Task-Oriented Cross-System Design for Timely and Accurate Modeling in the Metaverse

Zhen Meng, *Student Member*, *IEEE*, Kan Chen, *Student Member*, *IEEE*, Yufeng Diao, *Student Member*, *IEEE*, Changyang She, *Senior Member*, *IEEE*, Guodong Zhao, *Senior Member*, *IEEE*, Muhammad Ali Imran, *Fellow*, *IEEE*, and Branka Vucetic, *Life Fellow*, *IEEE*

Abstract

In this paper, we establish a task-oriented cross-system design framework to minimize the required packet rate for timely and accurate modeling of a real-world robotic arm in the Metaverse, where sensing, communication, prediction, control, and rendering are considered. To optimize a scheduling policy and prediction horizons, we design a Constraint Proximal Policy Optimization (C-PPO) algorithm by integrating domain knowledge from relevant systems into the advanced reinforcement learning algorithm, Proximal Policy Optimization (PPO). Specifically, the Jacobian matrix for analyzing the motion of the robotic arm is included in the state of the C-PPO algorithm, and the Conditional Value-at-Risk (CVaR) of the state-value function characterizing the long-term modeling error is adopted in the constraint. Besides, the policy is represented by a two-branch neural network determining the scheduling policy and the prediction horizons, respectively. To evaluate our algorithm, we build a prototype including a real-world robotic arm and its digital model in the Metaverse. The experimental results indicate that domain knowledge helps to reduce the convergence time and the required packet rate by up to 50%, and the cross-system design framework outperforms a baseline framework in terms of the required packet rate and the tail distribution of the modeling error.

Index Terms

Z. Meng, K. Chen, Y. Diao, G. Zhao, and M. A. Imran are with the James Watt School of Engineering, University of Glasgow, UK. (e-mail: z.meng.1@research.gla.ac.uk; k.chen.1@research.gla.ac.uk; y.diao.1@research.gla.ac.uk; guodong.zhao@glasgow.ac.uk; muhammad.imran@glasgow.ac.uk)

C. She and B. Vucetic are with the School of Electrical and Information Engineering, University of Sydney, Australia. (e-mail: shechangyang@gmail.com; branka.vucetic@sydney.edu.au)

Corresponding author: Changyang She.

Task-oriented cross-system design, scheduling, prediction, constraint deep reinforcement learning, Metaverse.

I. Introduction

As a digital world that mirrors the physical world and generates feedback for human users in real-time, the Metaverse can blur the lines between the physical and digital worlds and revolutionize how humans communicate and interact with each other [1]. To achieve this goal, the timely and accurate modeling of real-world devices/humans in the Metaverse is critical for user experience. The communication and computing latency and digital model distortion will lead to chaotic interactions and user dizziness [2]. For some mission-critical applications assisted by the Metaverse, such as remote healthcare and factory automation, slight out-of-synchronization between a real-world device and its digital model may cause serious consequences [3].

Efficiently replicating the real world in the Metaverse brings significant challenges in communications. While 5G networks have significantly improved latency, reliability, data rate, and connection density, it still falls short of satisfying the demands of the Metaverse [4]. One of the examples is that 5G New Radio can achieve 1 ms latency and 10^{-5} packet error probability in the radio access networks (RANs), but does not guarantee end-to-end delay and reliability. Jitter is another issue that can lead to inaccurate modeling, which may be caused by 5G RANs and 5G core networks. Furthermore, there is a mismatch between communication Key Performance Indicators (KPIs), i.e., latency, reliability, jitter, and throughput, and the KPI requirements of diverse tasks in the Metaverse, such as modeling error, haptic feedback distortion, and semantic segmentation errors, which will lead to poor user experience and low resource utilization efficiency.

On the other hand, the performance of modeling in the Metaverse is not solely determined by communication networks. Other systems, including sensing, prediction, control, and rendering, can also have significant impacts on End-to-End (E2E) latency and accuracy. Designing these systems separately results in strictly sub-optimal solutions and may fail to meet the task-oriented KPI requirements. Thus, cross-system design has been investigated in the recent literature, such as prediction and communication co-design [5], [6] and sampling and communication co-design [7], [8]. They have shown significant gains of the cross-system design, but they have also revealed potential issues. For example, cross-system models could be analytically intractable, and the

complexity of cross-system problems can be much higher than problems in separate design approaches. Thus, novel methodologies are needed for cross-system design.

The first step toward cross-system design is to formulate a problem that takes the relevant systems into account. Nevertheless, it could be difficult to obtain closed-form expressions of the objective function and the constraints as some of the KPIs are analytically intractable. Although we can use some approximations to formulate the problem, it is generally non-convex or NP-hard. This motivates us to develop data-driven deep learning approaches, where the policy to be optimized is represented by a neural network. Deep Reinforcement Learning (DRL) is a promising method for training the neural network. For example, Proximal Policy Optimization (PPO) is developed to optimize policies with a discrete action space [9]. More recently, the primal-dual method and Constraint-Rectified Policy Optimization (CRPO) were introduced into DRL for solving constrained problems [10]–[12]. It is worth noting that a straightforward implementation of DRL algorithms may not work [13]. Integrating domain knowledge from relevant systems into DRL algorithms is essential for the success of DRL [14] in practical applications.

A. Related Work

1) Gaps between 5G/6G and Metaverse: The concept of the Metaverse was initially introduced in Neil's book, *Snow Crash* [15], coinciding with the development of virtual physical fusion technology. Notable pioneering contributions have been made in various applications of the Metaverse, including the game networking [16], autonomous vehicles [17], Internet of Things (IoT) devices [18], and education [19]. The above work has yielded valuable insights that could be leveraged to advance the development of the Metaverse, particularly with regard to the conception and refinement of forthcoming communication systems and network architectures. However, there still exists a research gap between 5G/6G and the Metaverse: 1) The KPIs in 5G such as latency, reliability, and throughput cannot fulfill the requirements of diverse tasks in the Metaverse [1]. This misalignment leads to significant challenges in future communication system design for the Metaverse. For example, there is no closed-form relation between the KPIs in 5G and the requirement of tasks in the Metaverse. As a result, it is difficult to guarantee End-to-End (E2E) performance requirement [20]. 2) Future 6G networks should support a large number of devices with diverse KPIs in the Metaverse by leveraging distributed computing, storage, and communication resources in local devices, edge servers, and central servers [21]. A hierarchical

communication network architecture with a strong cloud server and multiple edge servers is promising, where edge servers generate quick responses to mobile devices and update critical messages to the cloud. 3) Existing HCI, sensing, communication, and computing systems are developed separately [22], which leads to sub-optimal solutions, brings extra communication overhead for coordinating multiple tasks and cannot meet task-oriented KPIs [23].

2) Timely, Efficiently and Accurate Modeling for Digital Twin and Metaverse: In 2012, NASA clarified the concept of Digital Twins and defined it as "integrated multiphysical, multiscale, probabilistic simulations of an as-built vehicle or system using the best available physical models, sensor updates, and historical data" [24]. However, the communication, computing, and storage resources in the existing IT and networking infrastructures are insufficient to support of diverse applications associated with digital twins and the Metaverse. Some existing studies focused on the timely, efficient, and accurate modeling of digital twins in the Metaverse. [18], [25]–[29]. In [28], a deep reinforcement learning approach was developed to solve the placement and migration problems of digital twins to minimize the synchronization delay with the help of edge computing. The authors of [18] proposed a resource allocation algorithm for synchronizing Internet of Things devices with their digital models in the Metaverse by using a game-theoretic framework. In [29], the authors proposed an edge continual learning framework that can accurately synchronize a physical object with evolving affinity with its digital twin. Instead of using a centralized framework, a distributed Metaverse framework was developed in [18] for synchronizing multiple digital twins, known as sub-Metaverses. The above work revealed some fundamental tradeoffs between resource utilization efficiency and KPIs in the Metaverse. Nevertheless, coordinating communications, computing, and storage sources for diverse Metaverse applications remains an open problem due to its high complexity, especially when the scale of the system is large.

3) Task-Oriented Cross-System Design For Metaverse: While 5G can support enhanced broadband services with high data rates, ultra-reliable and low-latency communications, and massive machine-type communications [4], it still falls short of satisfying task-oriented KPIs in the Metaverse, such as modeling error, haptic feedback distortion, and semantic segmentation error [30]. To fill the gap between the communication KPIs and the KPI requirements of specific tasks, task-oriented cross-system design is a promising approach. The authors of [31] considered an Age-of-Loop metric for the remote control of autonomous guided vehicles, and proposed a goal-oriented wireless solution that adjusts the data rate to achieve high control accuracy. Their results showed that with the goal-oriented KPI, it is possible to achieve higher accuracy

than the commonly used communication KPIs, such as Age-of-Information. In [32], the authors proposed a learning-based communication scheme that optimizes feature extraction, source coding, and channel coding in a task-oriented manner to achieve low-latency inference for image classification. The experimental results of this work indicated that task-oriented communication achieves a better rate-distortion tradeoff than baseline methods. More recently, the authors of [33] developed E2E task-oriented resource management by integrating sensing, computing, and communication processes into a joint design framework, where the artificial intelligence model is split and executed on edge servers for low-latency intelligent services. To improve the user experience in immersive Metaverse applications, the authors of [34] proposed a user-centered joint optimization approach to optimize frame generation location, transmission power, and channel access arrangement. These studies indicated that by task-oriented cross-system design, it is possible to provide a better user experience and achieve higher resource utilization efficiency. However, Metaverse is expected to handle large amounts of heterogeneous data from various sources and formats. How to develop task-oriented data representation and ontology models that enable efficient data exchange, integration, and interpretation within the Metaverse context still needs further investigation [35]. Furthermore, the cross-system problems are non-convex or NP-hard in general. Finding a near-optimal resource management solution with low complexity remains a challenging issue.

B. Contributions

In this paper, we aim to address the following fundamental issues: 1) How to eliminate the gap between traditional communication KPIs defined in the 5G standard and user-centered task-oriented KPIs in the Metaverse? 2) How to implement cross-system design in the Metaverse, including sampling, communication, prediction, control, and rendering? 3) How to utilize cross-system domain knowledge to guide the E2E training of DRL algorithms? The main contributions of this paper are summarized as follows:

- We establish a task-oriented cross-system design framework, where sensing, communication, prediction, control, and rendering are jointly considered for modeling a robotic arm in the Metaverse. The scheduling policy and the prediction horizon are jointly optimized to minimize the required packet rate to guarantee a modeling error constraint.
- We propose a Constraint Proximal Policy Optimization (C-PPO) algorithm by integrating domain knowledge into the advanced PPO algorithm, and further train the policy using the

C-PPO algorithm. Specifically, 1) the Jacobian matrix, which is widely used for analyzing the motion of robotic arms, is included in the state of DRL to improve the training efficiency of C-PPO. 2) The Conditional Value-at-Risk (CVaR) of the state-value function that characterizes the long-term modeling error is applied to formulate the constraint. 3) A two-branch neural network is developed to determine the scheduling policy and the prediction horizons.

- We build a prototype system including a real-world robotic arm and its digital model in the Metaverse, where the Nvidia Issac Gym platform is used. Extensive experiments are carried out in the prototype to evaluate the proposed task-oriented cross-system design approach. The experimental results show that our C-PPO outperforms several benchmark algorithms in terms of convergence time, stability, packet rate, and modeling error, and the cross-system design framework outperforms a baseline framework in terms of the required packet rate and the tail distribution of tracking errors.

The rest of this paper is organized as follows. In Section II, we propose the task-oriented cross-system design framework where all subsystems, i.e., sensing, communication, reconstruction, prediction, control, and rendering, are elaborated in detail. In Section III, we develop the C-PPO algorithm to optimize the scheduling and prediction policy while minimizing the communication load under the constraint of CVaR. Section IV describes the prototype and provides performance evaluations. Finally, Section V concludes this paper.

II. Task-Oriented Cross-System Design

In this section, we propose a task-oriented cross-system design framework for timely and accurate modeling in the Metaverse. The specific goal is to build a digital model of a real-world robotic arm for real-time monitoring and control.

A. Framework

The framework is shown in Fig. 1, where the real-world robotic arm with multiple joints is controlled by a user (a student or a trainee) for some tasks for example training in healthcare. The trajectory (the angles of all the joints) of the real-world robotic arm is measured by the built-in sensors. Then, the trajectory is sampled and transmitted to the Metaverse in a cloud server, where the sampled data are used to reconstruct the historical trajectory and predict the future trajectory. Here, the digital model in the Metaverse is controlled by the predicted trajectory

TABLE I

SUMMARY OF MAIN SYMBOLS

Symbol	Explanation	Symbol	Explanation
t_s	Duration of time slot	$\mathcal{P}(t)$	Pose of real-world robotic arm in the t-th time slot
I	The number of joints	$\boldsymbol{\eta}_c(t)$	Unit vector of rotation axis
$\mathcal{T}(t)$	Trajectory of the real-world robotic arm in the t-th time slot	$\psi_\eta(t)$	Angle of rotation
$\tau_i(t)$	Angle value of the i-th joint measured in the t-th time slot	$F_f(\cdot)$	Forward kinematics of real-world robotic
$x_i(t)$	Whether the i-th joint is scheduled to transmit in the t-th time slot	$\check{P}(t)$	Pose of virtual-world robotic arm in the t-th time slot
$X(t)$	Decision of the scheduler in the t-th time slot	$\mathbf{e}(t)$	Modeling error in the t-th time slot
$y_i(t)$	Indicators of whether packet arrivals in the t-th time slot	ω_1, ω_2	Weighting coefficients of the modeling error
$\dot{\mathbb{S}}(t)$	Set of angles received by the Metaverse	π_θ	Scheduling and prediction policy
W_l	Historical observation window for reconstruction	θ	Parameters of policy
$\bar{\mathcal{T}}(t_0)$	Reconstructed trajectory in the t_0 time slot	$\mathbf{a}_t$	Action taken in the t-th time slot
$F_l(\cdot,\theta_l)$	Function of reconstruction	$\mathbf{s}_t$	State
θ_l	Parameters of reconstruction	$a_{t,i}{}^{[1]}, a_{t,i}{}^{[2]}$	Action of the i-th joint taken in the t-th time slot
W_p	Input length of the prediction	$\rho_{t,i}{}^{[1]}, \rho_{t,i}{}^{[2]}$	Distribution of action in the t-th time slot
H	Output length of the prediction	$\mathcal{L}_c$, $\mathcal{L}_r$	Loss functions of policy network
$F_p(\cdot,\theta_p)$	Function of prediction	$A(\mathbf{s},\mathbf{a})$	Advantage function
θ_p	Parameters of the prediction function	$Q^{\pi_\theta}(\mathbf{s},\mathbf{a})$	State-action value under policy
L_p	MSE loss of prediction	$V^{\pi_\theta}(\mathbf{s})$	State value function
$\hat{\mathcal{T}}(t)$	Prediction trajectory in the t-th time slot	$\mathcal{J}(\cdot)$	Jacobian matrix
$\hat{\tau}_i(t)$	Prediction trajectory of the i-th joint in the t-th time slot	L_1, L_2, L_3	Length of joint links
$Z(t)$	Optimal prediction horizon	$\phi(t)$	Angle between x-axis and x'-axis
$z_i(t)$	Optimal prediction horizon of the i-th joint	$r(\mathbf{s}_t,\mathbf{a_t})$	Instantaneous reward
N_c	Time interval of control command generation	$c(\mathbf{s}_t,\mathbf{a_t})$	Instantaneous cost
N_r	Processing time of each image	R^{π_θ}	Long-term reward
k_p	Proportional parameter of PD controller	C^{π_θ}	Long-term cost
k_d	Derivative parameter of PD controller	γ	Discount factor
$\widetilde{\mathcal{T}}(t)$	Control results in the t-th time slot	Γ_c	Constraint of modeling error
$\widetilde{\tau}_i(t)$	Control result of the i-th joint in the t-th time slot	$\mathrm{CVaR}_\alpha(\cdot)$	CVaR function
$F_r(\cdot,\theta_r)$	Function of rendering	1-α	Confidence level of CVaR
θ_r	Parameters of rendering function	v	Multiplier of CVaR
$\check{\mathcal{T}}(k)$	The rendered trajectory	β	Learning rate of CVaR

rather than the reconstructed trajectory to compensate for delays caused by different components across systems. Finally, the digital model in the Metaverse is rendered and presented to another user (an expert or a trainer) via a computer screen or VR/AR headset. It is worth noting that each joint has its own state, and the states of the joints are interdependent. They need to collaborate with each other to accomplish the task. In addition, the total communication resources shared by all the joints are limited. Thus, it is possible to extend our system into multi-sensor scenarios.

The E2E Motion-To-Photon (MTP) latency is defined by the delay between the movement

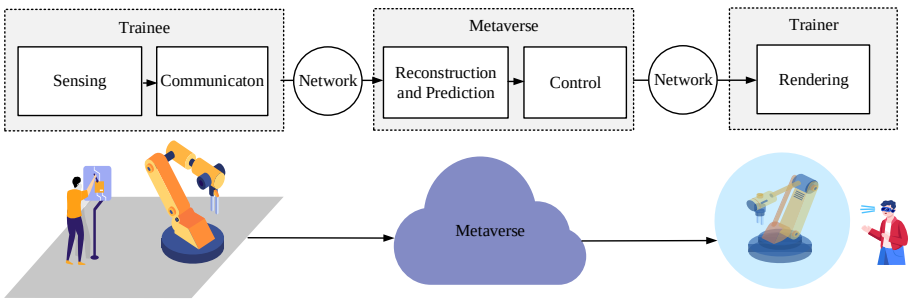


Fig. 1. Proposed task-oriented cross-system design framework for modeling a robotic arm in the Metaverse, where sensing, communication, reconstruction, predication, control and rendering are considered.

of the real-world robotic arm and the movement of its digital model in the Metaverse. Thus, it includes communication delay, computation delay, control delay, and rendering delay. By optimizing the prediction horizon and the scheduling policy[1], we minimize the communication overhead subject to constraints on the modeling accuracy and the MTP latency.

Fig. 2 illustrate the timing sequence of the proposed framework. The data is generated from the built-in sensors at the physical robotic arm. Then, the communication module conducts scheduling and sends the selected data to a computer server via a network. The server conducts data reconstruction and prediction, and then controls the digital model of robotic arm. Finally, the digital model was rendered[2] and presented to a human (trainer) via a VR headset (or a screen). In the following, we will introduce each component:

1) Sensing and Communications: Time is discretized into slots with a duration of t_s. The built-in sensors measure I joint angles in each time slot. Let $\mathcal{T}(t) = [\tau_1(t), ..., \tau_I(t)]$ be the trajectory of the real-world robotic arm, where $\tau_i(t)$, $i = 1, 2, 3, ..., I$, is the angle value of the i-th joint measured in the t-th time slot. We consider a scheduling policy in the communication system, where the indicator, $x_i(t)$, represents whether the i-th joint is scheduled for data transmission in the t-th time slot, $i = 1, ..., I$. If the i-th joint is not scheduled, then $x_i(t) = 0$. Otherwise, $x_i(t) = 1$, and one packet will be transmitted to the Metaverse. The decision of the scheduler in the t-th time slot is denoted by $X(t) = [x_1(t), x_2(t), ..., x_I(t)]$. The total number of packets

[1]The scheduling policy determines which joints will be scheduled for data transmission.

[2]To simplify the system, we assume that the rendering takes place at the server and the human user (trainer) directly interacts with the digital model via human-computer interface.

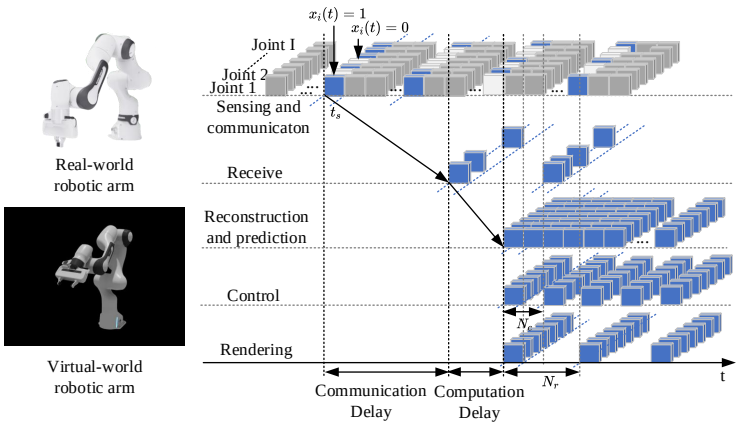


Fig. 2. The timing sequence of the proposed framework, where the modeling accuracy and the MTP latency need to be satisfied.

to be transmitted in the communication systems in the t-th time slot is given by $\sum_{i=1}^{I} x_i(t)$.

2) Reconstruction: To reconstruct the trajectory from sampled data, we use the linear interpolation and extrapolation method, which is widely used in the existing literature and can be easily implemented in our system [36]. The indicators of packet arrivals in the t-th time slot at the Metaverse are denoted by $y_i(t)$, $i = 1, 2, ..., I$. If a packet from the i-th joint arrived at the Metaverse in the t-th time slot, then $y_i(t) = 1$. Otherwise, $y_i(t) = 0$. From the arrived packets, the set of joint angles obtained by the Metaverse in the t-th time slot is denoted by $\mathbb{S}(t) = \{\tau_i(t) \mid y_i(t) = 1,\ i = 1, ..., I\}$. In a certain time slot t_0, the cloud server reconstructs the trajectory of the robotic arm from the received joint angles in a historical observation window with a duration of W_l time slots. The reconstruction algorithm is given by

$$\bar{\mathcal{T}}(t_0) = F_l(\mathbb{S}(t), \theta_l \mid t \in [t_0 - W_l, t_0 - W_l + 1, ..., t_0 - 1]), \tag{1}$$

where $\bar{\mathcal{T}}(t_0) \in \mathbb{R}^{1\times I}$ is the reconstructed trajectory, $F_l(\cdot, \theta_l)$ is the reconstruction function, and θ_l is the interpolation and extrapolation parameters.

3) Prediction: To compensate for the MTP latency, we propose to use the attention-mechanism-based predictor, referred to as *Informer*, to predict the future trajectory from the historical trajectory [37]. The lengths of the input and output trajectories are denoted by W_p and H. The values of W_p and H are determined by the auto-correlation coefficient of the trajectories [38]. We denote the prediction results for trajectory in the t-th time slot by $\hat{\mathcal{T}}(t) = [\hat{\tau}_1(t), \hat{\tau}_2(t), ..., \hat{\tau}_I(t)]$.

In a certain time slot t_1, the prediction algorithm can be expressed as follows,

$$[\hat{\mathcal{T}}(t_1+1), \hat{\mathcal{T}}(t_1+2), ..., \hat{\mathcal{T}}(t_1+H)] = F_p([\bar{\mathcal{T}}(t_1-W_p), \bar{\mathcal{T}}(t_1-W_p+1), ..., \bar{\mathcal{T}}(t_1)], \theta_p), \quad (2)$$

where $F_p(\cdot, \theta_p)$ is the prediction function and θ_p represents the parameters of this function. The loss function of the prediction algorithm is Mean Squared Error (MSE) between the output trajectory and the ground truth, which is given by

$$L_{\mathrm{p}} = \frac{1}{H} \sum_{n=1}^{H} \left(\hat{\mathcal{T}}(t_1+n) - \mathcal{T}(t_1+n) \right)^2. \quad (3)$$

We will optimize the prediction length $Z(t) = [z_1(t), z_2(t), ..., z_I(t)], z_i(t) \leq H$ for each joint in our cross-system design framework, which will be introduced in the next section.

4) Control: There are different algorithms we can use to control the virtual robotic arm in the Metaverse [39]. Without loss of generality, we utilize the joint-space position control and proportional–derivative (PD) controller [40]. Considering the limitations of the control frequency and the processing time, the target angle for each joint will be generated by the control algorithm and subsequently executed for every N_{c} time slots, which is denoted by $\widetilde{\mathcal{T}}(t) = [\widetilde{\tau}_1(t), ..., \widetilde{\tau}_I(t)]$. In the t_2-th time slot, for each joint i, the target joint position $\widetilde{\tau}_i(t_2)$ to be executed within the N_{c} time slots can be expressed as

$$\widetilde{\tau}_i(t_2) = k_p \cdot (\hat{\tau}_i(t_1 + z_i(t)) - \widetilde{\tau}_i(t_2 - N_c)) + k_d \cdot (\frac{\mathrm{d}\hat{\tau}_i(t_1 + z_i(t))}{\mathrm{d}t} - \frac{\mathrm{d}\widetilde{\tau}_i(t_2 - N_c)}{\mathrm{d}t}), \quad (4)$$

where k_p and k_d are the proportional and derivative parameters of the PD controller, respectively.

5) Rendering: In computer graphics, rendering refers to the process of generating controllable and photo-realistic images and videos of virtual scenes [41]. In our system, the processing time of each image is denoted by N_r time slots. In other words, the monitor or VR/AR glasses refresh the images at a refresh rate of $1/(N_r t_{\mathrm{s}})$ (times/second). The relationship between the trajectory of the digital model and the trajectory displayed to the user is given by

$$\check{\mathcal{T}}(t) = F_r(\widetilde{\mathcal{T}}(t), \theta_r), \quad (5)$$

where $F_r(\cdot, \theta_r)$ is the rendering function and θ_r represents the parameters for rendering.

B. KPIs and Communication Load

1) Task-Oriented KPI: The end effector of a robotic arm could be a gripper, a drill bit, or a sensor, depending on the specific task. We assume that the real-world end effector has seven degrees of freedom, and the pose of the end effector is denoted by

$$\mathcal{P}(t) = [l_{\mathrm{x,r}}(t), l_{\mathrm{y,r}}(t), l_{\mathrm{z,r}}(t), q_{\mathrm{x,r}}(t), q_{\mathrm{y,r}}(t), q_{\mathrm{z,r}}(t), q_{\mathrm{w,r}}(t)]. \quad (6)$$

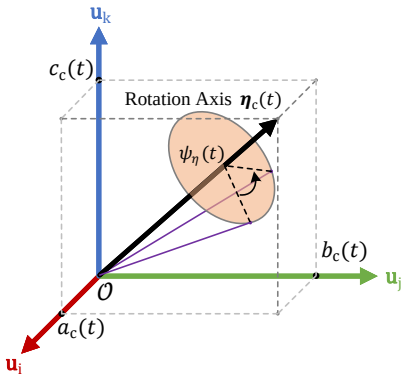


Fig. 3. Orientation of the end effector to the base coordinate system $\mathcal{O}$.

Specifically, $l_{\mathrm{x,r}}(t)$, $l_{\mathrm{y,r}}(t)$, $l_{\mathrm{z,r}}(t)$ are the coordinates of the end effector in a three-dimensional Cartesian coordinate system. $[q_{\mathrm{x,r}}(t),\ q_{\mathrm{y,r}}(t),\ q_{\mathrm{z,r}}(t),\ q_{\mathrm{w,r}}(t)]$ is the quaternion representing the orientation of the end effector [42]. Quaternions are preferred over other representations, such as Euler angles or rotation matrices, in our context because of their compact representation and their ability to avoid a particular limitation associated with 3D rotation systems, known as gimbal lock, which can cause a loss of degrees of freedom [43]. Please see Appendix B for more information. As shown in Fig. 3, the unit vector of the rotation axis, $\boldsymbol{\eta}_c(t) = [a_c(t), b_c(t), c_c(t)]$, and angle of rotation, $\psi_\eta(t)$, can be characterized by $q_{\mathrm{x,r}}(t)$, $q_{\mathrm{y,r}}(t)$, $q_{\mathrm{z,r}}(t)$, and $q_{\mathrm{w,r}}(t)$. In particular, the rotation axis is located in the coordinate system defined by three imaginary unit basic vectors, $\mathbf{u}_\mathrm{i}$, $\mathbf{u}_\mathrm{j}$, and $\mathbf{u}_\mathrm{k}$, which follow special multiplication rules [42]. The relationship among the quaternions, $\boldsymbol{\eta}_c(t)$, and $\psi_\eta(t)$ is expressed by

$$
\begin{aligned}
q_{\mathrm{x,r}}(t) &= \sin(\frac{\psi_\eta(t)}{2}) \cdot a_c(t), \\
q_{\mathrm{y,r}}(t) &= \sin(\frac{\psi_\eta(t)}{2}) \cdot b_c(t), \\
q_{\mathrm{z,r}}(t) &= \sin(\frac{\psi_\eta(t)}{2}) \cdot c_c(t), \\
q_{\mathrm{w,r}}(t) &= \cos(\frac{\psi_\eta(t)}{2}).
\end{aligned} \tag{7}
$$

Similarly, the quaternions of the virtual-world robotic arm follow the same rules.

From the I joint angles, $\mathcal{P}(t)$ is obtained from the forward kinematics according to

$$
\mathcal{P}(t) = F_f(\mathcal{T}(t)), \tag{8}
$$

where $F_f(\cdot)$ maps the joint angles to the pose of the end effector (positions and orientations). The expression of (8) depends on the structure and configuration of the robotic arm. Like $\mathcal{P}(t)$, the pose of the end effector displayed to the user also has seven degrees of freedom, denoted by $\check{P}(t) = [l_{\mathrm{x,v}}(t), l_{\mathrm{y,v}}(t), l_{\mathrm{z,v}}(t), q_{\mathrm{x,v}}(t), q_{\mathrm{y,v}}(t), q_{\mathrm{z,v}}(t), q_{\mathrm{w,v}}(t)]$. A task-oriented KPI is defined as the Euclidean distance between $\mathcal{P}(t)$ and $\check{P}(t)$,

$$\begin{aligned}\mathbf{e}(t) = \omega_1 \cdot \|(l_{\mathrm{x,r}}(t), l_{\mathrm{y,r}}(t), l_{\mathrm{z,r}}(t)), (l_{\mathrm{x,v}}(t), l_{\mathrm{y,v}}(t), l_{\mathrm{z,v}}(t))\|_2 \\ + \omega_2 \cdot \|(q_{\mathrm{x,r}}(t), q_{\mathrm{y,r}}(t), q_{\mathrm{z,r}}(t), q_{\mathrm{w,r}}(t)), (q_{\mathrm{x,v}}(t), q_{\mathrm{y,v}}(t), q_{\mathrm{z,v}}(t), q_{\mathrm{w,v}}(t))\|_2,\end{aligned} \tag{9}$$

where $\|\cdot\|_2$ is the L_2-norm defined as $\|\cdot\|_2 \triangleq \sqrt{\sum(\cdot)^2}$, and ω_1 and ω_2 are the weighting coefficients. The first term is the position error and the second is the orientation error. Depending on the accuracy requirements of different tasks in the Metaverse, ω_1 and ω_2 can be set to different values.

2) Communication Load: The Orthogonal Frequency-Division Multiplexing (OFDM) communication system is considered in our framework for it is widely deployed in cellular networks. The number of time and frequency resource blocks allocated to a packet is determined by the channel gain and the packet size. We assume that the channel gain and the packet size are two stationary random variables. The average number of resource blocks required in each slot is proportional to the average packet rate. To improve resource utilization efficiency, defined as the average number of resource blocks per slot, we minimize the average number of packets transmitted in each slot.

III. Constrained Reinforcement Learning for Cross system Design

In this section, we formulate an optimization problem that minimizes the communication load subject to a constraint on the CVaR of modeling error by optimizing the scheduling policy and the prediction horizon. In the cross-system design framework, there is no closed-form relationship between the KPIs and the optimization variables. To solve this problem, we develop a DRL algorithm by integrating domain knowledge into the advanced PPO algorithm.

A. Preliminary of PPO

PPO is an advanced reinforcement learning algorithm for solving problems with discrete action space. We chose Proximal Policy Optimization (PPO) as the baseline algorithm due to its simplicity, effectiveness, and high sample efficiency compared to other reinforcement learning

algorithms [44]. In addition, PPO maintains a balance between exploration and exploitation and avoids drastic policy updates, which is crucial for managing the complex dynamics of robotics and ensuring stable training [45]. We denote the state and action of PPO by $\mathbf{s}_t$ and $\mathbf{a}_t$, respectively. The policy is a mapping from the state to the probabilities of taking different actions, denoted by $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$, where θ are the training parameters of the policy network. With PPO, the parameters of the policy are updated according to the following expression,

$$\theta_{t+1} = \arg\max_{\theta} \mathop{\mathbb{E}}_{\mathbf{s}_t, \mathbf{a}_t \sim \pi_\theta} \mathcal{L}(\mathbf{s}_t, \mathbf{a}_t, \theta_t, \theta). \tag{10}$$

The loss function $\mathcal{L}(\mathbf{s}_t, \mathbf{a}_t, \theta_t, \theta)$ is given by

$$\begin{aligned} \mathcal{L}(\mathbf{s}_t, \mathbf{a}_t, \theta_t, \theta) = \min \Bigg(& \frac{\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)}{\pi_{\theta_t}(\mathbf{a}_t \mid \mathbf{s}_t)} A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t), \\ & \text{clip}\left(\frac{\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)}{\pi_{\theta_t}(\mathbf{a}_t \mid \mathbf{s}_t)}, 1-\epsilon, 1+\epsilon, \right) A^{\pi_{\theta_t}}(\mathbf{s}_t, \mathbf{a}_t) \Bigg), \end{aligned} \tag{11}$$

where $A(\mathbf{s}_t, \mathbf{a}_t)$ is the advantage function defined as the difference between the state-action value function, $Q^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)$, and the state value function, $V^{\pi_\theta}(\mathbf{s}_t)$,

$$A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi_\theta}(\mathbf{s}_t), \tag{12}$$

which estimates the advantage of taking action $\mathbf{a}_t$ in state $\mathbf{s}_t$, over other possible actions in the same state [46]. In the sequel, we develop our DRL algorithm by integrating domain knowledge into the PPO.

B. Knowledge-Assisted Problem Formulation

1) State: The state in the t-th time slot consists of two parts: the angles of the I joints and the Jacobian matrix of the real-world robotic arm, i.e., $\mathbf{s}_t = \{\mathcal{T}(t), \mathcal{J}(\mathcal{T}(t))\}$. In robotics, the Jacobian matrix is critical for analyzing and controlling the motion of robots. It characterizes the relationship between the velocity of the end effector and the velocities of all joints [47],

$$\frac{\Delta \mathcal{P}(t)}{\Delta t} = \mathcal{J}(\mathcal{T}(t)) \frac{\Delta \mathcal{T}(t)}{\Delta t}, \tag{13}$$

where $\frac{\Delta \mathcal{P}(t)}{\Delta t}$ is the velocity of the end effector, and $\frac{\Delta \mathcal{T}(t)}{\Delta t}$ is the angular velocities of I joints. In the t-th time slot, the Jacobian matrix can be obtained from $\mathcal{T}(t)$ and the kinematic properties of the robotic arm, e.g., Denavit–Hartenberg (D-H) parameters [47]. By multiplying Δt on both sides of (13), the relationship between $\Delta \mathcal{P}(t)$ and $\Delta \mathcal{T}(t)$ is expressed as follows,

$$\Delta \mathcal{P}(t) = \mathcal{J}(\mathcal{T}(t)) \Delta \mathcal{T}(t), \tag{14}$$

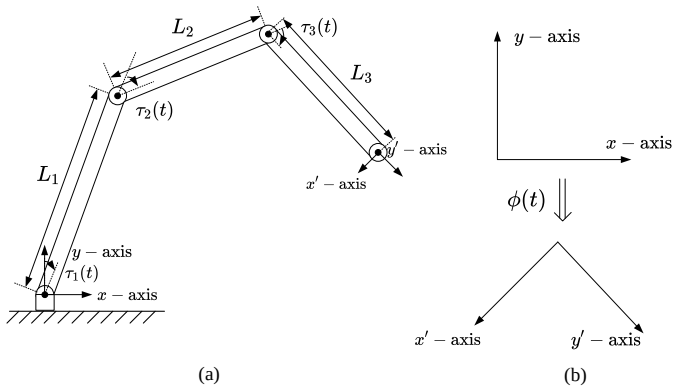


Fig. 4. Three-link two-dimensional robotic arm model.

where $\mathcal{J}(\mathcal{T}(t))$ shows how sensitive the modeling error of the end effector is to the errors of the I joints. Thus, we take the Jacobian matrix as one part of the state to improve the training efficiency of the DRL algorithm.

Let's take the three-link two-dimensional robotic arm as an example to show the Jacobian matrix [39]. With the example in Fig. 4(a), the forward kinematics in (8) can be expressed as follows,

$$\mathcal{P}(t) = \begin{bmatrix} l_{\mathrm{x,r}}(t) \\ l_{\mathrm{y,r}}(t) \\ \phi(t) \end{bmatrix} = \begin{bmatrix} L_1 \cdot \cos\tau_1(t) + L_2 \cdot \cos\left(\tau_1(t)+\tau_2(t)\right) + L_3 \cdot \cos\left(\tau_1(t)+\tau_2(t)+\tau_3(t)\right) \\ L_1 \cdot \sin\tau_1(t) + L_2 \cdot \sin\left(\tau_1(t)+\tau_2(t)\right) + L_3 \cdot \sin\left(\tau_1(t)+\tau_2(t)+\tau_3(t)\right) \\ \tau_1(t)+\tau_2(t)+\tau_3(t) \end{bmatrix}, \tag{15}$$

where L_1, L_2, and L_3 are the lengths of the three links, respectively. As shown in Fig. 4(b), $\phi(t)$ is the angle between x-axis and x'-axis in the clockwise direction. Then, the Jacobian matrix can be obtained by

$$\mathcal{J}(\mathcal{T}(t)) = \begin{bmatrix} \frac{\partial l_{\mathrm{x,r}}(t)}{\partial \tau_1(t)} & \frac{\partial l_{\mathrm{x,r}}(t)}{\partial \tau_2(t)} & \frac{\partial l_{\mathrm{x,r}}(t)}{\partial \tau_3(t)} \\ \frac{\partial l_{\mathrm{y,r}}(t)}{\partial \tau_1(t)} & \frac{\partial l_{\mathrm{y,r}}(t)}{\partial \tau_2(t)} & \frac{\partial l_{\mathrm{y,r}}(t)}{\partial \tau_3(t)} \\ \frac{\partial \phi}{\partial \tau_1(t)} & \frac{\partial \phi}{\partial \tau_2(t)} & \frac{\partial \phi}{\partial \tau_3(t)} \end{bmatrix} \tag{16}$$

where $\mathcal{J}(\mathcal{T}(t))$ consists of all partial derivatives of $\mathcal{P}(t)$. Specifically, the first two rows of the matrix are related to the partial derivatives of the position coordinates, while the last row is related to the partial derivatives of the angle of the end effector which is shown in Fig. 4(b). Thus, each element is calculated by

$$\frac{\partial l_{\mathrm{x,r}}(t)}{\partial \tau_1(t)} = -L_1 \cdot \sin \tau_1(t) - L_2 \cdot \sin\left(\tau_1(t) + \tau_2(t)\right) - L_3 \cdot \sin\left(\tau_1(t) + \tau_2(t) + \tau_3(t)\right), \tag{17}$$

$$\frac{\partial l_{\mathrm{x,r}}(t)}{\partial \tau_2(t)} = -L_2 \cdot \sin\left(\tau_1(t) + \tau_2(t)\right) - L_3 \cdot \sin\left(\tau_1(t) + \tau_2(t) + \tau_3(t)\right), \tag{18}$$

$$\frac{\partial l_{\mathrm{x,r}}(t)}{\partial \tau_3(t)} = -L_3 \cdot \sin\left(\tau_1(t) + \tau_2(t) + \tau_3(t)\right), \tag{19}$$

$$\frac{\partial l_{\mathrm{y,r}}(t)}{\partial \tau_1(t)} = -L_1 \cdot \sin \tau_1(t) - L_2 \cdot \sin\left(\tau_1(t) + \tau_2(t)\right) - L_3 \cdot \sin\left(\tau_1(t) + \tau_2(t) + \tau_3(t)\right), \tag{20}$$

$$\frac{\partial l_{\mathrm{y,r}}(t)}{\partial \tau_2(t)} = -L_2 \cdot \sin\left(\tau_1(t) + \tau_2(t)\right) - L_3 \cdot \sin\left(\tau_1(t) + \tau_2(t) + \tau_3(t)\right), \tag{21}$$

$$\frac{\partial l_{\mathrm{y,r}}(t)}{\partial \tau_3(t)} = -L_3 \cdot \sin\left(\tau_1(t) + \tau_2(t) + \tau_3(t)\right), \tag{22}$$

$$\frac{\partial \phi}{\partial \tau_1(t)} = 1, \qquad \frac{\partial \phi}{\partial \tau_2(t)} = 1, \qquad \frac{\partial \phi}{\partial \tau_3(t)} = 1. \tag{23}$$

From the above description, we can see that the modeling error of the end effector is more sensitive to the error of the joint that is far away from the end effector and less sensitive to the error of the joint that is close to the end effector. The robotic arm in our prototype has more than three joints and can move in a three-dimensional space. Thus, the Jacobian matrix could be more complicated than the two-dimensional robotic arm in Fig. 4.

The state, including joint angles and elements of the Jacobian matrix, needs to be normalized before feeding it into the neural network. We first find the maximum and minimum values of each joint angle and each element of the Jacobian matrix from the data set. Then, these values are employed to normalize the state within the range of $(0, 1)$.

2) Action: The action to be taken in the t-th time slot includes the joints to be scheduled, $X(t)$, and the optimal prediction horizons $Z(t)$. Although the prediction horizon needs to be transmitted to the server, $Z(t)$ is an integer ranging from 1 to 500. Thus, the communication overhead for updating $Z(t)$ is negligible compared to the update of the joint angle with high precision. We denote the action by $\mathbf{a}_t = [\mathbf{a}_t^{[1]}, \mathbf{a}_t^{[2]}] = [X(t), Z(t)]$, where the action of the i-th joint is denoted by $\mathbf{a}_{t,i}^{[1]} = x_i(t)$ and $\mathbf{a}_{t,i}^{[2]} = z_i(t)$.

3) Instantaneous Reward and Cost: Given the state and the action taken in the t-th time slot, the instantaneous reward, denoted by $r(\mathbf{s}(t), \mathbf{a}(t))$, is the communication load reduction compared with a benchmark that all joints are scheduled in every time slot. The instantaneous cost $c(\mathbf{s}(t), \mathbf{a}(t))$ is set to $\mathbf{e}(t)$ in (9).

4) Policy: The policy is represented by a neural network, $\pi_\theta(\mathbf{s}_t)$, where θ represents the training parameters. The network consists of multiple fully connected layers as shown in Fig. 5. In our study, the inputs to the policy networks include two different states: the angles of joints and the Jacobian matrix of the real-world robotic arm. The raw data of a joint angle has complete information and requires a complex neural network for feature extraction. The Jacobi matrix provides information that has been processed based on domain knowledge, and we can use a simple neural network for feature extraction. To handle different types of input information, we designed the two-branch neural network. Meanwhile, the two branches are designed to optimize two types of actions separately, i.e., the scheduling of a joint and the prediction horizon.

Specifically, the first two layers are designed for feature extraction, where the input denoted by $\{\mathcal{T}(t), \mathcal{J}(\mathcal{T}(t))\}$ is transformed into a more compact and informative representation that captures the underlying patterns. Then, we decouple the neural network into two parallel branches. The first branch is for the scheduling policy, $\pi_\theta^{[1]}$, and the second branch is for the policy of optimizing of the prediction horizons, $\pi_\theta^{[2]}$. After that, two branches are concatenated in the final linear layer. Followed by the Softmax activation function, the distribution of two actions, i.e., $\rho_t^{[1]}$ and $\rho_t^{[2]}$ are generated. Finally, two actions, i.e., $\mathbf{a}_t^{[1]}$ and $\mathbf{a}_t^{[2]}$ are sampled from $\rho_t^{[1]}$ and $\rho_t^{[2]}$, respectively.

Specifically, $\pi_\theta^{[1]}$ maps the state $\mathbf{s}_t$ to the distribution of $\mathbf{a}_{t,i}^{[1]}$, denoted by $\rho_t^{[1]} \in \mathbb{R}^{2\times I}$. The i-th column of $\rho_t^{[1]}$ is defined as follows,

$$\rho_{t,i}^{[1]} \triangleq \begin{pmatrix} \Pr\{a_{t,i}^{[1]} = 1\} \\ \Pr\{a_{t,i}^{[1]} = 0\} \end{pmatrix}. \tag{24}$$

Similarly, $\pi_\theta^{[2]}$ maps the state $\mathbf{s}_t$ to the distribution of $\mathbf{a}_{t,i}^{[2]}$, denoted by $\rho_t^{[2]} \in \mathbb{R}^{H\times I}$. The i-th column of $\rho_t^{[2]}$ is defined as follows,

$$\rho_{t,i}^{[2]} \triangleq \Big[\Pr\{a_{t,i}^{[2]} = 1\}, \Pr\{a_{t,i}^{[2]} = 2\}, ..., \Pr\{a_{t,i}^{[2]} = H\}\Big]^{\mathrm{T}}. \tag{25}$$

Once the distributions are obtained, $\mathbf{a}_{t,i}^{[1]}$ and $\mathbf{a}_{t,i}^{[2]}$ can be sampled from (24) and (25), respectively. Here, the probability of each action being sampled is based on the weight located in the corresponding elements [48]. The policies of different joints are represented by DNNs with the same structure. If there are more joints and sensors, they can reuse the DNN and fine-tune the parameters with few-shot learning. In this way, we can address the scalability issue.

5) Long-Term Reward and Cost: Following the policy π_θ, the long-term reward and long-term cost are defined as $R^{\pi_\theta} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}(t), \mathbf{a}(t)]$ and $C^{\pi_\theta} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t c(\mathbf{s}(t), \mathbf{a}(t)]$, respectively, where γ is the discount factor [12]. To estimate the long-term reward and long-term cost, we

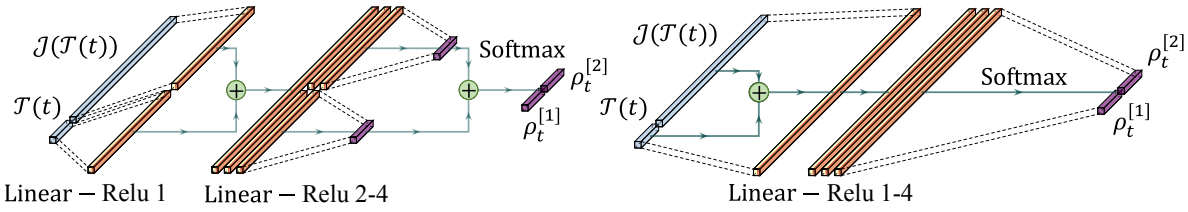


Fig. 5. Structures of neural networks: Two-branch neural network and fully-connected neural network.

can use the state-value function or the state-action value function. The state-value function and the state-action-value function of the long-term reward are defined as

$$V_r^{\pi_\theta}(\mathbf{s}) = \mathbb{E}[\sum\nolimits_{t=0}^{\infty} \gamma^t r(\mathbf{s}(t), \mathbf{a}(t)) \mid \mathbf{s}_0 = \mathbf{s},\ \pi_\theta], \tag{26}$$

$$Q_r^{\pi_\theta}(\mathbf{s}, \mathbf{a}) = \mathbb{E}[\sum\nolimits_{t=0}^{\infty} \gamma^t r(\mathbf{s}(t), \mathbf{a}(t)) \mid \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_0 = \mathbf{a},\ \pi_\theta], \tag{27}$$

respectively. The advantage function is given by $A_r^{\pi_\theta}(\mathbf{s}, \mathbf{a}) = Q_r^{\pi_\theta}(\mathbf{s}, \mathbf{a}) - V_r^{\pi_\theta}(\mathbf{s})$. Like the long-term reward, the state-value function and the state-action-value function of the long-term cost are defined as

$$V_c^{\pi_\theta}(\mathbf{s}) = \mathbb{E}[\sum\nolimits_{t=0}^{\infty} \gamma^t c(\mathbf{s}(t), \mathbf{a}(t)) \mid \mathbf{s}_0 = \mathbf{s},\ \pi_\theta], \tag{28}$$

$$Q_c^{\pi_\theta}(\mathbf{s}, \mathbf{a}) = \mathbb{E}[\sum\nolimits_{t=0}^{\infty} \gamma^t c(\mathbf{s}(t), \mathbf{a}(t)) \mid \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_0 = \mathbf{a},\ \pi_\theta], \tag{29}$$

respectively. The advantage function is given by $A_c^{\pi_\theta}(\mathbf{s}, \mathbf{a}) = Q_c^{\pi_\theta}(\mathbf{s}, \mathbf{a}) - V_c^{\pi_\theta}(\mathbf{s})$.

6) Modeling Accuracy Constraint: To guarantee the long-term modeling accuracy constraint, a straightforward approach is to evaluate C^{π_θ} by using $V_c^{\pi_\theta}(\mathbf{s})$ or $Q_c^{\pi_\theta}(\mathbf{s}, \mathbf{a})$. Note that the average long-term cost may not be applicable for mission-critical tasks in the Metaverse. For example, in haptic communications, users cannot recognize errors below a certain threshold, known as Just Noticeable Difference [49]. For mission-critical tasks, we propose to use CVaR of $Q_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)$ as the KPI. CVaR is a well-known risk measure used in financial portfolio analysis that depicts the cost in the tail of the risk distribution [11]. The expression of CVaR of $Q_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)$ is given by

$$\text{CVaR}_\alpha[Q_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)] = \min_{v \in \mathcal{R}} \left(v + \frac{1}{1-\alpha} \mathop{\mathbb{E}}_{\mathbf{s}_t, \mathbf{a}_t \sim \pi_\theta} \left\{ [Q_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) - v]^+ \right\} \right), \tag{30}$$

where $(x)^+ = \max(x, 0)$. $\alpha \in (0, 1)$ is the confidence level, and $Q_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)$ is equal to the average of the worst-case α-fraction of losses under optimal conditions [50].

Algorithm 1 C-PPO

Input: initial the parameters of neural network including policy network θ_0, initial state $\mathbf{s}_0$, step length β

for $t = 0, 1, 2, \ldots, T-1$ **do**
  Observe $\mathbf{s}_t$ and generate action from current policy $\pi_{\theta_t}([\mathbf{a}_t^{[1]}, \mathbf{a}_t^{[2]}] \mid \mathbf{s}_t)$
  Transmit the packets based on action $\mathbf{a}_t^{[1]}$
  Reconstruct the trajectory based on received packets by (1)
  Predict the trajectory based on action $\mathbf{a}_t^{[2]}$ by (2)
  Store state $\mathbf{s}_t$, action $\mathbf{a}_t$, reward r_t, cost c_t, and next state $\mathbf{s}_{t+1}$
  for $k = 1, 2, \ldots, K$ **do**
    Update $\text{CVaR}_\alpha[Q_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)]$ by (32)
  end for
  Compute the advantage function $A_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)$ and $A_r^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)$ based on (11), (33)
  if $\text{CVaR}_\alpha[Q_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)] \leq \frac{\Gamma_c}{1-\gamma}$ **then**
    Take one-step policy update towards maximizing $\mathcal{L}_r(\mathbf{s}_t, \mathbf{a}_t, \theta_t, \theta) : \theta_t \to \theta_{t+1}$
  else
    Take one-step policy update towards maximizing $\mathcal{L}_c(\mathbf{s}_t, \mathbf{a}_t, \theta_t, \theta)$: $\theta_t \to \theta_{t+1}$
  end if
end for

Output: Optimal policy π_θ^*

7) Problem Formulation: The goal is to find the optimal policy π_θ^* that maximizes the long-term reward R^{π_θ} subject to the constraint on CVaR of the long-term cost C^{π_θ}. Thus, the problem can be formulated as follows:

$$\pi_\theta^* = \arg\max_\theta Q_r^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \tag{31}$$

$$\text{s.t.} \quad \text{CVaR}_\alpha[Q_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)] \leq \frac{\Gamma_c}{1-\gamma}, \tag{31a}$$

where Γ_c is the modeling error depending on the requirements of specific tasks in the Metaverse.

C. C-PPO Algorithm

To guarantee the modeling accuracy constraint, we develop a C-PPO algorithm by integrating PPO and CVaR into the CRPO algorithm, which is a safe reinforcement learning algorithm with

convergence guarantee [12]. The basic idea of the CRPO algorithm is to maximize the long-term reward when the constraint is satisfied and minimize the long-term cost when the constraint is violated.

The details of the C-PPO algorithm can be found in Algorithm 1. In the t-th step, we first update the threshold of CVaR according to the current policy by the gradient descent,

$$v^{(k+1)} = v^{(k)} - \beta \frac{\Delta \text{CVaR}_\alpha(v^{(k)})}{\Delta v^{(k)}}, \tag{32}$$

where β is the learning rate and it takes K steps of gradient descent to update the threshold, $k = 1, ..., K$. Then, we validate whether the constraint can be satisfied or not. If the constraint in (31a) is satisfied, we maximize $\mathcal{L}_r(\mathbf{s}_t, \mathbf{a}_t, \theta_t, \theta)$ which is obtained by substituting $A_r^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)$ into (11). Otherwise, we minimize $\mathcal{L}_c(\mathbf{s}_t, \mathbf{a}_t, \theta_t, \theta)$ defined as follows,

$$\begin{aligned} \mathcal{L}_c(\mathbf{s}_t, \mathbf{a}_t, \theta_t, \theta) = \min \Bigg(& \frac{\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)}{\pi_{\theta_t}(\mathbf{a}_t \mid \mathbf{s}_t)} A_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t), \\ & \text{clip}\left(\frac{\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)}{\pi_{\theta_t}(\mathbf{a}_t \mid \mathbf{s}_t)}, 1-\epsilon, 1+\epsilon, \right) A_c^{\pi_{\theta_t}}(\mathbf{s}_t, \mathbf{a}_t) \Bigg), \end{aligned} \tag{33}$$

where $A_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)$ is obtained by Generalized Advantage Estimate (GAE) [46], [51]. With C-PPO, the parameters are updated according to the following expression,

$$\theta_{t+1} = \begin{cases} \theta_t + \alpha \nabla_\theta \mathcal{L}_r(\mathbf{s}_t, \mathbf{a}_t, \theta_t, \theta), & \text{CVaR}_\alpha[Q_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)] \leq \frac{\Gamma_c}{1-\gamma}, \\ \theta_t - \alpha \nabla_\theta \mathcal{L}_c(\mathbf{s}_t, \mathbf{a}_t, \theta_t, \theta), & \text{CVaR}_\alpha[Q_c^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)] > \frac{\Gamma_c}{1-\gamma}. \end{cases} \tag{34}$$

It is worth noting that the policy gradient and the CVaR gradient can be updated with different learning rates. To guarantee a stable CVaR constraint when performing the policy gradient, we update the threshold of CVaR with a higher frequency.

IV. Prototype Design and Performance evaluation

In this section, we demonstrate our cross-system prototype[3] design as shown in Fig. 6. Based on the prototype, we first evaluate the effectiveness of the proposed cross-system design framework and then compare the performance with different benchmarks.

[3]We will release our dataset and source code of C-PPO if this paper gets published.

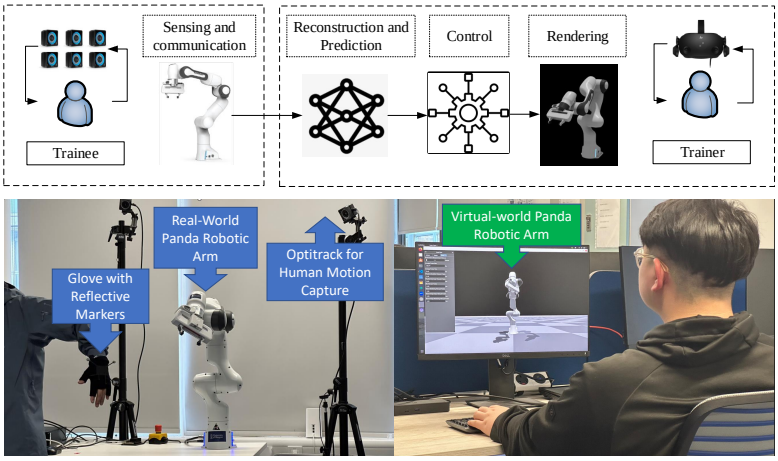


Fig. 6. Illustration of our prototype system. 1) Bottom left photo: A real-world robotic arm is controlled by a human trainee, 2) Bottom right photo: A digital model of the robotic arm in the Metaverse is rendered and presented to a trainer, 3) The diagram on the top of the two photos shows the system functions implemented at the two sides.

A. *Prototype Design*

1) Real-World Robotic Arm: We adopt an industrial-grade robotic arm system, Franka Emika Panda [52], in our prototype. The robotic arm has seven degrees of freedom (DoF) and can achieve up to 2 m/s end effector speed and ± 0.1 mm repeatability. In our prototype design, we use five DoF of the real-world robotic arm. The trainer wearing optical markers controls the robotic arm via the state-of-the-art motion capture system with six motion cameras, OptiTrack Prime-13 [53]. The reason why we use OptiTrack is because 1) It is a real-time motion capture system with high accuracy and low latency. This real-time high-fidelity tracking is essential for maintaining an accurate digital twin in the Metaverse and for evaluating the performance of our task-oriented, cross-system design framework. The use of OptiTrack also allows us to quantify the task-oriented KPI, i.e., average tracking error between the real-world robotic arm and its digital model in the Metaverse. 2) The OptiTrack motion tracking system can be scaled to accommodate various tracking volumes, from small studio setups to large outdoor environments. This scalability makes it versatile and adaptable for different types of motion capture objects in the Metaverse. 3) Our proposed framework is not limited to the use of OptiTrack as a motion capture device. Flexible positioning and motion tracking system can be used bsaed on the practical accuracy demands of the Metaverse applications.

The robotic arm receives the target pose from the motion capture system and then maps

TABLE II

SYSTEM PARAMETERS FOR PERFORMANCE EVALUATION

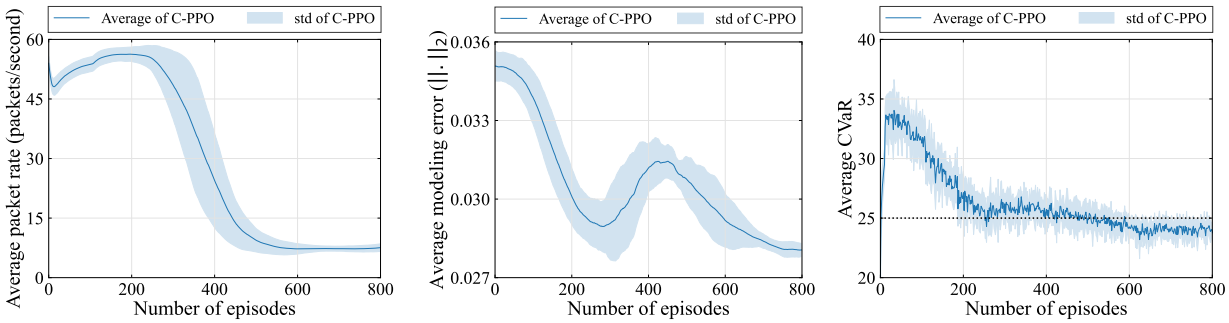
Prototype Setup		**Learning Setup**	
Parameters	**Values**	**Parameters**	**Values**
Duration of time slot	$1\,\mathrm{ms}$	Learning rate of actor network	3×10^{-4}
Number of joints I	5	Learning rate of critic network	3×10^{-4}
Input length of reconstruction function W_l	$2000\,\mathrm{ms}$	Learning rate of cost network	3×10^{-4}
Input length of prediction function W_p	$2000\,\mathrm{ms}$	Batch size	256
Prediction horizon of prediction function H	$500\,\mathrm{ms}$	Discount factor γ	0.99
Control interval N_c	$2\,\mathrm{ms}$	Clip ratio in the loss functions of C-PPO ϵ	0.2
Refresh rate of image $1/N_r t_s$	$60\,\mathrm{Hz}$	Total steps for updating CVaR K	500
Experimental time	$5 \times 10^4\,\mathrm{ms}$	Learning rate of constraint β	2×10^{-3}
Weighting coefficient of position ω_1	0.5	Confidence level of CVaR $1 - \alpha_c$	0.95
Weighting coefficient of orientation ω_2	0.5		

the pose to joint angles. After that, the robotic arm applies a proportional-integral-derivative method [54] for control, which converts the joint angles to a series of commands. Meanwhile, built-in sensors in the robotic arm measure joint angles, angular velocities, and inertial torque of each joint [52].

2) Virtual Robotic Arm in the Metaverse: We establish the Metaverse in the Nvidia Isaac Gym [55], a cutting-edge robotics simulation engine that uses state-of-the-art algorithms and physics engines to simulate the movement and behavior of robots in various environments. Meanwhile, we simulate the communication system between the real-world robotic arm and the Nvidia Isaac Gym by introducing a Gaussian-distributed communication latency. Its mean and the standard deviation are set to 10 ms and 1 ms, respectively.

B. System Setup

1) Parameters of the Prototype: For the prototype design, five joint angles of the real-world robotic arm are controlled by the trainee and measured by built-in sensors in each time slot. The measured data of the real-world and virtual-world robotic arms are recorded in the CSV format file. For the training process of the predictor, *informer*, we set the prediction input length W_p to 2000 ms and output length H to 500 ms. In Nvidia Issac Gym, we set the control interval N_c to 2 ms. The frequency at which the robotic arm interacts with the virtual environment is 500 Hz. The method for calculating the Jacobian Matrix can be found in Appendix A. The default parameters of the prototype and learning algorithm are listed in Table II, unless otherwise specified.



(a) Average packet rate in each episode.

(b) Average modeling error in each episode.

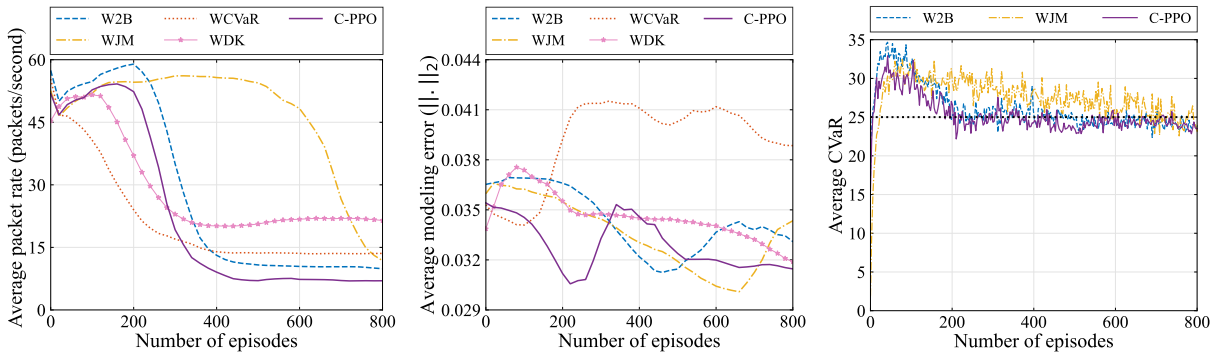
(c) Average value of CVaR in each episode.

Fig. 7. Performance evaluation.

2) Benchmarks: We build our C-PPO algorithm and four benchmarks in the well-known DRL library *Stable-Baselines3* [51]. The legends of the benchmarks are "W2B", "WJM", "WCVaR", and "WDK", respectively. (a) In W2B, the two-branch neural network is replaced with a fully-connected neural network; (b) In WJM, the Jacobian matrix is not included in the state; (c) In WCVaR, the CVaR of the modeling error is replaced with the average modeling error in the constraint; (d) WDK is a simplified C-PPO without using any domain knowledge (i.e., the two-branch neural network, Jacobian matrix, and CVaR of the modeling error). With the above benchmarks, we will illustrate the impact of different types of domain knowledge on the performance of our C-PPO algorithm.

C. Evaluation of C-PPO Algorithm

As shown in Fig. 7, the performance of C-PPO is evaluated by average packet rate, average modeling error, and average CVaR. We train C-PPO with 800 episodes to show the trends in performance changes. The average packet rate at the start point is defined as a baseline. The average packet rate reaches 8 packets/second, which saves 83.3% communication load than the baseline. In addition, optimizing under the restriction of the constraint CVaR, the average tracking error of the digital model achieves 0.0281, which is lower than the tracking error of the baseline (0.0347), as shown in Fig. 7(b). Meanwhile, the CVaR fluctuates around the constraint bound. The results show that C-PPO converges after 250 training episodes. Meanwhile, constraints and the average tracking error fluctuate slightly throughout training. In particular, C-PPO is stable and effective, since it performs consistently better in all ten training repetitions. It is also



(a) Average packet rate in each episode.

(b) Average modeling error in each episode.

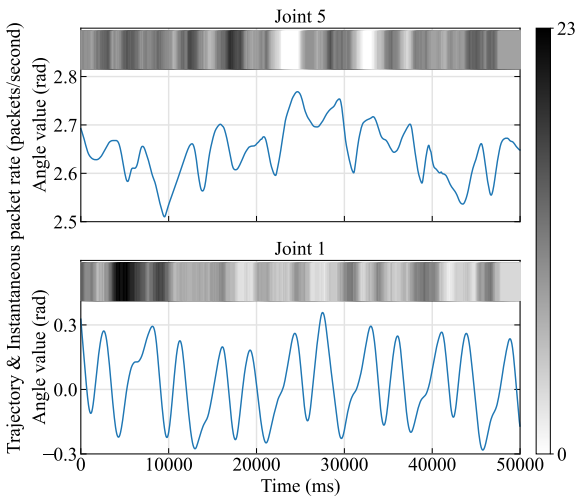
(c) Average value of CVaR in each episode.

Fig. 8. Ablation Study.

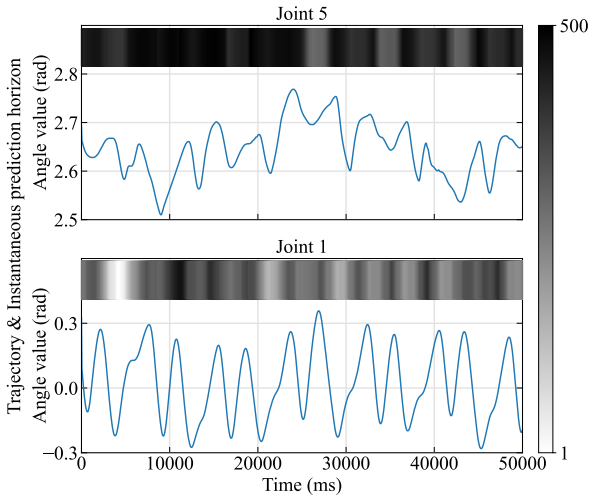
worth noting that although the prediction horizon needs to be transmitted to the server, $Z(t)$ is an integer ranging from 1 to 500. Thus, the communication overhead for updating $Z(t)$ is negligible compared to the update of the joint angle with high precision.

D. Ablation Study of C-PPO Algorithm

The performance of the C-PPO and the four benchmarks are illustrated in Fig. 8. In general, our C-PPO achieves the best performance in terms of convergence time, average packet rate, and average modeling error. The results in Fig. 8(a) show that the C-PPO can reduce the required average packet rate by around 50% compared to the benchmark without domain knowledge, WDK (from 17 packets/second to 8 packets/second). By comparing C-PPO with WJM, we can see that the Jacobian matrix can reduce the convergence time by 50% (from 800 episodes to 400 episodes). In Fig. 8(b), we evaluate the average modeling errors achieved by different algorithms. From C-PPO and WCVaR, we can observe that by using CVaR as the metric of the modeling error, the average modeling error is reduced from 0.041 to 0.032 (around 20% reduction), where the modeling error is defined in (9). The results 8(c) show that C-PPO, W2B, and WJM can guarantee the constraint on CVaR of the modeling error, that is, the dashed horizontal line. The other two benchmarks do not consider CVaR, and hence are not shown in this figure. From the performance of W2B in all these figures, we can see that the two-branch neural network converges faster than the fully-connected neural network and achieves better performance in terms of the average packet rate and average modeling error.



(a) Trajectories and instantaneous packet rates



(b) Trajectories and instantaneous prediction horizon

Fig. 9. Trajectories, instantaneous packet rates and instantaneous prediction horizon of two joints, where joint 5 is close to the end effector, and joint 1 is close to the base of the robotic arm.

E. Validation of Cross-System Design Framework

1) Dynamic scheduling in C-PPO: In Fig. 9(a), we provide an example to show how the proposed C-PPO algorithm changes the packet rates according to joint angles. The packet rates are represented by the grayscale intensity. As the grayscale intensity increases from white to black, the packet rates increase from 0 to 23 packets/second. The results imply that packet rates are correlated with fluctuations in joint angles. Besides, the joint that is far away from the end effector has higher average packet rates than the joint that is close to the end effector. This is because the modeling error of the end effector is less sensitive to modeling errors of the joints that are closer to it.

2) Dynamic prediction horizon in C-PPO: The effect of $Z(t)$ has already been demonstrated in the existing literature [56], [57]. The latency in the communication systems is stochastic, so we need to adjust the prediction horizon to compensate for the communication latency. In this way, we can reduce the modeling error in the Metaverse. In addition, we also provide the diagram of $Z(t)$ along with the trajectories of the real-world robotic arm. As shown in Fig. 9(b), the value of the prediction horizon is depicted through the grayscale intensity, where darker intensities correspond to longer prediction horizons. The results show that the prediction horizon is adjusted according to the mobility of the joints. Moreover, the prediction horizon of joint 5

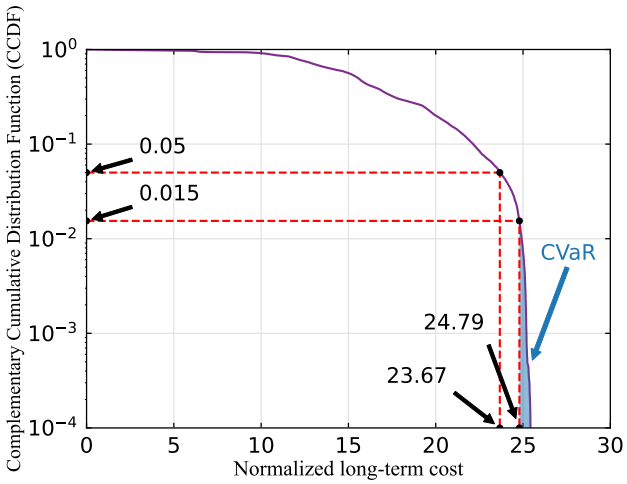


Fig. 10. CCDF of the long-term cost.

is much longer than that of joint 1. This is because the modeling accuracy of the end effector is less sensitive to the prediction errors of the joint that is closer to it compared to the joint that is farther from it.

3) CVaR: The Complementary Cumulative Distribution Function (CCDF) of the long-term cost is presented in Fig. 10. The results show that with the probability of 98.5%, the long-term cost is below the required threshold, which is set to 25 in the experiment. In addition, the probability (98.5%) is higher than the confidence level (95%). The result also indicates that the proposed C-PPO can significantly reduce the tail probability (i.e., the probability that the long-term cost is higher than the required threshold) of the long-term cost.

4) Performance Comparison: We compare our proposed cross-system design framework with a baseline framework: there is no prediction, and all joints transmit data packets in every time slot. In Fig. 11, we show the trajectories of the real-world robotic and two digital models obtained from our cross-system design framework and the baseline framework. The results show that with prediction, the cross-system design framework can model the virtual-world robotic arm in a timely manner. Without prediction, the user can recognize the modeling error caused by communication latency. In Fig. 12, we further illustrate the CCDF of modeling errors in the two frameworks. The results demonstrate that the cross-system design framework outperforms the baseline framework in terms of the tail distribution of the modeling error. Besides, with the baseline framework, all the joints transmit packets in every time slot. The packet rate is

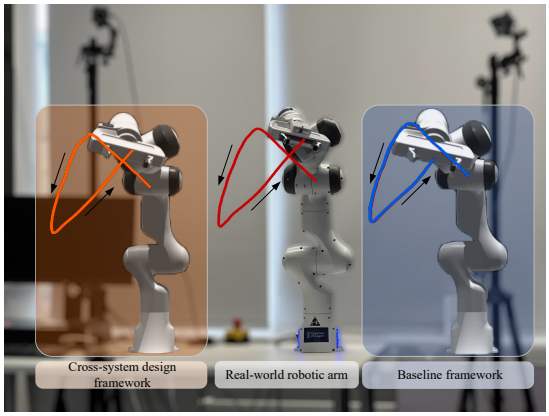


Fig. 11. Comparison among the trajectories of the real-world robotic arm (at the centre) and its digital models (on two sides, the left one is designed by the proposed framework while the right one is implemented by the baseline framework without cross-system design).

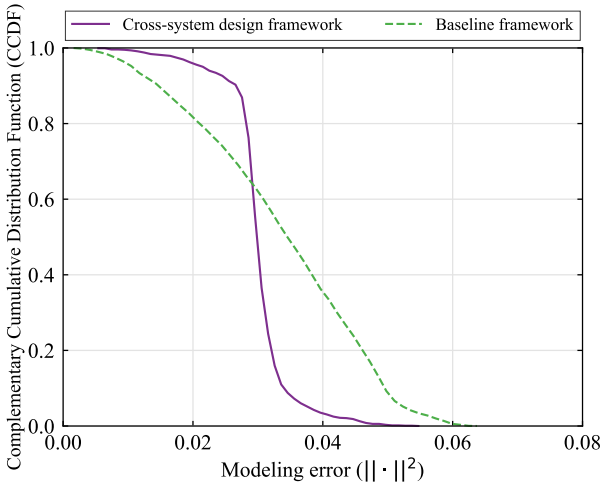


Fig. 12. Modeling errors of the proposed cross-system design framework and the baseline framework.

5000 packets/s, which is much higher than the cross-system design framework.

V. Conclusions

In this work, we established a task-oriented cross-system design framework to minimize the required packet rate to meet a constraint on the modeling error of a robotic arm in the Metaverse. To optimize the scheduling policy and the prediction horizons, we developed a C-PPO algorithm by integrating domain knowledge into the PPO. A prototype was built to evaluate the performance of the C-PPO and the cross-system design framework. Experimental results showed that the

domain knowledge helps reduce the required packet rate and the convergence time by up to 50%, and the cross-design framework outperforms a baseline framework in terms of the required packet rate and the tail distribution of the modeling error.

APPENDIX A
CALCULATION OF JACOBIAN MATRIX

For notational simplicity, the notations used in the appendix are different from the notations used in the main text.

To obtain the Jacobian matrix defined in Section III-B, one approach is to compute the partial derivatives with respect to each joint angle. The computation complexity of this approach could be high, and we introduce a low-complexity numerical method in this appendix to obtain the Jacobian matrix [47]. For a robotic arm with I rotation joints, the Jacobian matrix can be obtained from the following expression,

$$\mathcal{J} = \begin{bmatrix} {}^0_0R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times ({}^0_I\xi - {}^0_0\xi) & {}^0_1R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times ({}^0_I\xi - {}^0_1\xi) & \dots & {}^0_{I-1}R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times ({}^0_I\xi - {}^0_{I-1}\xi) \\ {}^0_0R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & {}^0_1R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \dots & {}^0_{I-1}R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} \in \mathbb{R}^{6\times I}, \tag{35}$$

where $\times$ is the cross product operation defined by: $\mathbf{a} = [x_1, y_1, z_1]$, $\mathbf{b} = [x_2, y_2, z_2]$, $\mathbf{a} \times \mathbf{b} = [y_1z_2 - y_2z_1,\ x_2z_1 - x_1z_2,\ x_1y_2 - x_2y_1]^{\mathrm{T}}$, ${}^0_{I-1}R \in \mathbb{R}^{3\times3}$ is the rotation matrix that describes the rotation of the coordinate frame $\{I-1\}$ in the coordinate frame $\{0\}$ which is the base coordinate and ${}^0_{I-1}\xi \in \mathbb{R}^{3\times1}$ is the translation vector that describes the translation of the origin of the coordinate frame $\{I-1\}$ in the coordinate frame $\{0\}$. In robotics, a coordinate frame is a system of reference used to describe the position and motion of robots in space. As shown in Fig. 13, the rotation center of a joint is commonly used as the reference point for setting up the coordinate frame $\{i\}$. Then, by concatenating the rotation matrix and the translation vector, the transformation matrix of ${}^0_{I-1}T$ is expressed as

$${}^0_{I-1}T = \left[\begin{array}{c|c} {}^0_{I-1}R & {}^0_{I-1}\xi \\ \hline [0,0,0] & 1 \end{array}\right] \in \mathbb{R}^{4\times4}, \tag{36}$$

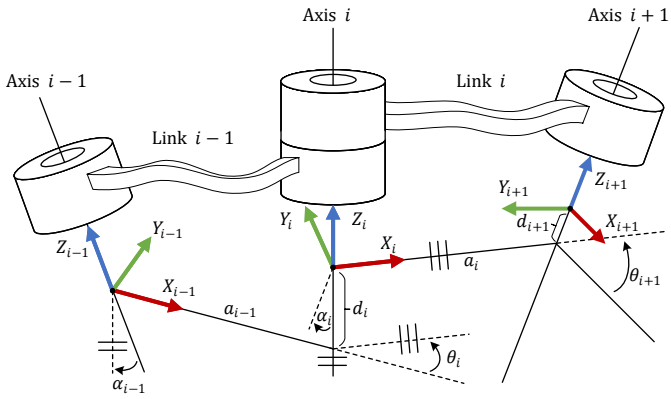


Fig. 13. The coordinate frame $\{i\}$ and D-H parameters.

which describes the relative position and orientation of coordinate frame $\{I-1\}$ with respect to the coordinate frame $\{0\}$.

One way to obtain the transformation matrix is deriving the D-H parameters [47]. D-H parameters provide a systematic way to describe the position and orientation of each link and joint in the robot in the joint space which is widely used by industrial manufacturers. The relationship between D-H parameters and transformation matrix ${}^{i-1}_{i}T$ can be expressed by

$$
{}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin(\alpha_{i-1})d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{37}
$$

As shown in Fig 13, θ_i is the angle value of i-th joint, α_{i-1}, a_{i-1}, d_i are the constant parameters determined by the mechanical system. Then, the transformation matrix ${}^{0}_{i}T$ between coordinate frames $\{0\}$ and $\{i\}$ can be obtained by the forward kinematic chain [47],

$$
{}^{0}_{i}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\cdots{}^{i-1}_{i}T \tag{38}
$$

In our prototype, Franka Emika Panda robotic arm is used. The corresponding D-H parameters are shown in Table III [58]. Thus, by substituting the D-H parameters into (38), we can obtain ${}^{0}_{i}T$, ${}^{0}_{i}R$ and ${}^{0}_{i}\xi$, $i=1,2,...,I$. Then, by substituting ${}^{0}_{i}R$ and ${}^{0}_{i}\xi$, $i=1,2,...,I$, into (35), we can obtain the Jacobian matrix $\mathcal{J}$. This completes the calculation of the Jacobian matrix.

APPENDIX B
FOUNDATION OF QUATERNION

Specifically, imaginary basis vectors follow the multiplication rules as

$$\boldsymbol{u}_{\mathrm{i}}^2 = \boldsymbol{u}_{\mathrm{j}}^2 = \boldsymbol{u}_{\mathrm{k}}^2 = -1, \qquad \boldsymbol{u}_{\mathrm{i}}\boldsymbol{u}_{\mathrm{j}}\boldsymbol{u}_{\mathrm{k}} = -1 \tag{39}$$

$$\boldsymbol{u}_{\mathrm{i}}\boldsymbol{u}_{\mathrm{j}} = -\boldsymbol{u}_{\mathrm{j}}\boldsymbol{u}_{\mathrm{i}} = \boldsymbol{u}_{\mathrm{k}}, \qquad \boldsymbol{u}_{\mathrm{j}}\boldsymbol{u}_{\mathrm{k}} = -\boldsymbol{u}_{\mathrm{k}}\boldsymbol{u}_{\mathrm{j}} = \boldsymbol{u}_{\mathrm{i}}, \qquad \boldsymbol{u}_{\mathrm{k}}\boldsymbol{u}_{\mathrm{i}} = -\boldsymbol{u}_{\mathrm{i}}\boldsymbol{u}_{\mathrm{k}} = \boldsymbol{u}_{\mathrm{j}}. \tag{40}$$

As shown in Fig. 3, according to the unit vector of the rotation axis $\boldsymbol{\eta}_{\mathrm{c}}$ and the rotation angle ψ_η, the quaternion can be obtained by

$$\begin{aligned}\boldsymbol{q} &= F_q(a_{\mathrm{c}}, b_{\mathrm{c}}, c_{\mathrm{c}}, \psi_\eta) \\ &= \sin(\frac{\psi_\eta}{2}) \cdot a_{\mathrm{c}} \cdot \boldsymbol{u}_{\mathrm{i}} + \sin(\frac{\psi_\eta}{2}) \cdot b_{\mathrm{c}} \cdot \boldsymbol{u}_{\mathrm{j}} + \sin(\frac{\psi_\eta}{2}) \cdot c_{\mathrm{c}} \cdot \boldsymbol{u}_{\mathrm{k}} + \cos(\frac{\psi_\eta}{2}).\end{aligned} \tag{41}$$

TABLE III
D-H PARAMETERS OF FRANKA EMIKA PANDA ROBOTIC ARM

Joint	a (m)	d (m)	α (rad)	θ (rad)
Joint 1	0	0.333	0	θ_1
Joint 2	0	0	$-\frac{\pi}{2}$	θ_2
Joint 3	0	0.316	$\frac{\pi}{2}$	θ_3
Joint 4	0.0825	0	$\frac{\pi}{2}$	θ_4
Joint 5	-0.0825	0.384	$-\frac{\pi}{2}$	θ_5
Joint 6	0	0	$\frac{\pi}{2}$	θ_6
Joint 7	0.088	0	$\frac{\pi}{2}$	θ_7

REFERENCES

[1] L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, and et al, "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

[2] 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.

[3] 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.

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

[5] A. M. Girgis, J. Park, M. Bennis, and M. Debbah, "Predictive control and communication co-design via two-way gaussian process regression and AoI-aware scheduling," *IEEE Trans. Commun.*, vol. 69, no. 10, pp. 7077–7093, 2021.

[6] 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.

[7] 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

[8] Y. Sun, Y. Polyanskiy, and E. Uysal, "Sampling of the wiener process for remote estimation over a channel with random delay," *IEEE Trans. Inf. Theory*, vol. 66, no. 2, pp. 1118–1135, 2020.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[10] 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

[11] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," 2015. [Online]. Available: https://arxiv.org/abs/1512.01629

[12] T. Xu, Y. Liang, and G. Lan, "CRPO: A new approach for safe reinforcement learning with convergence guarantee," in *Proc. Int. Conf. on Mach. Learn. (ICML)*, 2021, pp. 11 480–11 491.

[13] Z. Gu, C. She, W. Hardjawana, S. Lumb, D. McKechnie, T. Essery, and et al, "Knowledge-assisted deep reinforcement learning in 5G scheduler design: From theoretical framework to implementation," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2014–2028, 2021.

[14] C. She, C. Sun, Z. Gu, Y. Li, C. Yang, H. V. Poor, and et al, "A Tutorial on ultrareliable and low-latency communications in 6G: integrating domain knowledge into deep learning," *Proc. IEEE*, vol. 109, no. 3, pp. 204–246, 2021.

[15] N. Stephenson, *Snow crash: A novel*. Spectra, 2003.

[16] V.-P. Bui, S. R. Pandey, F. Chiariotti, and P. Popovski, "Game networking and its evolution towards supporting Metaverse through the 6G wireless systems," 2023. [Online]. Available: https://arxiv.org/abs/2302.01672

[17] M. Xu, D. Niyato, B. Wright, H. Zhang, J. Kang, Z. Xiong, and et al, "EPViSA: Efficient auction design for real-time physical-virtual synchronization in the Metaverse," 2022. [Online]. Available: https://arxiv.org/abs/2211.06838

[18] Y. Han, D. Niyato, C. Leung, D. I. Kim, K. Zhu, S. Feng, X. Shen, and C. Miao, "A dynamic hierarchical framework for iot-assisted digital twin synchronization in the metaverse," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 268–284, 2023.

[19] S. K. Jagatheesaperumal, K. Ahmad, A. Al-Fuqaha, and J. Qadir, "Advancing education through extended reality and internet of everything enabled Metaverses: Applications, challenges, and open issues," 2022. [Online]. Available: https://arxiv.org/abs/2207.01512

[20] H. Tataria, M. Shafi, M. Dohler, and S. Sun, "Six critical challenges for 6G wireless systems: A summary and some solutions," *IEEE Vehicular Technol. Mag.*, vol. 17, no. 1, pp. 16–26, 2022.

[21] K. Wang, J. Jin, Y. Yang, T. Zhang, A. Nallanathan, C. Tellambura, and B. Jabbari, "Task offloading with multi-tier computing resources in next generation wireless networks," 2022. [Online]. Available: https://arxiv.org/abs/2205.13866

[22] L. U. Khan, M. Guizani, D. Niyato, A. Al-Fuqaha, and M. Debbah, "Metaverse for wireless systems: Architecture, advances, standardization, and open challenges," 2023. [Online]. Available: https://arxiv.org/abs/2301.11441

[23] C. She, C. Yang, and T. Q. S. Quek, "Cross-layer optimization for ultra-reliable and low-latency radio access networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 127–141, 2018.

[24] E. Glaessgen and D. Stargel, "The digital twin paradigm for future NASA and U.S. air force vehicles," in *Proc. 53rd AIAA/ASME/ASCE/AHS/ASC Struct. Struct. Dyn.*, 2012, pp. 1–14.

[25] N. H. Chu, D. N. Nguyen, D. T. Hoang, K. T. Phan, E. Dutkiewicz, D. Niyato, and et al, "Dynamic resource allocation for Metaverse applications with deep reinforcement learning," 2023. [Online]. Available: https://arxiv.org/abs/2302.13445

[26] O. Hashash, C. Chaccour, W. Saad, K. Sakaguchi, and T. Yu, "Towards a decentralized Metaverse: Synchronized orchestration of digital twins and sub-Metaverses," 2022. [Online]. Available: https://arxiv.org/abs/2211.14686

[27] S. Zeng, Z. Li, H. Yu, Z. Zhang, L. Luo, B. Li, and et al, "HFedMS: Heterogeneous federated learning with memorable data semantics in industrial Metaverse," 2022. [Online]. Available: https://arxiv.org/abs/2211.03300

[28] Y. Lu, S. Maharjan, and Y. Zhang, "Adaptive edge association for wireless digital twin networks in 6G," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16 219–16 230, 2021.

[29] O. Hashash, C. Chaccour, and W. Saad, "Edge continual learning for dynamic digital twins over wireless networks," 2022.

[30] J. Yu, A. Alhilal, P. Hui, and D. H. K. Tsang, "6G mobile-edge empowered metaverse: Requirements, technologies, challenges and research directions," 2023.

[31] P. M. d. Santana, N. Marchenko, B. Soret, and P. Popovski, "Goal-oriented wireless communication for a remotely controlled autonomous guided vehicle," *IEEE Wireless Commun. Lett.*, 2023.

[32] J. Shao, Y. Mao, and J. Zhang, "Learning task-oriented communication for edge inference: An information bottleneck approach," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 197–211, 2021.

[33] D. Wen, P. Liu, G. Zhu, Y. Shi, J. Xu, Y. C. Eldar, and et al, "Task-oriented sensing, computation, and communication integration for multi-device edge AI," 2022. [Online]. Available: https://arxiv.org/abs/2207.00969

[34] W. Yu, T. J. Chua, and J. Zhao, "User-centric heterogeneous-action deep reinforcement learning for virtual reality in the Metaverse over wireless networks," 2023. [Online]. Available: https://arxiv.org/abs/2302.01471

[35] S. K. Jagatheesaperumal, Z. Yang, Q. Yang, C. Huang, W. Xu, M. Shikh-Bahaei, and Z. Zhang, "Semantic-aware digital twin for metaverse: A comprehensive review," 2023. [Online]. Available: https://arxiv.org/abs/2305.18304

[36] 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.

[37] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and et al, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[38] S. M. Ross, *Stochastic processes*. John Wiley & Sons, 1995.

[39] J. J. Craig, *Introduction to robotics*. Pearson Education, 2006.

[40] "Motion Generation," https://docs.omniverse.nvidia.com/app_isaacsim/app_isaacsim/ext_omni_isaac_motion_generation.html#isaac-sim-motion-generation, (accessed Mar. 2023).

[41] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, and et al, "Advances in neural rendering," 2021. [Online]. Available: https://arxiv.org/abs/2111.05849

[42] J. B. Kuipers, *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton Univ. Press, 1999.

[43] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, 01 2006.

[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[45] N. Heess, T. B. Dhruva, S. Sriram, J. Lemmon, and D. Silver, "Emergence of locomotion behaviours in rich environments," 2017.

[46] "Proximal Policy Optimization," https://spinningup.openai.com/en/latest/algorithms/ppo.html, (accessed Jan. 2020).

[47] 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.

[48] "torch.multinomial - Pytorch 2.0," https://pytorch.org/docs/stable/generated/torch.multinomial.html, (accessed Jul. 2023).

[49] S. Feyzabadi, S. Straube, M. Folgheraiter, E. A. Kirchner, S. K. Kim, and J. C. Albiez, "Human force discrimination during active arm motion for force feedback design," *IEEE Trans. Haptics*, vol. 6, no. 3, pp. 309–319, 2013.

[50] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *J. risk*, vol. 2, pp. 21–42, 2000.

[51] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[52] F. E. GmbH, "Franka Control Interface Documentation," https://frankaemika.github.io/docs/index.html, (accessed Sep. 2, 2022).

[53] I. D. O. NaturalPoint, "OptiTrack Prime 13," https://www.optitrack.com/cameras/primex-13.html, (accessed: Jan. 10, 2023).

[54] 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.

[55] "Isaac Gym," https://developer.nvidia.com/isaac-gym, (accessed Jul. 2022).

[56] 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, 2020.

[57] 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.

[58] F. E. GmbH, "Robot and interface specifications," https://frankaemika.github.io/docs/control_parameters.html, (accessed Mar. 21, 2023).