of time rounds with minimal resource requests is higher (20% for GPU, 40% for outbound bandwidth), the pre-rendering success percentage shows a noticeable decrease. Similar situations are observed in MA-PA-Delayed-DDPG and MA-PA-DDPG. Compared to MA-PA-TD3, our proposed MA-HyAR-TD3 approach attains more intelligent rendering resource leasing outcomes, and consequently, it achieves higher subjective evaluation and pre-rendering success percentage while utilizing fewer resources.

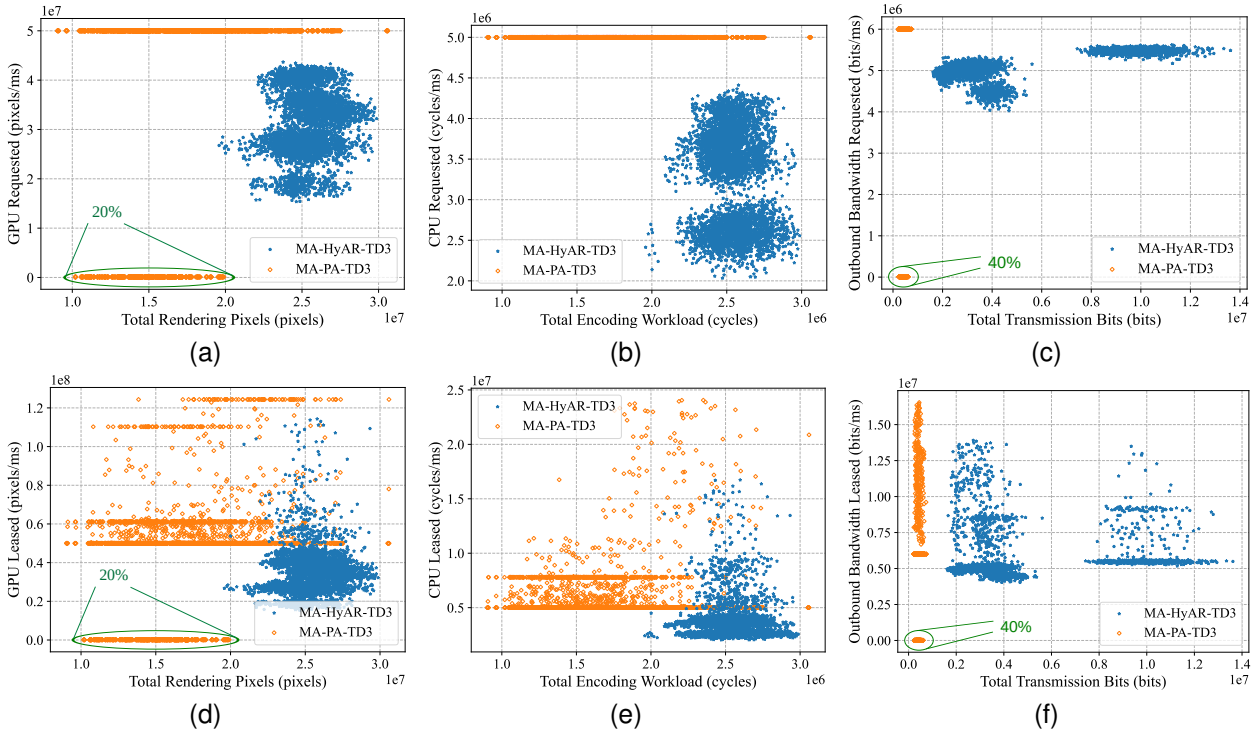

Fig. 14: Requested quantities of resources and actual leased quantities after auction *vs* the corresponding workloads.

VII. Conclusion

In this paper, we have constructed a QoE-oriented cooperative VR scene pre-rendering framework in edge-empowered Metaverse, where users can lease rendering resources from Planets for QoE enhancement in an on-demand manner. Under this framework, we have established a novel user QoE model which comprehensively considers both the pre-rendering delay and the subjective evaluation of visual quality. To dynamically, efficiently, and economically allocate the limited multidimensional rendering resources of Planets to users, we have proposed a novel MARL-based dynamic resource auction mechanism, where Planets act as auctioneers and users act as bidders. The proposed MARL method is named as MA-HyAR-TD3, which is specifically designed for multi-user discrete-continuous hybrid action scenario. Extensive evaluations have demonstrated the superiority of the proposed method over the representative schemes concerning user QoE and resource utilization efficiency. Regarding QoE, our method has exhibited a minimum 18-fold enhancement compared to other schemes, showcasing its ability to deliver users immersive Metaverse experience.

References

[1] N. Liu, T. H. Luan, Y. Wang, Y. Liu, and Z. Su, "Auction-based dynamic resource allocation in social metaverse," in *The 19th International Conference on Mobility, Sensing and Networking, MSN 2023, December 14-16, 2023*, pp. 669–676.

[2] H. Wang *et al.*, "A survey on the metaverse: The state-of-the-art, technologies, applications, and challenges," *IEEE Internet Things J.*, vol. 10, no. 16, pp. 14 671–14 688, 2023.

[3] W. Y. B. Lim *et al.*, "Realizing the metaverse with edge intelligence: A match made in heaven," *IEEE Wirel. Commun.*, vol. 30, no. 4, pp. 64–71, 2023.

[4] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, and X. Shen, "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 1, pp. 319–352, 2023.

[5] OpenAI. Video generation models as world simulators. Accessed: Feb. 15, 2024. [Online]. Available: https://openai.com/index/video-generation-models-as-world-simulators/

[6] BBC. Fortnite's travis scott virtual concert watched by millions. Accessed: Apr. 24, 2020. [Online]. Available: https://www.bbc.com/news/technology-52410647

[7] Y. Zhang, L. Pu, T. Lin, and J. Yan, "QoE-oriented mobile virtual reality game in distributed edge networks," *IEEE Trans. Multimedia*, vol. 25, pp. 9132–9146, 2023.

[8] J. Yu, A. Alhilal, T. Zhou, P. Hui, and D. H. Tsang, "Attention-based QoE-aware digital twin empowered edge computing for immersive virtual reality," *IEEE Trans. Wirel. Commun.*, 2024, early access, doi:10.1109/TWC.2024.3380820.

[9] X. Ren *et al.*, "Dual-level resource provisioning and heterogeneous auction for mobile metaverse," *IEEE. Trans. Mob. Comput.*, 2024, early access, doi:10.1109/TMC.2024.3377211.

[10] N. C. Luong, T. Le Van, S. Feng, H. Du, D. Niyato, and D. I. Kim, "Edge computing for metaverse: Incentive mechanism versus semantic communication," *IEEE. Trans. Mob. Comput.*, vol. 23, no. 5, pp. 6196–6211, 2024.

[11] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, N. Dai, and H.-S. Lee, "Furion: Engineering high-quality immersive virtual reality on today's mobile devices," *IEEE. Trans. Mob. Comput.*, vol. 19, no. 7, pp. 1586–1602, 2020.

[12] Y. Wang, Z. Su, and M. Yan, "Social metaverse: Challenges and solutions," *IEEE Internet Things Mag.*, vol. 6, no. 3, pp. 144–150, 2023.

[13] J. Zhao, L. Qian, and W. Yu, "Human-centric resource allocation in the metaverse over wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 3, pp. 514–537, 2024.

[14] Y. Jiang, J. Kang, X. Ge, D. Niyato, and Z. Xiong, "QoE analysis and resource allocation for wireless metaverse services," *IEEE Trans. Commun.*, vol. 71, no. 8, pp. 4735–4750, 2023.

[15] Y. Jiang *et al.*, "Joining edge-enabled metaverse services with network externality: A stackelberg game approach," in *2023 IEEE International Conference on Metaverse Computing, Networking and Applications, MetaCom 2023, June 26-28, 2023*, pp. 571–577.

[16] H. Du *et al.*, "Attention-aware resource allocation and QoE analysis for

metaverse xurllc services," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 7, pp. 2158–2175, 2023.

[17] N. H. Chu *et al.*, "Metaslicing: A novel resource allocation framework for metaverse," *IEEE. Trans. Mob. Comput.*, vol. 23, no. 5, pp. 4145–4162, 2024.

[18] Y. Chen, H. Kwon, H. Inaltekin, and M. Gorlatova, "VR viewport pose model for quantifying and exploiting frame correlations," in *IEEE Conference on Computer Communications, INFOCOM 2022, May 2-5, 2022*, pp. 1269–1278.

[19] Q. Zhang, J. Wei, S. Wang, S. Ma, and W. Gao, "Realvr: Efficient, economical, and quality-of-experience-driven VR video system based on MPEG OMAF," *IEEE Trans. Multimedia*, vol. 25, pp. 5386–5399, 2023.

[20] S. Wang *et al.*, "Robust saliency-driven quality adaptation for mobile 360-degree video streaming," *IEEE. Trans. Mob. Comput.*, vol. 23, no. 2, pp. 1312–1329, 2024.

[21] C. Xu, Z. Chen, M. Tao, and W. Zhang, "Edge-device collaborative rendering for wireless multi-user interactive virtual reality in metaverse," in *2023 IEEE Global Communications Conference, GLOBECOM 2023, December 4-8, 2023*, pp. 3542–3547.

[22] K. Meng, Y. Hui, R. Sun, N. Cheng, Z. Su, and T. H. Luan, "Environment-aware dynamic resource allocation for VR video services in vehicle metaverse," in *2023 IEEE 98th Vehicular Technology Conference, VTC2023-Fall, October 10-13, 2023*, pp. 1–5.

[23] Z. Chen, H. Zhu, L. Song, D. He, and B. Xia, "Wireless multiplayer interactive virtual reality game systems with edge computing: Modeling and optimization," *IEEE Trans. Wirel. Commun.*, vol. 21, no. 11, pp. 9684–9699, 2022.

[24] H. Du *et al.*, "Exploring attention-aware network resource allocation for customized metaverse services," *IEEE Netw.*, vol. 37, no. 6, pp. 166–175, 2023.

[25] X. Zhang, H. Zhang, K. Sun, K. Long, and Y. Li, "Human-centric irregular RIS-assisted multi-UAV networks with resource allocation and reflecting design for metaverse," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 3, pp. 603–615, 2024.

[26] Z. Long, H. Dong, and A. El Saddik, "Human-centric resource allocation for the metaverse with multi-access edge computing," *IEEE Internet Things J.*, vol. 10, no. 22, pp. 19 993–20 005, 2023.

[27] N. Q. Hieu, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Toward BCI-enabled metaverse: A joint learning and resource allocation approach," in *2023 IEEE Global Communications Conference, GLOBECOM 2023, December 4-8, 2023*, pp. 3342–3348.

[28] K. Boos, D. Chu, and E. Cuervo, "Flashback: Immersive virtual reality on mobile devices via rendering memoization," in *the 14th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys'16, June 26-30, 2016*, pp. 291–304.

[29] P. Reichl, S. Egger, R. Schatz, and A. D'Alconzo, "The logarithmic nature of QoE and the role of the weber-fechner law in QoE assessment," in *The 2010 IEEE International Conference on Communications, ICC 2010, May 23-27, 2010*, pp. 1–5.

[30] I. Lubashevsky, "Psychophysical laws as reflection of mental space properties," *Phys. Life Rev.*, vol. 31, pp. 276–303, 2019.

[31] B. Li *et al.*, "Hyar: Addressing discrete-continuous action reinforcement learning via hybrid action representation," in *The 10th International Conference on Learning Representations, ICLR 2022, April 25-29, 2022*, pp. 1–22.

[32] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *The 35th International Conference on Machine Learning, ICML 2018, July 10-15, 2018*, pp. 1587–1596.

[33] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *The 31st International Conference on Machine Learning, ICML 2014, June 21-26, 2014*, pp. 387–395.

[34] M. J. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," in *The 4th International Conference on Learning Representations, ICLR 2016, May 2-4, 2016*, pp. 1–12.

[35] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *The 4th International Conference on Learning Representations, ICLR 2016, May 2-4, 2016*, pp. 1–14.

[36] G. Alsuhli *et al.*, "Mobility load management in cellular networks: A deep reinforcement learning approach," *IEEE. Trans. Mob. Comput.*, vol. 22, no. 3, pp. 1581–1598, 2023.

[37] Y. Cui, T. Lv, W. Ni, and A. Jamalipour, "Digital twin-aided learning for managing reconfigurable intelligent surface-assisted, uplink, user-centric cell-free systems," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3175–3190, 2023.

[38] W. Wu, F. Yang, F. Zhou, Q. Wu, and R. Q. Hu, "Intelligent resource allocation for IRS-enhanced OFDM communication systems: A hybrid deep reinforcement learning approach," *IEEE Trans. Wirel. Commun.*, vol. 22, no. 6, pp. 4028–4042, 2023.