A Semantic-Aware Transmission with Adaptive Control Scheme for Volumetric Video Service

Yuanwei Zhu, Yakun Huang, Xiuquan Qiao, Zhijie Tan, Boyuan Bai, Huadong Ma, *Fellow, IEEE,* and Schahram Dustdar, *Fellow, IEEE*

Abstract— Volumetric video provides a more immersive holographic virtual experience than conventional video services such as 360-degree and virtual reality (VR) videos. However, due to ultra-high bandwidth requirements, existing compression and transmission technology cannot handle the delivery of real-time volumetric video. Unlike traditional compression methods and the approaches that extend 360-degree video streaming, we propose AITransfer, an AI-powered compression and semantic-aware transmission method for point cloud video data (a popular volumetric data format). AITransfer targets the semantic-level communication beyond transmitting raw point cloud video or compressed video with two outstanding contributions: (1) designing an integrated end-to-end architecture with two fundamental contents of feature extraction and reconstruction to reduce the bandwidth consumption and alleviate the computational pressure; and (2) incorporating the dynamic network condition into end-to-end architecture design and employing a deep reinforcement learning-based adaptive control scheme to provide robust transmission. We conduct extensive experiments on the typical datasets and develop a case study to demonstrate the efficiency and effectiveness. The results show that AITransfer can provide extremely efficient point cloud transmission while maintaining considerable user experience with more than 30.72x compression ratio under the existing network environments.

Index Terms—Point cloud video, reinforcement learning, adaptive transmission, semantic-aware

1 Introduction

Point cloud video, a representative volumetric video, can provide viewers with 6-DoF (degrees of freedom) immersive experiences, which has been widely used in many areas, including holographic communication, education, and healthcare [2], [3]. Point cloud video comprises a set of unstructured 3D points with attribute information, which is fundamentally different from 2D pixel-based video services with 3-DoF experiences, such as 360-degree panoramic video and virtual reality (VR) video [3], [4]. Naturally, delivering point cloud video is valuable but faces severe problems. The most typical difficulty is massive volumetric frames require ultra-high bandwidth at a Gbps scale, which surpasses the capability of current 5G networks. For example, a Microsoft Kinect for Windows v2 [5] camera captures 2.06 Gb of raw point cloud at 30 FPS [3], whereas a 16K 360-degree video requires only 100∼500 Mbps bandwidth. Furthermore, the amount of raw point cloud data will increase with more cameras placed at multiple angles, higher resolutions, and frame rates.

Existing point cloud video transmission techniques can be grouped into two categories. (i) Designing efficient compression methods to intuitively reduce the transmitted data volume [6], [7], [8], including 2D projection-based and 3D tree-based compression [6]. The former decomposes and projects the point cloud onto a 2D image with dense packing to extend existing full-fledged 2D video codecs. Typically, MPEG V-PCC [9] can optimize video compression with incremental differences between frames, providing a high compression ratio and real-time decoding. However, the encoding speed is very slow [6], commonly requiring 11 (lossy compression) to 42 (lossless compression) minutes to encode a one-second longdress video [10]. The latter method uses octree [11] or *k*d-tree [12] data structures to process each point cloud frame independently and efficiently exploit the sparsity of 3D data. However, the most representative PCL [13] and Draco [14] are tested on a desktop showing that both could only reduce the overall size by a ratio of 3.35× to 4.22× [6]. To summarize, these methods have either high coding latency or limited compression ratio, which are difficult when used to meet the real-time requirement. (ii) Extending techniques applied in 360-degree and VR video to volumetric video services to enhance adaptive transmission quality under different network conditions. For example, several existing systems modify the viewport prediction, tiling, and adaptive bitrate streaming (ABR) to selectively transfer the video content in users' field of view (FoV) [2], [4], [15], [16]. However, these approaches usually suffer from high mobile energy consumption and unacceptable processing latency on the receiving devices. Each transmitted tile is vulnerable to the fluctuating network and various packet losses in the reorganizing process.

Although these two types of technology are not contradictory and could be combined to provide a better performance [4], [6], it is still far from the bandwidth requirement for real-time point cloud video delivery. To this end, we

- *Y. Zhu, Y. Huang, X. Qiao, Z. Tan, and B. Bai are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China. E-mail: {zhuyw, ykhuang, qiaoxq, besttangent, baiboyuan}@bupt.edu.cn. (Corresponding author: Xiuquan Qiao)*
- *H. Ma is with the Beijing Key Lab of Intell. Telecomm. Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing, 100876, China. E-mail: mhd@bupt.edu.cn.*
- *S. Dustdar is with the Distributed Systems Group, Technische Universität Wien, Vienna, 1040, Austria. Email: dustdar@dsg.tuwien.ac.at.*

A preliminary version of this paper appears as the proceedings of the 29th ACM International Conference on Multimedia (MM 2021) [1].

propose AITransfer, an AI-powered and semantic-aware transmission technique, which remarkably and adaptively reduces the transmitted data volume for point cloud video service. AITransfer extracts deep semantic features from raw point cloud for efficient transmission and then performs point cloud reconstruction. The AI-based end-to-end design eschews cumbersome multiple processing such as compression and codec in conventional transmission schemes. Besides, AITransfer integrates dynamic network conditions into the above design and develops a deep reinforcement learning (DRL)-based adaptive control scheme to guarantee the transmission quality under different network environments. Nevertheless, there are two technical challenges to the design of AITransfer.

(i) How to extract as few but critical semantic features as possible from massive unordered raw point clouds? Compared with compressing all raw point clouds, only transferring a few deep semantic features can achieve a greater compression ratio. However, it is difficult to extract only a few features representing all original video information, due to the unordered and unstructured data characteristics. In addition, we also have to consider the impact of extracted features on subsequent reconstruction quality. The AI-powered approaches, while good at extracting features, will also bring expensive computation cost due to massive parameters. To address this challenge, we design an integrated neural network by fusing the feature extraction and reconstruction to achieve end-to-end training. In this way, we can directly render and replay the reconstructed result from the raw video acquisition on the premise of ensuring quality. At the data sending side, we extract the point clouds' deep semantic features inspired by PointNet++ [17], providing an extreme compression and energy-saving transmission. At the data receiving side, we employ a lightweight reconstruction neural network to recover the semantic features to the original video as closely as possible, providing a real-time point cloud reconstruction on the resource-constrained devices. We divide the trained model and deploy them onto the data sender and receiver respectively for distributed inference. Only a few extracted semantic features are transmitted, immensely reducing bandwidth consumption and protecting privacy.

(ii) How does the semantic-aware point cloud transmission mechanism adapt to dynamic network environments? Fluctuating network environments are inevitable. Conventional adaptive video streaming techniques are unavailable in this framework because AITransfer adopts a fundamentally different transmission mechanism from DASH-based (Dynamic Adaptive Streaming over HTTP) systems [2], [15]. To provide various compression ratios (i.e., bitrate) matching the dynamic network environment, AITransfer can only change the structure of the integrated neural network. Thus, we incorporate the network condition into the design of the end-to-end architecture and train multiple models with feature extraction and reconstruction. In practical transmission, AITransfer selects the corresponding model according to the network condition at that time. However, this strategy brings the challenge of overhead caused by model selection, which must be solved in milliseconds to support real-time transmission. To this end, we design a DRL-based adaptive control scheme, with the purpose of maximizing transmission performance to select the optimal model from an offline-trained model set. Boosted by the scheme, AITransfer can provide a quick-response and adaptive transmission.

We have implemented AITransfer using TensorFlow [18] and trained the model on a high-performance server equipped with eight Tesla V100 GPUs. We conduct extensive experiments on typical datasets and comparisons with baselines, showing a more than 30.72x compression ratio while maintaining considerable visual quality. We also conduct experiments on real-world point cloud videos to verify the effect of the adaptive control scheme, showing adaptive transmission under dynamic network conditions. In summary, the key contributions are summarized as follows:

- AITransfer provides AI-powered and semantic-aware transmission for real-time volumetric video, reducing bandwidth and energy consumption and changing the conventional transmission mechanism.
- We design and train an integrated end-to-end compression architecture, which transfers the semantic features instead of raw data, significantly reducing the transmitted data volume and protecting privacy.
- We design a DRL-based control scheme to underpin the adaptive transmission, monitoring realistic network conditions and matching the optimal inference model in millisecond-level decision time.
- Evaluation on typical datasets demonstrates promising results, and we develop a prototype deploying AITransfer into practice to verify its effectiveness.

2 Background and Motivation

We compare point cloud video with conventional video services in Table 1 to better understand the critical characteristics and differences.

TABLE 1: Comparisons with other video services [2]

	2D Video	360° Video	VR Video	Point Cloud Video
Freedom	2-DoF	3-DoF	3-DoF	6-DoF
Data Volume	∼1 Mbps	∼100 Mbps	∼100 Mbps	∼1 Gbps
Codec Status	mature	developing	developing	nascent
Key Techniques	adaptive streaming	tiling, viewport prediction	tiling, viewport prediction	tiling, viewport prediction, compression

Point cloud video is generated by simultaneous acquisition and fusion of multiple depth cameras from different angles. As illustrated in Table 1, point cloud video supports 6-DoF experiences, which differs from other video types in terms of data volume, coding research status, and key techniques. Delivering point cloud video requires ultra-high bandwidth consumption and more complex processing. At present, 2D pixel-based video codec technology has been quite mature, and some codecs can even compress VR content by a factor of 100 or 500. However, the development of point cloud video compression is still in its infancy. Existing representative 3D tree-based compression methods can only achieve several times the compression ratio [6].

Current point cloud compression methods are still difficult to support real-time transmission in the existing network environment, and not all points are needed to be transmitted intact. Hence, we can achieve a similar visual effect to the original point cloud video using AI technology for reconstruction when rendering and replaying at the receiving side. Deep learning is a promising method to extract semantic features, which encourages us to consider transferring the deep semantic features instead of raw point cloud video or compressed video, achieving a significant reduction of the transmitted data volume.

3 Design of AITransfer

In this section, we first introduce the overview of the point cloud video transmission system in Section 3.1 to help comprehend each component of AITransfer. Afterward, we design the integrated end-to-end transmission neural network architecture in Section 3.2, including feature extraction, feature reconstruction, and training loss function. Lastly, we design the adaptive control scheme into a DRL-based model and give its details in Section 3.3.

3.1 System Overview

We introduce the workflow to help understand the characteristics in Fig. 1, which consists of the following four core components.

(1) Multiple Camera Views. Generally, we employ multiple depth-cameras placed at different angles to capture raw point clouds and synchronize the point cloud streamings from each camera to a high-performance edge server using USB cables for pre-processing.

(2) Edge Server. Multiple point cloud streams from different views need to be spliced due to redundant overlapped information. Besides, the edge server plays the role of extracting deep semantic features from the spliced point cloud using feature extraction. Note that the edge server uses an adaptive control scheme to sense the connected terminals' network conditions before extracting and transferring features. The scheme then decides on the optimal inference model matching the current network condition.

(3) Base Station. AITransfer provides real-time point cloud video delivery in the current network environment. Point cloud semantic features are transferred over wireless connections to various terminals using existing base stations.

(4) Terminals. AITransfer is deployable on various terminals and usable in a wide range of scenarios. For instance, we use AITransfer to implement real-time holographic communication on a smartphone. A more immersive experience is to use head-mounted displays (HMDs) (e.g., Nreal [19], HoloLens [20]) to render the point cloud, and the users can interact with such immersive point cloud video.

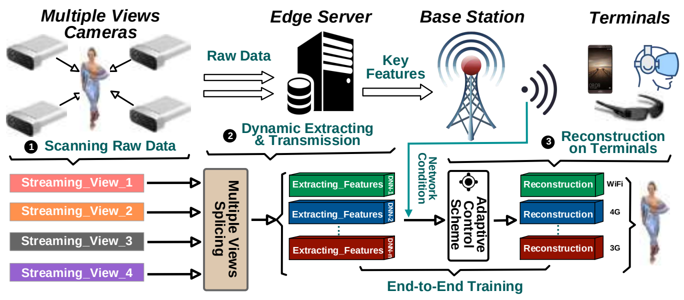

Fig. 1. System workflow and components of AITransfer

AITransfer's workflow consists of offline end-to-end neural network training and online resilient transmission.

Offline Phase. For the offline training, we use the spliced point cloud video as the input and ground truth. AITransfer infers reconstructed results and defines the loss function to complete end-to-end training. Note that the network condition in Fig. 1 is also a hyperparameter during the training, and plays the role in adjusting the volume of extracted semantic features for online transmission, which matches the dynamic network condition. After training a number of end-to-end neural networks with different network condition hyperparameters, we use dynamic network bandwidth as the input and train another DRL neural network in adaptive control scheme to select the optimal inference model from the candidate set.

Online Phase. During the online phase, AITransfer first processes various point cloud streamings to obtain a complete spliced result and then extracts semantic features using the selected inference model. Next, the terminals reconstruct the point cloud video using feature reconstruction to recover these transmitted features. To achieve instant network condition sensing, once the communication channel is established between the sender and the receiver, the adaptive control scheme senses the network condition and takes it into the DRL network to complete the forward inference. The scheme adaptively switches the optimal inference model (i.e., with the purpose of maximizing transmission performance) to execute semantic features' extraction and reconstruction. Overall, AITransfer's AI-powered transmission architecture with a DRL-based adaptive control scheme provides excellent capability to remarkably reduce network bandwidth and energy consumption in dynamic environments.

3.2 Design of the Transmission Network Architecture

3.2.1 *Hierarchical Feature Extraction*

As shown in the top half of Fig. 2, we design a hierarchical extraction architecture based on the backbone of PointNet++ [17]. We first explain why the PointNet++ architecture is leveraged as it only deals with point clouds and not videos directly. (i) PointNet++ has the capability of directly handling point cloud inputs for feature extraction. (ii) At present, existing 3D tree-based point cloud video streaming systems [2], [4], [6] transfer video as a sequence of individually compressed frames. This is because directly compressing the dynamic point cloud while considering the incremental differences between frames is very difficult. This may be a research point further studied in future work.

Specifically, we adopt the multiple layers called Set Abstraction Levels (SAL) for hierarchical feature learning due to the invariance of the unordered point set' permutation, capturing the local structure of the raw point cloud. Thus, the input point set is represented more abstractly with a smaller number of points and features when passing a basic SAL. After multiple layers of feature extraction, the point cloud is represented by a point-wise semantic feature matrix (N, M) for subsequent transmission. A basic SAL consists of three essential layers: Sampling Layer, Grouping Layer, and Mini-PointNet Layer [17]. More precisely, the Sampling Layer selects a point subset from the output of the previous layer, representing the centroids of local regions. The Grouping Layer constructs local region sets by finding a certain number of nearest neighbors around the centroids, and the Mini-PointNet Layer uses three 2D convolution layers and one max pooling layer to encode local region patterns into feature vectors. We set three SALs' number of centroids as 128, 64, and N for gradual extraction. Empirically, the numbers of out channels in the Mini-PointNet Layer are set as $[64, 64, 64]$, $[64, 64, 32]$, and $[32, 32, M]$. We keep other parameters as the default configuration as found in [17].

3.2.2 *Feature Expansion Reconstruction*

We design a relatively lightweight and efficient feature expansion module for point cloud reconstruction, considering that advanced GAN-based method like PU-GAN [21] is too heavy and cumbersome to be deployed in mobile or resource-constrained terminals. As mentioned in [22], employing the generator of PU-GAN can achieve an approximately smaller model than 10 MB. Hence, we adopt the Feature Expansion component and Point Set Generation component in the generator of PU-GAN, as shown in the bottom half of Fig. 2. The received semantic feature matrix (N, M) is first delivered to a multilayer perceptron (MLP) layer for unifying the dimension to $(N, 128)$. Besides, its features are expanded to $(256, 128)$ by an up-down-up expansion unit. This unit's design can produce more diverse point distributions to enhance the feature variations rather than a simple duplication strategy in PU-Net [23]. Then, the expanded feature is reconstructed into the point set with 3D coordinates by two MLP layers. Moreover, more details about parameter settings and the structure of the up-down-up expansion unit can be referred to in [21].

3.2.3 *End-to-End Training*

We first discuss two representative loss definitions in processing conventional point clouds, including the repulsion loss and uniform loss, and explain the non-universality for AITransfer. The repulsion loss avoids the generated points located near the original points, which is described in [23]:

$$\mathcal{L}_{\text{rep}} = \sum\nolimits_{i=1}^{\hat{N}} \sum\nolimits_{i' \in K(i)} \eta(\|x_{i'} - x_i\|) w(\|x_{i'} - x_i\|), \quad (1)$$

where $\hat{N}$ denotes the number of output points, $K(i)$ is the index set of the k-nearest neighbors of point x_i. $\eta(r)$ and $w(r)$ are the repulsion term and fast-decaying weight function. Also, the uniform loss is to improve the generative ability to generate point sets in a uniform distribution, which is described in [21]:

$$\mathcal{L}_{\text{uni}} = \sum\nolimits_{j=1}^{M} \frac{(|S_j| - \hat{n})^2}{\hat{n}} \cdot \sum\nolimits_{k=1}^{|S_j|} \frac{(|d_{j,k} - \hat{d}|)^2}{\hat{d}}. \quad (2)$$

The former and the latter terms account for the nonlocal and local distribution uniformity, respectively. S_j denotes a point subset in a patch, and $\hat{n}$ is the expected number. $d_{j,k}$ represents the distance from each point in S_j to its nearest neighbor. Note that it follows the chi-squared model to measure the deviation of $d_{j,k}$ from $\hat{d}$.

However, in this work, AITransfer aims to train an end-to-end neural network to minimize the distance between the outputs and inputs as much as possible, which is fundamentally different from the existing advanced methods' task. Therefore, we use the earth mover's distance (EMD) [24] as the reconstruction loss to encourage the generated points to lie on the target surface and be similar to original input data, which can be calculated by:

$$\mathcal{L}_{\text{rec}} = \min\nolimits_{\phi: P \to Q} \sum\nolimits_{p_i \in P} \|p_i - \phi(p_i)\|_2, \quad (3)$$

where $\phi : P \to Q$ represents the mapping from the input to the output. Furthermore, we provide the whole formulation of the training loss in AITransfer as follows:

$$\mathcal{L}(\theta) = \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \|\theta\|^2, \quad (4)$$

where λ_{rec} represents the weight, and $\|\theta\|^2$ represents regularization. In Section 4.2.2, we conduct experiments on the influence of the above loss functions to illustrate our design's effectiveness. Although the repulsion loss and uniform loss make a notable contribution to the point cloud upsampling task, they are not ideally suited to the point cloud video transmission scenario.

To train AITransfer with the discussed loss function, we employ the patch-based training strategy. More precisely, each 3D training mesh model is decomposed of 200 patches, and each patch occupies 5% of whole objects, grouping 256 points and normalized in a unit sphere. As for a point set in the testing phase, we follow the same strategy to use the farthest sampling and extract a local patch with a fixed number of points. Then the patches are fed into AITransfer, compressed, and reconstructed to a point set with the same size as the original input. Lastly, all patches of this point set are combined into the final output for replaying.

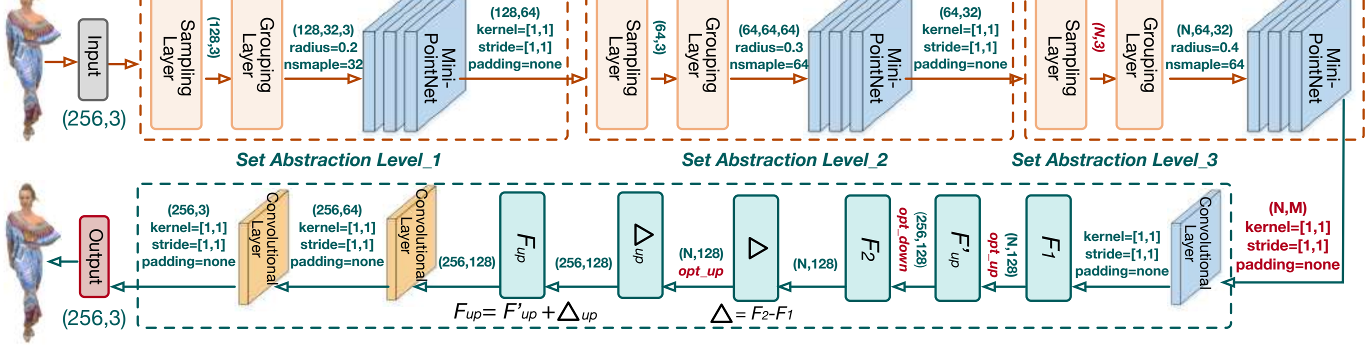

Fig. 2. Design of end-to-end network architecture

3.3 Design of the Adaptive Control Scheme

The adaptive control scheme is committed to matching the most appropriate inference model to various network conditions for providing the best transmission performance. Until now, there exists a significant variance in user preferences for volumetric video quality of experience (QoE) [2]. To the best of our knowledge, how to measure the QoE of point cloud video services still lacks research and clear definition. The current quality assessment tool for point cloud videos is a variant or extension of counterparts from conventional approaches, such as Peak Signal to Noise Ratio (PSNR). Notably, traditional PSNR cannot be directly used because it only represents color information of 2D video [25], and cannot represent position information of point cloud, especially since the number and position of points will change after our semantic-aware compression.

In this work, we conservatively consider two main objective metrics: the transmission latency and users' perceived video quality. To enable real-time point cloud video delivery with high reconstructed video quality, we consider establishing a relationship between the above two metrics to achieve an optimal trade-off. However, in previous research [1], the relationship between latency and quality is difficult to be precisely expressed as a simple formulation. Inspired by recent advances in DRL, we build an adaptive control scheme that learns to select inference models directly from experience. The question arises as to why additional effort should be expended to explore a new adaptive control scheme, when there is an existing one that we previously proposed [1]? The reason is that when the number of candidate models becomes wildly large, it will take an increased amount of time to infer the optimal solution. In Section 4.3.2, we compare these two schemes to illustrate the necessity and efficiency of the DRL-based adaptive control scheme.

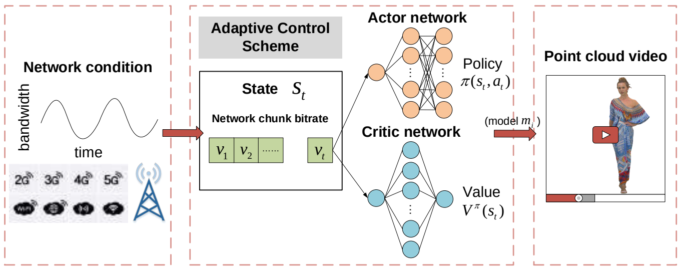

Fig. 3. Design of adaptive control scheme

A variety of DRL algorithms could be considered as the framework of the adaptive control scheme. We choose Asynchronous Advantage Actor-Critic (A3C) [26], because: (1) A3C is a state-of-the-art DRL algorithm, effectively solving the non-convergent problem of Actor-Critic; (2) A3C has been successfully applied to traditional video streaming applications such as adaptive-bitrate control and resource management [27]; (3) the asynchronous parallel training framework supports online training in which many users concurrently send their experience feedback to the agent; and (4) the asynchronous parallel training framework can accelerate the convergence of neural networks, better meeting the real-time requirement in our scenario.

Let $M_s = \{m_1, m_2, ..., m_N\}$ be the trained transmission inference model set, and each model m_i in M_s has a different size of the transmitted feature matrix $\mathcal{F}$. As shown in Fig. 3, the adaptive control scheme's training uses A3C which involves one central brain (global net) and multiple parallel workers (local nets), each worker has two types of neural networks: actor network and critic network. We describe the detailed functionalities below.

State: Adaptive control scheme's learning agent takes state input $s_t = v_t$ to its neural networks, where v_t is the network transmission bitrate at time t.

Action: After receiving s_t, the agent takes an action a_t that corresponds to the transmission inference model for the next video chunk (i.e., a group of point cloud frames), represented as $a_t = m_t$, where $m_t \in M_s$.

Reward: Supposing in one video chunk t, the edge server sends n_t frames of video to the receiver, then the transmission latency T_t is calculated as:

$$T_t = \frac{\mathcal{F}(m_t) \cdot n_t}{v_t}, \tag{5}$$

where $\mathcal{F}(m_t)$ represents the data volume of semantic feature matrix (N, M) for transmission.

Note that in this work, we only focus on transferring the coordinate (X, Y, Z) of the point cloud without taking (R, G, B) information into the design of AITransfer. Traditional PSNR is not suitable for measuring the quality of videos without color information under this transmission framework. Thus, we measure the users' perceived video quality by the point-to-point distance between input and output point cloud in each frame like in [21], meaning the similarity from the reconstructed video to the original video.

We define A_t as the reconstruction accuracy representing the video quality of chunk t. Intuitively, the larger of the size of the feature matrix $\mathcal{F}(m_i)$ has, the more accurate the reconstruction performance becomes. This is because a large size of $\mathcal{F}(m_i)$ means more informative semantic features extracted from the original point cloud can be retained for subsequent decompression. Also, the accuracy is influenced by the data volume, and by the specific values in it.

We calculate the accuracy A_t by adopting two kinds of commonly used point-to-point distances, Chamfer distance (CD) [28] and Hausdorff distance (HD) [29], which are respectively defined as:

$$\begin{aligned} d_{CD}(S_1, S_2) = & \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 \\ & + \frac{1}{|S_2|} \sum_{y \in S_2} \min_{x \in S_1} \|y - x\|_2^2 \end{aligned} \tag{6}$$

$$\begin{aligned} & d_{HD}(S_1, S_2) = \max(h(S_1, S_2), h(S_2, S_1)) \\ & \begin{cases} h(S_1, S_2) = \max\limits_{x \in S_1} \min\limits_{y \in S_2} \|x - y\| \\ h(S_2, S_1) = \max\limits_{y \in S_2} \min\limits_{x \in S_1} \|y - x\| \end{cases}, \end{aligned} \tag{7}$$

where S_1 and S_2 are two point sets. The smaller the metric values are, the better the reconstruction results are. Let $\mathcal{J}$ be the number of point cloud frames in each video chunk, the average reconstruction accuracy A_t is represented as:

$$A_t = \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} d_{CD}(S_1, S_2), \tag{8}$$

or

$$A_t = \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} d_{HD}(S_1, S_2). \tag{9}$$

In reinforcement learning, the design of reward plays an important role in helping neural networks achieve convergence fast and obtain a global optimal solution. As a matter of experience, we define the QoE in Eq. (10) as the reward, where $w_T, w_A \in \mathbb{R}_+$ represent the weights of latency and accuracy, respectively:

$$QoE_t = -w_T T_t - w_A A_t. \tag{10}$$

Actor network: The agent takes an action a_t upon receiving s_t based on a policy, which is a probability distribution over actions: $\pi(s_t, a_t) = P(a_t|s_t) \rightarrow [0, 1]$. In practical problems, the transmission rate is a continuous real number and there will be too many pairs of (s_t, a_t). To overcome this, we use a neural network (actor network) to output the policy, because it can take input directly from observation without any hand-crafted features and the number of policy parameters is easily manageable by the neural network. As is well-known, the primary purpose of a DRL agent is to maximize the expected cumulative discounted reward received from the environment. A3C algorithm trains its policy based on the policy gradient method [26]. The gradient of the cumulative discounted reward with respect to (w.r.t) the policy parameters θ is calculated as:

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r_t] = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) A^{\pi_\theta}(s, a)], \tag{11}$$

where γ is the discounted factor. $A^{\pi_\theta}(s, a)$ is the advantage function, which represents the difference between the expected total reward deterministically taking action a in state s and the expected reward for actions drawn from policy π_θ. The common activation function of policy function $\pi_\theta(s, a)$ is a softmax function. Building on actor-critic, A3C also adds an entropy regularization term $H(\cdot)$ to the actor's update rule for helping the agent converge to a better policy [26]. In summary, the actor's accumulative gradient update is:

$$\theta \leftarrow \theta + \sum_t \nabla_\theta \log \pi_\theta(s_t, a_t) A(s_t, a_t) + c\nabla_\theta H(\pi(s_t; \theta)). \tag{12}$$

Critic network: A critic network merely helps to train the actor network. In online testing, only the actor network is required to output the optimal transmission inference model. To calculate $A(s_t, a_t)$, we need an estimation of the value function $V^{\pi_\theta}(s)$, the expected total reward starting at state s and following the policy π_θ. The critic network will learn the estimation of $V^{\pi_\theta}(s)$ from empirically observed rewards. The mean-square loss function is used to update the critic network parameter θ_v.

$$\theta_v \leftarrow \theta_v - \sum_t \nabla_{\theta_v}(r_t + \gamma V^{\pi_\theta}(s_{t+1}; \theta_v) - V^{\pi_\theta}(s_t; \theta_v))^2 \tag{13}$$

Since A3C is asynchronous and multi-threaded, we describe one of the threads in Algorithm 1 to clarify the details.

Algorithm 1: Train the adaptive control scheme.

```
Input  : Global net parameters $\theta$, $\theta_v$;
         local net parameters $\theta'$, $\theta'_v$;
         maximum episode $T_{max}$;
Output: Global net parameters $\theta$, $\theta_v$.
1  $T \leftarrow 0$;
2  while $T \leq T_{max}$ do
3      Trace ← [];                      /* Clear trace list;
4      $t \leftarrow 1$;
5      $d\theta \leftarrow 0$; $d\theta_v \leftarrow 0$;          /* Reset gradients;
6      $\theta' \leftarrow \theta$; $\theta'_v \leftarrow \theta_v$;          /* Synchronize thread;
7      while $s_t \neq$ 'terminal' do
8          $a_t \leftarrow \pi(a_t|s_t; \theta')$;     /* Take $a_t$ by policy;
9          $(s_{t+1}, r_t) \leftarrow a_t$;        /* Receive $r_t$ and $s_{t+1}$;
10         $t \leftarrow t + 1$;
11     Trace ← $[s_1, a_1, r_2, ..., r_t, s_t]$;
12     $\theta' \leftarrow$ AccGrad($\theta'$);
         /* Accumulate gradients w.r.t $\theta'$ by Eq. (12);
13     $\theta'_v \leftarrow$ AccGrad($\theta'_v$);
         /* Accumulate gradients w.r.t $\theta'_v$ by Eq. (13);
14     $\theta \leftarrow \theta'$; $\theta_v \leftarrow \theta'_v$;
15     $T \leftarrow T + 1$;
16 return $\theta$, $\theta_v$;
```

4 Evaluation

We evaluate AITransfer from the system-level performance and in-depth analysis to answer the following three questions: (1) Does AITransfer enable a high compression ratio while guaranteeing a considerable experience? (2) How AITransfer performs when compared with conventional point cloud compression methods? (3) Can AITransfer provide adaptive sensing and transmission under a dynamic network environment? We first introduce the datasets, the baselines, and then the detailed experimental settings.

Datasets. (1) We use 145 3D mesh models from the released dataset by PU-GAN [21], including a variety of diverse objects with different point set sizes. We select 40 simple, 40 medium, and 40 complex models for training AITransfer and the rest of the 25 models are used for testing. (2) To train the A3C network in the adaptive control scheme and test its performance, we collect 4 large-scale dynamic point cloud video datasets [10]. There are four sequences in the dataset, known as *longdress*, *loot*, *redandblack*, and *soldier*. In each sequence, the full body of a human subject is captured by 42 RGB cameras at 30 FPS, over a 10 s period. For simplicity of evaluating the reconstruction accuracy of each video frame, we uniformly preprocess all human subjects to 100,000 points by using the farthest point sampling technique.

Baselines. Several existing point cloud video streaming systems [2], [4], [15], [16] adopt the 3D tree-based compression method to independently process each point cloud frame, and the total video is transmitted as a sequence of individually compressed frames. More importantly, they target the viewport prediction and tiling mechanism, which adaptively transfers parts of the video content. For fair comparability, we pay more attention to conducting comparisons on compression method rather than adaptive mechanisms for system-level evaluation. We compare AITransfer with two conventional compression methods: the octree-based method [30], Draco [14], and a representative deep learning-based approach, Geo-CNN [31]. Conventional compression methods generally specialize in data structure and remove redundancy by compressing the number of data bits. We unify the original point cloud coordinates into 15 bits before executing the compression for a fair comparison.

- Octree [30] is a representative compression method for point sampled models based on an octree decomposition of space, which is applied in PCL. We use Oct (d=5) and Oct (d=10) to denote the depth of 5 and 10, meaning 3x and 1.5x quantization compression ratios.
- Draco [14] is a popular library for compressing and decompressing 3D geometric meshes and point clouds. We use Draco (d=5) and Draco (d=10) to represent the quantization parameter (qp) of 5 and 10, meaning 3x and 1.5x compression ratios. We set the compression level (cl) and other parameters as default.
- Geo-CNN [31] is a data-driven geometry compression for static point clouds based on learned convolutional transforms and uniform quantization. Since the training and testing datasets need to be converted to a voxel grid with a fixed size before compression, we retrained the network using the default 64 resolution. It compresses a 64^3 voxel grid into an 8^3 feature with 32 channels, meaning a 16x compression ratio.

Experimental Settings. We implement AITransfer using TensorFlow and train it on a high-performance server with eight Tesla V100-PCIE-32GB GPUs. (1) For the end-to-end network training settings, we set the training epoch as 200 and define the parameter β as 0.9 with an optimizer of Adam. The initial learning rate is set as 0.001 and reduced by a decay rate of 0.7 per 50k iterations until 10^{-6}. The batch size is set as 28, λ_{rec} is set as 100. (2) For the DRL network training settings, we set the number of threads as 5, discount factor γ as 1, and the weight of entropy c as 0.001. The weights of latency and accuracy in QoE are 0.6 and 0.4. One video chunk t is 1 s, and n_t is 10. The learning rates of the actor network and critic network are set to 0.0005.

4.1 System-Level Evaluation

4.1.1 *Qualitative Comparisons*

We compare the reconstruction results of AITransfer (5-5) and AITransfer (15-15) with baselines. Since each patch of the object is composed of (256, 3) information, AITransfer (5-5) and AITransfer (15-15) denote 30.72x and 3.41x compression ratios, respectively. We give some representative visual comparison examples including Star (2502 points), Tiger (58370 points), Gramme-aligned (249366 points), Statue-rome-aligned (500506 points), and Statue-dragon-aligned (997892 points) in Fig. 4.

We observe that: (1) Oct (d=10) and Draco (qp=10) output uniform results that are similar to the inputs in all cases, while Oct (d=5) and Draco (qp=5) perform a nonuniform and ”blocky” phenomenon. This is because when the number of each point’s bits is 10, there is still enough precision to represent the coordinate information. When the depth drops to 5, some points close to each other overlap, and the number of points with diacritical coordinates reduces exponentially. (2) Geo-CNN’s outputs look fuzzy and distorted with a resolution of 64. It tends to produce more noisy points and loses many surface details at the edge of objects. (3) For AITransfer, when dealing with sparse point sets, AITransfer (5-5) may produce fewer irregular points and lightly lose some fine-grained details, such as Star’s angles and Tiger’s tail. This is because a (256, 3) point set is compressed into just a (5, 5) feature matrix, and the spatial distribution information of the original input point set is transferred with high limitation. AITransfer (15-15) further alleviates this kind of distortion. Mainly, when dealing with denser point sets, the reconstruction results show that AITransfer can acquire all uniform and almost undistorted point clouds. In summary, the qualitative evaluation demonstrates that AITransfer can achieve at least a

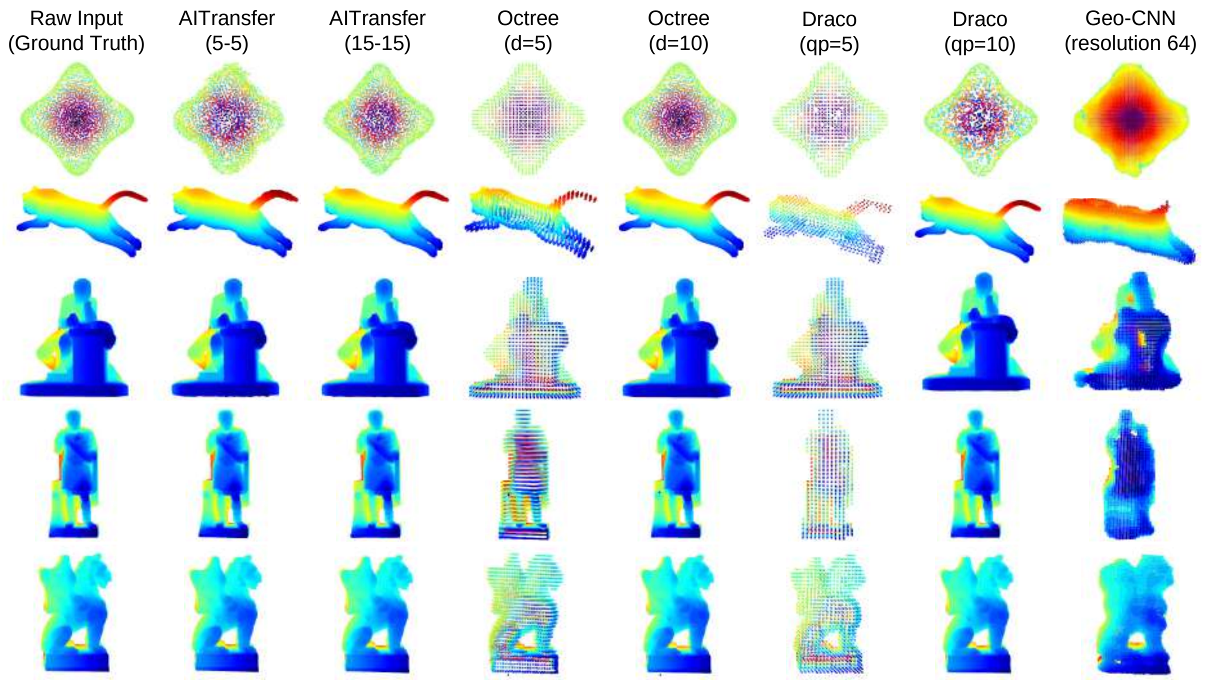

Fig. 4. Qualitative comparisons on the reconstruction results with other methods

30.72x compression ratio while ensuring a visually similar reconstruction result against the ground truth on sparse point sets and nearly lossless results on denser ones. This is promising to transfer intensive point cloud video across the existing network environment including 5G networks.

4.1.2 *Quantitative Comparisons*

We employ three commonly used metrics: (1) Chamfer distance (CD) [28], (2) Hausdorff distance (HD) [29] and (3) point-to-surface distance (P2F) [32] for quantitative evaluation. The number of points in the input and output of Geo-CNN is not precisely consistent, resulting in the inability to calculate these metrics. Thus, we compare AITransfer with Octree and Draco in this part. We conduct comparisons on 25 testing models, sorted by the number of points from the smallest (1023) to the largest (997892) in Table 2.

The results show that: (1) Octree achieves the lowest CD in almost all cases, and the Octree (d=10) is a little bit lower. Compared with Draco (qp=5), AITransfer (15-15) achieves more than 3x compression ratio while performing well on nine of the selected datasets. (2) In view of HD, AITransfer (15-15) achieves lower values than Octree (d=5) on eleven of the selected datasets, and AITransfer (15-15) evaluates lower values than Draco (d=5) on eight of the selected datasets. (3) As for P2F, Draco gets the highest value in each case, and Oct (d=10) is the lowest. At an approximately equal compression ratio, AITransfer (15-15) gets lower results than Oct (d=5) on the last eight of the selected datasets, and these datasets contain a relatively large number of points. AITransfer (5-5) acquires better performance than Oct (d=5) under this circumstance, demonstrating it can obtain more uniform and better 3D surface reconstruction quality on the denser point sets even at an extremely high compression ratio.

To sum up, the quantization results of AITransfer are sometimes inferior to the other two comparison methods, especially on sparse point cloud datasets with the same compression ratio. This is because the compression mechanism in AITransfer is based on semantic-level features, and the output object point cloud is reconstructed by extending these features. When the input point is sparse, each point contributes vital distributed information to a complete 3D model. This compression mechanism will inevitably bring spatial deviation and some noise points. However, Octree and Draco are based on traditional signal-processing mechanisms. Each point in the decompressed point set is offset in its original position, and if the point cloud is sparse, this offset may not result in overlap between points. The above quantitative metrics CD and HD are defined with the point-to-point distance, thus Octree and Draco can sometimes achieve shallow values of CD and HD. Consequently, AITransfer achieves a relatively even and stable performance, and it prefers to be employed in intensive point cloud video transmission tasks.

4.1.3 *Performance of Encoding and Decoding*

AITransfer can significantly reduce the transmitted data volume and latency. Besides, we verify the efficiency of compression and decompression. Since AITransfer has different compression mechanism from other methods, we record the time of feature extraction and reconstruction, respectively, as the encoding and decoding time. For a fair comparison, we compare AITransfer (10-10) with Octree (d=10) and Draco (qp=10) on the *longdress* dataset. The experiment was repeated 1000 times on the Tesla V100 GPU-enabled server. The average time required to encode and decode one raw *longdress* point cloud frame is shown in Table 3.

TABLE 3: Comparisons on average encoding/decoding time

	Octree(d=10)	Draco(qp=10)	AITransfer(10-10)
Enc (s)	0.535663	0.500624	0.092860
Dec (s)	3.225866	0.204626	0.038606

TABLE 2: Quantitative comparisons on the reconstruction results with other methods

Metric	CD (E-02)						HD						P2F					
Dataset	AIT (5-5)	AIT (15-15)	Oct (d=5)	Oct (d=10)	Draco (d=5)	Draco (d=10)	AIT (5-5)	AIT (15-15)	Oct (d=5)	Oct (d=10)	Draco (d=5)	Draco (d=10)	AIT (5-5)	AIT (15-15)	Oct (d=5)	Oct (d=10)	Draco (d=5)	Draco (d=10)
m333	2.0832	1.8319	0.0867	0.0024	1.5944	1.6959	0.1455	0.1597	0.0063	0.0088	0.0734	0.0778	0.0744	0.0531	0.0176	0.0007	0.1998	0.1996
m32	2.2048	1.0666	0.0835	0.0270	1.5143	1.5489	0.1145	0.0663	0.3141	0.2799	0.0710	0.0734	0.0598	0.0366	0.0116	0.0010	0.2973	0.2973
m60	0.7643	0.6505	0.0428	0.0011	1.0653	1.0698	0.0957	0.1037	0.0083	0.0098	0.1101	0.0945	0.0303	0.0201	0.0099	0.0004	0.2502	0.2506
m88	4.3116	4.2842	0.0510	0.0326	1.7599	1.8343	0.2798	0.2873	0.1041	0.0796	0.0852	0.0852	0.0671	0.0393	0.0114	0.0007	0.3065	0.3058
casting	0.7800	0.4989	0.0881	0.0071	0.5243	0.5165	0.0470	0.0303	0.0800	0.0753	0.0483	0.0596	0.0425	0.0266	0.0128	0.0006	0.1409	0.1422
chair	0.6819	0.2338	0.0908	0.0069	0.1887	0.1219	0.0669	0.0395	0.1157	0.1109	0.0074	0.0046	0.0493	0.0243	0.0182	0.0009	0.3971	0.3975
elephant	0.4918	0.3174	0.0776	0.0113	0.1541	0.0946	0.0476	0.0241	0.0630	0.0701	0.0050	0.0057	0.0410	0.0285	0.0173	0.0008	0.1527	0.1525
genus3	0.6637	0.3740	0.1106	0.0037	0.2549	0.1582	0.0617	0.0434	0.0598	0.0814	0.0099	0.0102	0.0503	0.0310	0.0200	0.0008	0.2300	0.2303
eight	0.5122	0.2848	0.0735	0.0014	0.1833	0.0863	0.0450	0.0205	0.0025	0.0008	0.0067	0.0047	0.0421	0.0253	0.0161	0.0005	0.0586	0.0581
elk	1.5497	0.4790	0.0775	0.0006	0.2288	0.2065	0.1025	0.0342	0.0054	0.0064	0.0193	0.0200	0.0588	0.0376	0.0192	0.0007	0.3854	0.3855
kitten	0.5785	0.2363	0.0771	0.0013	0.2548	0.2176	0.0367	0.0327	0.0340	0.0242	0.0105	0.0092	0.0355	0.0209	0.0167	0.0006	0.1444	0.1444
camel	0.5089	0.4981	0.0707	0.0075	0.2113	0.2073	0.0366	0.0350	0.1279	0.1168	0.0114	0.0172	0.0278	0.0164	0.0134	0.0007	0.1618	0.1613
coverrear	0.8680	0.3515	0.1278	0.0482	1.1847	1.0059	0.0686	0.0219	0.0060	0.0025	0.0612	0.0540	0.0393	0.0231	0.0113	0.0005	0.0929	0.0934
cow	0.5226	0.2692	0.0714	0.0007	0.3069	0.2640	0.0530	0.0188	0.0027	0.0024	0.0120	0.0102	0.0363	0.0206	0.0146	0.0005	0.0956	0.0954
duck	0.8099	0.4639	0.1752	0.0017	0.2153	0.1471	0.0657	0.0622	0.0118	0.0225	0.0088	0.0089	0.0436	0.0267	0.0235	0.0008	0.2620	0.2623
horse	0.4363	0.2818	0.0766	0.0128	0.2046	0.1682	0.0397	0.0309	0.1592	0.1712	0.0116	0.0089	0.0386	0.0265	0.0168	0.0008	0.3000	0.2994
m329	0.8154	0.5713	0.1059	0.0044	0.3535	0.2986	0.0428	0.0289	0.0894	0.0996	0.0176	0.0159	0.0501	0.0335	0.0188	0.0007	0.1479	0.1473
m355	0.6433	0.4282	0.1190	0.0140	0.2808	0.2432	0.0605	0.0245	0.0040	0.0085	0.0095	0.0089	0.0337	0.0194	0.0201	0.0007	0.1771	0.1771
star	0.4048	0.3064	0.2580	0.0007	0.2951	0.1964	0.0290	0.0106	0.0079	0.0012	0.0118	0.0106	0.0332	0.0152	0.0289	0.0009	0.2518	0.2514
Panda	2.1960	1.9519	0.0699	0.0005	1.9038	2.0126	0.1111	0.0948	0.1238	0.1467	0.0955	0.0909	0.0078	0.0049	0.0210	0.0007	0.1645	0.1644
Tiger	0.8715	0.6563	0.0305	0.0001	0.4805	0.5057	0.0976	0.0692	0.0451	0.0484	0.0452	0.0488	0.0048	0.0023	0.0118	0.0004	0.1213	0.1211
Gramme	0.0753	0.1057	0.0559	0.0001	0.2821	0.2371	0.0046	0.0053	0.0022	0.0005	0.0133	0.0110	0.0022	0.0012	0.0162	0.0006	0.2155	0.2156
rome	0.0200	0.0150	0.0397	0.0001	0.1716	0.0935	0.0014	0.0010	0.0213	0.0219	0.0164	0.0130	0.0010	0.0005	0.0114	0.0004	0.0733	0.0724
ramesses	0.9777	0.8256	0.0440	0.0001	0.6962	0.6875	0.1107	0.1016	0.0021	0.0040	0.0754	0.0677	0.0008	0.0005	0.0132	0.0004	0.1520	0.1519
dragon	0.0787	0.0681	0.0563	0.0001	0.5976	0.5632	0.0065	0.0060	0.0312	0.0315	0.0677	0.0592	0.0009	0.0006	0.0159	0.0006	0.1775	0.1774

The result shows that AITransfer consumes less time on both encoding and decoding than conventional compression methods. The reason is that AI-powered compression can leverage GPUs to accelerate the efficiency of the codec. The inference time of AITransfer is closely tied to the structure of neural networks, so it can be further optimized by network lightweight technologies.

4.2 Understanding AITransfer In-depth

4.2.1 *Analysis of the Compression Ratio*

We trained a variety of models with different sizes of transmitted feature matrix (N, M). The row N and column M of the matrix $\mathcal{F}(m)$ can be arbitrarily set according to actual situations. Hence, we just set the matrix to a square matrix ranging from (05,05) to (20,20) with 5×5 interval as a group of examples, representing 30.72x, 7.68x, 3.41x, and 1.92x compression ratios, roughly exploring the relationship between the reconstruction accuracy and compression ratio. In practice, the relationship between them can be fitted by training more combinations. We show the average performance on 25 testing datasets in Table 4.

TABLE 4: Quantitative comparisons with different matrixes

(N, M)	CD	HD	P2F/avg	P2F/std
(05,05)	0.009540	0.070830	0.034846	0.028539
(10,10)	0.007111	0.057459	0.023528	0.021022
(15,15)	0.006820	0.054102	0.021391	0.019903
(20,20)	0.006624	0.053033	0.020900	0.019328

We conclude that: (1) As the compression ratio reduces from 30.72x to 1.92x, the three metrics CD, HD, and P2F on average, all decrease. This illustrates that the more features our system transfers, the more similar the decompressed point set is to the ground truth, and the higher surface reconstruction performance of AITransfer achieves. (2) We observe that CD, HD, and P2F reduce by 25.46%, 18.88%, and 32.48%, respectively, when changing the size of the (5,5) to (10,10). When compared with that of (10,10), the CD, HD, and P2F with the size of (15,15) further reduce by 4.09%, 5.84%, and 9.08%, respectively. Compared with (15,15), the size of (20,20) only reduces by 2.88%, 1.97%, and 2.30% of the three metrics. (3) The P2F standard deviation shows a similar characteristic of change, illustrating the model's stability with more transmission features. When our system transfers more information from the original point cloud, more features can be retained and received to support the reconstruction on the terminals. (4) The negative correlation may not be a linear gradient strictly. This is because the accuracy is influenced by transmitted data volume and the specific values in it. Note that the training processes among different models are independent of each other, and there is no inclusion relationship between the feature matrixes. Hence, we know that the adaptive control scheme chooses different compression ratios for future video chunk transmission based on forecasts of available bandwidth. Therefore, the model selected by the adaptive control scheme is up to the actual network condition.

4.2.2 *Analysis of the Loss Function*

We validate the EMD reconstruction loss' effectiveness for training AITransfer, comparing it with other representative loss functions in Table 5. We conduct a comparison study taking the (15,15) transmitted semantic feature matrix to evaluate the influence of different loss functions, including CD loss, repulsion loss, and uniform loss. For experimental settings, we replace the EMD with CD as reconstruction loss, add the repulsion loss to the total loss function, and add the uniform loss to the total loss function. We also follow [21] and [23] to set the $w(r)$ in Eq. (1), set the expected percentage in Eq. (2) as $\{0.4\%, 0.6\%, 0.8\%, 1.0\%, 1.2\%\}$, and λ_{rep} and λ_{uni} are set as 1 and 10, respectively.

TABLE 5: Quantitative comparisons with different losses

Loss function	CD	HD	P2F/avg	P2F/std
EMD	0.006820	0.054102	0.021391	0.019903
CD	0.006815	0.053670	0.023411	0.021209
EMD+rep	0.006768	0.056922	0.021842	0.020001
EMD+uni	0.006887	0.055006	0.023659	0.028517

We can see that EMD loss achieves an increase of 0.07% and 0.80% on the CD and HD metrics, and a decrease of 8.63% on the P2F, compared with using CD as the loss function. When compared with using EMD+repulsion as the loss function, we find using a single EMD loss increases by 0.76% of the CD metric, while it decreases by 4.95% and 2.06% on the HD and P2F, respectively. Besides, EMD loss improves 0.97% on the CD metric and decreases 1.64% and 9.58% on the HD and P2F compared with EMD+uniform loss. Also, using EMD loss achieves the smallest standard deviation of P2F. We conclude that employing a single EMD as total loss can achieve considerable performance. Replacing EMD with CD and adding repulsion loss and uniform loss, respectively, into the total loss will not improve the performance in all cases and might even have a negative impact. This is because the repulsion loss and uniform loss are designed for point cloud upsampling to expand the number of points on a sparse point set while making the output look uniform. Besides, adding the latter two losses also increases the training time of the network.

4.3 Demonstrating Adaptive Control Scheme In-depth

In this subsection, we verify whether the adaptive control scheme can provide compliant transmission under a dynamic network environment. For comparison purposes and better convergence when training the DRL network, we first explore the relationship between the reconstruction accuracy and compression ratio by training more inference models. We trained 8 additional inference models, whose transmitted semantic feature matrix sizes and corresponding compression ratios are shown in Table 6.

TABLE 6: Mapping from feature matrix to compression ratio

(N, M)	(06,06)	(08,08)	(10,10)	(12,12)
Ratio	21.33	12.00	7.68	5.33
(N, M)	(14,14)	(16,16)	(18,18)	(20,20)
Ratio	3.92	3.00	2.37	1.92

Due to CD and HD being point-to-point distances with the same order of magnitude, we just measure the reconstruction accuracy of one video frame by taking the average of CD and HD. Then, we calculate the accuracy of each frame in the 4 point cloud video datasets (1200 frames in total) and fitted the relation between accuracy and compression ratio by a polynomial. Without loss of generality, we simulate an environment from a realistic bandwidth trace (0∼100 Mbps) [33] in Fig. 5, where the network condition is constantly changing dynamically as time passes.

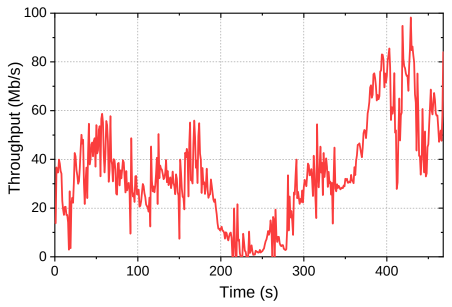

Fig. 5. A realistic example of a bandwidth trace

4.3.1 *Effectiveness of Adaptive Control Scheme*

On the one hand, to test the effectiveness of the adaptive control scheme, we have trained the A3C network and used the trained actor network to infer the transmission model for a new point cloud video. We draw the decision results in 1∼30 s as an example, shown in Fig. 6 (a). The black solid line represents the change of bandwidth over time, and the red points represent the inference models over time selected by the adaptive control scheme.

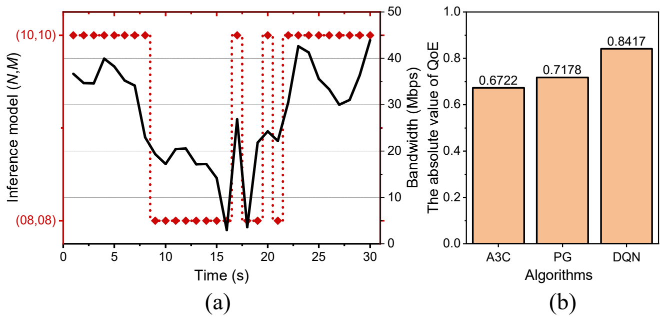

Fig. 6. (a) Selection of inference model as bandwidth changes; (b) Comparison with different RL algorithms

In Fig. 6 (a), when the bandwidth is sufficient, the adaptive control scheme will select an inference model with a large feature matrix to provide a more satisfactory visual quality. Otherwise, it will select a model with a high compression ratio. This kind of adjustment with the same trend as the dynamic change of network condition demonstrates the effectiveness of the adaptive control scheme.

4.3.2 *Efficiency of Adaptive Control Scheme*

On the other hand, to test the efficiency of the adaptive control scheme, we compare the adaptive control scheme based on DRL with the online adapter based on a simple formulation in our previous work [1]. To develop a case study, we experiment on a PC equipped with an AMD Ryzen 5 2600X Six-Core Processor (3.60 GHz), 8 GB RAM, and an Nvidia GTX 1060 6 GB GPU. We record the running time of selecting the inference model in Table 7, where *model-num* denotes the number of inference models; *oa1-sum* and *oa2-sum* respectively denote the total time (for 30 seconds) of the online adapter in [1] and DRL-based adaptive control scheme; *oa1-mean* and *oa2-mean* respectively denote the average time (for once). Note that the results of the DRL-based adaptive control scheme in Table 7 contain the GPU startup and warm-up time. To alleviate the effect by this time and randomness, we conduct each experiment 1000 times to obtain the average.

TABLE 7: Running time of two adaptive control schemes

model-num	8	64	256	1024
oa1-sum	0.004004	0.032029	0.183167	1.548417
oa2-sum	0.023011	0.023195	0.023787	0.023385
oa1-mean	0.000133	0.001068	0.006106	0.051614
oa2-mean	0.000767	0.000773	0.000793	0.000779

As observed in Table 7, when there are only 8 candidate models, the time of *oa1-sum* and *oa1-mean* are respectively less than *oa2-sum* and *oa2-mean*. The reason is that the number of models is so few that it can be quickly traversed once. However, when the number of candidate models increases to 64, 256, 1024, or more, the running time using the online adapter in [1] will grow non-linearly. This is because the previous scheme is based on the sort algorithm, having at least $O(nlogn)$ time complexity. The time using the DRL-based adaptive control scheme does not increase as the number of models increases, because the time is only related to the depth and structure of the neural network.

Compared with our previous work, the DRL-based adaptive control scheme proposed in this paper has the advantages of employing a neural network for direct forward inference to make decisions and leveraging GPUs to accelerate the calculation process, which demonstrates the efficiency of the DRL-based adaptive control scheme.

4.3.3 *Evaluation of A3C Framework*

We compare A3C with the other two RL algorithms: Deep Q Network (DQN) and Policy Gradients (PG) to verify the superiority of A3C. For fairness, we adjust three models' parameters including the learning rate, and set the neural node number of the hidden layer to 96, then we train three algorithms to ensure neural network convergence. We show the absolute value of QoE (i.e., Eq. (10)) evaluated on the testing video in Fig. 6 (b). The smaller the value, the better the QoE. The results show that A3C can help the adaptive control scheme provide a more satisfactory QoE than DQN and PG. Considering the characteristics of A3C mentioned in Section 3.3, we finally choose it in this bandwidth-aware point cloud video delivery scenario.

4.4 Case Study of Holographic Communication

We prototype AITransfer into a case study for enabling holographic communications in Fig. 7. We use four Kinect V2 [5] depth-cameras, respectively, placed at the front left, back left, front right, and back right of interest to collect a surrounding real-time point cloud, and fuse different angles of the camera by using the PCL [13] tool. We use the nearest edge cloud server equipped with the NVIDIA Tesla V100 GPUs to match the network condition and extract point cloud semantic features. We use an Aruba WiFi router to broadcast compressed features by WiFi to the receivers, a laptop and a HoloLens 2. The devices feed the received features into the reconstruction part to complete the inference, render and play the inference result. The playback result shown on the laptop can be seen in Fig. 7.

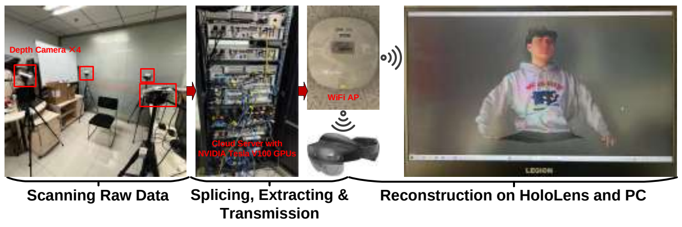
Depth Camera ×4
Cloud Server with NVIDIA Tesla V100 GPUs
WiFi AP
Scanning Raw Data
Splicing, Extracting & Transmission
Reconstruction on HoloLens and PC

Fig. 7. A case study of holographic communication

5 Related Work

Point Cloud Compression. The most intuitive way is to compress point clouds and reduce the transmitted data volume, including geometry compression, attribute compression, and motion-compensated compression [34], [35]. Most of these focus on static *k*d-tree and octree-based solutions. Typical examples include PCL [13] and Draco [14]. Besides, transforming 3D point clouds to 2D maps and compressing them with conventional algorithms may lead to a loss of some key features [36]. Recently, deep learning-based geometry compressions [8], [37] divide raw point clouds into 3D voxels, which results in large consumption for converting raw data to voxels due to a sparse Euclidean space. The existing point cloud compression methods explore compressing from space and pixel characteristics to reduce storage memory, ignoring the point cloud's transmission features. More importantly, the computational overhead of conventional compression techniques is also unacceptable for real-time transmission. Our AITransfer explores compressing the point clouds by extracting key features instead of raw point data and providing real-time transmission services.

Point Cloud Video Streaming. Point cloud video services have attracted growing interest in academia and industry [4], [38], [39], [40]. Most recent works optimize the transmission by extending VR streaming techniques, such as viewport prediction, bitrate adaptation, and tiling mechanism [41], [42]. PCC-DASH [15] is a standards-compliant method for HTTP adaptive streaming of scenes comprising multiple dynamic point clouds. Hosseini and Timmerer [43] reduce the number of point clouds using various sampling schemes and provide dynamic adaptive streaming with octree-based representation. Narwhal [42] maximizes the viewing experiences based on the optimization of the computational and communication resources. ViVo [4] is the first practical three visibility-aware volumetric video streaming method for mobile devices. Zhang *et al.* [44] propose a lightweight edge differential privacy preservation framework, effectively protecting user privacy and ensuring service delay, which is of great significance to the security of point cloud video communications. These schemes have optimized conventional frameworks of video delivery, but do not fully exploit the point cloud video features and degrade the transmission. Our work is fundamentally different from these efforts, which uses AI technology to deeply analyze the point cloud's semantic features for efficient transmission.

Reinforcement Learning on Video Streaming. Reinforcement learning (RL) has been a powerful means of resource management and bitrate adaptation in 2D video streaming. Pensieve [27] is a system that generates ABR algorithms using RL, it trains a neural network to select bitrates for future video chunks based on observations collected by client video players. In most recent years, RL has been further extended to 360-degree video and VR video streaming [45], [46], [47]. DRL360 [45] is a DRL-based framework for 360-degree video streaming, which helps improve the system performance by jointly optimizing multiple QoE objectives across a broad set of dynamic features. Kan *et al.* [46] propose a DRL-based rate adaptation algorithm for adaptive 360-degree video streaming. Du *et al.* [47] propose a DRL-based approach to learn the optimal viewport rendering offloading and transmit power control policies for high-quality immersive VR video services. Revisiting the above works, RL has been widely analyzed in conventional video streaming, but to the best of our knowledge, it is now more seldom to find research avenues focused on point cloud video streaming. Deploying an RL-based optimizer in real-time systems has to confront additional challenges.

6 Discussion

We discuss the superiority and some limitations of AITransfer. First, AITransfer achieves a higher compression ratio and real-time requirements with flexibility than other point cloud video streaming systems. On the one hand, AITransfer extracts and transfers the deep semantic features instead of geometrical raw data, obtaining a significant improvement in transmission with a small data volume. On the other hand, AITransfer fuses the training process of reconstructing the features back to the original point cloud into the feature extracting phase for an end-to-end training. Second, AITransfer adapts to different network conditions and matches the optimal model within an impressive response time. Third, we find that AITransfer cannot provide high quantitative performance for some sparse point clouds. This illustrates that sparse point cloud contains more informative key features, and AITransfer has difficulty recovering the complete point cloud from the insufficient features in the reconstruction stage. A potential way is to explore a well-designed neural network to address the shortcomings of the reconstructed model, closing the gap between the output and raw input point cloud. Besides, this paper fundamentally provides a basic AI-powered transmission framework and can be extended with other techniques such as feature extraction and expansion that can actually be adjusted by other advanced deep learning network modules.

7 Conclusion

In this paper, we design and implement AITransfer to explore a semantic-aware transmission mechanism different from traditional bulky video transmission frameworks. It allows the critical transmission of point cloud semantic features at the sending side, significantly reducing the amount of transmitted data and making it more suitable in existing network environments. Also, it provides lightweight point cloud reconstruction on the receiver side to obtain a visual result similar to the original point cloud. Furthermore, AITransfer considers the dynamic and unstable nature of the network environment, incorporates it into the end-to-end network design, and provides an adaptive transmission control scheme to balance the trade-off between latency and quality. In future work, we will explore more lightweight and efficient decoding technologies than AITransfer to deploy them on mobile devices in a wide range of fields.

Acknowledgments

The authors would like to thank Jacky Cao for his diligent proofreading of this article. This research was supported in part by the Funds for International Cooperation and Exchange of NSFC under Grant 61720106007, in part by the National Natural Science Foundation of China under Grant 62202065, in part by the Project funded by China Postdoctoral Science Foundation 2022TQ0047 and 2022M710465, in part by the Fundamental Research Funds for the Central Universities under Grant 2019XD-A03-3, in part by the BUPT Excellent Ph.D. Students Foundation under Grant CX2022128.

References

[1] Y. Huang, Y. Zhu, X. Qiao, Z. Tan, and B. Bai, "Aitransfer: Progressive ai-powered transmission for real-time point cloud video streaming," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 3989–3997.

[2] Z. Liu, Q. Li, X. Chen, C. Wu, S. Ishihara, J. Li, and Y. Ji, "Point cloud video streaming: Challenges and solutions," *IEEE Network*, vol. 35, no. 5, pp. 202–209, 2021.

[3] A. Clemm, M. T. Vega, H. K. Ravuri, T. uters, and F. De Turck, "Toward truly immersive holographic-type communication: Challenges and solutions," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 93–99, 2020.

[4] B. Han, Y. Liu, and F. Qian, "Vivo: Visibility-aware mobile volumetric video streaming," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–13.

[5] "Azure kinect dk. online [2021]," https://azure.microsoft.com/en-us/services/kinect-dk/.

[6] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim, "Groot: a real-time streaming system of high-fidelity volumetric videos," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[7] X. Sun, S. Wang, and M. Liu, "A novel coding architecture for multi-line lidar point clouds based on clustering and convolutional lstm network," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[8] T. Huang and Y. Liu, "3d point cloud geometry compression on deep learning," in *Proceedings of the 27th ACM International Conference on Multimedia (MM)*, 2019, pp. 890–898.

[9] E. S. Jang, M. Preda, K. Mammou, A. M. Tourapis, J. Kim, D. B. Graziosi, S. Rhyu, and M. Budagavi, "Video-based point-cloud-compression standard in mpeg: from evidence collection to committee draft [standards in a nutshell]," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 118–123, 2019.

[10] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies-a voxelized point cloud dataset," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, vol. 7, p. 8, 2017.

[11] R. Schnabel and R. Klein, "Octree-based point-cloud compression." in *PBG@ SIGGRAPH*, 2006, pp. 111–120.

[12] E. Hubo, T. Mertens, T. Haber, and P. Bekaert, "The quantized kd-tree: Efficient ray tracing of compressed point clouds," in *2006 IEEE Symposium on Interactive Ray Tracing*. IEEE, 2006, pp. 105–113.

[13] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2011, pp. 1–4.

[14] "Draco 3d graphics compression. online[2021]," https://google.github.io/draco/.

[15] J. van der Hooft, T. Wauters, F. De Turck, C. Timmerer, and H. Hellwagner, "Towards 6dof http adaptive streaming through point cloud compression," in *Proceedings of the 27th ACM International Conference on Multimedia (MM)*, 2019, pp. 2405–2413.

[16] L. Wang, C. Li, W. Dai, J. Zou, and H. Xiong, "Qoe-driven and tile-based adaptive streaming for point clouds," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 1930–1934.

[17] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[19] "Nreal - building mixed reality for everyone. online [2021]," https://www.nreal.ai/.

[20] "Microsoft hololens. mixed reality technology for business. online [2021]," https://www.microsoft.com/en-us/hololens.

[21] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: a point cloud upsampling adversarial network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7203–7212.

[22] Y. Qian, J. Hou, S. Kwong, and Y. He, "Deep magnification-arbitrary upsampling over 3d point clouds," *arXiv preprint arXiv:2011.12745*, 2020.

[23] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: Point cloud upsampling network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2790–2799.

[24] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[25] S. Winkler and P. Mohandas, "The evolution of video quality measurement: From psnr to hybrid metrics," *IEEE transactions on Broadcasting*, vol. 54, no. 3, pp. 660–668, 2008.

[26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[27] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2017, pp. 197–210.

[28] M. A. Butt and P. Maragos, "Optimum design of chamfer distance transforms," *IEEE Transactions on Image Processing*, vol. 7, no. 10, pp. 1477–1484, 1998.

[29] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva, "A benchmark for surface reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 2, pp. 1–17, 2013.

[30] R. Schnabel and R. Klein, "Octree-based point-cloud compression." *Spbg*, vol. 6, pp. 111–120, 2006.

[31] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 4320–4324.

[32] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *IEEE Visualization, 2002. VIS 2002.* IEEE, 2002, pp. 163–170.

[33] J. Van Der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "Http/2-based adaptive streaming of hevc video over 4g/lte networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[34] M. Krivokuća, P. A. Chou, and M. Koroteev, "A volumetric approach to point cloud compression–part ii: Geometry compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 2217–2229, 2019.

[35] P. A. Chou, M. Koroteev, and M. Krivokuća, "A volumetric approach to point cloud compression—part i: Attribute compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 2203–2216, 2019.

[36] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5828–5839.

[37] A. F. Guarda, N. M. Rodrigues, and F. Pereira, "Point cloud geometry scalable coding with a single end-to-end deep learning model," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 3354–3358.

[38] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1079–1092, 2018.

[39] Y. Zhang, G. Cui, S. Deng, F. Chen, Y. Wang, and Q. He, "Efficient query of quality correlation for service composition," *IEEE Transactions on Services Computing*, vol. 14, no. 3, pp. 695–709, 2018.

[40] Y. Zhang, K. Wang, Q. He, F. Chen, S. Deng, Z. Zheng, and Y. Yang, "Covering-based web service quality prediction via

neighborhood-aware matrix factorization," *IEEE Transactions on Services Computing*, vol. 14, no. 5, pp. 1333–1344, 2019.

[41] F. Qian, B. Han, J. Pair, and V. Gopalakrishnan, "Toward practical volumetric video streaming on commodity smartphones," in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2019, pp. 135–140.

[42] J. Li, C. Zhang, Z. Liu, W. Sun, W. Hu, and Q. Li, "Narwhal: a dash-based point cloud video streaming system over wireless networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 1326–1327.

[43] M. Hosseini and C. Timmerer, "Dynamic adaptive point cloud streaming," in *Proceedings of the 23rd Packet Video Workshop*, 2018, pp. 25–30.

[44] Y. Zhang, J. Pan, L. Qi, and Q. He, "Privacy-preserving quality prediction for edge-based iot services," *Future Generation Computer Systems*, vol. 114, pp. 336–348, 2021.

[45] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "Drl360: 360-degree video streaming with deep reinforcement learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1252–1260.

[46] N. Kan, J. Zou, K. Tang, C. Li, N. Liu, and H. Xiong, "Deep reinforcement learning-based rate adaptation for adaptive 360-degree video streaming," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 4030–4034.

[47] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, "Mec-assisted immersive vr video streaming over terahertz wireless networks: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9517–9529, 2020.

Yuanwei Zhu is currently working towards a Ph.D. degree at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. His current research interests include point clouds, video streaming transmission, and deep reinforcement learning.

Yakun Huang received a Ph.D degree in computer science from Beijing University of Posts and Telecommunications, in 2021. He has authored or co-authored over 10 technical papers in international journals and at conferences, including the IEEE Transactions on Mobile Computing, the IEEE Transactions on Service Computing, the IEEE Network, INFOCOM, ICDCS, MM. His current research interests include video streaming, mobile computing, edge computing, and distributed deep learning.

Xiuquan Qiao is currently a Full Professor at the Beijing University of Posts and Telecommunications, Beijing, China, where he is also the Deputy Director of the Key Laboratory of Networking and Switching Technology, Network Service Foundation Research Center of State. He has authored or co-authored over 60 technical papers in international journals and at conferences, including the IEEE Communications Magazine, Proceedings of IEEE, Computer Networks, IEEE Internet Computing, the IEEE Transactions On Automation Science and Engineering, and the ACM SIGCOMM Computer Communication Review. His current research interests include the future Internet, services computing, computer vision, distributed deep learning, augmented reality, virtual reality, and 5G networks. Dr. Qiao was a recipient of the Beijing Nova Program in 2008 and the First Prize of the 13th Beijing Youth Outstanding Science and Technology Paper Award in 2016. He served as the associate editor for the magazine Computing (Springer) and the editor board of China Communications Magazine.

Zhijie Tan is currently working towards an M.S. degree at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. His research area now focuses on simultaneous localization and mapping (SLAM), sensor fusion, and computer vision.

Boyuan Bai is currently working towards a Ph.D. degree at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. His current research interests include point clouds, video streaming transmission, and deep reinforcement learning.

Huadong Ma (M99-SM16, IEEE Fellow) received a Ph.D. degree in computer science from the Institute of Computing Technology in 1995. He is a Professor at the Beijing University of Posts and Telecommunications, China. His current research focuses on multimedia system and networking, Internet of Things, and sensor networks. He has published over 300 papers in prestigious journals (such as ACM/IEEE Transactions) and conferences (such as ACM SIGCOMM, MobiCom, IEEE INFOCOM). He is an Editorial Board Member of the IEEE TMM, IEEE IoT Journal, ACM T-IoT, and MTAP. He serves for Chair of ACM SIGMOBILE China.

Schahram Dustdar (Fellow, IEEE) was an Honorary Professor of Information Systems at the Department of Computing Science, University of Groningen, Groningen, The Netherlands, from 2004 to 2010. From 2016 to 2017, he was a Visiting Professor at the University of Sevilla, Sevilla, Spain. In 2017, he was a Visiting Professor at the University of California at Berkeley, Berkeley, CA, USA. He is currently a Professor of Computer Science with the Distributed Systems Group, Technische Universität Wien, Vienna, Austria. Dr. Dustdar was an elected member of the Academy of Europe, where he is the Chairman of the Informatics Section. He was a recipient of the ACM Distinguished Scientist Award in 2009, the IBM Faculty Award in 2012, and the IEEE TCSVC Outstanding Leadership Award for outstanding leadership in services computing in 2018. He is the Co-Editor-in-Chief of the ACM Transactions on Internet of Things and the Editor-in-Chief of Computing (Springer). He is also an Associate Editor of the IEEE TRANSACTIONS ON SERVICES COMPUTING, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the ACM Transactions on the Web, and the ACM Transactions on Internet Technology. He serves on the Editorial Board of IEEE INTERNET COMPUTING and the IEEE Computer Magazine.