DRT-RL: Residual Reinforcement Learning for Delay-Resilient Robotic Teleoperation

Kaize Deng Zewen Yang

Technical University of Munich, 80333 Munich, Germany
(e-mail: kaize.deng, zewenyang@tum.de).

Abstract: Stochastic communication delays in teleoperation introduce signal discontinuities that undermine control stability and degrade control performance. Consequently, the conventional reinforcement learning (RL) methods struggle with the delayed observations due to the delay-induced observations, leading to high-frequency chattering. To address this, we propose a hybrid control framework, DRT-RL, integrating a state estimator utilizing Long Short-Term Memory (LSTM) with a residual RL policy, which is resilient to stochastic delays. The LSTM reconstructs smooth, continuous state estimates from delayed observations, enabling the RL agent to learn a residual torque compensation policy that balances tracking accuracy with velocity smoothness. Experimental validation on Franka Panda robots demonstrates that our approach significantly outperforms the state-of-the-art baselines, ensuring robust and stable teleoperation even under high-variance stochastic delays.

Keywords: teleoperation system, networked systems, stochastic delay, reinforcement learning, deep neural network

1. INTRODUCTION

Teleoperation, a critical technology in modern robotics, enables human operators to control remote manipulators across geographical distances via networked communication. This capability facilitates expert intervention in environments where direct physical presence is impractical or hazardous, spurring transformative applications in domains such as remote surgery (Choi et al., 2018), distributed manufacturing (Manupati et al., 2017), and even space robotic coordination (Ruoff, 1994).

Architecturally, a teleoperation system comprises two primary components: a local leader device, which captures the human operator's commands, and a remote follower manipulator, which is tasked with executing the corresponding movements in the distant environment (see Fig. 1). First, the networked communication channel between the local and remote sites introduces a stochastic, time-varying state transmission delay. Specifically, the state signal of the leader is sent over the network to the remote reinforcement learning (RL) agent and arrives with a delay $\omega_s^t \in \mathbb{R}_{\geq 0}$ at time $t \in \mathbb{R}_{>0}$. This information gap fundamentally transforms the problem from a fully observable Markov Decision Process (MDP) into a Partially Observable MDP (POMDP), in which an optimal action cannot be computed directly from the past states. Second, the agent–follower link is subject to inherent bidirectional delays. The observation delay $\omega_o^t \in \mathbb{R}\geq 0$ affects the sensor feedback from the follower and directly induces delay observability. Therefore, at any time t, the controller has access only to delayed state information corresponding to $t - \omega_o^t$, rather than the true current state of the follower robot. Third, stochastic variability in the action delay $\omega_a^t \in \mathbb{R}_{\geq 0}$, commonly referred to as jitter, disrupts the command stream along the same agent–follower link.

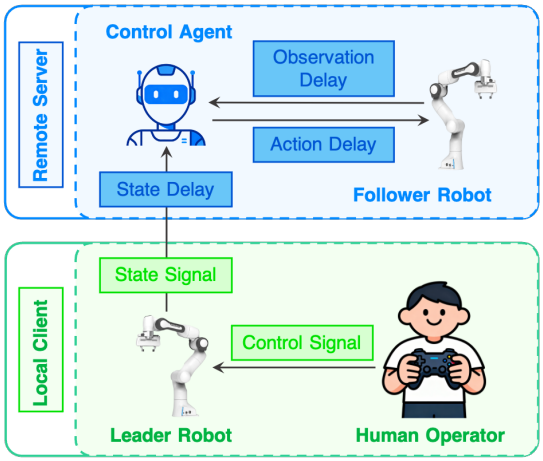


Fig. 1. Teleoperation architecture with stochastic communication delays. The time-varying state delay ω_s^t is induced by client-server transmission latency; the action delay ω_a^t and observation delay ω_o^t arise from the bidirectional delays within the agent-follower control loop.

All these factors are particularly detrimental to low-level Proportional–Derivative (PD) control laws, i.e., the asynchronous arrival of commands injects artificial variability into the derivative of the tracking error, thereby inducing high-frequency oscillations, mechanical vibration, and chattering in the closed-loop system.

1.1 Related Work

Prior work has addressed the challenges induced by communication delays in teleoperation through several paradigms. Passivity-based control frameworks leverage

energy conservation principles to establish formal stability guarantees (Anderson and Spong, 1989; Niemeyer and Slotine, 1991; Mujčić and Oračević, 2019). By mapping impedance signals into scattering variables, these methods ensure passivity of the communication channel and, by extension, closed-loop stability irrespective of constant delay magnitude. However, the theoretical stability is not guaranteed when delays are stochastic and time-varying rather than constant in real-world networks. Moreover, the requisite signal transformation compromises system transparency, leading to degraded tracking fidelity and attenuated force reflection. Model-based predictive strategies, such as the Smith Predictor (Smith, 1957), attempt to explicitly compensate for latency by simulating the system response forward in time. However, this approach is predicated on accurate a priori knowledge of both plant dynamics and delay characteristics. In scenarios where delays exhibit stochastic variation or significant model–plant mismatch exists, predictive performance deteriorates substantially.

More recent state-of-the-art (SOTA) methods have leveraged RL to tackle these uncertainties. Bypassing the need for such precise, model-based prediction, one category of work uses RL to autonomously tune controller gains (song WANG et al., 2007; Lee et al., 2020; Zhang et al., 2021). These methods, however, typically model the problem as a MDP, which fundamentally assumes the agent can observe the complete, current state to make an optimal decision. This core assumption is violated by observation delays, where the agent receives only outdated states. Policies trained under this false MDP assumption are thus acting on past information, leading to suboptimal and potentially unstable decisions (Huang et al., 2019; Katsikopoulos and Engelbrecht, 2003).

To address this POMDP, researchers have developed delay-aware policies, a common strategy being state augmentation (Nath et al., 2021). This approach involves stacking a history of past observations and actions to create an expanded state representation. While this can improve performance, it introduces significant challenges. As the history length required to compensate for longer delays increases, the state space expands dramatically, invoking the curse of dimensionality and rendering the learning problem computationally intractable. Furthermore, this fixed-history approach is often brittle, struggling to generalize across different or highly stochastic delay patterns. Another methodology is model-based Reinforcement Learning (MBRL). The core idea is to learn a predictive model of the environment's dynamics to compensate for latency (Barde et al., 2020). A naive application, Action-Buffer based State Prediction (ABSP), is computationally prohibitive as it requires re-simulating the entire future trajectory at every time step. To address this inefficiency, the Predictive Model Delay Correction (PMDC) framework (McCutcheon and Fallah, 2023) introduced the more efficient State-Buffer based State Prediction (SBSP). SBSP maintains a buffer of predicted future states and updates it incrementally, making it a computationally viable solution for real-time control.

However, despite this computational advance, the SOTA PMDC framework exhibits critical limitations when confronted with high-variance stochastic delays. While PMDC addresses the POMDP through state prediction, it does not explicitly mitigate the root cause of control-loop vibration. The framework relies on adaptive PD gain tuning, yet the underlying PD controller remains fundamentally vulnerable to discontinuous, jittery input signals induced by high delay variance. This signal aperiodicity introduces spurious variance into the derivative term, destabilizing the control loop regardless of gain adaptation. Furthermore, PMDC's state prediction lacks explicit velocity learning, resulting in imprecise derivative estimates that compound with delay uncertainty under high-variance conditions. Finally, validation of these advanced predictive methods has been constrained primarily to simulated environments; their robustness and practical applicability on real-world robotic hardware under high-variance network conditions remain largely undemonstrated.

1.2 Contributions

To address these persisting research gaps, this thesis proposes a generalized hybrid RL framework named DRT-RL, for teleoperation under high-variance stochastic delays. Our primary contributions are:

- Autoregressive State Estimation for Signal Continuity: We develop a LSTM-based state estimator that leverages recurrent cell states as learnable temporal memory to capture robot motion dynamics. Unlike conventional single-step predictors, our autoregressive architecture generates smooth, continuous state trajectories across the delay horizon, fundamentally eliminating the signal discontinuities that induce spurious velocity spikes and destabilize low-level controllers.
- Residual Reinforcement Learning for Robust Compensation: We propose a hybrid control architecture in which a residual RL agent learns corrective torque terms to compensate for model uncertainties and external disturbances. By decoupling the nominal model-based controller from the learned residual policy, our framework ensures stability and accuracy tracking and achieves superior robustness compared to existing SOTA methods.
- Sim-to-Real Validation under Diverse Network Conditions: We validate our framework on a physical Franka Panda manipulator across three distinct delay scenarios: (1) low delay with low variance, (2) high delay with low variance, and (3) high delay with high variance. This comprehensive evaluation demonstrates practical robustness under varying stochastic communication conditions and addresses a critical gap in the experimental validation of delay-resilient teleoperation methods.

2. PRELIMINARY AND PROBLEM SETTING

2.1 Teleoperation System

This work considers a homogeneous teleoperation system where the leader (l) and follower (f) robots share the same embodiment. We model each agent as a rigid-body mechanical system with n degrees of freedom, whose robot motion equations are described by

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{D}\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{d}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{\tau}, \quad (1)$$

where the $\boldsymbol{q} \in \mathbb{R}^n$ is the joint positions, the matrices $\boldsymbol{M}(\boldsymbol{q})$, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, and $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ are the inertia, Coriolis/centrifugal, and damping matrices, respectively. The vector function $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ denotes the gravitational torques, and the control torque vector is $\boldsymbol{\tau} \in \mathbb{R}^n$.The external disturbances and model uncertainties are considered in the model as a vector function $\boldsymbol{d}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$.

2.2 Control Objective

The primary control objective is to achieve high-fidelity motion synchronization between the leader and follower robots. Let $\boldsymbol{q}_l(t) \in \mathbb{R}^n$ and $\boldsymbol{q}_f(t) \in \mathbb{R}^n$ denote the vector of joint angular positions for the leader robot and the follower robot at time t, respectively. Since the human operator only controls the end-effector position, the control signal in Fig. 1 is considered as task-space motion. Here, we define the $\boldsymbol{x}_l(t) = \boldsymbol{f}_k(\boldsymbol{q}_l(t)) \in \mathbb{R}^m$ as the end-effector positions of the local robot, where $\boldsymbol{f}_k : \mathbb{R}^n \to \mathbb{R}^m$ is the forward kinematics mapping.

Therefore, we formally define the joint tracking error $\boldsymbol{e}(t)$ as the instantaneous deviation between the leader's and follower's configurations:

$$\boldsymbol{e}(t) = \boldsymbol{f}_k^{-1}(\boldsymbol{x}_l(t)) - \boldsymbol{q}_f(t). \tag{2}$$

The goal is to design a control policy that minimizes the magnitude of the error $\boldsymbol{e}(t)$ under stochastic delays, including state delay $\omega_s \in \mathbb{R}_{>0}$, action delay $\omega_a \in \mathbb{R}_{>0}$, and observation delay $\omega_o \in \mathbb{R}_{>0}$.

2.3 State Estimator

Due to the inevitable communication delays between the leader and the follower, obtaining the real-time positions is infeasible. Consequently, the true leader state $\boldsymbol{q}_l(t), \dot{\boldsymbol{q}}_l(t)$ is unobservable at the current time step. The control agent only has access to the delayed states: $\boldsymbol{q}_l(t - \omega_s^t)$, $\dot{\boldsymbol{q}}_l(t - \omega_s^t)$. However, using this delayed signal in a feedback loop would result in instability. To mitigate this latency and approximate the ground-truth state of the leader robot, we employ a LSTM-based state estimator to reconstruct the real-time leader robot state: $\hat{\boldsymbol{q}}_l(t)$ and $\hat{\dot{\boldsymbol{q}}}_l(t)$, which denote the predicted leader states provided by the estimator in Section 3.1.

2.4 Controller Design

To track the desired motion of the local leader, a nominal controller based on the computed torque control method is proposed as follows

$$\boldsymbol{\tau}_{\text{nom}} = \boldsymbol{M}(\boldsymbol{q}_f)\ddot{\boldsymbol{q}}_{\text{ref}} + \boldsymbol{C}(\boldsymbol{q}_f, \dot{\boldsymbol{q}}_f)\dot{\boldsymbol{q}}_f + \boldsymbol{D}\dot{\boldsymbol{q}}_f + \boldsymbol{g}(\boldsymbol{q}_f), \tag{3a}$$

$$\ddot{\boldsymbol{q}}_{\text{ref}} = \hat{\ddot{\boldsymbol{q}}}_l + \boldsymbol{K}_d\left(\hat{\dot{\boldsymbol{q}}}_l - \dot{\boldsymbol{q}}_f\right) + \boldsymbol{K}_p\left(\hat{\boldsymbol{q}}_l - \boldsymbol{q}_f\right), \tag{3b}$$

where $\boldsymbol{K}_p, \boldsymbol{K}_d \in \mathbb{R}^{n \times n}$ are the positive-definite proportional and derivative gain matrices.

Although the nominal controller provides a robust theoretical basis for trajectory tracking by compensating for known rigid-body dynamics, its efficacy is inherently constrained by unmodeled dynamics and state estimation errors. Inspired by (Johannink et al., 2019), we train a reinforcement learning agent to learn the residual torque compensation term $\boldsymbol{\tau}_{\text{rl}}$. Therefore, the final control input $\boldsymbol{\tau}_{\text{cmd}}$ applied to the follower robot is formulated as the summation of the nominal baseline and the learned residual, i.e.,

$$\boldsymbol{\tau}_{\text{cmd}} = \boldsymbol{\tau}_{\text{nom}} + \boldsymbol{\tau}_{\text{rl}}, \tag{4}$$

where the $\boldsymbol{\tau}_{\text{rl}}$ is introduced in Section 3.2.

3. METHODOLOGY

3.1 LSTM-based State Estimator

A critical issue in teleoperation under stochastic delays is the discrete jumps in the feedback signal from the local robot. Such discontinuities result in unbounded derivative estimates, causing the controller to generate aggressive torque transients that destabilize the robot system. Thus, the estimator must guarantee smooth trajectory reconstruction in addition to minimizing state error.

To this end, we leverage the LSTM network to address this issue. Its recurrent structure provides the temporal memory necessary to resolve the partially observable states induced by stochastic delays. Furthermore, the LSTM inherently enforces kinematic continuity, effectively eliminating signal discontinuities that would otherwise compromise controller stability.

Inertial Memory Learning Let set $\mathcal{Z}(t)$ denote the sequence of the N^t most recent delayed states available at time t:

$$\mathcal{Z}(t) = \{\mathbf{s}_i^t\}_{i=1,\ldots,N^t}, \tag{5}$$

where the element $\mathbf{s}_i^t = [\boldsymbol{q}_l(t)^\top, \dot{\boldsymbol{q}}_l(t)^\top, \omega_s^t]^\top$ represents the concatenation of the joint positions $\boldsymbol{q}_l(t)$, velocities $\dot{\boldsymbol{q}}_l(t)$ and the state delay magnitude ω_s^t.

The LSTM processes the input sequence to update its internal cell state $\mathbf{c}_k$. We interpret this state as a learnable inertial memory derived from the underlying physical laws of robot motion. Consider the generalized discrete-time evolution of the robot. The transition from state $\mathbf{s}_{k-1}$ to $\mathbf{s}_k$ is governed by a non-linear function ℓ representing the rigid body dynamics and integration scheme:

$$\mathbf{s}_k^t = \mathbf{s}_{k-1}^t + \ell(\mathbf{s}_k^t)\Delta t \approx \mathbf{s}_{k-1}^t + \Delta\mathbf{s}_{dyn}, \tag{6}$$

where $\Delta\mathbf{s}_{dyn}$ represents the complex state change induced by velocity, acceleration, Coriolis forces, and gravity over the interval Δt.

The LSTM cell state update mechanism shares a structural isomorphism with the accumulation process. The cell update is defined as

$$\mathbf{c}_k = (\mathbf{f}_k \odot \mathbf{c}_{k-1}) + (\mathbf{i}_k \odot \tilde{\mathbf{c}}_k), \tag{7}$$

where $\mathbf{f}_k$ is the forget gate, $\mathbf{i}_k$ is the input gate, and $\tilde{\mathbf{c}}_k$ is the candidate cell state. By learning to maintain $\mathbf{f}k \approx \mathbf{1}$ during training, the LSTM effectively suppresses the forgetting mechanism, ensuring that the prior state $\mathbf{c}_{k-1}$ is preserved without decay. This preservation allows the cell to function as a temporal accumulator, structurally isomorphic to the system state $\mathbf{s}_{k-1}$ in Eq. (6). Specifically, the cell state $\mathbf{c}_{k-1}$ acts as the accumulator maintaining historical information, while the gated update term $\mathbf{i}_k \odot \tilde{\mathbf{c}}_k$ approximates the nonlinear dynamics term $\Delta\mathbf{s}_{dyn}$. Therefore, Eq. (7) functionally replicates the accumulation structure of the physical system, allowing the cell state to act as a learnable momentum term that captures the inertial dynamics of the robot.

Autoregressive State Estimation To generate dense, continuous state estimates over extended delay periods, we employ an autoregressive prediction strategy. This mechanism leverages the learned inertial memory embedded in the LSTM cell state to simulate the robot's forward motion from the most recent anchor observation $\hat{\mathbf{s}}_k^{(t-\omega_s^t)}$ up to the current time t. The prediction horizon is determined by the maximum observed system delay $\omega_s^{\max}$. During the delay interval, the estimator performs iterative recursive rollouts at high frequency. In this autoregressive phase, the model uses its own previous output as input for the subsequent prediction step:

$$\hat{\mathbf{s}}_k^{(t-\omega_s^t)} = \text{LSTM}(\hat{\mathbf{s}}_{k-1}^{(t-\omega_s^t)}, \hat{\mathbf{s}}_{k-2}^{(t-\omega_s^t)}, \dots), \tag{8}$$

where $\hat{\mathbf{s}}_{k-1}^{(t-\omega_s^t)}$ denotes the predicted state from the previous iteration. This recursive feedback loop propagates the learned dynamics forward in time without requiring new observations, generating a smooth, kinematically consistent trajectory that connects the delayed state $\hat{\mathbf{s}}_k^{(t-\omega_s^t)}$ to the current estimated state $\hat{\boldsymbol{q}}_l(t)$ and velocity $\hat{\dot{\boldsymbol{q}}}_l(t)$.

3.2 Residual RL Control

Observation Space The presence of stochastic time-varying delay ω_s^t inherently renders the environment partially observable, as the agent cannot access the immediate true state of the leader robot. This effectively transforms the standard MDP into a POMDP. A naive approach is State Augmentation (stacking history states into the agent observation space) (Nath et al., 2021), however, this leads to the curse of dimensionality as the required augmentation history length N grows with delay magnitude. To avoid this computational intractability, we adopt a Model-Based Reinforcement Learning approach. We rely on the LSTM estimator (Section 3.1) to estimate real-time states $\hat{\boldsymbol{q}}_l(t)$ and $\hat{\dot{\boldsymbol{q}}}_l(t)$. This effectively transforms the POMDP into a standard MDP process for the control policy. The RL agent receives an augmented observation vector $\boldsymbol{o}_t \in \mathbb{R}^{d_{obs}}$ comprising the follower robot state $[\boldsymbol{q}_f, \dot{\boldsymbol{q}}_f]^T$, the LSTM-predicted leader state $[\hat{\boldsymbol{q}}_l, \hat{\dot{\boldsymbol{q}}}_l]^T$, the tracking errors $[\boldsymbol{e}_q, \boldsymbol{e}_{\dot{q}}]^T$, and the follower state history $\{\boldsymbol{q}_f(t-k), \dot{\boldsymbol{q}}_f(t-k)\}_{k=1}^N$.

SAC Algorithm In this work, we employ Soft Actor-Critic (SAC) for policy optimization due to its sample efficiency and stability in continuous control tasks.

Following the standard objective $J(\theta)$ of RL policy is defined as the expectation of the cumulative reward $R(\tau) = \sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)$ over the time horizon as

$$J(\theta) = \mathbb{E}_{\tau \sim P_{\pi_\theta}}[R(\tau)]. \tag{9}$$

To optimize the parameters θ, the gradient $\nabla_\theta J(\theta)$ is derived via the Policy Gradient Theorem:

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{\tau \sim P_{\pi_\theta}}[\nabla_\theta \log P_{\pi_\theta}(\tau) R(\tau)] \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \hat{Q}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}),\end{aligned} \tag{10}$$

where N denotes the batch size. To evaluate the policy, we need to utilize standard Value Function $V^\pi(\mathbf{s})$ and Action-Value Function $Q^\pi(\mathbf{s}, \mathbf{a})$:

$$V^\pi(\mathbf{s}) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid \mathbf{s}_t = \mathbf{s} \right], \tag{11}$$

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a} \right]. \tag{12}$$

These functions serve as the objective performance metric, allowing the algorithm to distinguish between optimal and suboptimal behaviors by quantifying the expected long-term return of specific states and actions. In SAC, the Action-Value function serves as the Critic, providing the gradient signal necessary to update the policy.

SAC augments the standard objective with an entropy regularization term $\mathcal{H}(\pi(\cdot|\mathbf{s}_t))$ to encourage exploration.

$$J_{\text{SAC}}(\theta) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T} \gamma^t \left(r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot|\mathbf{s}_t)) \right) \right], \tag{13}$$

where α is the temperature parameter. The entropy term is particularly critical, by maximizing the policy entropy, the agent is encouraged to explore all optimal modes of behavior rather than collapsing into a single deterministic policy, preventing overfitting to specific delay patterns and thereby enhancing robustness against the uncertainty.

Multi-Objective Reward Design The control policy optimization is driven by a composite reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, formulated to balance asymptotic tracking stability with control effort minimization and strict safety constraints. We employ a linear scalarization of these objectives, defined at each time step t as:

$$r_t(\mathbf{s}_t, \mathbf{a}_t) = r_{\text{track}} + r_{\text{reg}} + r_{\text{safe}}. \tag{14}$$

The **tracking fidelity** term r_{track} penalizes the weighted squared Euclidean distance between the follower state and the estimated leader intent in the joint space topology:

$$r_{\text{track}} = -\lambda_p \|\hat{\mathbf{q}}_l - \mathbf{q}_f\|_2^2 - \lambda_v \|\hat{\dot{\mathbf{q}}}_l - \dot{\mathbf{q}}_f\|_2^2, \tag{15}$$

where $\lambda_p, \lambda_v \in \mathbb{R}_{>0}$ are weighting coefficients governing the trade-off between position accuracy and velocity synchronization.

The **action regularization** term r_{safe} discourages large residual torques:

$$r_{\text{reg}} = -\lambda_a \frac{1}{n} \sum_{i=1}^{n} (a_{t,i})^2, \tag{16}$$

where $a_{t,i}$ is the residual torque applied to joint i.

The **safety penalty** term r_{safe} enforces hard constraints on the agent's behavior. we define the r_{safe} as the sum of individual penalty terms:

$$r_{\text{safe}} = P_{\text{joint}} + P_{\text{track}} + P_{\text{est}}, \tag{17}$$

where each penalty is activated upon violation of its corresponding conditions:

$$\begin{cases} P_{\text{joint}} = -100, & \text{if } \exists i, q_{f,i} \notin [q_{\min}^i, q_{\max}^i], \\ P_{\text{track}} = -100, & \text{if } \|\boldsymbol{e}_{\text{track}}\| > 0.5, \\ P_{\text{est}} = -100, & \text{if } \|\boldsymbol{e}_{\text{pred}}\| > 0.3, \end{cases} . \tag{18}$$

The $[q_{\min}^i, q_{\max}^i]$ denotes the physical limits of the i-th joint. The tracking error e_{track} is defined as the L2 norm between leader joint state $\boldsymbol{q}_l(k)$ and follower joint state $\boldsymbol{q}_f(k)$ at step k:

$$e_{\text{track}}(k) = \|\boldsymbol{q}_l(k) - \boldsymbol{q}_f(k)\|_2. \tag{19}$$

Similarly, the estimation error e_{est} is defined as the L2 norm between the LSTM-predicted leader state $\hat{\boldsymbol{q}}_l(k)$ and the true leader state $\boldsymbol{q}_l(k)$ at step k:

$$e_{\text{est}}(k) = \|\hat{\boldsymbol{q}}_l(k) - \boldsymbol{q}_l(k)\|_2. \tag{20}$$

4. EXPERIMENTAL RESULTS

4.1 Experimental Setup

All experiments employ a seven-degree-of-freedom Franka Panda robot arm integrated with ROS2 Humble middleware. We systematically compare our method against two baseline approaches representing the progression of delay compensation techniques:

- **PMDC (SOTA RL-based method)** (McCutcheon and Fallah, 2023): Utilizes a learned dynamics model and State-Buffer based State Prediction (SBSP) to explicitly estimate the real-time state from delayed observations.
- **Vanilla PD (Baseline method)**: A standard inverse dynamics controller operating directly on delayed observations without prediction or compensation, serving as the performance lower bound.

To quantify the tracking performance, a desired reference trajectory for the end-effector is commanded to the leader robot, which the follower must replicate under delay. The reference trajectory in the task space is defined as

$$\boldsymbol{x}_l^{\text{ref}}(k) = \begin{bmatrix} 0.2\sin(3k) \\ 0.2\sin(4k + \frac{\pi}{2}) \\ 0.02\sin(k) \end{bmatrix}. \tag{21}$$

Moreover, the teleoperation system architecture deploys the RL agent co-located with the follower robot, resulting in constant action delay $\omega_o^t = \omega_a^t = 50$ ms between the agent's torque commands and the follower robot's execution for all t. Stochasticity arises from the network communication between the leader robot and the RL agent, introducing variable observation delay $\omega_s^t \sim \mathcal{U}(\omega_{\min}, \omega_{\max})$. And the sampling period is set as $\Delta t = 4$ ms. We evaluate three delay configurations shown in Fig. 2 as follows:

- **Low delay with low variance**: $d_{\text{obs}} \sim \mathcal{U}(120, 160)$ ms, total: 170–210 ms
- **High delay with low variance**: $d_{\text{obs}} \sim \mathcal{U}(200, 240)$ ms, total: 250–290 ms
- **High delay with high variance**: $d_{\text{obs}} \sim \mathcal{U}(40, 240)$ ms, total: 90–290 ms

The first three configurations enable direct comparison with PMDC, while the high-variance profile (200 ms variance) validates robustness under extreme network conditions where prior methods fail.

4.2 Result Analysis

We evaluate the tracking performance by the norm of the Cartesian tracking error $\|\boldsymbol{e}(t)\|$ between the leader and follower end-effectors. In Fig. 3, we compare results across three delay configurations. Under low-delay, low-variance conditions (170–210 ms), DRT-RL achieves the lowest tracking error, outperforming both PMDC and the Vanilla PD controller, which exhibits substantial oscillations (peaks ≈ 0.4 m); this demonstrates the superiority

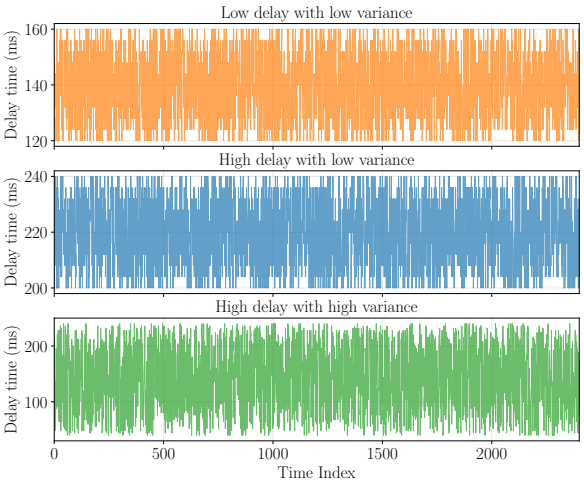


Fig. 2. State delay patterns in teleoperation tasks. Low delay with low variance (top), high delay with low variance (middle), high delay with high variance (bottom).

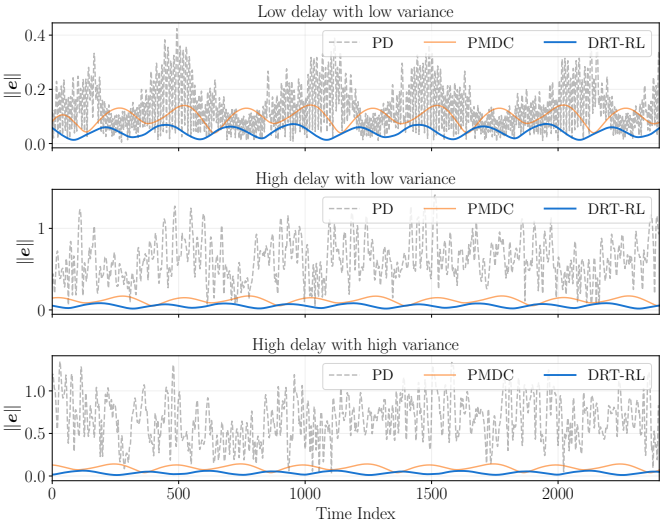
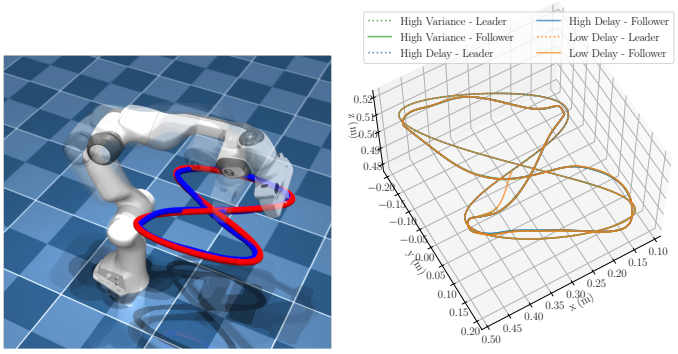


Fig. 3. Tracking error $\|\boldsymbol{e}\|$ comparison between DRT and PD methods under different network delay conditions.

of LSTM-based autoregressive estimation over feedforward networks used in PMDC. As delay magnitude increases to 250–290 ms, DRT-RL maintains stable, low-error performance, whereas PMDC deteriorates and the PD controller fails completely with error peaks exceeding 1 m. The most pronounced differentiation emerges under the challenging high-delay, high-variance scenario (90–290 ms): the PD controller exhibits severe instability with consistent oscillation, and PMDC degrades significantly due to the inability of its memoryless feedforward network to adapt to variable delays from fixed-length inputs.

As illustrated in Fig. 4, the DRT-RL method demonstrates near-perfect trajectory tracking in the MuJoCo simulation environment. In contrast, physical experiments reveal an increased tracking error, highlighting the impact of unmodeled dynamics and real-world sensor noise.



(a) MuJoCo environment. (b) Real experiment.

Fig. 4. Visualization of spatial trajectories using DRT-RL in a high-delay, high-variance scenario. (a) The red line is the local leader, the blue line denotes the remote follower. (b) The dashed line is the leader, while the solid line is the follower.

5. DISCUSSION

While our framework employs an LSTM-based state estimator, recent advances in sequence modeling have established powerful attention-based architectures, notably the Transformer Vaswani et al. (2017), which could theoretically be applied to this prediction task. However, several practical constraints make LSTMs more suitable for real-time teleoperation. Because Transformer-based architectures scale quadratically with sequence length ($\mathcal{O}(N^2)$) and large data requirements make them impractical for millisecond-scale control loops. In contrast, the LSTM achieves constant-time inference ($\mathcal{O}(1)$), making it a suitable choice for real-time teleoperation.

A critical challenge in learning-based teleoperation remains the sim-to-real gap due to unmodeled dynamics or sensor noise. Future research should prioritize online adaptation mechanisms capable of updating the estimator and policy in real-time as network conditions fluctuate. Additionally, extending the framework to bilateral teleoperation scenarios incorporating haptic feedback, and exploring hybrid neural architectures such as attention-augmented LSTMs or state-space models.

6. CONCLUSION

This paper introduces DRT-RL, a hybrid learning-based control framework for robust teleoperation under high-variance stochastic delays. At its core is an autoregressive state estimator that leverages LSTM-based temporal memory to resolve delay-induced partial observability, while a coupled residual reinforcement learning policy compensates for unmodeled dynamics to ensure stable and precise operation. Experimental validation on a physical Franka Panda demonstrates that our framework significantly outperforms SOTA baselines, achieving superior tracking performance and stability.

REFERENCES

Anderson, R.J. and Spong, M.W. (1989). Bilateral Control of Teleoperators with Time Delay. *IEEE Transactions on Automatic Control*, 34(5), 494–501. doi: 10.1109/9.24239.

Barde, P., Roy, J., de La Saulece, É., Calauzènes, C., and Moinard, V. (2020). At human speed: Deep reinforcement learning with action delay. In *International Conference on Learning Representations*.

Choi, P.J., Oskouian, R.J., and Tubbs, R.S. (2018). Telesurgery: Past, Present, and Future. *Cureus*, 10(5).

Huang, J., Chen, J., and Sun, C. (2019). Reinforcement learning in robotic teleoperation with time delay: A survey. *Annual Reviews in Control*, 48, 189–203. doi: 10.1016/j.arcontrol.2019.06.005.

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J.A., Solowjow, E., and Levine, S. (2019). Residual Reinforcement Learning for Robot Control. In *2019 International Conference on Robotics and Automation (ICRA)*, 6023–6029. doi: 10.1109/ICRA.2019.8794127.

Katsikopoulos, K.V. and Engelbrecht, S.E. (2003). Markov decision processes with delays and asynchronous cost collection. *IEEE Transactions on Automatic Control*, 48(4), 568–574. doi:10.1109/TAC.2003.809800.

Lee, D., Lee, S.J., and Yim, S.C. (2020). Reinforcement Learning-Based Adaptive PID Controller for DPS. *Ocean Engineering*, 216, 108053.

Manupati, V.K., Krishnan, M.G., Varela, M.L.R., and Machado, J. (2017). Telefacturing based distributed manufacturing environment for optimal manufacturing service by enhancing the interoperability in the hubs. *Journal of Engineering*, 2017, 1–14. doi: 10.1155/2017/9305989.

McCutcheon, L. and Fallah, S. (2023). Adaptive PD Control Using Deep Reinforcement Learning for Local-Remote Teleoperation with Stochastic Time Delays. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7046–7053. doi: 10.1109/IROS55552.2023.10341953.

Mujčić, E. and Oračević, A. (2019). Internet Based Teleoperations With Using Wave Variables. *Not specified in paper*. Appears to be a conference or journal publication.

Nath, S., Baranwal, M., and Khadilkar, H. (2021). Revisiting State Augmentation Methods for Reinforcement Learning with Stochastic Delays. In *Proceedings of the Association for Computing Machinery*, 1346–1355. doi: 10.1145/3459637.3482386.

Niemeyer, G. and Slotine, J.J. (1991). Stable Adaptive Teleoperation. *IEEE Journal of Oceanic Engineering*, 16(1), 1619–1625.

Ruoff, C. (1994). *Teleoperation and Robotics in Space (Ingenieria Mecanica y Maquinaria)*. American Institute of Aeronautics & Astronautics.

Smith, O.J.M. (1957). Closed Control of Loops with Dead Time. *Chemical Engineering Progress*, 53(5), 217–219.

song WANG, X., hu CHENG, Y., and SUN, W. (2007). A proposal of adaptive pid controller based on reinforcement learning. volume 17, 40–44. Elsevier. doi: 10.1016/S1006-1266(07)60009-1.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is All you Need. *Advances in neural information processing systems*, 30.

Zhang, H., Yang, Y., and Jiang, Y. (2021). Reinforcement learning-based control with communication delays: Theory and applications. *IEEE Transactions on Cybernetics*, 51(9), 4368–4381. doi:10.1109/TCYB.2020.2988820.