Sampling, Communication, and Prediction Co-Design for Synchronizing the Real-World Device and Digital Model in Metaverse

Zhen Meng, Changyang She, Guodong Zhao, and Daniele De Martini

***Abstract*—The metaverse has the potential to revolutionize the next generation of the Internet by supporting highly interactive services with the help of Mixed Reality (MR) technologies; still, to provide a satisfactory experience for users, the synchronization between the physical world and its digital models is crucial. This work proposes a sampling, communication and prediction co-design framework to minimize the communication load subject to a constraint on tracking the Mean Squared Error (MSE) between a real-world device and its digital model in the metaverse. To optimize the sampling rate and the prediction horizon, we exploit expert knowledge and develop a constrained Deep Reinforcement Learning (DRL) algorithm, named Knowledge-assisted Constrained Twin-Delayed Deep Deterministic (KC-TD3) policy gradient algorithm. We validate our framework on a prototype composed of a real-world robotic arm and its digital model. Compared with existing approaches: (1) When the tracking error constraint is stringent (MSE $= 0.002^\circ$), our policy degenerates into the policy in the sampling-communication co-design framework. (2) When the tracking error constraint is mild (MSE $= 0.007^\circ$), our policy degenerates into the policy in the prediction-communication co-design framework. (3) Our framework achieves a better trade-off between the average MSE and the average communication load compared with a communication system without sampling and prediction. For example, the average communication load can be reduced up to 87% when the average track error constraint is 0.002°. (4) Our policy outperforms the benchmark with the static sampling rate and prediction horizon optimized by exhaustive search, in terms of the tail probability of the tracking error. Furthermore, with the assistance of expert knowledge, the proposed algorithm KC-TD3 achieves better convergence time, stability, and final policy performance.**

***Index Terms*—Sampling, communication, prediction, constraint deep reinforcement learning, metaverse.**

I. Introduction

Facilitated by the rapid development of Augmented Reality (AR), Virtual Reality (VR) and Mixed Reality (MR), the metaverse is expected to change our daily lives in different aspects, such as shopping, social interaction, healthcare [1], education [2] and gaming [3]. One of the main challenges is providing an immersive and highly interactive experience [4], which requires synchronization between the physical and virtual worlds to ensure smooth user motion tracking and timely feedback. Indeed, poor synchronization can lead to chaotic interactions and dizziness [5]. Besides, in mission-critical applications assisted by the metaverse, even slight out-of-synchronization between a real-world device and the digital model may cause serious consequences [6].

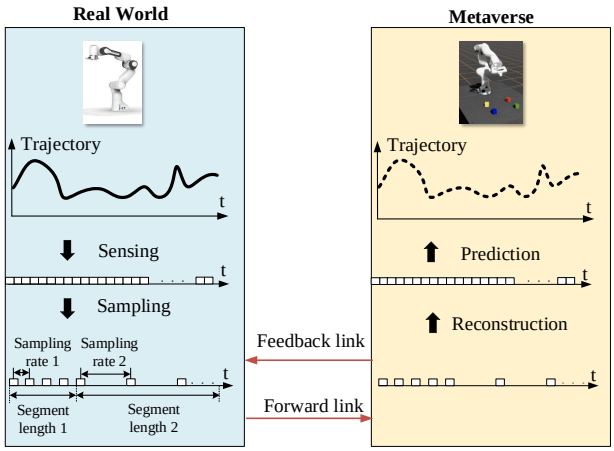


Fig. 1. Proposed co-design framework to synchronize a real-world device and its digital model in the metaverse.

The synchronization performance can be characterized by three key metrics: Motion-To-Photon (MTP) latency, data rate, and packet-loss rate. MTP latency refers to the time between a user's action and the corresponding effect displayed in the virtual world [7]. In applications that require haptic feedback, the required MTP should be less than $1\,\mathrm{ms}$ [8]. The data rate, i.e. the maximum rate of data transfer across the network, is the bottleneck for most multimedia applications [9], especially when a large amount of data is generated by multimedia sources – e.g. High-Definition video, AR/VR/MR, gaming, massive sensor networks, etc. Further, the packet-loss rate, i.e. the percentage of packets sent by a transmitter but not received by the receiver, is crucial for mission-critical applications, such as remote robotic control, smart-factories monitoring and online healthcare [10].

It is very challenging to meet the three key performance metrics in practice. Although the three use cases have been considered the fifth-generation (5G) cellular networks, Enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-

Z. Meng and G. Zhao are with James Watt School of Engineering, University of Glasgow, UK. (e-mail: z.meng.1@glasgow.ac.uk; guodong.zhao@glasgow.ac.uk)

C. She is with the School of Electrical and Information Engineering, University of Sydney, Australia. (e-mail: shechangyang@gmail.com)

D. De Martini is with Oxford Robotics Institute, University of Oxford, UK. (e-mail: daniele@robots.ox.ac.uk)

Corresponding author: Changyang She.

Latency Communication (URLLC) and Massive Machine Type Communication (mMTC), the requirements in metaverse cannot be fulfilled; indeed, while 5G New Radio would allow a sub-millisecond delay in the radio access network, the End-to-End (E2E) delay is still far from the MTP requirement [11]. Moreover, to support interactive applications globally, the metaverse requires an extremely high data rate, far beyond the capabilities of 5G networks at 50 Gbits per second [12].

Recently, researchers started to investigate interdisciplinary approaches beyond conventional communication system designs. The existing literature has two branches of related work: sampling-communication co-design [13]–[18] and prediction-communication co-design [19]–[22]. (1) Sampling-Communication Co-Design uses Age of Information (AoI), mutual information, or goal-oriented semantics to down-sample the information/packets at the transmitter side; thus, the task is completed with less communication load after reconstructing the original information at the receiver side. (2) Prediction-Communication Co-Design, instead, predicts a device's future state based on its historical states; if the prediction horizon equals the communication delay, the user's experienced delay is zero [23]. Further, if the communication presents packet losses, the same approach can infer the missing states.

Considering that sampling, communication, and prediction are closely interconnected and interdependent, we argue that it is possible to improve the performance compared with the above existing work by combining the three. For example, suppose that a predictor can estimate missing information in a longer prediction horizon; in that case, the overall reliability is less sensitive to packet losses in communications, and the sampling rate can be reduced. Nevertheless, prediction and reconstruction errors may deteriorate the user experience; still, the impact on the synchronization in the metaverse remains unclear, and a coherent design framework that combines sampling, communication and prediction is not available in the existing literature.

A. Related Work

We discuss both sampling-communication and prediction-communication co-designs as present in the existing literature.

1) Sampling-Communication Co-design: AoI is a performance metric widely used in co-design communication systems and sampling policies (also called state update policies) [13]–[15]. In [13], the authors optimized the sensing and updating policy for an air pollution monitoring application by minimizing the weighted sum of the AoI and the total energy consumption of the device. By adjusting the weighting coefficients of the AoI and tuning the energy consumption manually, it is possible to achieve the target trade-off. To collect new data from power-constrained sensors in an Industrial Internet of Things (IIoT) network, the authors of [14] optimized a scheduling algorithm by decoupling the multi-sensor problem into single-sensor problems. In [15], the authors considered a status update problem over an error-prone wireless channel, where the average cost of sampling and communications was minimized subject to average AoI constraints.

Since AoI is an intermediate performance metric that does not fully capture the requirement of a specific task and is not aware of the burstiness of the source, it is not a good metric [16]. For example, when the state of an environment or device is stationary, there is no need to update the state frequently. When the state changes rapidly, the source should generate packets and update frequently. For this reason, different performance metrics and design frameworks have been considered to improve sampling efficiency, e.g. goal-oriented communications [17] and mutual information [18]. The authors of [17] developed a goal-oriented sampling and communication policy for status updates over an unreliable wireless channel, where only effective samplings for lowing real-time reconstruction errors were allowed to be transmitted to the actuator. The results show that the proposed strategy can significantly improve the effective updates and reduce the cost of actuation errors. [18], instead, used the mutual information between the real-time source values and the samples delivered to the receiver to optimize the sampling policy, proposing a transmitter that maximizes the expected mutual information by sending a new packet once the latter is below a threshold.

2) Prediction-Communication Co-design: Prediction plays an essential role in reducing the user-experienced delay in URLLC. To reduce round-trip delay in a VR application, the authors of [19] proposed to predict, pre-render and cache VR videos in an edge server, where Long Short-Term Memory (LSTM) and Multi-Layer Perception (MLP) neural networks predict body and head motion, respectively. In [20], the authors considered an AR robotic telesurgical application, where, with the help of prediction, they could reduce the task completion time by 19% without increasing the manipulation error rate. The authors of [21] jointly optimized the communication and packetized predictive control system to minimize the wireless resource consumption under the control outage probability constraint. The results in [22] indicated that prediction and communication co-design can achieve a better trade-off between reliability and latency than traditional communication systems without prediction.

B. Contributions

In this paper, we investigate how to synchronize the trajectories of a real-world device and its digital model in the metaverse. The main contributions of this paper are summarized as follows:

- We establish a sampling, prediction, and communication co-design framework for synchronizing the trajectories of a device in the real world and its digital model in the metaverse. The sampling rate and the prediction horizon are jointly optimized to minimize the communication load subject to a Mean Squared Error (MSE) constraint between the trajectory of the device and its digital model.
- In the co-design framework, we propose a Knowledge-assisted Constrained Twin-Delayed Deep Deterministic (KC-TD3) algorithm by combining Deep Reinforcement

Learning (DRL) techniques with expert knowledge on sampling, communication and prediction. Specifically, the following learning techniques are applied to improve the reinforcement learning algorithm: 1) extension of double Q-Learning, 2) state-space reduction, 3) interdependent action normalization, and 4) accelerated primal-dual policy optimization (APDO).
- We build a prototype comprising a real-world robotic arm and its digital model in the metaverse. The experimental results show that our proposed algorithm achieves good convergence time and stability. Compared with a communication system without sampling and prediction, the sampling, communication, and prediction co-design framework can reduce the average tracking error and the communication load by $87.5\,\%$ and $87\,\%$, respectively. Besides, the co-design framework works well in communication systems with high packet loss probabilities, $1\,\%$ to $10\,\%$.

The rest of this paper is organized as follows. In Section II, we propose the co-design framework and formulate a joint design problem that includes sampling, communication, and prediction components in one optimization problem. In Section III, we develop the KC-TD3 algorithm to optimize the sampling and prediction policy while minimizing the communication load. Section IV describes the prototype we used to verify our method and Section V provides performance evaluations. Finally, Section VI concludes the paper.

II. Sampling, Communication, and Prediction Co-design Framework

In this section, we describe the proposed framework and formulate the co-design problem as the foundation of our algorithm.

A. Co-Design Framework

As shown in Figure 1, we consider the synchronization between a real-world device and its digital model in the metaverse. Specifically, a sensor of the device first measures its trajectory[1], Then, the data is sampled, i.e. decimated, and transmitted to the metaverse, where it can be reconstructed and used to predict the future trajectory. To reduce the latency between the real-world devices and the digital model in the metaverse, the digital model of the device follows the predicted trajectory and feeds back the prediction results to the real-world devices. Finally, the device compares its trajectory with the predicted one and adjusts the sampling rate and the prediction horizon.

As shown in Figure 2, time is discretized into slots. The E2E delays in the forward and feedback links of long distance communication networks are denoted by D_d and D_f, respectively. The unit is defined as the number of time slots, where the duration of each time slot is set to 1 ms in our experiments. We assume that E2E delay includes both computing delay and communication delay. If the delay of a packet is longer than a delay bound $D_{\max}$, the packet is outdated and will be dropped by the system.

The trajectory of the device is measured by the sensor in each time slot. We can subdivide the raw sensor measurements into sequences, which we call *segments*; we denote the k-th segment by $\mathcal{T}(k) = \{\tau_k(i) \mid i = 1, \dots W_k\}$, where $\tau_k(i)$ are the sensor readings and W_k is the length of the segment $\mathcal{T}(k)$. We assume that the length of each segment is larger than the communication delay. As such, the k-th segment can arrive at the receiver by the end of the $(k+1)$-th segment. Thus, we have the following constraint

$$W_k \geq D_{\max}. \tag{1}$$

Since the raw trajectory samples are highly correlated in the time domain – and thus the data packets – they are highly redundant. For this reason, to reduce the communication load, we can subsample the raw data from the sensor and then transmit them to the cloud server via the long distance communication network. Each trajectory segment is sampled at a fixed rate $r(k)$. Let $\tilde{\mathcal{T}}(k) = \{\tilde{\tau}_k(i) \mid i = 1, \dots n_k\}$ be the sampled trajectories of the k-th segment, where n_k is the number of packets sent by the transmitter.

After reception, the cloud server reconstructs the original trajectory from the sampled one – we denote it by $\bar{\mathcal{T}}(k) = \{\bar{\tau}_k(i) \mid i = 1, \dots W_k\}$ – and, given the last historical trajectories $\bar{\mathcal{T}}(1), \bar{\mathcal{T}}(2), \dots \bar{\mathcal{T}}(k)$, predicts the future trajectory of the device, i.e. $\hat{\mathcal{T}}(k+1)$ and $\hat{\mathcal{T}}(k+2)$ of length W_{k+1} and W_{k+2}, respectively. As such, the prediction horizon needs to be

$$H(k+2) = W_{k+1} + W_{k+2}. \tag{2}$$

Considering that the communication delay in the feedback link does not exceed $D_{\max}$, which is shorter than W_{k+2}, the device obtains $\hat{\mathcal{T}}(k+2)$ by the end of the $(k+2)$-th segment, $\mathcal{T}(k+2)$. By measuring the average tracking error, which is defined as the MSE between the true trajectory, $\mathcal{T}(k+2)$, and the predicted trajectory, $\hat{\mathcal{T}}(k+2)$, the system can adjust the sampling rate and the length of the $(k+3)$-th segment W_{k+3}. According to (2), in our optimization framework, instead of optimizing prediction horizon directly, we can optimize the length of each segment. Besides, the sensing data of the real-world device are displayed to the user by different equipment (e.g., High-Definition Screen, AR, VR glasses) via local communication networks (e.g., High-Definition Multimedia Interface (HDMI) or other outputs ports).

B. Sampling, Reconstruction, and Prediction

1) Sampling: The transmitter only sends the sampled packets to the receiver, residing on the remote server. Denoting the number of samples of the k-th segment by n_k, the sampling rate of the k-th segment is given by

$$r(k) = \frac{n_k}{W_k}. \tag{3}$$

[1]The trajectory could be any observations such as pressure, temperature, humidity, etc. The reason why we do not use "state" in the system design is that the definition of "state" in our reinforcement learning algorithm is not the same as "state" (trajectory) measured by sensors. To avoid misunderstanding, we use "trajectory" in our system design.

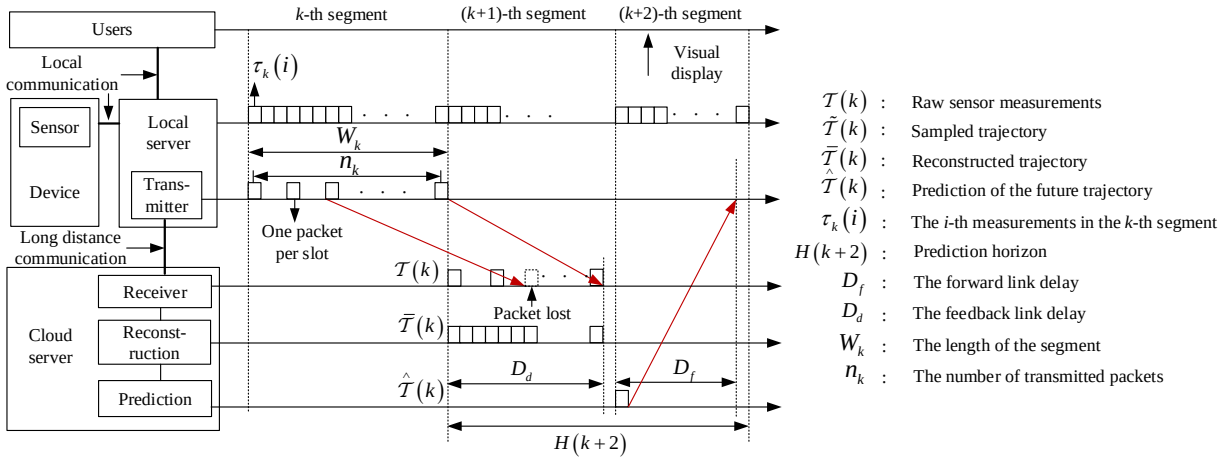


Fig. 2. The timing sequence of the proposed co-design framework (The senor belongs to the real world device for data generation and the transmitter belongs to a local server. The receiver and functions for construction and predication are deployed at the cloud server that operates the metaverse).

As a result, the sampled trajectory can be obtained as follows,

$$\tilde{\tau}_k(i) = \tau_k\left(\left\lfloor \frac{W_k}{n_k} \right\rfloor \cdot (i-1)+1\right), i = 1, ..., n_k. \quad (4)$$

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

2) Reconstruction: In digital signal processing, there are several up-sampling methods for state reconstruction, such as inverse fast Fourier transform, repetitions of the last received state, linear interpolation, and barycentric interpolation [24]. Without loss of generality, due to its wide adoption, and its simplicity, we adopt linear interpolation [25]. The relationship between the reconstructed trajectory and the sampled trajectory is given by

$$\bar{\mathcal{T}}(k) = f_{\mathrm{I}}(\tilde{\mathcal{T}}(k); \theta_{\mathrm{I}}), \quad (5)$$

where θ_{I} are the parameters for the interpolation function.

3) Prediction: We utilize a MLP that takes the historical trajectory as its input and generates the predicted one. The relationship between the input and output of the predictor in the t_{k+1}-th slot can be expressed as

$$\begin{aligned}&[\hat{\tau}_{k+1}(t_{k+1} - W_{k+1} + 1), ..., \hat{\tau}_{k+1}(t_{k+1} + W_{k+2})] \\ &= f_{\mathrm{P}}(\bar{\tau}(t_{k+1} - L_{\mathrm{in}} - W_{k+1}), \bar{\tau}(t_{k+1} - L_{\mathrm{in}} - W_{k+1} + 1), \\ &\quad ..., \bar{\tau}(t_{k+1} - W_{k+1}); \theta_{\mathrm{P}}), \end{aligned} \quad (6)$$

where L_{in} is the length of the historical trajectory and θ_{P} represents the MLP parameters. We ignore the subscript k of the $\bar{\tau}$ since the input of the predictor includes multiple trajectory segments. It is worth noting that the value of L_{in} is determined by the temporal correlation of the trajectory and does not depend on the length of the k-th segment, W_k.

C. Tracking Error and Communication Load

1) Tracking Error: The tracking error of the k-th segment, denoted by $e(k)$, is given by

$$\begin{aligned} e(k) &= \mathrm{MSE}(\mathcal{T}(k), \hat{\mathcal{T}}(k)) \\ &= \frac{1}{|W_k|}\sum_{t=1}^{W_k}(\tau_k(i) - \hat{\tau}_k(i))^2, \end{aligned} \quad (7)$$

where $\tau_k(i)$ is the trajectory measured at the i-th time slot in the k-th segment, and $\hat{\tau}_k(i)$, for $i = 1, ..., W_k$, is the prediction trajectory.

2) Communication Load: We assume that the communication load is proportional to the packet rate of the system. For example, in Orthogonal Frequency-Division Multiplexing (OFDM) communication systems (adopted in the 4-th generation and 5-th generation cellular networks), the time-frequency resource blocks occupied by a packet are determined by the packet size and the channel gain. If the packet size and the channel gain are stationary, the average resource blocks allocated to a device are proportional to the packet rate (i.e., the sampling rate in our system). Since the number of packets to be transmitted for the k-th segment is n_k, the required radio resource blocks to transmit the k-th segment of the trajectory are proportional to n_k. Therefore, to minimize the required radio resource blocks for this service, we minimize the sampling rate subject to a constraint on the MSE in (7). It is also worth noting that the sampling frequency in our system is different from the sampling frequency of OFDM systems. We assume that only one packet can be transmitted in one slot (i.e., transmission time interval in OFDM systems). By adjusting the sampling rate, we adjust the packet rate in the communication system. For example, if the sampling rate is 0.1 per slot, the transmitter only sends one packet every ten slots and remains silence in the other nine slots.

D. Problem Formulation

To reduce the communication load subject to the MSE constraint, we optimize the sampling rate and the prediction horizon. The problem can be formulated as follows,

$$\min_{H(k),n_k} \lim_{N\to+\infty} \frac{1}{N}\sum_{k=1}^{N} r(k) \tag{8}$$

$$\text{s.t.} \quad \lim_{N\to+\infty} \frac{1}{N}\sum_{k=1}^{N} e(k) \leq \Gamma_c, \tag{8a}$$

where N represents the number of segments and Γ_c is the required average MSE threshold; this latter is dependant on specific applications and is considered fixed for both training and deployment.

III. Constrained Deep Reinforcement Learning for Optimizing Sampling, Prediction, and Communication

The problem (8) is a sequential decision problem and can be re-formulated as a Constrained Markov Decision Processes (CMDP). Typically, the weighted sum reward is used for handling multi-criterion optimization (sampling rate and MSE). However, for the weighted sum reward, we need to adjust the weighting coefficient manually. With the constraint reinforcement learning, we can find the weighting coefficient by optimizing the Lagrangian multiplier in the dual domain. In other words, the constraint reinforcement learning algorithm can obtain a suitable weighting coefficient that can guarantee the MSE constraint. Therefore, to solve the problem, we integrate expert knowledge into the primal-dual Deep Deterministic Policy Gradient (DDPG) algorithm and develop the KC-TD3 algorithm.

A. CMDP Formulation

State: The state $\mathbf{s}_k$ observed by the device by the end of the k-th segment (the t_k-th time slot) includes the last two trajectory segments measured by the device, $\mathcal{T}(k-1)$ and $\mathcal{T}(k)$, and those predicted by the cloud sever, $\hat{\mathcal{T}}(k-1)$ and $\hat{\mathcal{T}}(k)$. The predicted trajectory segments depend on the reconstructed historical trajectory $\bar{\mathcal{T}}_{in}(k) = \bar{\tau}(t_k - L_{\text{in}} - W_k - W_{k-1}), ..., \bar{\tau}(t_k - W_k - W_{k-1})$. As shown in (4) and (5), $\bar{\mathcal{T}}_{in}(k)$ is determined by the true trajectory from the $(t_k - L_{\text{in}} - W_k - W_{k-1})$-th slot to the $(t_k - W_k - W_{k-1})$-th slot, as well as the sampling rate and the prediction horizon.

Action: The action to be taken by the end of the k-th segment includes the length of the $(k+1)$-th segment, W_{k+1}, and the number of samples to be transmitted n_{k+1} (equivalent to the sampling rate r_{k+1}). Thus, the action is denoted by $\mathbf{a}_k = (W_{k+1}, n_{k+1})$. For convenience, we denote two actions as a vector denoted by $\mathbf{a}_k = [\mathbf{a}_k{}^{[1]}, \mathbf{a}_k{}^{[2]}]$, where $\mathbf{a}_k{}^{[1]} = W_{k+1}$ and $\mathbf{a}_k{}^{[2]} = n_{k+1}$. Based on this definition, we have $\mathbf{a}_k{}^{[1]} \in \{D_{\max}, ..., W_{\max}\}$ and $\mathbf{a}_k{}^{[2]} = 1, ..., \mathbf{a}_k{}^{[1]}$.

Instantaneous Reward and Cost: Given the state and the action at the end of the k-th segment, the instantaneous reward is the negative of sampling rate and the cost is the MSE of the $(k+1)$-th segment. According to (3) and (7), we have $r_k = -r(k+1)$ and $c_k = e(k+1)$.

Policy: The agent follows a deterministic policy denoted by $\mu : \mathbf{a}_k = \mu(\mathbf{s}_k|\theta)$, where θ represents the parameters of the policy.

Long-Term Reward and Long-Term Cost: Following the policy $\mu(\cdot|\theta)$, the long-term discounted reward is given by

$$R^{\mu(\cdot|\theta)} = \mathbb{E}[\sum\nolimits_{k=0}^{\infty} \gamma^k r_k], \tag{9}$$

where the γ is the discount factor. Similarly, the long-term discounted cost under the policy $\mu(\cdot|\theta)$ is given by

$$C^{\mu(\cdot|\theta)} = \mathbb{E}[\sum\nolimits_{k=0}^{\infty} \gamma^k c_k]. \tag{10}$$

CMDP Formulation: The goal is to find the optimal policy $\mu^*(\cdot|\theta^*)$ that maximizes the long-term reward $R^{\mu(\cdot|\theta)}$ subject to the constraint on the long-term cost $C^{\mu(\cdot|\theta)}$. Thus, the problem can be reformulated as follows:

$$\mu^*(\cdot|\theta^*) = \arg\max_{\mu(\cdot|\theta)} R^{\mu(\cdot|\theta)} \tag{11}$$

$$\text{s.t.} \quad C^{\mu(\cdot|\theta)} \leqslant \frac{\Gamma_c}{1-\gamma}. \tag{11a}$$

To utilize DRL to solve the problem, we first prove that the transitions of the system follow an MDP (see the proof in Appendix A).

The Lagrangian function of the constrained optimization problem is defined as follows [26],

$$\Gamma(\mu(\cdot|\theta), \lambda) = R^{\mu(\cdot|\theta)} - \lambda(C^{\mu(\cdot|\theta)} - \frac{\Gamma_c}{1-\gamma}), \tag{12}$$

where λ is the Lagrangian multiplier. Then, the constrained problem can be converted to the following unconstrained problem,

$$(\mu^*(\cdot|\theta^*), \lambda^*) = \arg\min_{\lambda\geqslant 0} \max_{\mu(\cdot|\theta)} \Gamma(\mu(\cdot|\theta), \lambda). \tag{13}$$

B. Preliminary of Primal-Dual DDPG

Primal-dual DDPG is an off-policy method to solve CMDP [27]. It combines the DDPG algorithm with the primal-dual method to find the optimal policy and the dual variable. The policy, the long-term reward, and the long-term cost are represented by three neural networks, and we denote them by $\mu(\cdot|\theta)$, $Q^R(\cdot|\phi^R)$ and $Q^C(\cdot|\phi^C)$, where θ, ϕ^R, and ϕ^C are the parameters of these three neural networks, respectively.

During the training process, the corresponding action generated by the actor network is given by

$$\mathbf{a}_k = clip\left(\mu(\mathbf{s}_k|\theta) + clip(\varepsilon, -c, c), \mathbf{a}_{\text{Low}}, \mathbf{a}_{\text{High}}\right), \tag{14}$$

where ε is the White Gaussian Noise with distribution $\varepsilon \sim \mathcal{N}(0, \sigma)$, $[\mathbf{a}_{\text{Low}}, \mathbf{a}_{\text{High}}]$ is the action space, and

$$clip(x, c_1, c_2) = \min\left(\max\left(x, c_1\right), c_2\right). \tag{15}$$

After taking action $\mathbf{a}_k$ at the k-th step (i.e., selecting W_k and n_k in the last time slot of the k-th trajectory segment in our system), the system observes the instantaneous reward and cost, and transits from $\mathbf{s}_k$ to $\mathbf{s}_{k+1}$. The transition is

denoted by $\mathcal{D}_k \triangleq \langle \mathbf{s}_k, \mathbf{a}_k, r_k, c_k, \mathbf{s}_{k+1} \rangle$, which is stored in the replay memory, $\mathcal{M}$. In each training step, a number of transitions (mini-batch) are randomly selected from replay memory $\mathcal{M}$ and are used to optimize θ, ϕ^R, and ϕ^C. We denote $\mathcal{D}_{k_i} = \langle \mathbf{s}_{k_i}, \mathbf{a}_{k_i}, r_{k_i}, c_{k_i}, \mathbf{s}_{k_i+1} \rangle, i = 1, 2, ..., N_{\text{batch}}$ as the i-th transition in the k-th episode of the training stage, where N_{batch} is the batch size. To optimize the critic and cost neural networks, the Bellman equation is utilized. The target reward and cost functions can be expressed as follows:

$$Q^R(\mathbf{s}_{k_i}, \mathbf{a}_{k_i}) = r + \gamma Q^R\left(\mathbf{s}_{k_i+1}, \mu(\mathbf{s}_{k_i+1}|\theta)|\phi^R\right), \quad (16)$$

$$Q^C(\mathbf{s}_{k_i}, \mathbf{a}_{k_i}) = r + \gamma Q^C\left(\mathbf{s}_{k_i+1}, \mu(\mathbf{s}_{k_i+1}|\theta)|\phi^C\right). \quad (17)$$

Then, the long-term reward and long-term reward cost neural networks are updated by using the mean-squared Bellman error (MSBE) loss function which are derived as follows

$$L\left(\phi^R\right) = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \left[\left(Q^R(\mathbf{s}_{k_i}, \mathbf{a}_{k_i}) - Q^R\left(\mathbf{s}_{k_i}, \mathbf{a}_{k_i}|\phi^R\right)\right)^2\right], \quad (18)$$

$$L\left(\phi^C\right) = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \left[\left(Q^C(\mathbf{s}_{k_i}, \mathbf{a}_{k_i}) - Q^C\left(\mathbf{s}_{k_i}, \mathbf{a}_{k_i}|\phi^C\right)\right)^2\right]. \quad (19)$$

In primal-dual DDPG, the actor policy is updated by maximizing the Lagrangian function in (12), where the long-term reward and the long-term cost are replaced by the critic network and the cost network, respectively, i.e.,

$$\max_{\theta} \mathbb{E}\left[Q^R(\mathbf{s}_{k_i}, \mathbf{a}_{k_i}) - \lambda Q^C(\mathbf{s}_{k_i}, \mathbf{a}_{k_i})\right]. \quad (20)$$

After that, the dual variable λ is updated by gradient descent to minimize the Lagrangian function according to

$$\lambda^{(k+1)} = \left[\lambda_i^{(k)} + \beta_k \left(Q^C\left(\mathbf{s}_{k_i}, \mu(\mathbf{s}_{k_i}|\theta)|\phi^C\right) - \frac{\Gamma_c}{1-\gamma}\right)\right]^+, \quad (21)$$

where β_k is the step size and $[x]^+ = \max\{0, x\}$.

C. KC-TD3 Design

The straightforward application of primal-dual DDPG in our problem can cause several issues (to be discussed in the following). To improve the performance in the training process of primal-dual DDPG, we propose to exploit advanced reinforcement learning techniques and expert knowledge on sampling, communication, and prediction including 1) extension of double Q-Learning, 2) state-space reduction, 3) interdependent action normalization, and 4) APDO. The resulting approach is KC-TD3.

1) Extension of Double Q-Learning: The overestimation bias of the critic network will lead to poor performance when optimizing the actor network [28]. Similarly, with primal-dual DDPG, if the cost network is underestimated, the constraint cannot be satisfied. In our problem, we have a constraint on the average synchronization error between the virtual model and real-world device. As in double Q-learning critic networks are trained to approximate the state-action value function, where the target value of the Bellman equation is the smaller one of the two critic networks. Thus, the Bellman equation can be expressed as follows,

$$Q^R(\mathbf{s}_{k_i}, \mathbf{a}_{k_i}) = r + \gamma \min_{l=1,2} Q^R_{\phi_l^R}\left(\mathbf{s}_{k_i+1}, \mu(\mathbf{s}_{k_i+1}|\theta)|\phi_l^R\right), \quad (22)$$

where $Q^R_{\phi_l^R}$ is the estimated state-action reward function from the l-th reward network with parameter ϕ_l^R. Based on the knowledge of the synchronization error constraint that the synchronization error should be smaller than a required threshold, two cost networks are trained to approximate each state-action cost function. The target cost value is estimated by the larger one of the two cost networks. Thus, the Bellman equation can be expressed as follows,

$$Q^C(\mathbf{s}_{k_i}, \mathbf{a}_{k_i}) = r + \gamma \max_{l=1,2} Q^C_{\phi_l^C}\left(\mathbf{s}_{k_i+1}, \mu(\mathbf{s}_{k_i+1}|\theta)|\phi_l^C\right), \quad (23)$$

where $Q^C_{\phi_l^C}$ is the estimated state-action cost function of the l-th cost network with parameter ϕ_l^C.

2) State-Space Reduction: The original state $\mathbf{s}_k$ consists of two trajectory segments measured by the device, $\mathcal{T}(k-1)$ and $\mathcal{T}(k)$, and that predicted by the cloud server, $\hat{\mathcal{T}}(k-1)$ and $\hat{\mathcal{T}}(k)$. According to the expert knowledge of the our prediction algorithm, the dimension of the state $\mathbf{s_k}$ is dynamic and depends on the action W_k. However, the dimension of the input of the actor network is fixed, and thus we cannot feed the state $\mathbf{s}_k$ to the actor network directly. One possible approach is to replace the trajectory segments with the input of the prediction algorithm, $\bar{\mathcal{T}}_{\text{in}}$, which relies on the measured trajectory segments and determines the predicted trajectory segments. Although the dimension of $\bar{\mathcal{T}}_{\text{in}}$ is fixed, it can be large when the correlation time of the trajectory is large (up to a few seconds) and the state generation rate is high (1000 samples/s in our prototype). This may lead to a long training time and require a large number of samples. To improve the learning efficiency, we replace the original state with the MSE in (6),

$$\dot{s}_k = e(k). \quad (24)$$

In this way, we can reduce the input of the actor network to a scalar.

3) Interdependent Action Normalization: Considering the aforementioned action in Section III-A, we design the action with two elements, $\mathbf{a}_k = [\mathbf{a}_k{}^{[1]}, \mathbf{a}_k{}^{[2]}]$, where $\mathbf{a}_k{}^{[1]} \in \{D_{\max}, ..., W_{\max}\}$ and $\mathbf{a}_k{}^{[2]} = 1, ..., \mathbf{a}_k{}^{[1]}$. According to the knowledge that the sampling interval cannot exceed the length

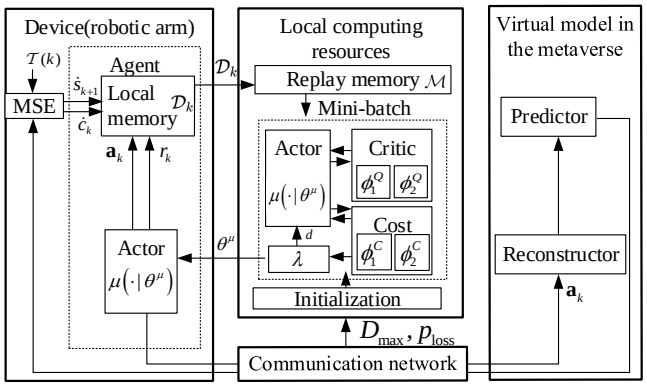


Fig. 3. Illustration of proposed KC-TD3 architecture.

of one trajectory segment, the feasible region of the second element depends on the first element. Based on that we normalize $\mathbf{a}_k{}^{[2]}$ by using W_k (inverse of $\mathbf{a}_k{}^{[1]}$), i.e.,

$$\dot{\mathbf{a}}_k^{[2]} = \frac{\mathbf{a}_k{}^{[2]}}{W_k}. \tag{25}$$

With this normalization, we have $\dot{\mathbf{a}}_k^{[2]} \in [0,1]$ and use a sigmoid function in the output layer of the actor network.

4) APDO: The dual variable updating procedure only utilizes on-policy samples, which leads to low sampling efficiency. To address this issue, we proposed to apply APDO [27], where the dual variable is updated by the dual gradient ascent every d_λ iteration (shown in the steps 16-18 in Algorithm 1). With this approach, the historical data samples stored in the replay buffer are utilized to update λ and help to improve the sample efficiency and the convergent speed.

D. KC-TD3 Training Architecture

The proposed KC-TD3 structure can be implemented in the real-world system with the architecture in Fig. 3, which mainly consists of a device (such as a robotic arm), local computing resource (a desktop/local server connected to the robotic arm), the communication network, and the metaverse.

1) Communication Network Initialization: The delay, $D_{\max}$, and the packet loss probability, p_{loss}, in the communication network are measured in the initialization stage. With linear reconstruction in (5), we need at least two samples to reconstruct each trajectory segment. Otherwise, the system is in outage. We denote the outage probability by $\emptyset$. For a given requirement on the outage probability, such as 10^{-5}, and a packet loss probability in the communication network (up to $10\,\%$ in our experiments), the minimum number of samples the transmitter should update for each segment is denoted by $N_{\min}$. To meet the outage probability requirement, $N_{\min}$ can be obtained from the following expression,

$$p_{\text{loss}}^{N_{\min}} + N_{\min} p_{\text{loss}}^{N_{\min}-1}(1-p_{\text{loss}}) \leq \emptyset. \tag{26}$$

Algorithm 1 KC-TD3

Input: Initialize parameters of actor, critic and cost networks, θ, ϕ_1^R, ϕ_2^R, ϕ_1^C, ϕ_2^C. Measure the latency and packet loss probability in the communication network, $D_{\max}$ and p_{loss}. Obtain the minimal number of samples the device needs to update for each trajectory segment from (26).

Output: Optimal $\lambda^*$and optimal policy $\mu^*(\cdot|\theta^*)$.

1: Initialize the target networks: $\theta_{\text{arg}} \leftarrow \theta$, $\phi_{\text{arg},1}^R \leftarrow \phi_1^R$, $\phi_{\text{arg},2}^R \leftarrow \phi_2^R$, $\phi_{\text{arg},1}^C \leftarrow \phi_2^C$, $\phi_{\text{arg},2}^C \leftarrow \phi_2^C$.
2: Initialize the Lagrangian multiplier $\lambda = 0$, actor update delay d_{a}, dual variable update delay d_λ.
3: **for** episode m = 1,... **do**
4: Observe the average tracking error $\dot{s}_k$.
5: Generate an action based on (14) and execute the action.
6: Observe the reward, cost, the next state, and store $\langle \dot{s}_{k_i}, \mathbf{a}_{k_i}, r_{k_i}, c_{k_i}, \dot{s}_{k_i+1} \rangle$ in memory $\mathcal{M}$.
7: **for** j in range (from 1 to the maximal number of updates) **do**
8: Randomly sample a batch of transitions $\langle \dot{s}_{k_i}, \mathbf{a}_{k_i}, r_{k_i}, c_{k_i}, \dot{s}_{k_i+1} \rangle$ from $\mathcal{M}$.
9: Updating Q^R-functions by one step of gradient descent based on (22), (18).
10: Updating Q^C-functions by one step of gradient descent based on (23), (19).
11: **if** j mod d_{a} = 0 **then**
12: Update the actor policy by by one step of gradient ascent based on (20).
13: **end if**
14: **if** k mod d_λ = 0 **then**
15: Update dual variable by one step of gradient ascent based on (21).
16: **end if**
17: Update target network by:

$$\begin{aligned} \phi_{\text{arg}}^R &\leftarrow \rho\phi_{\text{arg},i}^R + (1-\rho)\phi_i^R; \\ \phi_{\text{arg}}^C &\leftarrow \rho\phi_{\text{arg},i}^C + (1-\rho)\phi_i^C; \\ \theta_{\text{arg}} &\leftarrow \rho\theta_{\text{arg}} + (1-\rho)\theta; \\ i &= \{1,2\}. \end{aligned}$$

18: **end for**
19: **end for**

2) Training Algorithm: The details of the algorithm can be found in Algorithm 1. With the observed average tracking error, the agent at the device takes actions according to the output of the actor network. Then, the average tracking error of the next trajectory segment is measured and saved in the replay memory. After that, the transitions in the replay memory are randomly selected to optimize the critic and cost networks (referred to as one iteration of the gradient descent optimization). Finally, the actor network and the dual variable λ are updated every d_{a} and d_λ iterations (referred to as update delay in Algorithm 1), respectively.

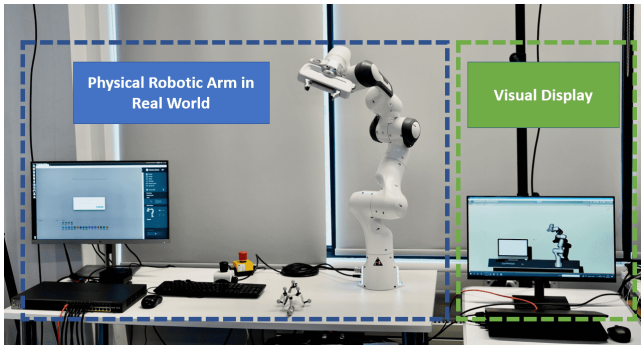


Fig. 4. Our prototype system (the movements of a physical robotic arm and the visual display. The digital model in the metaverse to be synchronized is in the remote which is not shown in the graph).

IV. Prototype Design and Data Collection

A. Prototype Design

We build a prototype[2] as shown in Fig. 4, where a virtual robotic arm needs to synchronize with a physical robotic arm in the real world. This is essential for many future use cases, such as education, healthcare, Industry 4.0, etc.

Physical Robotic Arm in Real World: An industrial-grade robotic arm system, Franka Emika Panda [29], is used in our prototype. It has seven degree of freedoms (DoFs) achieving up to 2 m/s end-effector speed and +/-0.1 mm repeatability. The robotic arm receives the target end-effector position from a controller, and then conducts inverse-kinematics calculation to map the target end-effector position to the seven joint angle positions of the robotic arm. After that, the robotic arm applies a proportional-integral-derivative method [30] for control, which converts the joint angle positions to a series of commands on the angular velocity of each joint. Multiple types of sensing data including joint angle values, joint angular velocities and inertial torque of joints are obtained by the application programming interface provided by libfranka, which is a C++ implementation of the Franka Control Interface [29].

Virtual Robotic Arm in the Metaverse: Unity software is used to generate the virtual robotic arm in the metaverse [31]. Specifically, we construct the digital model of the physical robotic arm, Franka Emika Panda, with the same number of DoFs deployed on a cloud server. The virtual robotic arm needs to obtain the angle position of each joint in real time. Such that the virtual robotic arm can be synchronized with the physical robotic arm. In the communication system, our prototype uses User Datagram Protocol to connect the virtual robotic arm to the physical one. Besides, the sensing data of the robotic arm are displayed to the user by High-Definition Screen via HDMI.

[2]In our prototype, we simplify the system by considering the one-way synchronization from a single physical robotic arm to a virtual robotic arm, and test our co-design framework and KC-TD3 algorithm. The proposed co-design can be extended to multi-user cases.

Fig. 5. Illustration of our data collection via an experiment, where a human operator controls the physical robotic arm to draw the "star" shape in the air (The demonstration video of our data collection is available at https://youtu.be/LCqSGtkrgug)

TABLE I
System Parameters for Performance Evaluation

Parameters	Values
Slot duration	1 ms
Transmission time interval T	1 ms
Time delay bound $D_{\max}$	10 ms
The range of segment length $\mathcal{T}(k)$	10 ms to 100 ms
The range of sampling rate $n(k)$	20 Hz to 1000 Hz
Experimental time	2×10^4 ms

B. Data Collection

As shown in Fig. 5, a human operator controls the physical robotic arm via a motion capture system [32], drawing a "star" shape in the air for 20 seconds.

Specifically, six OptiTrack Prime-13 motion capture cameras are deployed in a 4×4 m^2 area, where the human operator holds a tool with seven passive markers [32]. The motion capture system constructs the seven markers as a rigid body and outputs the position (3 DoFs) and orientation (3 DoFs) of the rigid body at the frequency of 120 Hz. The robotic arm receives the position and orientation as the target position of its end-effector. By controlling the angles of the seven joints, the physical robotic arm is able to move its end-effector towards the target position. In this way, the end-effector of the robotic arm tracks the human operator's hand trajectory in real-time. In this work, we use the data of the first joint from the base to demonstrate our design. During the movement of the robot arm, the device measures the joint angles at a frequency of 1kHz, and saves the data in csv format files, which will be used in the subsequent neural network training[3]

V. Performance Evaluation

In this section, we first evaluate the training performance of our KC-TD3, and then compare the performance of the

[3]Unless otherwise specified, the settings of the experimental parameters are listed in TABLE I.

proposed sampling, communication, and prediction co-design framework with different benchmarks.

A. Evaluation of KC-TD3 Algorithm

1) KC-TD3 Training Performance: The average KC-TD3 training performance and standard deviation are illustrated in Figs. 6(a)-6(c). In each training experiment, we provide the results in 800 episodes. Then, we repeat the same experiment 10 times. In Fig. 6(a), we provide the normalized average communication load, where the communication load without sampling is $100\,\%$. In Figs. 6(b) and 6(c), the average tracking error and the value of the dual variable are presented. The results in Figs. 6(a)-6(c), show that the KC-TD3 algorithm converges to a stable policy after 300 episodes. The normalized average communication load is 13%, and the average tracking error is MSE $= 0.007°$, which satisfies the average tracking error constraint.

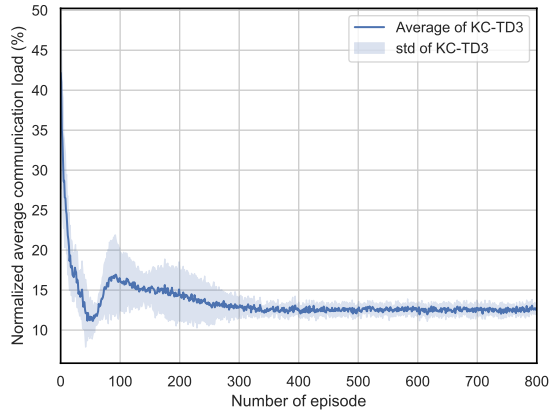
2) Ablation Study on Expert Knowledge: In Figs. 7(a)-7(c), we illustrate the impacts of different expert knowledge on the training performance of the proposed KC-TD3. When only partial expert knowledge is available, we obtained three benchmarks: (a) KC-TD3 without extension of double Q-Learning; (b) KC-TD3 without state-space reduction; (c) KC-TD3 without interdependent action normalization.

From Figs. 7(a)-7(c), we observe that the proposed strategy with full expert knowledge has the best performance in terms of stability and convergence. It also achieves the lowest communication load (in Fig. 7(a)), and satisfies the average tracking error constraint (in Fig. 7(b)). By comparing the proposed KC-TD3 algorithm with the three benchmarks, we can obtain the following insights: (1) without the extended double Q-Learning, the algorithm can hardly meet the average tracking error constraint; (2) without state-space reduction, the algorithm cannot converge to the optimal solution, where the average tracking error is far below the constraint at the cost of a higher communication load. (3) Without the interdependent action normalization, the obtained policy can meet the average tracking error constraint, but it does not minimize the communication load.

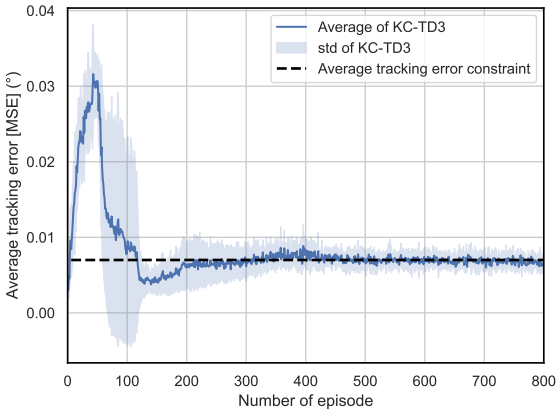
B. Validation of the Sampling, Communication, and Prediction Co-design Framework

1) Fixed Sampling Policy: We first evaluate the performance of a benchmark policy with fixed sample rate and prediction horizon. Specifically, Fig. 8 shows the normalized average communication load versus the number of samples in each segment. The results show that the normalized average communication load grows as the number of samples in each segment increases, but decreases as the prediction horizon increases. This means that a higher sampling rate (the ratio of the number of samples in each segment to the duration of the segment, which is half of the prediction horizon) leads to a higher communication load.

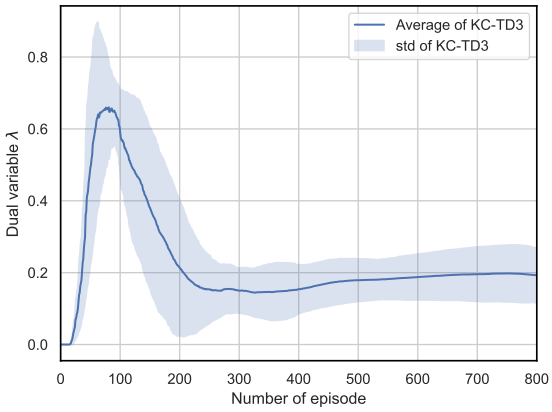
We further provide the average tracking errors with different sampling rates in Fig. 9. The results show that the average tracking error decreases dramatically as the number of samples



(a) Normalized average communication load in each episode (If the normalized average communication load is 100%, it means that the average communication load is the same as the system without sampling).

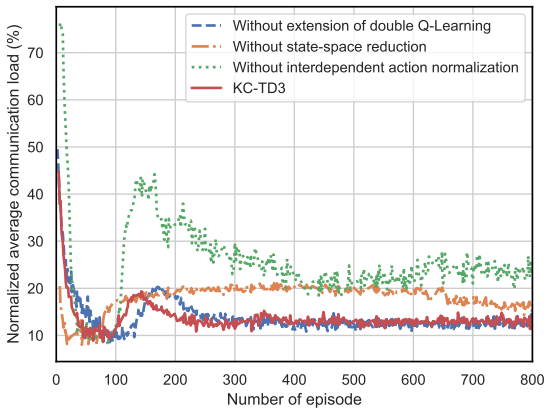


(b) Average tracking error in each episode.

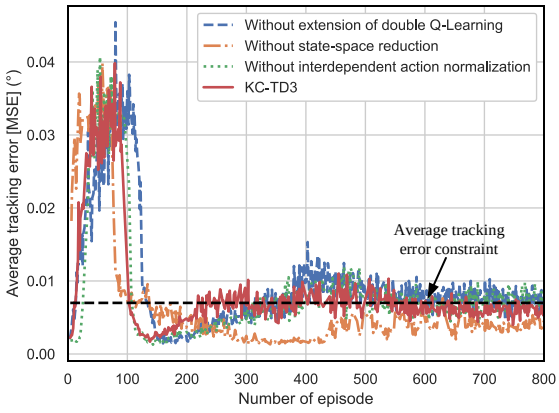


(c) Dual variable in each episode.

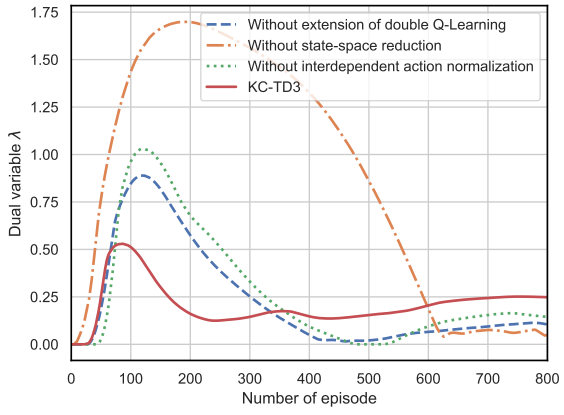
Fig. 6. The average KC-TD3 training performance and standard deviation.



(a) Normalized average communication load in each episode.



(b) Average tracking error in each episode.



(c) Dual variable in each episode.

Fig. 7. Ablation Study of different expert knowledge on the training performance of the proposed KC-TD3.

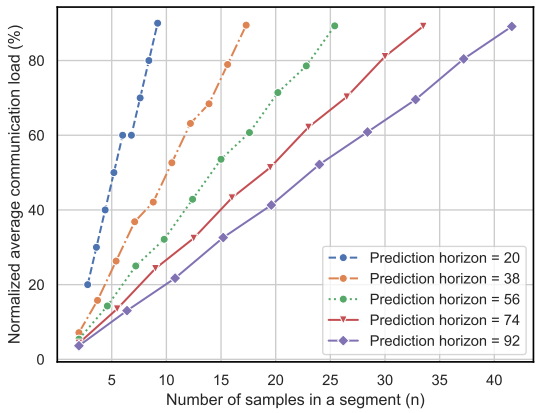


Fig. 8. Normalized average communication load under different prediction horizons and sampling numbers.

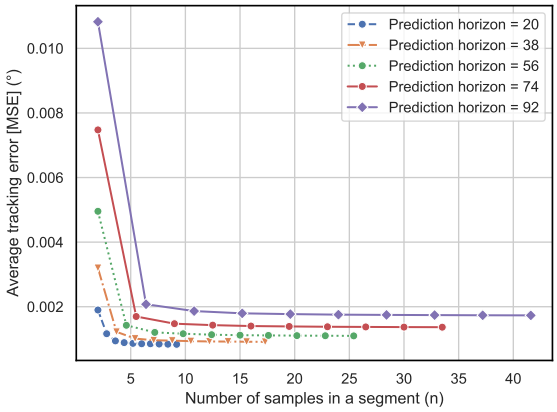


Fig. 9. Average tracking error under different prediction horizons and the number of samples.

in each segment n increases from 2 to 10. When $n > 10$, the average tracking error is nearly constant. In addition, a longer prediction horizon leads to a higher average tracking error. Therefore, the number of samples in each segment and the segment length should be optimized to obtain the minimum normalized average communication load subject to the average tracking error constraint.

2) Dynamic Sampling in KC-TD3 : We provide an example in Fig. 10 to illustrate how the proposed KC-TD3 dynamically adjusts the segment length (which is equivalent to adjusting the length of prediction horizons) and the number of samples in each segment. Compared with existing approaches: (1) When the average tracking error constraint is stringent (MSE $= 0.002°$), our policy degenerates into the policy obtained from the sampling-communication co-design framework. (2) When the average tracking error constraint is mild (MSE $= 0.007°$), our policy degenerates into the policy obtained from the prediction-communication co-design framework. This indicates that our proposed strategy is a more

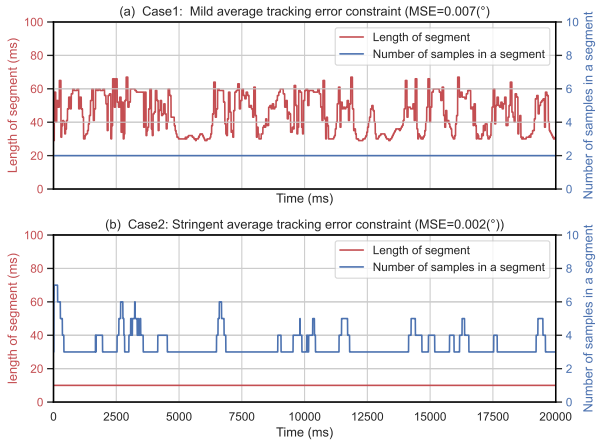


Fig. 10. A demonstration of dynamic sampling of the proposed KC-TD3.

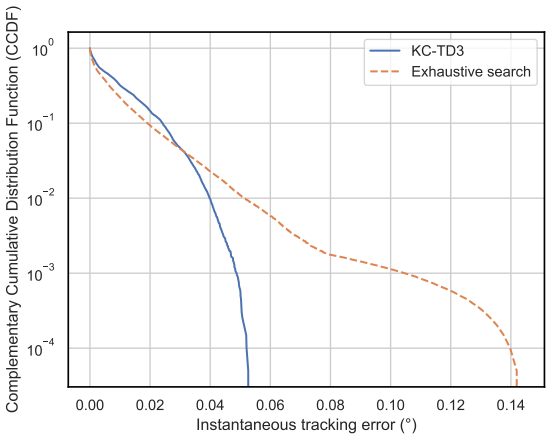


Fig. 12. CCDF comparison of exhaustive search and proposed KC-TD3, where the E2E latency is 50 ms and the average MSE constraint is $0.007°$.

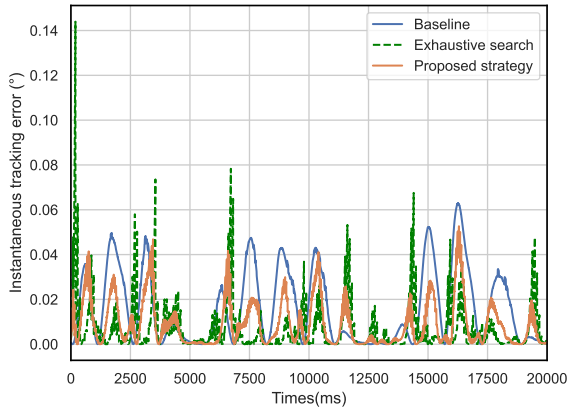


Fig. 11. Instantaneous tracking error of Baseline, exhaustive search (with optimized static n_k and $H(k)$), and proposed KC-TD3 (with optimized dynamic n_k and $H(k)$), where the E2E latency is 50 ms and the average MSE constraint is $0.007°$.

general design framework compared with the two existing frameworks in the existing work. The results also indicate that the prediction-communication co-design framework cannot find the optimal policy when the tracking error constraint is stringent (MSE $= 0.002°$), and the sampling-communication co-design framework cannot find the optimal policy when the average tracking error constraint is mild (MSE $= 0.007°$).

3) Overall Performance: Fig. 11 compares the tracking errors of different design approaches, where the delay bound is $D_{\max} = 50$ ms and the average tracking error constraint is MSE = $0.007°$. The "Baseline" approach transmits all samples to the receiver with the communication load of 100% and there is no prediction at the receiver side. The "Exhaustive search" approach is obtained by searching a fixed sampling rate and a fixed prediction horizon that minimize the normalized average communication load subject to the constraint on the average tracking error. We propose it here only as a performance benchmark, since it is not feasible for a practical

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT DESIGN

	Baseline	Exhaustive search	KC-TD3
Average tracking error constraint = $0.002°$			
Normalized average communication load (%)	100%	27%	27%
Average tracking error (°)	0.016	0.0020	0.0019
Average tracking error constraint = $0.007°$			
Normalized average communication load (%)	100%	13%	13%
Average tracking error (°)	0.016	0.0070	0.0069

online application. The instantaneous tracking errors achieved by the proposed strategy and the above two approaches are provided in Fig. 11, which shows that the proposed KC-TD3 can significantly reduce the instantaneous tracking error. This means that the synchronization between the virtual robotic arm and the physical one can be effectively improved.

Table II compares KC-TD3 with "Baseline" and "Exhaustive search" approaches. Compared with "Baseline" approach, KC-TD3 reduces the normalized average communication load by 73% and improves the average tracking error by 87.5% (when the average tracking error constraint is $0.002°$). When the average tracking error constraint is $0.007°$, KC-TD3 reduces the normalized average communication load by 87% and improves the average tracking error by 56.2% compared with "Baseline" approach. Besides, the normalized average communication load and the average tracking error achieved by KC-TD3 are the same as "Exhaustive search" approach.

Furthermore, we compared our policy that dynamically adjust sampling rate and prediction horizon according to the MSE with the exhaustive search approach that optimizes a static sampling rate and a static prediction horizon. We demonstrate the Complementary Cumulative Distribution Function

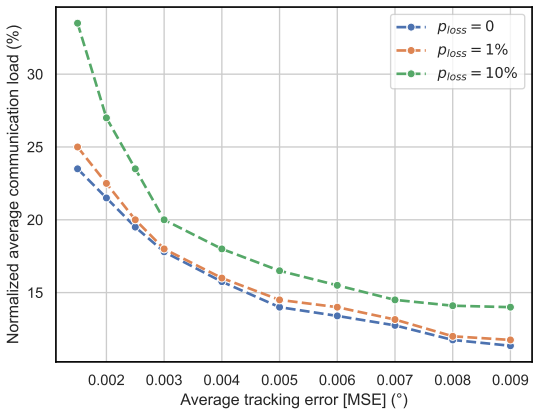


Fig. 13. Trade-off between normalized average communication load and average tracking error with different packet loss probabilities p_{loss} = 0, 1% and 10%.

(CCDF) of the tracking error in Fig. 12. The results indicate that the CCDF achieved by "Exhaustive search" has a longer tail compared with the CCDF achieved by KC-TD3. This means that by adjusting the sampling rate and the prediction horizon dynamically, KC-TD3 can effectively reduce the tail probability of the tracking error. In other words, KC-TD3 enjoys more stable tracking error performance.

Fig. 13 further demonstrates the trade-off between the normalized average communication load and the average tracking error achieved by KC-TD3, where different packet loss probabilities in the communication system are considered, i.e., $p_{\text{loss}} = 0, 1\%$, and 10%. The results reveal trade-off between the normalized average communication load and the average tracking error. In addition, with a smaller packet loss probability, it is possible to achieve a better trade-off between the normalized average communication load and the average tracking error. Furthermore, compared with "Baseline" approach, KC-TD3 can reduce up to 75% of normalized average communication load subject to a $0.002°$ average tracking error constraint even when the packet loss probability in the communication system is as high as 10%.

VI. Conclusions

In this paper, we demonstrated how to synchronize a real-world robotic arm and its digital model in the metaverse by sampling, communication, and prediction co-design. We established a framework for minimizing the average communication load under the constraint on the average tracking error between a real-world robotic arm and its digital model. Then, we proposed the KC-TD3 algorithm to adjust the sampling rate and the prediction horizon, where expert knowledge and advanced reinforcement learning techniques are exploited. In addition, we built a prototype of the proposed real-time robotic control system with a digital model in the metaverse. The results of our experiments showed that the proposed co-design framework can significantly reduce the communication load in the practical scenarios with communication packet losses.

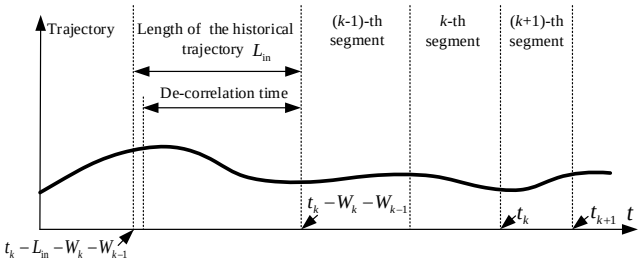


Fig. 14. The relationship between the de-correlation time and length of the historical trajectory.

Compared with several benchmarks, our KC-TD3 algorithm converges faster and can guarantee the average tracking error constraint.

Appendix A
Proof of the Markov Property

According to the definition, the k-th state includes, $\mathcal{T}(k-1)$, $\mathcal{T}(k)$, $\hat{\mathcal{T}}(k-1)$ and $\hat{\mathcal{T}}(k)$. As shown in (6), the predicted trajectory segments, $\hat{\mathcal{T}}(k-1)$ and $\hat{\mathcal{T}}(k)$, depend on the reconstructed historical trajectory, i.e.,

$$\bar{\mathcal{T}}_{\text{in}}(k) = [\bar{\tau}(t_k - L_{\text{in}} - W_k - W_{k-1}), ..., \bar{\tau}(t_k - W_k - W_{k-1})].$$

Further considering sampling and reconstruction in (4) and (5), $\bar{\mathcal{T}}_{in}(k)$ is determined by the actions and trajectory from the $(t_k - L_{\text{in}} - W_k - W_{k-1})$-th slot to the $(t_k - W_k - W_{k-1})$-th slot. Therefore, $\mathcal{T}(k-1)$ and $\mathcal{T}(k)$ in the k-th state are determined by the states and actions in the past $(t_k - L_{\text{in}} - W_k - W_{k-1})$ slots, which are available at the transmitter (the agent).

Similarly, the $(k+1)$-th state depends on the states and actions in the past $(t_{k+1} - L_{\text{in}} - W_{k+1} - W_k)$ slots, i.e., which is highly overlapped with that from the $(t_k - L_{\text{in}} - W_k - W_{k-1})$-th slot to the t_k-th slot. The new trajectory information by the end of the $(k+1)$-th state includes $[\tau(t_{k+1} - W_{k+1}), ..., \tau(t_{k+1})]$ and the k-th action is $\mathbf{a}_k$. The dimension of the input of the prediction algorithm, L_{in}, is the de-correlation time of the trajectory. Thus, $[\tau(t_{k+1} - W_{k+1}), ..., \tau(t_{k+1}))]$ only depends on the trajectory $[\tau(t_k - L_{\text{in}} - W_k - W_{k-1}), ..., \tau(t_k)]$ (with the assumption $W_{k+1} < W_k + W_{k-1}$) [4] and action $\mathbf{a}_k$. In Fig. 14, we illustrates the relationship between de-correlation time and length of historical trajectory (observation horizon). Given the k-th state-action pair, if the observation horizon is longer than the de-correlation time of the system, the $(k+1)$-th state does not depend on the states and actions before the $(t_k - L_{\text{in}} - W_k - W_{k-1})$-th slot. According to the definition of Markov decision process, the Markov property holds in our problem.

[4]This assumption holds in most cases in our system since the prediction horizons of two consecutive segments are highly correlated and do not vary rapidly.

REFERENCES

[1] D. Holloway, "Virtual worlds and health: Healthcare delivery and simulation opportunities," in *Virtual worlds and metaverse platforms: New communication and identity paradigms*. IGI Global, 2012, pp. 251–270.

[2] C. Collins, "Looking to the future: Higher education in the metaverse," *Educause Rev.*, vol. 43, no. 5, pp. 51–63, 2008.

[3] S. Bardzell and K. Shankar, "Video game technologies and virtual design: a study of virtual design teams in a metaverse," in *Proc. Int. Conf. Virtual Real. (ICVR)*, 2007, pp. 607–616.

[4] C. Choi, J. Jun, J. Heo, and K. Kim, "Effects of virtual-avatar motion-synchrony levels on full-body interaction," in *Proc. 34th Annu. ACM Symp. Appl. Comput. (SAC)*, 2019, pp. 701–708.

[5] S. Ellis, F. Breant, B. Manges, R. Jacoby, and B. Adelstein, "Factors influencing operator interaction with virtual objects viewed via head-mounted see-through displays: viewing conditions and rendering latency," in *Proc. IEEE Annu. Int. Symp. Virtual Real. (VRAIS)*, 1997, pp. 138–145.

[6] H. Laaki, Y. Miche, and K. Tammi, "Prototyping a digital twin for real time remote control over mobile networks: Application of remote surgery," *IEEE Access*, vol. 7, pp. 20 325–20 336, 2019.

[7] K. Mania, B. D. Adelstein, S. R. Ellis, and M. I. Hill, "Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity," in *Proc. 1st Symp. Appl. Perception Graph. Visual. (APGV)*, 2004, pp. 39–47.

[8] A. Aijaz, M. Dohler, A. H. Aghvami, V. Friderikos, and M. Frodigh, "Realizing the tactile internet: Haptic communications over next generation 5g cellular networks," *IEEE Trans. Wireless Commun.*, vol. 24, no. 2, pp. 82–89, 2017.

[9] I. Oliver, A. Miller, and C. Allison, "Mongoose: Throughput redistributing virtual world," in *Proc. 21st IEEE Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2012, pp. 1–9.

[10] M. Dianatfar, J. Latokartano, and M. Lanz, "Review on existing vr/ar solutions in human–robot collaboration," *Procedia CIRP*, vol. 97, pp. 407–411, 2021.

[11] L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, and P. Hui, "All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda," 2021. [Online]. Available: https://arxiv.org/abs/2110.05352

[12] "*Study on scenarios and requirements for next generation access technologies*," document 3GPP, TSG RAN TR38.913 R14, Jun. 2017.

[13] S. Wang, M. Chen, W. Saad, C. Yin, S. Cui, and H. V. Poor, "Reinforcement learning for minimizing age of information under realistic physical dynamics," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2020, pp. 1–6.

[14] H. Tang, J. Wang, L. Song, and J. Song, "Minimizing age of information with power constraints: Multi-user opportunistic scheduling in multi-state time-varying channels," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 854–868, 2020.

[15] E. Fountoulakis, M. Codreanu, A. Ephremides, and N. Pappas, "Joint sampling and transmission policies for minimizing cost under aoi constraints," 2021. [Online]. Available: https://arxiv.org/abs/2103.15450

[16] M. Kountouris and N. Pappas, "Semantics-empowered communication for networked intelligent systems," *IEEE Commun. Mag.*, vol. 59, no. 6, pp. 96–102, 2021.

[17] N. Pappas and M. Kountouris, "Goal-oriented communication for real-time tracking in autonomous systems," in *Proc. IEEE Int. Conf. Auton. Syst. (ICAS)*, 2021, pp. 1–5.

[18] Y. Sun and B. Cyr, "Information aging through queues: A mutual information perspective," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2018, pp. 1–5.

[19] X. Hou and S. Dey, "Motion prediction and pre-rendering at the edge to enable ultra-low latency mobile 6dof experiences," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 1674–1690, 2020.

[20] F. Richter, Y. Zhang, Y. Zhi, R. K. Orosco, and M. C. Yip, "Augmented reality predictive displays to help mitigate the effects of delayed telesurgery," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 444–450.

[21] X. Tong, G. Zhao, M. A. Imran, Z. Pang, and Z. Chen, "Minimizing wireless resource consumption for packetized predictive control in real-time cyber physical systems," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2018, pp. 1–6.

[22] Z. Hou, C. She, Y. Li, L. Zhuo, and B. Vucetic, "Prediction and communication co-design for ultra-reliable and low-latency communications," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 1196–1209, 2019.

[23] Z. Hou, C. She, Y. Li, D. Niyato, M. Dohler, and B. Vucetic, "Intelligent communications for tactile internet in 6g: Requirements, technologies, and challenges," *IEEE Commun. Mag.*, vol. 59, no. 12, pp. 82–88, 2021.

[24] J. F. Steffensen, *Interpolation*. New York: Courier Corporation, 2006.

[25] G. Scaglia, A. Rosales, L. Quintero, V. Mut, and R. Agarwal, "A linear-interpolation-based controller design for trajectory tracking of mobile robots," *Control. Eng. Pract.*, vol. 18, no. 3, pp. 318–329, 2010.

[26] D. P. Bertsekas, *Nonlinear programming*. Belmont, MA, USA: Athena Scientific, 1999.

[27] Q. Liang, F. Que, and E. Modiano, "Accelerated primal-dual policy optimization for safe reinforcement learning," 2018. [Online]. Available: https://arxiv.org/abs/1802.06480

[28] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1587–1596.

[29] F. E. GmbH. Requirements on communication interface for panda research robot. https://frankaemika.github.io/docs/requirements.html (accessed: Feb. 10, 2022).

[30] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 4, pp. 559–576, 2005.

[31] J. Craighead, J. Burke, and R. Murphy, "Using the unity game engine to develop sarge: a case study," in *Proc. Simul. Workshop Int. Conf. Intell. Robots Syst. (IROS Workshops)*, vol. 4552, 2008.

[32] I. D. O. NaturalPoint, "Optitrack prime 13," https://www.optitrack.com/cameras/primex-13.html, (accessed: Feb. 10, 2022).