Digital-Twin-Based 3-D Map Management for Edge-Assisted Device Pose Tracking in Mobile AR

Conghao Zhou, *Member, IEEE*, Jie Gao, *Senior Member, IEEE*, Mushu Li, *Member, IEEE*, Nan Cheng, *Senior Member, IEEE*, Xuemin (Sherman) Shen, *Fellow, IEEE*, and Weihua Zhuang, *Fellow, IEEE*

***Abstract*—Edge-device collaboration has the potential to facilitate compute-intensive device pose tracking for resource-constrained mobile augmented reality (MAR) devices. In this article, we devise a 3-D map management scheme for edge-assisted MAR, wherein an edge server constructs and updates a 3-D map of the physical environment by using the camera frames uploaded from an MAR device, to support local device pose tracking. Our objective is to minimize the uncertainty of device pose tracking by periodically selecting a proper set of uploaded camera frames and updating the 3-D map. To cope with the dynamics of the uplink data rate and the user's pose, we formulate a Bayes-adaptive Markov decision process problem and propose a digital twin (DT)-based approach to solve the problem. First, a DT is designed as a data model to capture the time-varying uplink data rate, thereby supporting 3-D map management. Second, utilizing extensive generated data provided by the DT, a model-based reinforcement learning algorithm is developed to manage the 3-D map while adapting to these dynamics. Numerical results demonstrate that the designed DT outperforms Markov models in accurately capturing the time-varying uplink data rate, and our devised DT-based 3-D map management scheme surpasses benchmark schemes in reducing device pose tracking uncertainty.**

***Index Terms*—3-D, augmented reality (AR), deep variational inference, digital twin (DT), edge-device collaboration, model-based reinforcement learning (MBRL).**

I. Introduction

In the sixth-generation (6G) networks, immersive communications are anticipated to transcend the existing communication paradigm by offering users highly realistic and interactive experiences [2]. Augmented reality (AR), as a representative form of immersive communications, aims to seamlessly integrate virtual objects into the surrounding physical environments users, thereby enabling them to interact with virtual objects in a lifelike manner [1], [3]. Despite decades of development, AR has not been adopted in our daily lives on a large scale due to limitations, such as device size [4], [5]. With rapid advancement in mobile devices, including smartphones and smart glasses, mobile AR (MAR) technology is expected to penetrate various fields in the 6G era, unlocking opportunities for a wide range of applications, such as immersive learning and tourism [6], [7].

Tracking the time-varying pose of each MAR device is indispensable for MAR applications. Generally, to geometrically align the virtual objects with the physical environment within the Field of View (FoV) of each MAR device in a 3-D manner, the spatial relationship between the MAR device and the physical environment needs to be determined [8]. Nowadays, the real-time information on the required 3-D spatial relationship can be provided by the simultaneous localization and mapping (SLAM) technique, which can be used to estimate the 3-D position and orientation, jointly referred to as the *3-D device pose*, of an MAR device relative to the physical environment within its FoV [9]. As a result, SLAM-based 3-D device pose tracking[1] is anticipated to be a common module used by emerging MAR platforms, e.g., ARKit [10] and ILLIXR [11], for supporting the development of various MAR applications. Despite the capability of SLAM in 3-D alignment for MAR applications, limited resources hinder the widespread implementation of SLAM-based 3-D device pose tracking on MAR devices. The primary limitation arises from the excessive resources demanded by SLAM or its variant techniques, beyond what are typically available on MAR devices [12]. Specifically, to achieve accurate 3-D device pose tracking, SLAM techniques need the support of a 3-D map that consists of a large number of distinguishable landmarks in the physical environment. However, obtaining and maintaining such a 3-D map for continuously updating previously device poses consume excessive storage and computing resources [13].

Cloud/edge-assisted device pose tracking offers a promising solution to address the resource limitations of MAR

Manuscript received 8 November 2023; revised 5 January 2024; accepted 23 January 2024. Date of publication 20 February 2024; date of current version 9 May 2024. This work was supported by a Research Grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada. This article was presented in part at IEEE/CIC International Conference on Communications in China (ICCC), Dalian, China, 2023, pp. 1–6 [DOI: 10.1109/ICCC57788.2023.10233293]. *(Corresponding author: Jie Gao.)*

Conghao Zhou, Xuemin (Sherman) Shen, and Weihua Zhuang are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: c89zhou@uwaterloo.ca; sshen@uwaterloo.ca; wzhuang@uwaterloo.ca).

Jie Gao is with the School of Information Technology, Carleton University, Ottawa, ON K1S 5B6, Canada (e-mail: jie.gao6@carleton.ca).

Mushu Li is with the Department of Electrical, Computer, and Biomedical Engineering, Toronto Metropolitan University, Toronto, ON M5B 2K3, Canada (e-mail: mushu1.li@torontomu.ca).

Nan Cheng is with the School of Telecommunications Engineering, Xidian University, Xi'an 710071, China (e-mail: nancheng@xidian.edu.cn).

Digital Object Identifier 10.1109/JIOT.2024.3360414

[1]"Device pose tracking" is also called "device localization" in some works.

devices by leveraging network resources [14], [15]. From cloud-computing-assisted tracking to the recently prevalent mobile-edge-computing-assisted tracking, researchers have explored resource-efficient approaches for network-assisted tracking from different perspectives. Research works in one category are from the perspective of device pose tracking primarily, which focus on refining SLAM system design to facilitate cloud/edge-device collaboration for MAR [8], [16]. However, these research works tend to overlook the impact of network dynamics by assuming time-invariant communication resource availability or delay constraints. Meanwhile, studies in another category have delved into cloud/edge computing task offloading and scheduling from a networking perspective, considering dynamic service demand and resource availability [17], [18], [19]. Treating device pose tracking as a computing task, these approaches are apt to optimize networking-related performance metrics, such as delay, but do not capture the impact of computing task offloading and scheduling on the performance of device pose tracking. Consequently, despite considerable research efforts from both perspectives, network-assisted device pose tracking that *natively* adapts to network dynamics with optimal device pose tracking performance remains a significant challenge for MAR.

To fill the gap between the aforementioned two categories of research works, we investigate *network dynamics-aware* 3-D map management for network-assisted tracking in MAR. Specifically, we consider an edge-assisted SALM architecture, in which an MAR device conducts real-time device pose tracking locally and uploads the captured camera frames to an edge server. The edge server constructs and updates a 3-D map using the uploaded camera frames to support the local device pose tracking. We optimize the performance of device pose tracking in MAR by managing the 3-D map, which involves uploading camera frames and updating the 3-D map. There are three key challenges to 3-D map management for individual MAR devices. First, an MAR device must select only a portion of the collected information, more specifically camera frames, on its physical environment to update the 3-D map for its 3-D device pose tracking, given the computing and storage resource constraints at the edge server [20]. Second, the camera frame uploading at an MAR device must adapt to the time-varying uplink data rate of the MAR device, which determines the maximum number of camera frames that can be uploaded per unit time for 3-D map update [21]. Third, a new performance metric different from tracking accuracy for evaluating the device pose tracking performance becomes necessary when the network perspective is integrated into 3-D map management, due to the lack of ground truth for the real-time 3-D pose of an MAR device in practice [11].

To address these challenges, we introduce a digital twin (DT)-based approach to effectively cope with the dynamics of the uplink data rate and the device pose. Building upon the DT architecture delineated in our previous work [2], we establish a DT for an MAR device to create a data model that can infer the unknown dynamics of its uplink data rate. Subsequently, we propose an artificial intelligence (AI)-based method, which utilizes the data model provided by the DT to learn the optimal policy for 3-D map management in the presence of device pose variations. The main contributions of this article are as follows.

1) We introduce a new performance metric, termed pose estimation uncertainty, to indicate the long-term impact of 3-D map management on the performance of device pose tracking, which adapts conventional device pose tracking in MAR to network dynamics.
2) We establish a user DT (UDT), which leverages deep variational inference to extract the latent features underlying the dynamic uplink data rate. The UDT provides these latent features to simplify 3-D map management and support the emulation of the 3-D map management policy in different network environments.
3) We develop an adaptive and data-efficient 3-D map management algorithm featuring model-based reinforcement learning (MBRL). By leveraging the combination of real data from actual 3-D map management and emulated data from the UDT, the algorithm can provide an adaptive 3-D map management policy in highly dynamic network environments.

The remainder of this article is organized as follows. Section II provides an overview of related works. Section III describes the considered scenario and system models. Section IV presents the problem formulation and transformation. Section V introduces our UDT, followed by the proposed MBRL algorithm based on the UDT in Section VI. Section VII presents the simulation results, and Section VIII concludes this article.

II. Related Works

In this section, we first summarize existing works on edge/cloud-assisted device pose tracking from the MAR or SLAM system design perspective. Then, we present some related works on computing task offloading and scheduling from the networking perspective.

A. Cloud/Edge-Assisted Device Pose Tracking

Existing studies on edge/cloud-assisted MAR applications can be classified based on their approaches to aligning virtual objects with physical environments. Specifically, there are image retrieval-based, deep learning-based, and localization-based approaches [16].

The image retrieval-based approaches utilize a preconstructed database comprising labeled images, deployed at a cloud/edge server [22]. Given a captured camera frame, an MAR device searches and retrieves the most similar labeled image from the database. Subsequently, the information from this retrieved labeled image is utilized to support the 3-D alignment of virtual objects with this captured camera frame. Deep learning-based approaches in MAR can be viewed as an advancement over image retrieval-based approaches. To overcome the low efficiency of image retrieval-based approaches, deep learning-based approaches leverage deep neural networks (DNNs), e.g., convolutional neural networks, to find the most similar labeled image [23]. Both image retrieval-based and deep learning-based approaches are only suitable for lightweight MAR applications that do not need large databases [16]. Since a physical object can be viewed from various angles and distances, these approaches require a large set of distinct labels. In addition, the accuracy of both approaches in 3-D alignment is limited for existing MAR applications [24].

Currently, both industries and academia have shifted their focus toward localization-based approaches, e.g., Visual-SLAM [11]. By establishing 3-D maps for the physical environments, localization-based approaches can estimate the 3-D poses of individual MAR devices with high accuracy. Instead of identifying physical objects based on their appearance, localization-based approaches can leverage location-related information of physical objects to facilitate accurate and resource-efficient 3-D alignment. Chen et al. [16] utilized a cloud server to calibrate the localization of a local MAR device. The MAR device uploads recent camera frames when there is a significant discrepancy between the localization result from the cloud server and that from the MAR device. Ben Ali et al. build an edge-device collaboration system for Visual-SLAM, with 3-D map management on the edge server and 3-D pose estimation on the local MAR device. Following [12], Chen et al. [21] investigated the impact of radio resource constraints and introduce pose estimation uncertainty in edge-assisted Visual-SLAM. Extending edge-device collaboration to support multiple MAR users, the works in [24] and [25] focus on the coordinate synchronization to guarantee spatial consistency across different MAR devices. Ren et al. [26] investigated the computing and communication resource allocation to support coordinate synchronization. Despite the existing efforts toward SLAM system design in cloud/edge-assisted MAR, the impact of network dynamics on device pose tracking performance remains open.

We employ a Visual-SALM technique, as a localization-based approach, to enhance edge-assisted 3-D pose tracking in MAR. Different from conventional localization-based approaches that often overlook network dynamics and assume 3-D maps of unlimited size, we emphasize the long-term impact of network dynamics on 3-D map management and propose a DT-based approach to adapt to the dynamics of the uplink data rate and the user's pose, while considering a limited-size 3-D map given the resource constraints at the edge server.

B. Computing Task Offloading and Scheduling

By treating the tracking of the device pose for each camera frame as a computing task, the process of uploading camera frames and updating a 3-D map is closely related to the computing task offloading and scheduling in networking [6]. Depending on the chosen performance metrics, existing approaches to computing task offloading and scheduling differ significantly.

Many studies concentrate on improving the delay performance of computing task offloading in a specific network scenario, including space-air-ground integrated networks [27], [28] and vehicular networks [29]. Meanwhile, some researchers investigate computing task offloading or scheduling schemes for specific applications. Li et al. [30] focused on virtual reality applications and aim to reduce the camera frame missing rate in dynamic network environments. Considering surveillance and search-and-rescue-related applications with unmanned aerial vehicles (UAVs), Luo et al. [31] proposed to improve the reliability of target search results by properly offloading the search tasks of UAVs according to UAV trajectories. With the advent of AI-related applications, researchers have started to investigate computing task offloading or scheduling strategies to optimize the accuracy of AI-related applications. The studies in [17] and [32] focus on the inference accuracy of DNNs utilized for AR and Internet of Things, respectively. To facilitate federated learning, Du et al. [18] proposed a task scheduling scheme for distributed devices according to their data qualities and channel conditions.

Different from the aforementioned works on computing task offloading and scheduling, our approach incorporates device pose estimation uncertainty as a performance metric to evaluate camera frame uploading and 3-D map update in MAR applications. Furthermore, we prioritize camera frames when updating the 3-D map to accommodate user pose variations given the time-varying uplink data rate.

III. System Model

A. Considered Scenario

Let one MAR device run an MAR application. While the 3-D pose (i.e., the position and orientation) of the MAR device changes over time, the physical environment (e.g., a living room) of the MAR device does not change [33]. To establish the spatial alignment between virtual objects from the MAR application and the physical environment, the MAR device needs to periodically capture camera frames for tracking its 3-D pose as it moves while updating a 3-D map of the physical environment. A *3-D map* consists of a set of captured camera frames and the corresponding set of 3-D map points in these camera frames. Each 3-D map point is referred to as a feature point, which corresponds to a distinctive spot or characteristic (e.g., a corner of wall) of the physical environment [10].

Generally, the device pose tracking for MAR applications comprises two modules: 1) a lightweight pose calculation module for real-time 3-D pose calculation and 2) a resource-intensive mapping module for managing a 3-D map of the physical environment [20], [34]. To calculate the device pose corresponding to a particular camera frame, the feature points detected in this camera frame need to be matched with feature points in previously captured camera frames, which are stored in the 3-D map. The 3-D map management in the mapping module involves constructing and updating the 3-D map as the reference for 3-D pose calculation.

Due to the limited computing capability and battery of the MAR device, we adopt an edge-device collaborative framework, as shown in Fig. 1, to support the MAR application. Specifically, an edge server at a base station (BS) is equipped with the mapping module for 3-D map management, and the MAR device is equipped with the pose calculation module for local 3-D pose calculation.

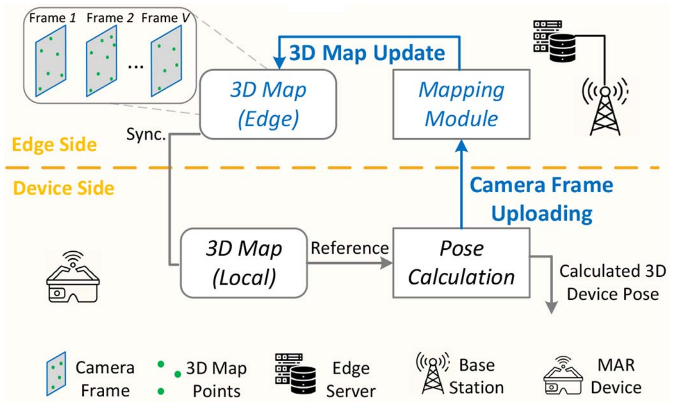


Fig. 1. Considered scenario of edge-assisted MAR.

B. Workflow of Edge-Assisted Device Pose Tracking

The general workflow of device pose tracking for edge-device collaborative MAR applications includes four steps.

1) *3-D Pose Calculation:* The MAR device calculates its 3-D pose corresponding to each camera frame by matching the 3-D map points (i.e., feature points of the physical environment) detected in this camera frame with those contained in the local 3-D map, shown as the "3-D Map (Local)" block in Fig. 1.
2) *Camera Frame Uploading:* The MAR device uploads a subset of recently captured camera frames to the edge server depending on its available uplink communication resource.
3) *3-D Map Update:* The edge server updates the 3-D map, shown as the "3-D Map (Edge)" block in Fig. 1, by processing the camera frames uploaded by the MAR device.
4) *Synchronization:* The edge server periodically sends the updated 3-D map back to the MAR device as references to facilitate 3-D pose calculation [12].

The mapping module at the edge server and the pose calculation module at the MAR device operate on two different time scales. Specifically, the 3-D pose calculation (Step 1) is conducted for each camera frame and takes as short as several milliseconds to complete, while the 3-D map management (Steps 2-4) generally operates on a larger time scale (e.g., over several seconds) [20]. In this article, we focus on *camera frame uploading* (Step 2) and *3-D map update* (Step 3) corresponding to the blue arrows in Fig. 1, which are detailed in Sections III-D and III-E, respectively. For brevity, the term "3-D map" in the remainder of this article denotes "the 3-D map managed by the edge server" unless otherwise stated.

C. 3-D Map Model

The edge server updates the 3-D map per F camera frames, referred to as a time slot. Denote the set of time slots and the set of camera frames captured across all time slots by $\mathcal{K}$ and $\mathcal{F}$, respectively. We illustrate the corresponding timeline of 3-D map management in Fig. 2. Each camera frame, denoted by $f \in \mathcal{F}$, contains a set of 3-D map points, denoted by $\mathcal{M}_f$.

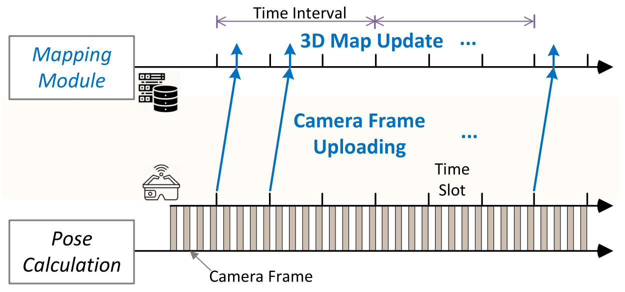


Fig. 2. Timeline of 3-D map management.

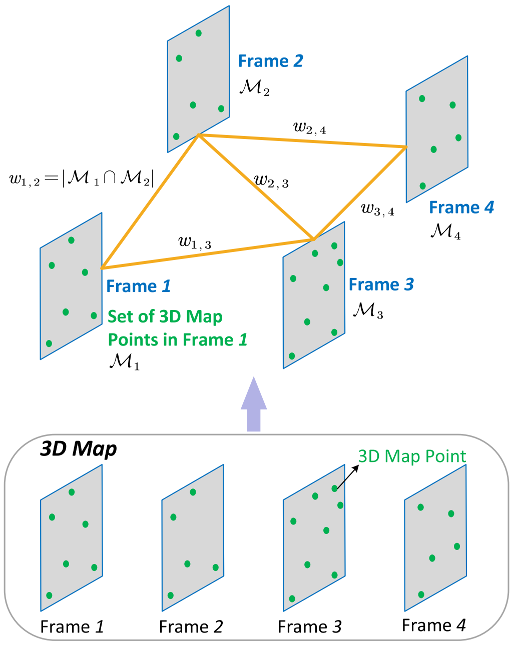


Fig. 3. Illustration of the 3-D map model.

We model the 3-D map as a weighted undirected graph to capture the relationships among the camera frames forming the 3-D map. The model for a 3-D map, including four camera frames, is illustrated in Fig. 3. Denote the 3-D map at the beginning of time slot $k \in \mathcal{K}$ by $\mathcal{G}_k^{\mathrm{e}} = (\mathcal{V}_k^{\mathrm{e}}, \mathcal{E}_k^{\mathrm{e}})$, where $\mathcal{V}_k^{\mathrm{e}} \subset \mathcal{F}$ denotes the set of camera frames contained in the 3-D map at the beginning of time slot k, and $\mathcal{E}_k^{\mathrm{e}}$ denotes the set of relationships between every pair of camera frames in $\mathcal{V}_k^{\mathrm{e}}$. For edge $e = (f, f') \in \mathcal{E}_k^{\mathrm{e}}$ connecting frames $f \in \mathcal{V}_k^{\mathrm{e}}$ and $f' \in \mathcal{V}_k^{\mathrm{e}}$, we define its weight as follows:

$$w_{f,f'} = |\mathcal{M}_f \cap \mathcal{M}_{f'}| \quad \forall f, f' \in \mathcal{V}_k^{\mathrm{e}} \tag{1}$$

where $|\cdot|$ represents the cardinality of a set and $\cap$ denotes the intersection of two sets. If the sets of 3-D map points contained in camera frames f and f' are similar, the weight of edge, $w_{f,f'}$, will be large. As shown in Fig. 3, the set of 3-D map points corresponding to each camera frame is the collection of the corresponding green points, and the edges are depicted as the orange lines between camera frames.

D. Camera Frame Uploading

As the MAR user moves around, the MAR device uploads its newly captured camera frames to the edge server for updating the 3-D map. The uplink data rate varies over time due to time-varying communication resource availability or communication link quality [35], and the camera frame uploading must adapt to such variations. As shown in Fig. 2, we introduce an additional time scale, named the time interval (over a few minutes) so that the dynamics of the uplink data rate is stationary within each time interval. Each time interval consists of K consecutive time slots, and the set of time slots within time interval t by $\mathcal{K}_t = \{k|(t-1)K < k \leq tK \ \forall k \in \mathcal{K}\}$. Within time interval t, we denote the uplink data rate at time slot $k \in \mathcal{K}_t$ by random variable d_k, which follows an N-state Markov chain. The state transition matrix of the N-state Markov chain is assumed to be stationary within each time interval but can vary across time intervals. We introduce random variable x_t to represent the temporal variation of the state transition matrix of the Markov chain across time intervals, i.e., state transition probability $P(d_{k+1}|d_k, x_t)$ is conditioned on x_t and can vary across time intervals.

Without loss of generality, suppose that the MAR device uploads its newly captured camera frames within each time slot to the edge server at the end of the time slot, shown as the blue arrows between the edge server and the MAR device in Fig. 2. Due to the bandwidth limit for uplink transmissions, a subset of camera frames captured at the end of each time slot needs to be selected for uploading. Denote the set of all camera frames captured during time slot $k \in \mathcal{K}$ and the subset of camera frames selected for uploading by $\mathcal{F}_k \subseteq \mathcal{F}$ and $\mathcal{U}_k \subseteq \mathcal{F}_k$, respectively. Let the amount of transmitted data (in bits) for uploading each camera frame, denoted by α, be identical for uploading each camera frame. In any time slot, camera frame uploading should satisfy the following constraint:

$$\alpha|\mathcal{U}_k| \leq d_k D^{\text{req}} \quad \forall k \in \mathcal{K} \tag{2}$$

where D^{req} denotes the maximum tolerable transmission delay for uploading the selected camera frames. The value of D^{req} in (2) can be set flexibly according to the overall performance requirement of device pose tracking in MAR.

E. 3-D Map Update

Generally, the mapping module at the edge server involves updating the 3-D map and solving a 3-D map optimization problem [12]. In this section, we model the 3-D map update and the impact of the computing and storage resource limitation for solving a 3-D map optimization problem on 3-D map update.

1) Impact of Resource Limitation for Solving 3-D Map Optimization Problem: Given a 3-D map $\mathcal{G}_k^{\text{e}}$, the mapping module is responsible for jointly estimating the 3-D device poses corresponding to all camera frames $f \in \mathcal{V}_k^{\text{e}}$ by solving a 3-D map optimization problem [20].[2] However, the computing and storage resources required for solving this 3-D map optimization problem increases exponentially with the size of 3-D map, while these resources are usually limited at the edge server for any individual MAR device. Considering the resource limitation, we denote the maximum size of the 3-D map by $V^{\max}$. Given the set of camera frames in the 3-D map, i.e., $\mathcal{V}_k^{\text{e}}$, the size of 3-D map should satisfy the following constraint:

$$|\mathcal{V}_k^{\text{e}}| \leq V^{\max} \quad \forall k \in \mathcal{K}. \tag{3}$$

Meanwhile, due to the resource limitation, the 3-D map cannot store all the camera frames ever uploaded [8], [20], resulting in the need of removing some camera frames regularly to update the 3-D map.

2) 3-D Map Update: In each time slot, the set of camera frames stored in the 3-D map is updated after the edge server receives newly uploaded camera frames.[3] Due to the limited size of the 3-D map, a set of camera frames, denoted by $\mathcal{C}_k \subseteq \mathcal{V}_k^{\text{e}}$, are removed from the 3-D map. Given the set of newly uploaded camera frames $\mathcal{U}_k$, the set of camera frames in the 3-D map in time slot $k+1$, i.e., $\mathcal{V}_{k+1}^{\text{e}}$, evolves as follows:

$$\mathcal{V}_{k+1}^{\text{e}} = \left\{\mathcal{U}_k \cup \mathcal{V}_k^{\text{e}}\right\} \backslash \mathcal{C}_k \quad \forall k, k+1 \in \mathcal{K}. \tag{4}$$

The evolution of set $\mathcal{V}_k^{\text{e}}$ affects both the nodes and the edges of the graph representing the 3-D map. Following the 3-D map update, the edge server sends the 3-D map back to the MAR device as references for supporting the local 3-D pose calculation at the MAR device within the subsequent time slot.

From Sections III-D and III-E, it can be seen that 3-D map management decisions in each time slot involve both $\mathcal{U}_k$ for camera frame uploading and $\mathcal{C}_k$ for 3-D map update.

F. Pose Estimation Uncertainty

A performance metric is required to measure the impact of 3-D map management decisions on the performance of device pose tracking [17]. The tracking accuracy is widely used as the metric to evaluate the performance of pose calculation in MAR [12], [16]. However, due to the lack of ground truth for 3-D poses of an MAR device in real time, a different metric is needed for guiding real-time 3-D map management. Recent studies have concentrated on the metric of pose estimation uncertainty, which captures how the quality of a 3-D map affects the robustness of MAR or SLAM. For a given 3-D map, the pose estimation uncertainty can be obtained in real time. A lower uncertainty represents a higher reliability of the 3-D pose estimation, and in turn a higher pose calculation accuracy by reducing cumulative errors [36]. Existing works have shown the suitability of this metric in facilitating SLAM, especially for tracking accuracy improvement [21].

Since the pose estimation uncertainty affects the reliability of device pose tracking in MAR and can be calculated without requiring ground truth for 3-D poses of an MAR device [37], we adopt it as the performance metric to guide 3-D map management. Generally, pose estimation uncertainty, which is unitless, characterizes the impact of the relationship among camera frames in a given 3-D map on the error covariance of estimated 3-D poses based on this 3-D map [38]. Given 3-D map $\mathcal{G}_k^{\text{e}}$, the pose estimation uncertainty is calculated

[2]Given a 3-D map, solving a 3-D map optimization problem requires finding the maximum likelihood estimations for the 3-D device pose corresponding to each camera frame by comparing the feature points contained in every pair of two camera frames in the 3-D map [20].

[3]The phrases "update a 3-D map" and "update the set of frames contained in a 3-D map" are used interchangeable in this article.

according to the node connectivity and edge weights in the 3-D map, as follows [21], [36]:

$$u(\mathcal{G}_k^{\mathrm{e}}) = -\log\Big(\det\Big(\hat{\boldsymbol{L}}(\mathcal{G}_k^{\mathrm{e}}) \otimes \boldsymbol{\Pi}\Big)\Big) \quad \forall k \in \mathcal{K} \tag{5}$$

where $\hat{\boldsymbol{L}}(\mathcal{G}_k^{\mathrm{e}})$ denotes the reduced Laplacian matrix of the graph representing the 3-D map $\mathcal{G}_k^{\mathrm{e}}$ [39], $\otimes$ represents the Kronecker product, and $\det(\cdot)$ denotes the determinant of a matrix [40]. Matrix $\boldsymbol{\Pi}$ in (5) has a dimension of 6×6 due to the six Degrees of Freedom (DoF) of a 3-D pose, and the value of $\boldsymbol{\Pi}$ is related to the camera settings of the MAR device and can usually be assumed as a constant.

IV. Problem Formulation and Transformation

In this section, we first formulate a 3-D map management problem with the objective of minimizing the pose estimation uncertainty. Then, we transform the problem into a Markov decision process (MDP) problem.

A. Problem Formulation

To capture the impact of the 3-D map updated in time slot k on the device pose calculation for the camera frames captured in the subsequent time slot $k+1$, we define the average pose estimation uncertainty over all camera frames in set $\mathcal{F}_{k+1}$ as υ_k. Given the set of camera frames $\mathcal{F}_{k+1}$ captured within time slot $k+1$, the value of υ_k is given by

$$\upsilon_k = |\mathcal{F}_{k+1}|^{-1} \sum_{f \in \mathcal{F}_{k+1}} u\big(\mathcal{G}_k^{\mathrm{e}} \cup \{f\}\big) \quad \forall k, k+1 \in \mathcal{K} \tag{6}$$

where

$$\mathcal{G}_k^{\mathrm{e}} \cup \{f\} := \big(\mathcal{V}_k^{\mathrm{e}} \cup \{f\}, \mathcal{E}_k^{\mathrm{e}} \cup \{e = (f, f') | f' \in \mathcal{V}_k^{\mathrm{e}}\}\big) \tag{7}$$

in which $\{e = (f, f') | f' \in \mathcal{V}_k^{\mathrm{e}}\}$ denotes the set of newly generated edges due to adding camera frame f to 3-D map $\mathcal{G}_k^{\mathrm{e}}$.

To minimize the pose estimation uncertainty over all time slots, we formulate the following optimization problem:

$$\text{P1:} \min_{\{\mathcal{U}_k, \mathcal{C}_k\}_{k \in \mathcal{K}}} \sum_{k \in \mathcal{K}} \upsilon_k \tag{8a}$$

$$\text{s.t.} \quad (2), (3), (4) \tag{8b}$$

$$\mathcal{U}_k \subseteq \mathcal{F}_k \quad \forall k \in \mathcal{K} \tag{8c}$$

$$\mathcal{C}_k \subseteq \mathcal{V}_k^{\mathrm{e}} \cup \mathcal{U}_k \quad \forall k \in \mathcal{K}. \tag{8d}$$

The optimization variables in Problem P1 are the set of selected camera frames for uploading, i.e., $\mathcal{U}_k$, and the set of camera frames removed from the original 3-D map, i.e., $\mathcal{C}_k$, in each time slot. Solving Problem P1 is challenging due to three reasons. First, managing the 3-D map for any given time slot is an NP-hard fixed-cardinality maximization problem [21]. Regular iterative methods cannot be applied to Problem P1 due to the unknown *a priori* information on the set of camera frames captured within subsequent time slots. Second, 3-D map management across multiple time slots results in a sequential decision-making problem. Decisions made for the 3-D map in one time slot inevitably influence those in subsequent time slots. Making decisions for each time slot independently, without considering their cumulative effect, is not likely to yield the optimal long-term 3-D map management. Third, the stochastic uplink data rate is nonstationary across multiple time intervals, subject to the influence of the random variable x_t, which exacerbates the challenges.

B. Problem Transformation

In this section, we first analyze the characteristics of pose estimation uncertainty to reduce the solution space for solving Problem P1, and then transform the problem into an MDP problem.

1) Cardinality of the Optimal Solution Set: We present the following lemma to show that the pose estimation uncertainty decreases when the number of camera frames forming a 3-D map increases.

Lemma 1: Given a connected 3-D map $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the pose estimation uncertainty $u(\mathcal{G})$ monotonously decreases with the value of $|\mathcal{V}|$ when $\det(\boldsymbol{\Pi}) \geq 1$.

Proof: See the Appendix. ■

Lemma 1 allows us to reduce the solution space of Problem P1. The cardinalities of the optimal $\mathcal{U}_k$ and $\mathcal{C}_k$ for time slot $k \quad \forall k \in \mathcal{K}$ are derived in Theorem 1.

Theorem 1: For time slot k, the cardinalities of the optimal $\mathcal{U}_k$ and $\mathcal{C}_k$, i.e., $|\mathcal{U}_k|$ and $|\mathcal{C}_k|$, are given by

$$|\mathcal{U}_k| = \lfloor \alpha^{-1} D^{\mathrm{req}} d_k \rfloor \quad \forall k \in \mathcal{K} \tag{9}$$

and

$$|\mathcal{C}_k| = V^{\max} - |\mathcal{U}_k| \quad \forall k \in \mathcal{K} \tag{10}$$

respectively, where $\lfloor \cdot \rfloor$ represents the floor function.

Proof: Based on Lemma 1 and (3), the optimal number of camera frames contained in the 3-D map, i.e., $|\mathcal{V}_k^{\mathrm{e}}|$, should satisfy

$$|\mathcal{U}_k| + |\mathcal{V}_k^{\mathrm{e}}| - |\mathcal{C}_k| = V^{\max} \quad \forall k \in \mathcal{K}. \tag{11}$$

For a given value of $|\mathcal{C}_k|$, the objective function value decreases when the number of uploaded camera frames, i.e., $|\mathcal{U}_k|$, increases. According to (2), the optimal value of $|\mathcal{U}_k|$ is given by (9). Given a fixed value $|\mathcal{U}_k|$, the objective function value decreases when fewer camera frames are removed from the 3-D map. Therefore, the optimal value of $|\mathcal{U}_k|$ is given by (10). ■

According to Theorem 1, uploading as many camera frames as possible and removing as few camera frames as possible from the 3-D map can decrease the pose estimation uncertainty of 3-D map management for MAR. While Theorem 1 gives the cardinalities of the optimal $\mathcal{U}_k$ and $\mathcal{C}_k$, the optimal sets of camera frames for uploading and for removing are yet to be determined. Consequently, there is a need for an approach to determine the optimal $\mathcal{U}_k$ and $\mathcal{C}_k$, which takes the long-term impacts of time-varying device pose and uplink data rate into account.

2) Bayes-Adaptive MDP: Next, we reformulate Problem P1 as an MDP problem. Denote the state space and the action space of the MDP by $\mathcal{S}$ and $\mathcal{A}$, respectively. Denote $\boldsymbol{\mathcal{G}}_k^{\mathrm{e}} = [\mathcal{G}_i^{\mathrm{e}}]_{k-\tau \leq i \leq k}$, $\boldsymbol{\mathcal{F}}_k = [\mathcal{F}_k]_{k-\tau \leq i \leq k}$, and $\boldsymbol{d}_k = [d_k]_{k-\tau \leq i \leq k}$. Let $\boldsymbol{s}_k = [\boldsymbol{\mathcal{G}}_k^{\mathrm{e}}, \boldsymbol{\mathcal{F}}_k, \boldsymbol{d}_k] \in \mathcal{S}$ and $\boldsymbol{a}_k = [\mathcal{U}_k, \mathcal{C}_k] \in \mathcal{A}$ denote the state and the action, i.e., 3-D map management decision, at the beginning of time slot t, respectively. Denote the state transition function by $P(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k)$. In addition, we define the reward function for time slot k as the negative of the long-term pose estimation uncertainty defined in (6), given by

$$r_k = -\upsilon_k \quad \forall k \in \mathcal{K}. \tag{12}$$

With the MDP model and Theorem 1, we reformulate Problem P1 as the following discounted MDP problem for sequential 3-D map management decision making in the presence of time-varying device pose and uplink data rate:

$$\text{P2:} \max_{\{\boldsymbol{a}_k\}_{k\in\mathcal{K}}} \sum_{k\in\mathcal{K}} \gamma^k r_k \tag{13a}$$

$$\text{s.t.} \quad (8b)-(8d), \ (9)\ (10) \tag{13b}$$

where $\gamma \in (0, 1)$ is the discount factor for quantifying the long-term impact of an action on the rewards obtained in future time slots [19]. Our goal is to find a policy, i.e., π, for making proper 3-D map management decisions in each state.

The dynamics in Problem P2 encompass both variations in device pose and uplink data rate. The device pose is determined solely by human behavior, whereas uplink data rate is mostly determined by network conditions. Therefore, the variations in device pose and uplink data rate can be considered independent. As mentioned in Section III-D, due to the unknown random variable x_t, the specific parameters of the N-state Markov chain may vary across time intervals, thereby resulting in a nonstationary uplink data rate. For tractability, we make the assumption that only the dynamics of uplink data rate is nonstationary in problem P2, rather than device pose variations. Correspondingly, Problem P2 becomes a Bayes-adaptive MDP (BAMDP) problem, and the transition probabilities corresponding to the uplink data rate $P(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k, x_t), k \in \mathcal{K}_t$, are time-varying, where $x_t \sim p(x)$ follows an unknown distribution with some latent parameters [41]. To solve the BAMDP problem, we establish a DT for capturing the unknown distribution $p(x)$, presented in Section V, and propose a model-based deep reinforcement learning (DRL) method using the DT to adapt to both the time-varying uplink data rate and the device pose.

V. User Digital Twin

In this section, we create a DT for an individual MAR device, referred to as a UDT, to establish a data model that can capture the unknown distribution $x_t \sim p(x)$ in approximating $P(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k, x_t)$. Our UDT design evolves from the framework presented in [2], with a specific focus on assisting 3-D map management in MAR. The UDT, consisting of an *MAR device data profile* and several *UDT functions*, is located at the BS. A network controller is responsible for maintaining and updating the MAR device data profile through the execution of the following four UDT functions: 1) real experience collection; 2) latent feature extraction; 3) artificial experience generation; and 4) UDT update. We illustrate the workflow of the designed UDT in Fig. 4. In the "UDT" segment (to the left of the dashed vertical line in Fig. 4), the real experience of 3-D map management,

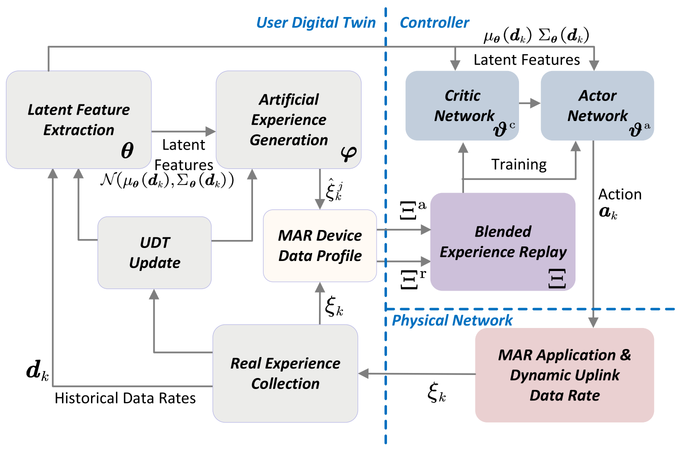

User Digital Twin
Controller
$\mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k)\ \Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$
Latent Features
Latent Feature Extraction $\boldsymbol{\theta}$
Latent Features
$\mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k), \Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k))$
Artificial Experience Generation $\boldsymbol{\varphi}$
$\hat{\xi}_k^j$
Critic Network $\boldsymbol{\vartheta}^{\rm c}$
Actor Network $\boldsymbol{\vartheta}^{\rm a}$
Training
Action $\boldsymbol{a}_k$
$\Xi^{\rm a}$
$\Xi^{\rm r}$
UDT Update
MAR Device Data Profile
Blended Experience Replay Ξ
ξ_k
Physical Network
$\boldsymbol{d}_k$
Historical Data Rates
Real Experience Collection
ξ_k
MAR Application & Dynamic Uplink Data Rate

Fig. 4. Workflow of the designed UDT and UDT-based 3-D map management.

including state, action, reward, and next state, is collected and stored in the MAR device data profile at the end of each time slot. Based on the collected real experiences, a deep variational inference method is used to extract latent features from each real experience and generate artificial experiences based on the extracted latent features, which correspond to the aforementioned UDT functions (2) and (3), respectively. The generated artificial experiences are also stored in the MAR device data profile. Meanwhile, the UDT update function, i.e., UDT function (4), is used to update the parameters of other UDT functions, such as the weights of the DNNs. Details of the four UDT functions are presented below.

A. Real Experience Collection

A real experience collected at the beginning of time slot $k+1$ is the tuple $\xi_k = (\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$. In any given time slot, the UDT contains the real experiences of 3-D map management collected in preceding time slots for learning the 3-D map management policy in subsequent time slots. Let $\Xi^{\rm r}$ denote the set of collected real experiences contained in the UDT, which the UDT can update per time slot by collecting a new real experience. As shown in Fig. 4, the collected data in $\Xi^{\rm r}$ are stored in the MAR device data profile for supporting the other three UDT functions and the decision making on 3-D map management.

B. Latent Feature Extraction

Since the distribution $x_t \sim p(x)$ is unknown *a priori*, calculating $p(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k, x_t)$ is not possible. Therefore, we adopt a deep variational inference method to capture the unknown distribution $p(x)$.

Define a Z-dimensional variable $\boldsymbol{z}$, which follows a normal distribution, i.e., $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_Z)$. Given $\boldsymbol{z}$, we introduce a function $q(\boldsymbol{z}|\boldsymbol{d}_k, \boldsymbol{d}_{k+1}; \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ to approximate the probability $p(\boldsymbol{z}|\boldsymbol{d}_k, \boldsymbol{d}_{k+1}, x_t)$ and a function $q(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k, \boldsymbol{z}; \boldsymbol{\varphi})$ parameterized by $\boldsymbol{\varphi}$ to approximate probability $p(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k, \boldsymbol{z})$. Since directly maximizing the likelihood $p(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k; \boldsymbol{\varphi})$ is intractable, our goal is to maximize a lower bound of $p(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k;\boldsymbol{\varphi})$, given by

$$\log p(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k;\boldsymbol{\varphi}) \geq \mathbb{E}_{z\sim q(z|\boldsymbol{d}_k,\boldsymbol{d}_{k+1};\boldsymbol{\theta})}\left[\log q(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k,z;\boldsymbol{\varphi})\right] - l_{\mathrm{KL}}(q(z|\boldsymbol{d}_k,\boldsymbol{d}_{k+1};\boldsymbol{\theta})||p(z)) \tag{14}$$

where $l_{\mathrm{KL}}(q(z|\boldsymbol{d}_k,\boldsymbol{d}_{k+1};\boldsymbol{\theta})||p(z))$ denotes the Kullback–Leibler (KL) divergence

$$l_{\mathrm{KL}}(q(z|\boldsymbol{d}_k,\boldsymbol{d}_{k+1};\boldsymbol{\theta})||p(z)) = \mathbb{E}_{z\sim q(z|\boldsymbol{d}_k,\boldsymbol{d}_{k+1};\boldsymbol{\theta})}\left[\log q(z|\boldsymbol{d}_k,\boldsymbol{d}_{k+1};\boldsymbol{\theta}) - \log p(z)\right]. \tag{15}$$

We omit the full derivation of the formula in (14) and refer interested readers [42].

Considering that a multidimensional normal distribution can be used to capture various data distributions, we adopt a normal distribution $\mathcal{N}(z|\mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k),\Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k))$ to approximate $q(z|\boldsymbol{d}_k,\boldsymbol{d}_{k+1};\boldsymbol{\theta})$ so that the term $l_{\mathrm{KL}}(q(z|\boldsymbol{d}_k,\boldsymbol{d}_{k+1};\boldsymbol{\theta})||p(z))$ in (15) has a closed form [43]. Specifically, we leverage a DNN to output the parameters of the Z-dimensional normal distribution $(z|\boldsymbol{d}_k,\boldsymbol{d}_{k+1};\boldsymbol{\theta})$, i.e., mean $\mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$ and variance $\Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$. Given parameter $\boldsymbol{\theta}$, we refer to the output $\mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$ and $\Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$ as the *latent features* representing the unknown distribution $x_t \sim p(x)$, extracted from input $\boldsymbol{d}_k$. In addition, we approximate the function $q(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k,z;\boldsymbol{\varphi})$ through another DNN parameterized by $\boldsymbol{\varphi}$. Given $\boldsymbol{d}_k$, the only element in $\boldsymbol{d}_{k+1}$ that needs to be predicted is d_{k+1}. The value of d_{k+1}, denoted by $\hat{d}_{k+1}$, can be output as follows:

$$\hat{d}_{k+1} = \phi_{\boldsymbol{\varphi}}(\tilde{z}) \quad \forall k \in \mathcal{K}_t \tag{16}$$

where $\phi_{\boldsymbol{\varphi}}(\cdot)$ denotes the DNN parameterized by $\boldsymbol{\varphi}$ with a Softmax activation function, and vector $\tilde{z}$ is a sample from the distribution $\mathcal{N}(z|\mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k),\Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k))$. Consequently, the outputs of the DNNs with parameters $\boldsymbol{\vartheta}$ and $\boldsymbol{\varphi}$ are the extracted latent features, i.e., $\mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$ and $\Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$, and the value of $\hat{d}_{k+1}$, respectively. Both outputs are used to facilitate our proposed DRL methods for solving Problem P2, as shown in Fig. 4.

C. Artificial Experience Generation

The artificial experience generation function of the UDT can generate data samples for $\boldsymbol{d}_{k+1}$ by leveraging $\boldsymbol{d}_k$ and the extracted latent features $\mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$ and $\Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$ at time slot k. Upon feeding $\boldsymbol{d}_k$ into the two DNNs parameterized by $\boldsymbol{\vartheta}$ and $\boldsymbol{\varphi}$ sequentially, the output of the Softmax activation function in the DNN represented by (16) yields a probability vector. This vector encapsulates the likelihood of each potential state among the N states in the Markov chain pertaining to the uplink data rate, as mentioned in Section III-D. As a result, the artificial experience generation function of the UDT can generate a set of data samples, each of which represents a possible uplink data rate within time slot $k+1$, denoted by $\hat{d}^j_{k+1}$. Let $\hat{\boldsymbol{d}}^j_{k+1} = [\boldsymbol{d}_k, \hat{d}^j_{k+1}]$ denote a generated data sample according to the collected real data $\boldsymbol{d}_k$.

Based on each generated sample of the uplink data rate, this UDT function can generate an artificial experience, denoted by $\hat{\xi}^j_k = (\hat{\boldsymbol{s}}^j_k, \hat{\boldsymbol{a}}^j_k, \hat{r}^j_k, \hat{\boldsymbol{s}}^{j+1}_k)$. Specifically, given state $\hat{\boldsymbol{s}}^j_k$, this UDT function can randomly select an action, denoted by $\hat{\boldsymbol{a}}^j_k$, which satisfies the constraints in Problem P2 and is used for the emulation of 3-D map management. Subsequently, this UDT function can emulate the 3-D map at time slot $k+1$, denoted by $\hat{\mathcal{G}}^{\mathrm{e},j}_{k+1}$, as (4) given action $\hat{\boldsymbol{a}}^j_k$ taken in state $\hat{\boldsymbol{s}}^j_k$ and calculate the reward using (12). As a result, given any tuple of real experience $\xi_k = (\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1}) \in \Xi^{\mathrm{r}}$, this UDT function can generate J artificial experiences with the jth tuple given $(\boldsymbol{s}_k, \hat{\boldsymbol{a}}^j_k, \hat{r}^j_k, \hat{\boldsymbol{s}}^{j+1}_k)$ where

$$\hat{\boldsymbol{s}}^{j+1}_k = \left[\hat{\mathcal{G}}^{\mathrm{e},j}_{k+1}, \mathcal{F}_{k+1}, \hat{\boldsymbol{d}}^j_{k+1}\right]. \tag{17}$$

We define the set of these generated artificial experiences as $\Xi^{\mathrm{a}} = \{\hat{\xi}^j_k|\forall 0 < j \leq J, \xi_k \in \Xi^{\mathrm{r}}\}$. As shown in Fig. 4, the data of generated artificial experiences are also stored in the MAR device data profile and can be used to support making 3-D map management decisions.

Algorithm 1: UDT Update

```
1 Input: $\Xi^{\mathrm{r}}$
2 Initialization: $\boldsymbol{\theta}$, $\boldsymbol{\varphi}$
3 $\boldsymbol{\theta}_*$, $\boldsymbol{\varphi}_*$ ← Update $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ by optimizing (18) based on
  real experiences $\Xi^{\mathrm{r}}$;
4 $\Xi^{\mathrm{a}}$ ← Update with the newly generated
  tuples $\left\{\hat{\xi}^j_k|\forall 0 < j \leq J, \xi_k \in \Xi^{\mathrm{r}}\right\}$ using (17);
5 Output: $\boldsymbol{\theta}_*$, $\boldsymbol{\varphi}_*$, $\Xi^{\mathrm{a}}$
```

D. UDT Update

Both the latent feature extraction and the artificial experience generation functions require online optimization of parameters $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$. The optimization is conducted by minimizing the loss function given by the right-hand side of (14) as follows:

$$L(\boldsymbol{\theta},\boldsymbol{\varphi}) = \mathbb{E}_{z\sim q(z|\boldsymbol{d}_k,\boldsymbol{d}_{k+1};\boldsymbol{\theta})}\left[\log q(\boldsymbol{d}_{k+1}|\boldsymbol{d}_k,z;\boldsymbol{\varphi})\right] - l_{\mathrm{KL}}(q(z|\boldsymbol{d}_{k+1},\boldsymbol{d}_k;\boldsymbol{\theta})||p(z)). \tag{18}$$

The gradient descent method can be employed to find the optimal parameters $\boldsymbol{\theta}_*$ and $\boldsymbol{\varphi}_*$, utilizing the reparametrization trick in [44], that minimize the gradients of $L(\boldsymbol{\theta},\boldsymbol{\varphi})$ for DNN backward propagation. We introduce the update of DNNs with parameters $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ and the generation of artificial experiences in Algorithm 1. In Line 3, we optimize the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ by minimizing the loss function in (18). In Line 4, given a newly collected set of real experiences, i.e., Ξ^{r}, the UDT generates a new set of artificial experiences for adapting to the dynamic network uplink data rate. Our UDT design allows for flexible adjustments of parameters $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ according to the newly collected real experiences, which happens once every W time slots.

VI. Model-Based DRL Method

While the designed UDT can capture the dynamics of the uplink data rate, we still need a method to adapt to temporal variations in the device pose and the uplink data rate when solving Problem P2. Conventional DRL-based methods can be used to solve MDP problems characterized by unknown but stationary environmental dynamics [41]. However, they cannot be directly applied to solving Problem P2 due to the nonstationary nature of the MDP across time intervals.

Therefore, we propose a *model-based* DRL (MBRL) method by leveraging extensive organized data provided by the UDT, including the extracted latent features of the uplink data rate and the generated artificial experiences, to solve the BAMDP problem. Our proposed method is built upon an off-policy model-free DRL algorithm, in which historical experiences can be used offline for training the DNNs [19], [27]. Next, we will present the model-free DRL framework and our designs of UDT-assisted DRL after that.

A. Model-Free DRL

Given that the UDT can extract latent features regarding dynamic uplink data rate through the approximation of random variable x_t, we extend the originally state $\boldsymbol{s}_k$ by defining the augmented state as $\dot{\boldsymbol{s}}_k = [\boldsymbol{s}_k, \mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k), \Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k)]$. The latent features extracted by using the UDT can represent the information on the random variable x_t, which results in the nonstationary uplink data rate. Therefore, adopting the augmented state can transform the BAMDP into an MDP, thereby enabling us to apply a DRL-based method for learning a 3-D map management policy.

Using the augmented state, we adopt an actor-critic framework to learn the optimal policy π^*. Define a Q-value function of state $\dot{\boldsymbol{s}}_k$ and action $\boldsymbol{a}_k$ as the accumulated discounted reward, as follows:

$$Q(\dot{\boldsymbol{s}}_k, \boldsymbol{a}_k) = \sum_{j=1}^{J} \gamma^j r_{k+j+1} \quad \forall k \in \mathcal{K} \tag{19}$$

where the Q-value quantifies the long-term impact of each action on subsequent states and actions [19]. However, calculating the Q-value directly is impossible due to unknown state transitions in the future. Thus, we derive the Q-value calculated for time slot k by using the obtained reward at time slot k and the Q-value calculated for time slot $k+1$, as follows:

$$Q(\dot{\boldsymbol{s}}_k, \boldsymbol{a}_k) = r_k + \gamma Q(\dot{\boldsymbol{s}}_{k+1}, \boldsymbol{a}_{k+1}) \quad \forall k \in \mathcal{K}. \tag{20}$$

Given (20), model-free DRL methods approximate the Q-value function by minimizing the temporal difference between the Q-values for different time slots [27], [31]. Specifically, a DNN (the critic network) with the parameter $\boldsymbol{\vartheta}^{\mathrm{c}}$ is leveraged to approximate the Q-value function, i.e., $Q(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\vartheta}^{\mathrm{c}})$. The loss function for optimizing the parameter $\boldsymbol{\vartheta}^{\mathrm{c}}$ is given by

$$L(\boldsymbol{\vartheta}^{\mathrm{c}}) = \frac{1}{|\Xi|} \sum_{\xi_k \in \Xi} \left(r_k + \gamma Q\left(\dot{\boldsymbol{s}}_{k+1}, \pi(\dot{\boldsymbol{s}}_{k+1}); \boldsymbol{\vartheta}^{\mathrm{c}}\right) - Q\left(\dot{\boldsymbol{s}}_k, \boldsymbol{a}_k; \boldsymbol{\vartheta}^{\mathrm{c}}\right)\right)^2 \tag{21}$$

where Ξ denotes a batch of tuples ξ_k selected from historical experiences for training and $|\Xi|$ represents the batch size. Note that we consider an off-policy DRL framework, wherein the policy $\pi(\dot{\boldsymbol{s}}_k)$ employed for Q-value approximation in (21) may be different from the policy used for actual action execution in practice.

Algorithm 2: AMM Algorithm

```
1  Input: $W$, $I$, $|\Xi|$
2  Initialization: $\boldsymbol{\vartheta}^{\mathrm{c}}$, $\boldsymbol{\vartheta}^{\mathrm{a}}$, $\boldsymbol{\theta}$, $\boldsymbol{\varphi}$, $\Xi^{\mathrm{r}}$, $\Xi^{\mathrm{a}}$, $\boldsymbol{s}_1$
3  for $k \in \mathcal{K}$ do
4      $\mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$, $\Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$ $\leftarrow$ the UDT extracts the latent features as Section V-B;
5      Select action $\boldsymbol{a}_k = \pi(\dot{\boldsymbol{s}}_k; \boldsymbol{\vartheta}^{\mathrm{a}})$;
6      $r_k$, $\boldsymbol{s}_{k+1}$ $\leftarrow$ take action $\boldsymbol{a}_k$ on state $\boldsymbol{s}_k$;
7      Combine $I$ tuples randomly selected from $\Xi^{\mathrm{r}}$ and $|\Xi| - I$ tuples randomly selected from $\Xi^{\mathrm{a}}$ as $\Xi$;
8      Update $\boldsymbol{\vartheta}^{\mathrm{c}}$ by minimizing (21);
9      Update $\boldsymbol{\vartheta}^{\mathrm{a}}$ by using policy gradient descent in (22);
10     $\Xi^{\mathrm{r}}$ $\leftarrow$ Update with the latest tuples of real experiences, i.e., $\{\xi_j | k - |\Xi^{\mathrm{r}}| < j \leq k\}$;
11     if $k \mod W = W - 1$ then
12         $\boldsymbol{\theta}_*$, $\boldsymbol{\varphi}_*$, $\Xi^{\mathrm{a}}$ $\leftarrow$ Run Algorithm 1 for UDT update;
13     end
14     $k$, $\boldsymbol{s}_k$ $\leftarrow$ $k+1$, $\boldsymbol{s}_{k+1}$;
15 end
16 Output: $\pi(\dot{\boldsymbol{s}}_k; \boldsymbol{\vartheta}^{\mathrm{a}}_*)$
```

Given an approximated Q-value function, our goal is to find a policy $\pi(\dot{\boldsymbol{s}}_k)$ for taking an action in each state with the consideration of the long-term impact of the action. Model-free DRL can be used to find the parameters of the optimal policy through parameterizing the policy $\pi(\dot{\boldsymbol{s}}_k)$. Specifically, we approximate the 3-D map management policy $\pi(\dot{\boldsymbol{s}}_k; \boldsymbol{\vartheta}^{\mathrm{a}})$ by using another DNN (the actor network) with parameter $\boldsymbol{\vartheta}^{\mathrm{a}}$. Given $Q(\dot{\boldsymbol{s}}_k, \boldsymbol{a}_k; \boldsymbol{\vartheta}^{\mathrm{c}})$, the policy gradient calculated for optimizing the policy $\pi(\dot{\boldsymbol{s}}_k; \boldsymbol{\vartheta}^{\mathrm{a}}_*)$ is given as follows:

$$\nabla_{\boldsymbol{\vartheta}^{\mathrm{a}}} \Omega(\boldsymbol{\vartheta}^{\mathrm{a}}) = \frac{1}{|\Xi|} \sum_{\xi_k \in \Xi} \nabla_{\mathbf{w}} Q(\dot{\boldsymbol{s}}_k, \boldsymbol{a}_k; \boldsymbol{\vartheta}^{\mathrm{c}})\big|_{\pi(\dot{\boldsymbol{s}}_k; \boldsymbol{\vartheta}^{\mathrm{a}})} \nabla_{\boldsymbol{\vartheta}^{\mathrm{a}}} \pi(\dot{\boldsymbol{s}}_k; \boldsymbol{\vartheta}^{\mathrm{a}}) \tag{22}$$

where $\Omega(\boldsymbol{\vartheta}^{\mathrm{a}})$ represents the value of the objective function (13a) achieved with the 3-D map management policy $\pi(\dot{\boldsymbol{s}}_k; \boldsymbol{\vartheta}^{\mathrm{a}})$.

B. Blended Experience Replay

With state augmentation, the expansive state space poses a challenge for a model-free DRL-method in finding the optimal policy. This is because the required volume of collected real experiences for policy learning significantly increases with the state space size. Therefore, our idea behind the proposed MBRL method is using both the collected real experiences and the generated artificial experiences provided by the UDT for accelerating the policy learning. Specifically, the proposed MBRL method incorporates a new mechanism, called *blended experience replay*, to accelerate the training of the actor and critic networks by using both types of experiences from the UDT. Correspondingly, the batch used for DNN training in (21) and (22) can be selected from Ξ^{a}, Ξ^{r}, or both.

We propose an adaptive map management (AMM) algorithm using the UDT data in Algorithm 2. In Line 4, we use the UDT to extract the latent features $\mu_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$ and $\Sigma_{\boldsymbol{\theta}}(\boldsymbol{d}_k)$

TABLE I
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
D^{req}	0.5 second	α	5 Mbits
$d_k^{\max}$	80 Mbits/second	$d_k^{\min}$	40 Mbits/second
$V^{\max}$	25-45 frames	F	60 frames

for enabling state augmentation as mentioned in Section V-B. This procedure corresponds to the gray arrows crossing from the "UDT" segment to the "Controller" segment in Fig. 4. In Lines 5-6, the decision on the output action $\boldsymbol{a}_k$ is made by the actor network given state $\dot{\boldsymbol{s}}_k$, and the corresponding reward and the next state are observed. Next, we train the actor and critic networks by using the blended experience replay, as shown in Lines 7-9. Specifically, we randomly select I tuples of collected real experiences from Ξ^{r} and $|\Xi| - I$ tuples of generated artificial experiences from Ξ^{a}, respectively, and combine them in a batch to update the parameters of the actor and the critic networks. The procedure of using the UDT to support blended experience replay corresponds to the gray arrows from the "MAR Device Data Profile" block to the "Blended Experience Replay" block in Fig. 4. The parameter I controls the ratio of the amount of artificial experiences to the overall batch size in training. In Line 11, the UDT updates the set of real experiences Ξ^{r} by adding the newly collected tuples into the set. This step corresponds to the gray arrow crossing from the "Physical Network" segment to the "UDT" segment in Fig. 4.

In addition to training the DNNs for MBRL, we adopt a resource-efficient online training method to update the UDT as given in lines 11-13. We introduce a hyperparameter W for UDT update, and optimize the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ per W time slots as mentioned in Section V. Meanwhile, the UDT generates new artificial experiences according to the newly collected real experiences, which increases the adaptivity of the proposed MBRL method.

VII. NUMERICAL RESULTS

In our simulations, we use the "westgate-playroom" camera frame sequence in the SUN3D data set [45], which contains data collected in a real indoor environment. The set of 3-D map points detected in each camera frame $\mathcal{M}_f$ is obtained using the open-source ORB-SLAM framework [20]. Important parameter settings are listed in Table I.

We utilize two long-short-term-memory (LSTM) layers with 300 neurons, followed by four fully connected layers with (256, 128, 128, 32) neurons, to build the DNN with parameter $\boldsymbol{\theta}$ for latent feature extraction in the UDT. Meanwhile, we use four fully connected layers with (256, 128, 64, 16) neurons to build the DNN with parameter $\boldsymbol{\varphi}$ for artificial experience generation in the UDT.

For the actor and the critic DNNs used in the MBRL scheme, we leverage two graph convolutional networks (GCNs) as embedding layers to capture the relationship among camera frames in the 3-D map. Each GCN consists of two graph convolutional layers with (128, 32) neurons. Following the embedding layer, three fully connected layers with (64, 32, 32) neurons and four fully connected layers with (64, 32, 16, 4) neurons are used for building the critic and the actor DNNs, respectively.

We adopt the following 3-D map management schemes in MAR as benchmark [14], [20], [21].

1) *Latest Frame First (LFF):* The 3-D map is periodically updated by adding the lastly captured camera frames and removing the camera frames captured the earliest.
2) *Periodical Uploading (PU):* The camera frames to upload are selected uniformly from all camera frames captured within each time slot, and the earliest captured camera frames are removed from the 3-D map.
3) *ADAPT [21]:* The set of camera frames contained in the 3-D map is selected from all camera frames captured within each time slot, by using an optimization method proposed to minimize the uncertainty.

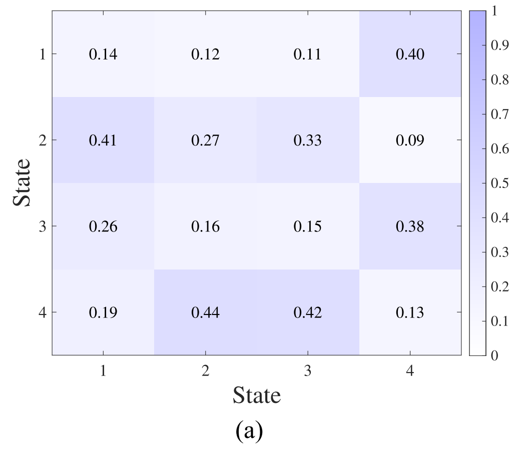
A. Performance of UDT

In this section, we compare the performance of the proposed UDT-based approach with that of LSTM-based and Markov model-driven approaches in capturing the dynamics of the uplink data rate.

We first show the accuracy of the designed UDT in capturing the dynamics of the time-varying uplink data rate within one time interval. In Fig. 5, a comparison is made between the estimated and actual values of the state transition matrix for a 4-state Markov chain within one time interval. Each state in the Markov chain corresponds to a distinct uplink data rate, i.e., d_k, and each value in Fig. 5 shows a state transition probability. We can observe that the values estimated by the UDT are very close to the actual values of the state transition matrix, which underlines the effectiveness of the UDT in capturing the stationary uplink data rate within one time interval.

Next, we compare the performance of two designed UDTs with latent features of different dimensions, i.e., Z, with that of a data-driven method (labeled as "LSTM") in capturing the nonstationary dynamics of the uplink data rate across multiple time intervals. Specifically, three time intervals are considered, each of which corresponds to a unique state transition matrix. As shown in Fig. 6, we plot the error of the estimated state transition matrix for the three schemes versus the number of states in the Markov chain, i.e., N. The value of each bar represents the average error over 20 independent simulation runs. We can observe that the designed UDT outperforms the LSTM-based method in all scenarios since the LSTM-based method simply makes deterministic predictions rather than capturing the underlying state transition probabilities. Additionally, given the UDT, the error increases with N since a larger value of N results in more parameters to approximate.

In Fig. 7, we compare the performance of the designed UDT with that of a Markov model-driven approach (labeled as "Markov model-driven") across 15 time intervals, in the scenarios with stationary and nonstationary uplink data rates. For this model-driven approach, we predefine a mathematical



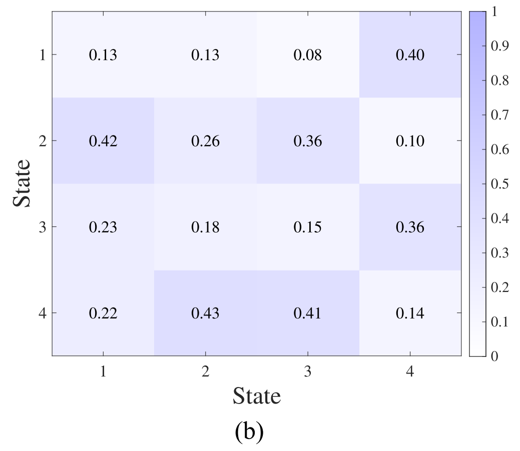
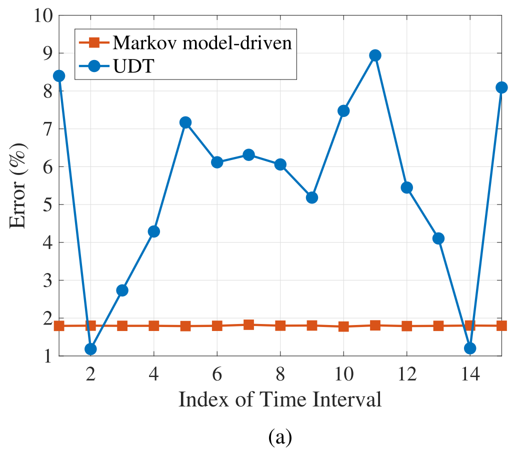


Fig. 5. Estimated and the actual values of the state transition matrix of a 4-state Markov chain within one time interval. (a) Estimated values using the UDT. (b) Actual values.



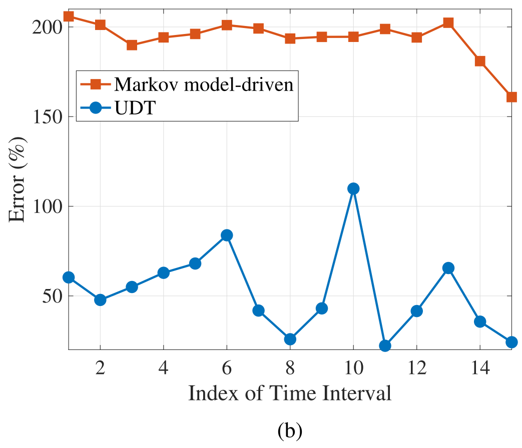


Fig. 7. Performance comparison between the UDT and the Markov model-driven approach across 15 time intervals. (a) Stationary uplink data rate. (b) Nonstationary uplink data rate.

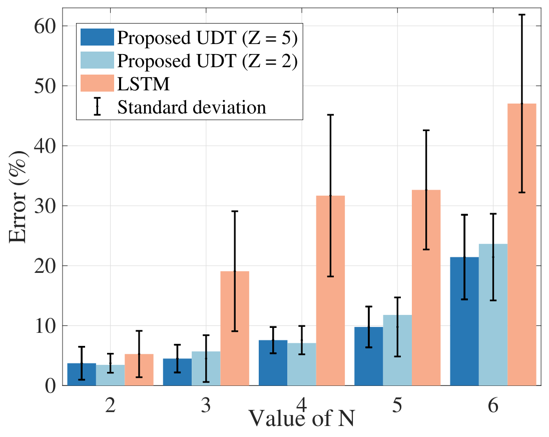


Fig. 6. Performance comparison between the UDT and the LSTM-based prediction across three time intervals.

model, i.e., N state Markov chain, and estimate its state transition matrix as model parameters according to the statistics of collected data. In Fig. 7(a), when the time-varying uplink data rate is stationary across all the time intervals, i.e., the state transition matrix is constant, the model-driven approach slightly outperforms the designed UDT in terms of error. This is because the model-driven approach operates on a known *a priori* mathematical model rather than using a data model to approximate the mathematical model. However, in Fig. 7(b), when the state transition matrix underlying the Markov chain varies across time intervals, the UDT-based approach significantly outperforms the model-driven approach since the designed UDT can capture the time-varying dynamics across time intervals using a data model with DNNs.

B. MBRL for 3-D Map Management

In this section, we evaluate the performance of the proposed MBRL scheme for 3-D map management using the UDT.

In Figs. 8 and 9, we compare the performance of the proposed MBRL scheme in two cases, one using the UDT (labeled as "Proposed (UDT)") and the other using LSTM to generate artificial experiences (labeled as "Proposed (LSTM)"), with that of the three benchmark 3-D map management schemes in one time interval. The uplink data rate follows a two-state Markov chain, with each point representing the average over 15 independent simulation runs. In Fig. 8, by setting different transition matrices of the two-state Markov chain, we change the ratio of the time slots corresponding to the high-rate state to all time slots. We observe that the proposed MBRL scheme for 3-D map management achieves a lower pose estimation uncertainty than the three benchmark schemes. This is because the MBRL scheme, with the help

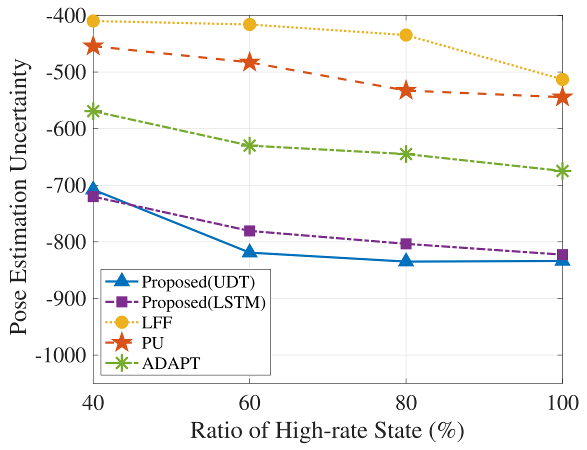


Fig. 8. Pose estimation uncertainty versus the ratio of high-rate state when $N = 2$.

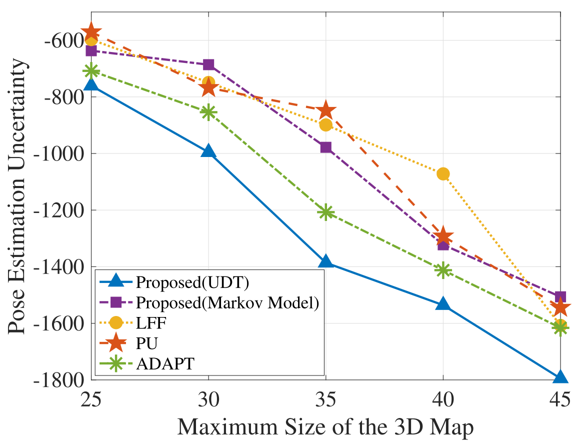


Fig. 10. Pose estimation uncertainty versus the 3-D map size.

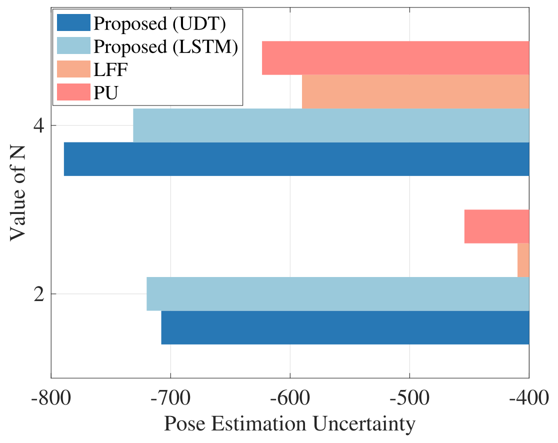


Fig. 9. Performance comparison between MBRL and conventional 3-D map management schemes when $N = 2$ and $N = 4$.

of the UDT, can learn a policy that prioritizes the camera frames for 3-D map management by considering their long-term impacts, as opposed to the myopic 3-D map management adopted by the three benchmark schemes. This allows the proposed scheme to cope with the dynamics of the uplink data rate and the user's pose. In addition, we can observe that the "Proposed (UDT)" and the "Proposed (LSTM)" schemes have similar performance when $N = 2$, but the former outperforms the latter when $N = 4$. This is because the error of LSTM significantly decreases with value of N as shown in Fig. 6, thereby reducing the accuracy of the generated artificial experiences.

In Fig. 9, we evaluate the impact of the time-varying uplink data rate on the performance of 3-D map management. Specifically, we examine two scenarios, in which the expected uplink data rate is the same while the state transition matrix for the Markov chain has 2 and 4 states, respectively. Compared with the scenario with 4 states, the variance of the uplink data rate across time slots is larger in the scenario with 2 states. We can observe that the performance advantage of the proposed MBRL scheme expands with the value of N. This is because dealing with a large variance in the uplink data rate requires the policy to take into account the long-term impact of 3-D map management decision in each time slot on subsequent time slots, which cannot be achieved by the benchmark schemes.

In Fig. 10, we compare the performance of the proposed MBRL scheme with that of the three benchmark schemes versus the maximum 3-D map size, $V^{\max}$, which ranges from 25 to 45 camera frames. For each map size, three time intervals are simulated, and the uplink data rate is nonstationary across the three time intervals. In addition to "LFF," "PU," and "ADAPT" schemes, we use the proposed MBRL scheme that employs a Markov model (instead of the UDT) to capture x_t as a benchmark scheme, which is labeled as "Proposed (Markov model)." We have two observations from Fig. 10. First, given different sizes of 3-D maps, the proposed MBRL using the UDT scheme can select an appropriate set of camera frames for uploading and updating the 3-D map based on their long-term impacts. Thus, it outperforms the "LFF," the "PU," and the "ADAPT" schemes, which make decisions myopically. For example, the "ADAPT" scheme solves a myopic uncertainty minimization problem for each time slot rather than a sequential decision-making problem considering the long-term impact of 3-D map management decisions on subsequent time slots. Second, the proposed MBRL scheme using the UDT outperforms the "Proposed (Markov model)" scheme in terms of pose estimation uncertainty as well. This is because using a fixed Markov model to capture the nonstationary uplink data rate can be inaccurate, as shown in Fig. 7(b), thereby significantly hampering the capability of the MBRL scheme in learning the optimal 3-D map management policy. In contrast, the UDT can cope with the nonstationary uplink data rate and facilitate the proposed MBRL scheme.

VIII. Conclusion and Future Work

In this article, we have designed a UDT-based 3-D map management scheme to facilitate edge-assisted device pose tracking for MAR applications. The UDT established for the

MAR device can extract the latent features from the time-varying uplink data rate, thereby supporting the emulation of 3-D map management. By using the collected and generated data from the UDT, our MBRL scheme learns a 3-D map management policy to prioritizing camera frames for uploading to update the 3-D map, which minimizes pose estimation uncertainty. Numerical results have demonstrated the effectiveness of the UDT in capturing the dynamics of the uplink data rate and the adaptivity of the MBRL scheme in coping with the variations in the uplink data rate and the device pose. The designed network dynamics-aware scheme establishes a foundation for customizing UDTs to optimize 3-D map management policies based on the distinct network conditions of MAR devices. In the future, we will target efficient resource reservation at an edge server to support 3-D map management for multiple MAR devices, considering not only the uplink data rate but also the impacts of different device pose variation patterns on the computing, data storage, and communication resource demands.

Appendix
Proof of Lemma 1

Given a 3-D map $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the pose estimation uncertainty $u(\mathcal{G})$ can be calculated according to (5), given by [40]

$$\begin{aligned} u(\mathcal{G}) &= -\log\Big(\det\Big(\hat{\boldsymbol{L}}(\mathcal{G}) \otimes \boldsymbol{\Pi}\Big)\Big) \\ &= -\log\Big(\det\Big(\hat{\boldsymbol{L}}(\mathcal{G})\Big)^6 \det(\boldsymbol{\Pi})^{|\mathcal{V}|-1}\Big) \end{aligned} \tag{23}$$

where $|\mathcal{V}|$ denotes the number of camera frames in 3-D map $\mathcal{G}$ and the dimension of $\hat{\boldsymbol{L}}(\mathcal{G})$ is $(|\mathcal{V}| - 1) \times (|\mathcal{V}| - 1)$.

According to the Kirchhoff's Matrix-Tree Theorem [39], we can calculate the value of $\det(\hat{\boldsymbol{L}}(\mathcal{G}))$ for graph $\mathcal{G}$ based on its weighted number of a spanning tree, i.e., the weighted sum of all edges in a tree that connect all nodes in the graph without forming any cycles. Since adding a new edge to a connected graph always increases the weighted number of a spanning tree if the resulting graph remains connected [46], the following inequality holds:

$$\kappa(\mathcal{G}) < \kappa(\mathcal{G} \cup \{f\}),\ f \notin \mathcal{V} \tag{24}$$

where f denotes a newly added node corresponding to a newly uploaded camera frame, which creates at least one new edge in the 3-D map $\mathcal{G}$. According to the Kirchhoff's Matrix-Tree Theorem and (24), we can derive the following inequality:

$$\begin{aligned} u(\mathcal{G}) &= -\log\Big(\kappa(\mathcal{G})^6 \det(\boldsymbol{\Pi})^{|\mathcal{V}|-1}\Big) \\ &> -\log\Big(\kappa(\mathcal{G} \cup \{f\})^6 \det(\boldsymbol{\Pi})^{|\mathcal{V}|}\Big) \\ &= u(\mathcal{G} \cup \{f\}) \end{aligned} \tag{25}$$

where $\det(\boldsymbol{\Pi}) \geq 1$ when cameras are high-resolution and high-accuracy and can provide extensive and reliable information for device pose tracking (e.g., an identity matrix is adopted in [21]). Therefore, Lemma 1 is proved based on (25).

References

[1] C. Zhou, J. Gao, M. Li, N. Cheng, X. Shen, and W. Zhuang, "Digital twin-based 3D map management for edge-assisted mobile augmented reality," in *Proc. IEEE/CIC ICCC*, Dalian, China, 2023, pp. 1–6, doi: 10.1109/ICCC57788.2023.10233293.

[2] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–30, 1st Quart., 2021.

[3] H. Zhang, S. Mao, D. Niyato, and Z. Han, "Location-dependent augmented reality services in wireless edge-enabled metaverse systems," *IEEE Open J. Commun. Soc.*, vol. 4, pp. 171–183, 2023.

[4] Y. Yang, "Multi-tier computing networks for intelligent IoT," *Nature Electron.*, vol. 2, no. 1, pp. 4–5, 2019.

[5] L. Zhang, X. Wu, F. Wang, A. Sun, L. Cui, and J. Liu, "Edge-based video stream generation for multi-party mobile augmented reality," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 409–422, Jan. 2024.

[6] X. Shen et al., "Toward Immersive communications in 6G," *Front. Comput. Sci.*, vol. 4, Jan. 2023, Art. no. 1068478.

[7] Z. Tan, H. Qu, J. Zhao, S. Zhou, and W. Wang, "UAV-aided edge/fog computing in smart IoT community for social augmented reality," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4872–4884, Jun. 2020.

[8] X. Ran, C. Slocum, M. Gorlatova, and J. Chen, "ShareAR: Communication-efficient multi-user mobile augmented reality," in *Proc. 18th ACM HotNets*, 2019, Princeton, NJ, USA, pp. 109–116.

[9] J.-C. Piao and S.-D. Kim, "Real-time visual–inertial SLAM based on adaptive keyframe selection for mobile AR applications," *IEEE Trans. Multimedia*, vol. 21, no. 11, pp. 2827–2836, Nov. 2019.

[10] J. Linowes and K. Babilinski, *Augmented Reality For Developers: Build Practical Augmented Reality Applications With Unity, ARCore, ARKit, and Vuforia*. Birmingham, U.K.: Packt Publ. Ltd., 2017.

[11] M. Huzaifa et al., "ILLIXR: Enabling end-to-end extended reality research," in *Proc. IEEE IISWC*, Storrs, CT, USA, 2021, pp. 24–38.

[12] A. J. Ben Ali, M. Kouroshli, S. Semenova, Z. S. Hashemifar, S. Y. Ko, and K. Dantu, "Edge-SLAM: Edge-assisted visual simultaneous localization and mapping," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 1, pp. 1–31, 2022.

[13] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1160–1192, 2nd Quart., 2021.

[14] K. Apicharttrisorn et al., "Characterization of multi-user augmented reality over cellular networks," in *Proc. IEEE SECON*, 2020, pp. 1–9.

[15] Y. Han, Y. Chen, R. Wang, J. Wu, and M. Gorlatova, "Intelli-AR preloading: A learning approach to proactive hologram transmissions in mobile AR," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17714–17727, Sep. 2022.

[16] K. Chen, T. Li, H.-S. Kim, D. E. Culler, and R. H. Katz, "MARVEL: Enabling mobile augmented reality with low energy and low latency," in *Proc. 16th ACM SenSys*, 2018, Shenzhen, China, pp. 292–304.

[17] G. Pan, H. Zhang, S. Xu, S. Zhang, and X. Chen, "Joint optimization of video-based AI inference tasks in MEC-assisted augmented reality systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 2, pp. 479–493, Apr. 2023.

[18] J. Du, B. Jiang, C. Jiang, Y. Shi, and Z. Han, "Gradient and channel aware dynamic scheduling for over-the-air computation in federated edge learning systems," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1035–1050, Apr. 2023.

[19] C. Zhou, J. Gao, M. Li, X. Shen, and W. Zhuang, "Digital twin-empowered network planning for multi-tier computing," *J. Commun. Inf. Netw.*, vol. 7, no. 3, pp. 221–238, Sep. 2022.

[20] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[21] Y. Chen, H. Inaltekin, and M. Gorlatova, "AdaptSLAM: Edge-assisted adaptive SLAM with resource constraints via uncertainty minimization," in *Proc. IEEE INFOCOM*, New York, NY, USA, 2023, pp. 1–13.

[22] W. Zhang, B. Han, and P. Hui, "SEAR: Scaling experiences in multi-user augmented reality," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 5, pp. 1982–1992, May 2022.

[23] Z. Liu, G. Lan, J. Stojkovic, Y. Zhang, C. Joe-Wong, and M. Gorlatova, "CollabAR: Edge-assisted collaborative image recognition for mobile augmented reality," in *Proc. ACM/IEEE IPSN*, Sydney, Australia, 2020, pp. 301–312.

[24] X. Ran, C. Slocum, Y.-Z. Tsai, K. Apicharttrisorn, M. Gorlatova, and J. Chen, "Multi-user augmented reality with communication efficient and spatially consistent virtual objects," in *Proc. ACM CoNEXT*, New York, NY, USA, 2020, pp. 386–398.
[25] A. Dhakal, X. Ran, Y. Wang, J. Chen, and K. Ramakrishnan, "SLAM-share: Visual simultaneous localization and mapping for real-time multi-user augmented reality," in *Proc. 18th ACM CoNEXT*, Rome, Italy, 2022, pp. 293–306.
[26] P. Ren et al., "Edge AR X5: An edge-assisted multi-user collaborative framework for mobile Web augmented reality in 5G and beyond," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2521–2537, 2020, Oct.–Dec. 2022.
[27] N. Cheng et al., "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.
[28] Z. Ji, S. Wu, and C. Jiang, "Cooperative multi-agent deep reinforcement learning for computation offloading in digital twin satellite edge networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3414–3429, Nov. 2023.
[29] Y. Hui et al., "Collaboration as a service: Digital-twin-enabled collaborative and distributed autonomous driving," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 18607–18619, Oct. 2022.
[30] M. Li, J. Gao, C. Zhou, X. Shen, and W. Zhuang, "User dynamics-aware edge caching and computing for mobile virtual reality," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 5, pp. 1131–1146, Sep. 2023.
[31] Q. Luo, T. H. Luan, W. Shi, and P. Fan, "Deep reinforcement learning based computation offloading and trajectory planning for multi-UAV cooperative target search," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 504–520, Feb. 2023.
[32] S. Hu, M. Li, J. Gao, C. Zhou, and X. Shen, "Adaptive device-edge collaboration on DNN inference in AIoT: A digital twin-assisted approach," *IEEE Internet Things J.*, early access, Nov. 28, 2023, doi: 10.1109/JIOT.2023.3336600.
[33] V. Cozzolino, L. Tonetto, N. Mohan, A. Y. Ding, and J. Ott, "Nimbus: Towards latency-energy efficient task offloading for AR services," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1530–1545, Apr.–Jun. 2023.
[34] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.
[35] J. Tan, Y.-C. Liang, L. Zhang, and G. Feng, "Deep reinforcement learning for joint channel selection and power control in D2D networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1363–1378, Feb. 2021.
[36] K. Khosoussi, M. Giamou, G. S. Sukhatme, S. Huang, G. Dissanayake, and J. P. How, "Reliable graphs for SLAM," *Int. J. Robot. Res.*, vol. 38, nos. 2–3, pp. 260–298, 2019.
[37] M. L. Rodríguez-Arévalo, J. Neira, and J. A. Castellanos, "On the importance of uncertainty representation in active SLAM," *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 829–834, Jun. 2018.
[38] Y. Chen, S. Huang, L. Zhao, and G. Dissanayake, "Cramér–Rao bounds and optimal design metrics for pose-graph SLAM," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 627–641, Apr. 2021.
[39] C. Godsil and G. F. Royle, *Algebraic Graph Theory*. New York, NY, USA: Springer, 2001.
[40] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*, Tech. Univ. Denmark, Lyngby, Denmark, Nov. 2012. [Online]. Available: http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html
[41] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar, "Bayesian reinforcement learning: A survey," *Found. Trends Mach. Learn.*, vol. 8, nos. 5–6, pp. 359–483, 2015.
[42] N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi, "Sequential variational autoencoders for collaborative filtering," in *Proc. 12th ACM WSDM*, 2019, pp. 600–608.
[43] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.
[44] L. Zintgraf et al., "VariBAD: A very good method for Bayes-adaptive deep RL via meta-learning," in *Proc. ICLR*, 2020, pp. 1–20
[45] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using SfM and object labels," in *Proc. IEEE ICCV*, Sydney, NSW, Australia, 2013, pp. 1625–1632.
[46] K. Khosoussi, S. Huang, and G. Dissanayake, "Novel insights into the impact of graph structure on SLAM," in *Proc. IEEE/RSJ IROS*, Chicago, IL, USA, 2014, pp. 2707–2714.

Conghao Zhou (Member, IEEE) received the B.Eng. degree from Northeastern University, Shenyang, China, in 2017, the M.Sc. degree from the University of Illinois Chicago, Chicago, IL, USA, in 2018, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2022.

He is currently a Postdoctoral Fellow with the University of Waterloo. His research interests include space–air–ground integrated networks, network slicing, and machine learning for wireless networks.

Jie Gao (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical engineering from the University of Alberta, Edmonton, AB, Canada, in 2009 and 2014, respectively.

He was a Postdoctoral Fellow with Toronto Metropolitan (formerly Ryerson) University, Toronto, ON, Canada, from 2017 to 2019 and a Research Associate with the University of Waterloo, Waterloo, ON, Canada, from 2019 to 2020. He was an Assistant Professor with the Department of Electrical and Computer Engineering, Marquette University, Milwaukee, WI, USA, from 2020 to 2022 and is currently an Assistant Professor with the School of Information Technology, Carleton University, Ottawa, ON, Canada. His research interests include machine learning for communications and networking, cloud and multiaccess edge computing, Internet of Things (IoT) and Industrial IoT solutions, and B5G/6G networks in general.

Mushu Li (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2021.

She is currently a Postdoctoral Fellow with Toronto Metropolitan University, Toronto, ON, Canada. She was a Postdoctoral Fellow with the University of Waterloo from 2021 to 2022. Her research interests include mobile edge computing, the system optimization in wireless networks, and machine learning-assisted network management.

Dr. Li was the recipient of Natural Science and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowship (2022) and NSERC Canada Graduate Scholarship (2018).

Nan Cheng (Senior Member, IEEE) received the B.E. and M.S. degrees from the Department of Electronics and Information Engineering, Tongji University, Shanghai, China, in 2009 and 2012, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2016.

He was a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, from 2017 to 2019. He is currently a Professor with State Key Laboratory of ISN and with the School of Telecommunications Engineering, Xidian University, Xi'an, Shaanxi, China. He has published over 90 journal papers in IEEE transactions and other top journals. His current research focuses on B5G/6G, AI-driven future networks, and space–air–ground integrated network.

Prof. Cheng serves as an Associate Editor for IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, and *Peer-to-Peer Networking and Applications*, and serves/served as a Guest Editor for several journals.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks.

Dr. Shen received the "West Lake Friendship Award" from Zhejiang Province in 2023, the President's Excellence in Research from the University of Waterloo in 2022, the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society (ComSoc), the Technical Recognition Award from Wireless Communications Technical Committee (2019), and the AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award in 2003 from the Province of Ontario, Canada. He serves/served as the General Chair for the 6G Global Conference'23 and ACM Mobihoc'15, the Technical Program Committee Chair/Co-Chair for IEEE Globecom'24, 16, and 07, IEEE Infocom'14, and IEEE VTC'10 Fall, and the Chair for the IEEE ComSoc Technical Committee on Wireless Communications. He is the President of the IEEE ComSoc. He was the Vice President for Technical and Educational Activities, the Vice President for Publications, the Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and the member of IEEE Fellow Selection Committee of the ComSoc. He served as the Editor-in-Chief for the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and *IET Communications*. He is a Registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.

Weihua Zhuang (Fellow, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from Dalian Marine University, Dalian, China, in 1982 and 1985, respectively, and the Ph.D. degree in electrical engineering from the University of New Brunswick, Fredericton, NB, Canada, in 1993.

She is a University Professor and a Tier I Canada Research Chair of Wireless Communication Networks with the University of Waterloo, Waterloo, ON, Canada. Her research focuses on network architecture, algorithms, and protocols, and service provisioning in future communication systems.

Dr. Zhuang is the recipient of the 2021 Women's Distinguished Career Award from IEEE Vehicular Technology Society, the 2021 Technical Contribution Award in Cognitive Networks from IEEE Communications Society, the 2021 R.A. Fessenden Award from IEEE Canada, and the 2021 Award of Merit from the Federation of Chinese Canadian Professionals in Ontario. She was the Editor-in-Chief of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY from 2007 to 2013, the General Co-Chair of 2021 IEEE/CIC International Conference on Communications in China, the Technical Program Chair/Co-Chair of IEEE VTC2017/2016-Fall, the Technical Program Symposia Chair of 2011 IEEE Globecom, and an IEEE Communications Society Distinguished Lecturer from 2008 to 2011. She is an Elected Member of the Board of Governors and the President of the IEEE Vehicular Technology Society. She is a Fellow of Royal Society of Canada, Canadian Academy of Engineering, and Engineering Institute of Canada.