MATER: Mutually Aware Framework for Teleoperated-robot with Extended Reality

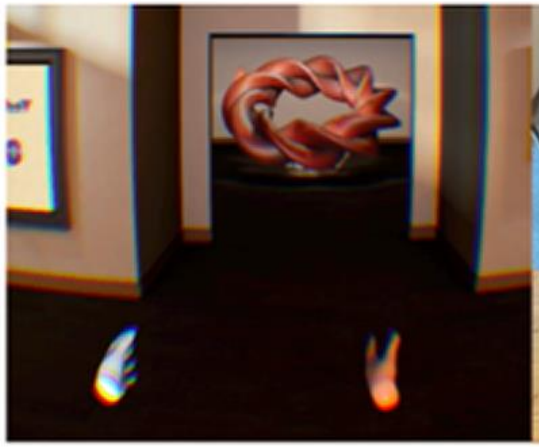
Ziliang Zhang, Cong Liu, Hyoseung Kim
University of California, Riverside

XR Teleoperation

What is **XR Teleoperation**?

- User operates a remote robot with an Extended Reality (XR) device
- User views robot state in a 3D space and controls with direct user motion – like Avatar!

User view



Remote robot

XR Teleoperation **use cases**

- Operate robot in hazardous environment
- Remote control, surgery, automation

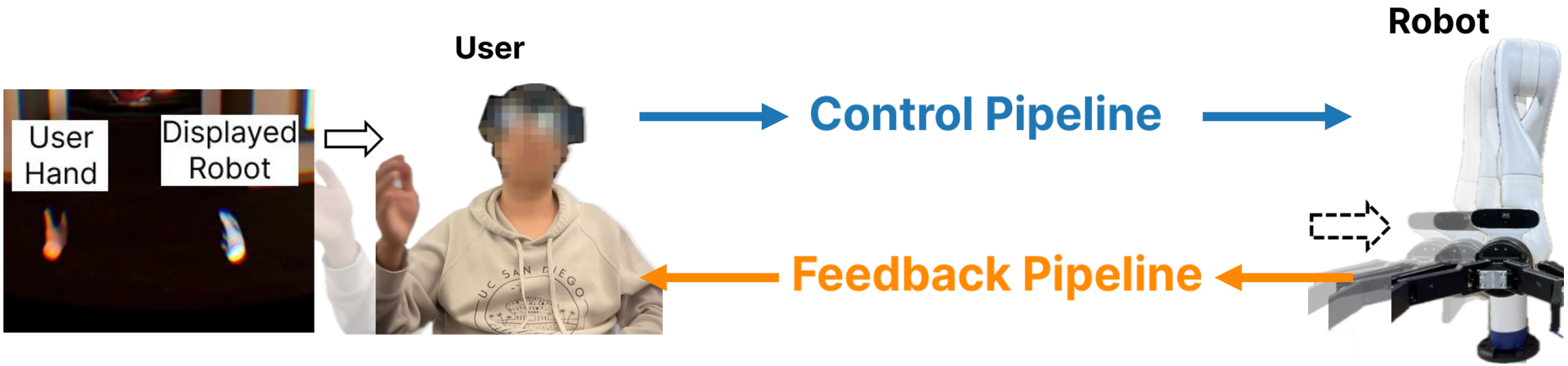
Fire fighter uses XR Teleoperation*

Da Vinci Robotic Surgery**

* ChatGPT 5.5, Fire-fighter operating robot with XR (Generated)
** OHC, What is Da Vinci Robotic Surgery https://ohcare.com/what-is-da-vinci-robotic-surgery/

XR Teleoperation pipeline

- **Control Pipeline**: User motion is sampled by XR sensors and transmitted to robot for control
- **Feedback Pipeline**: Robot feeds back real-time pose and environment data for user to view and conduct subsequent motion



XR Teleoperation Challenges

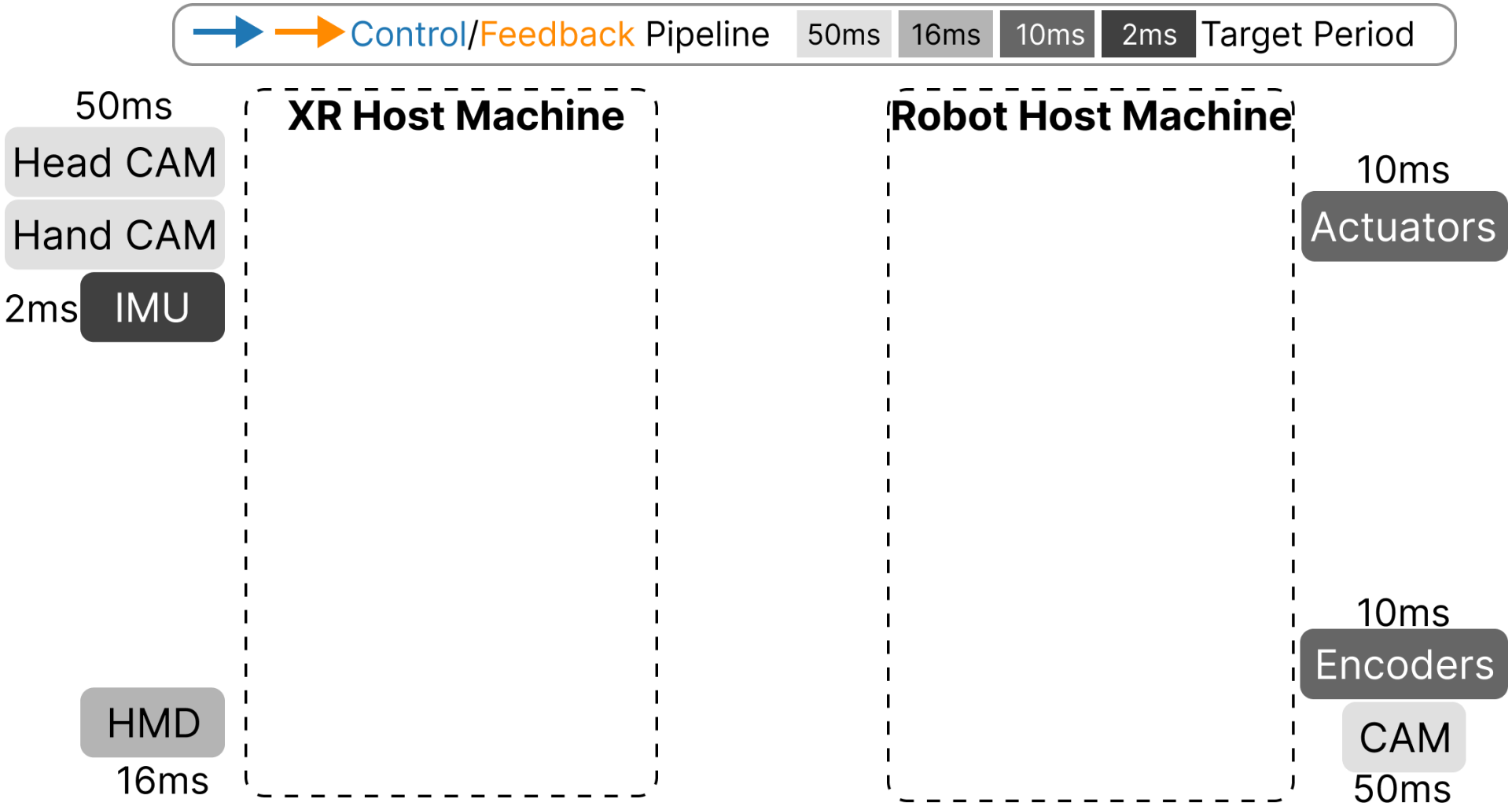
- **Communication Latency**: User always experience one round-trip latency
- **Teleoperation Accuracy**: User always observes robot lags behind and has to stop constantly for robot to catch up – **teleoperation error**

XR Teleoperation Challenges

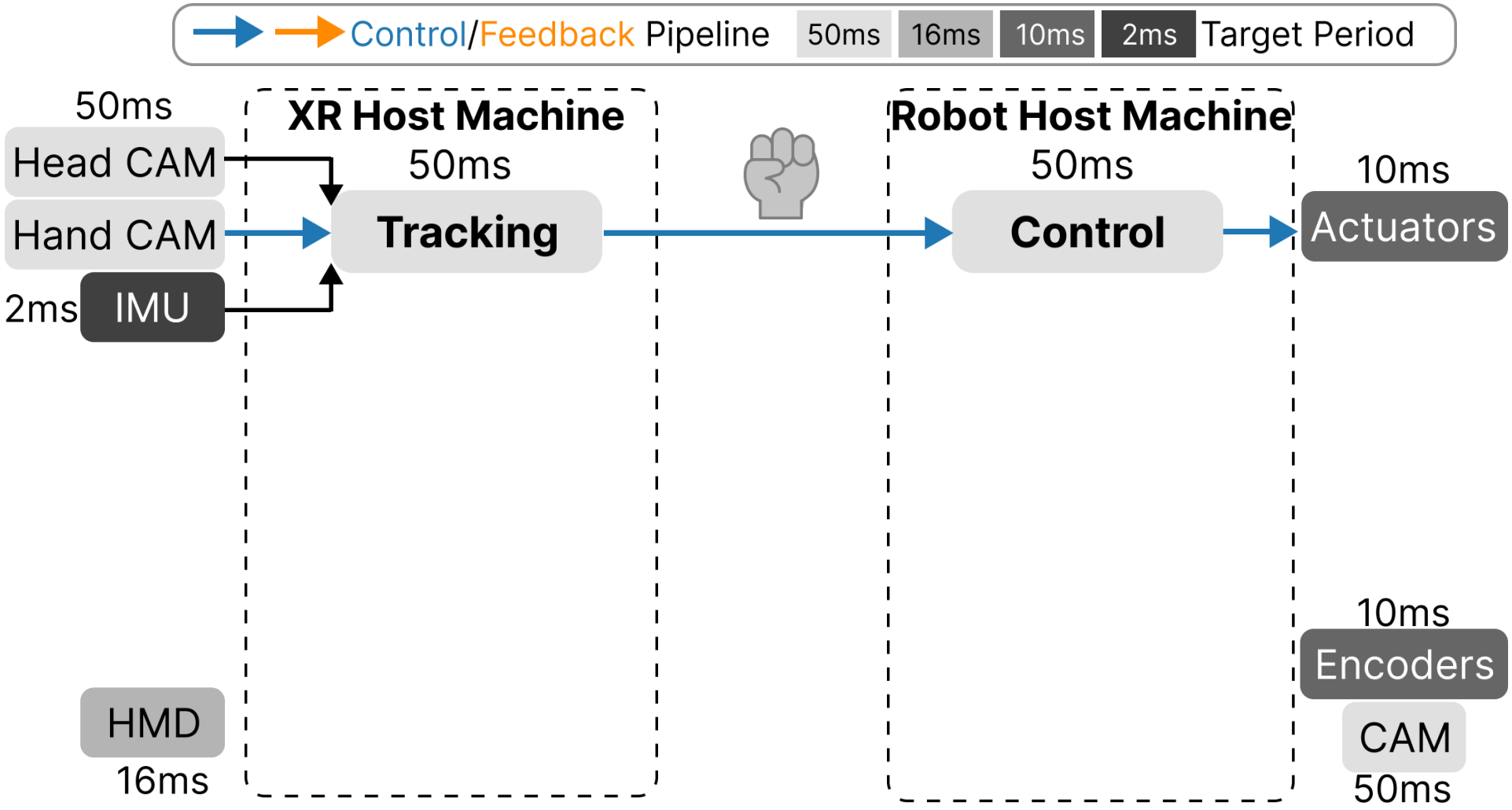
- **Communication Latency**: User always experience one round-trip latency
- **Teleoperation Accuracy**: User always observes robot lags behind and has to stop constantly for robot to catch up – **teleoperation error**

Let's figure out why in the system!

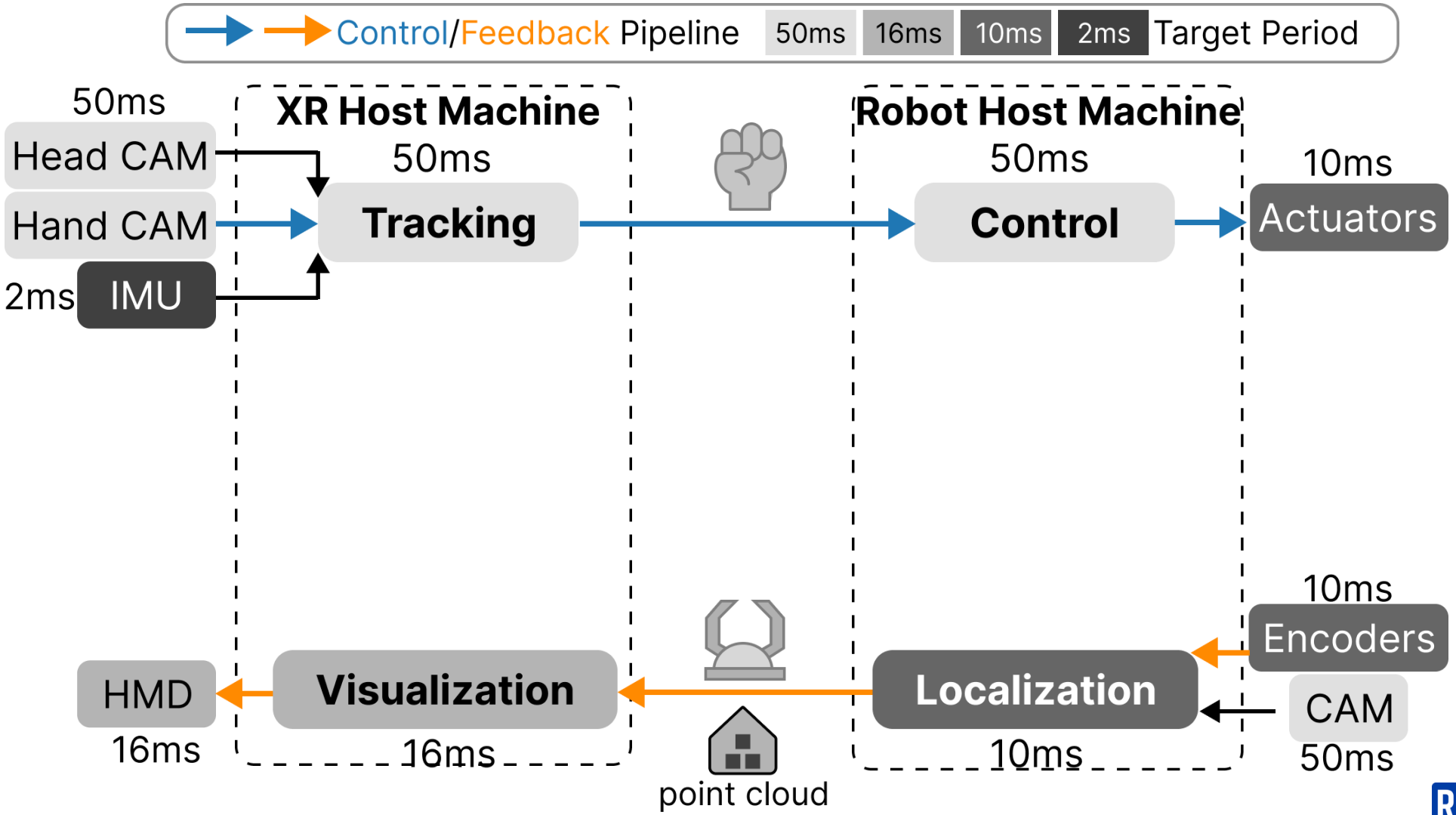
XR Teleoperation System Framework



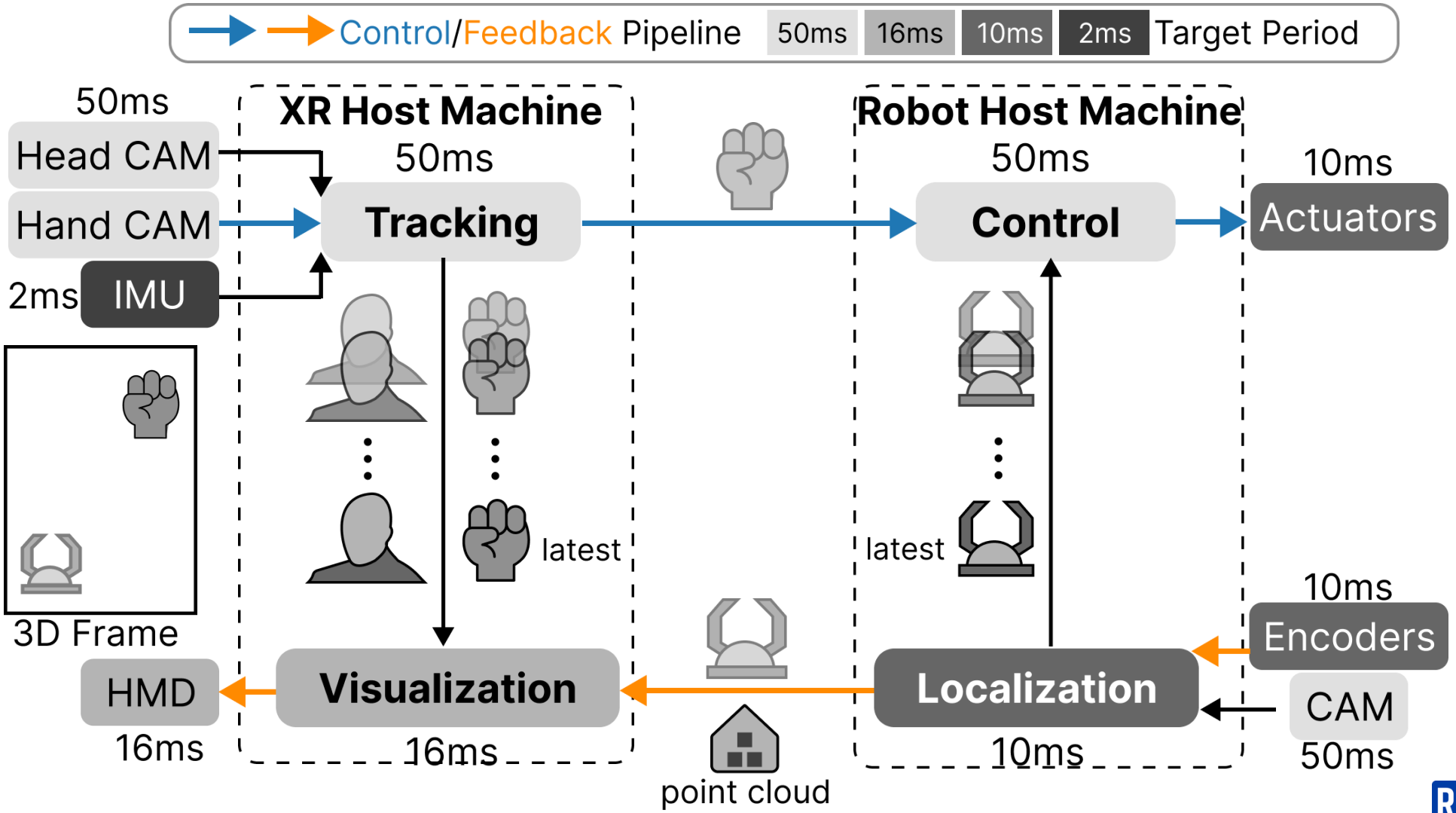
XR Teleoperation System Framework



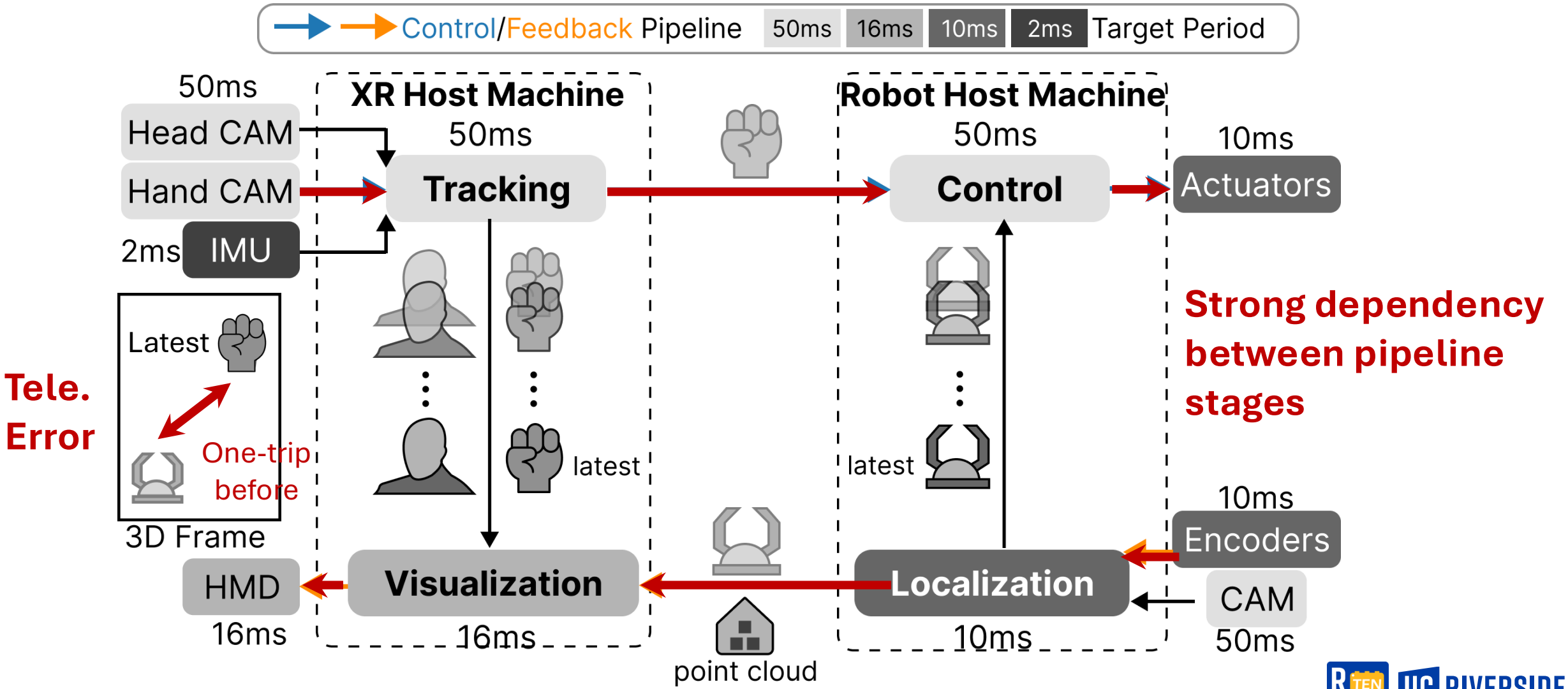
XR Teleoperation System Framework



XR Teleoperation System Framework



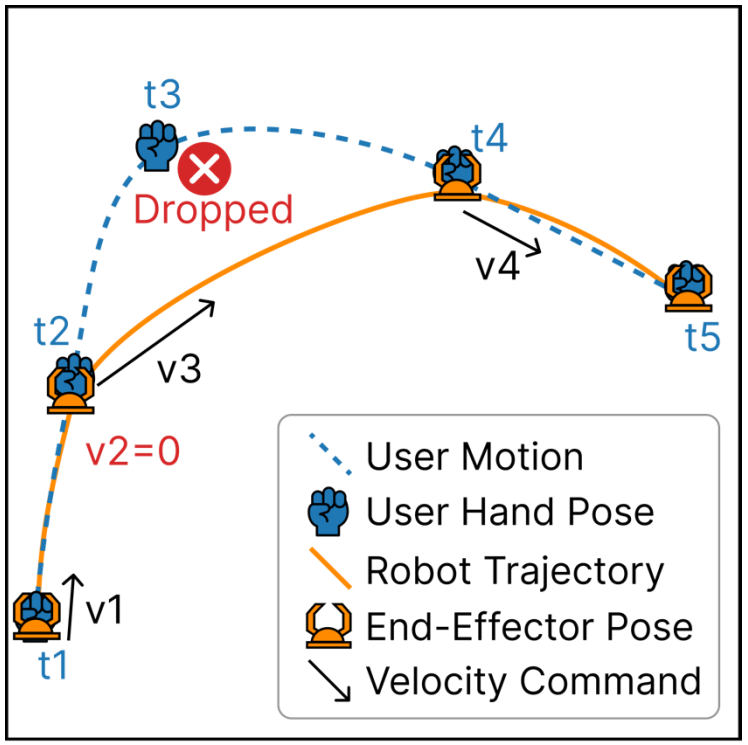
XR Teleoperation System Framework



However, it is hard to solve the problem…

C1: Network-dependent Teleoperation Performance

- Pose Drop: robot skips t3 and produces irrecoverable deviation

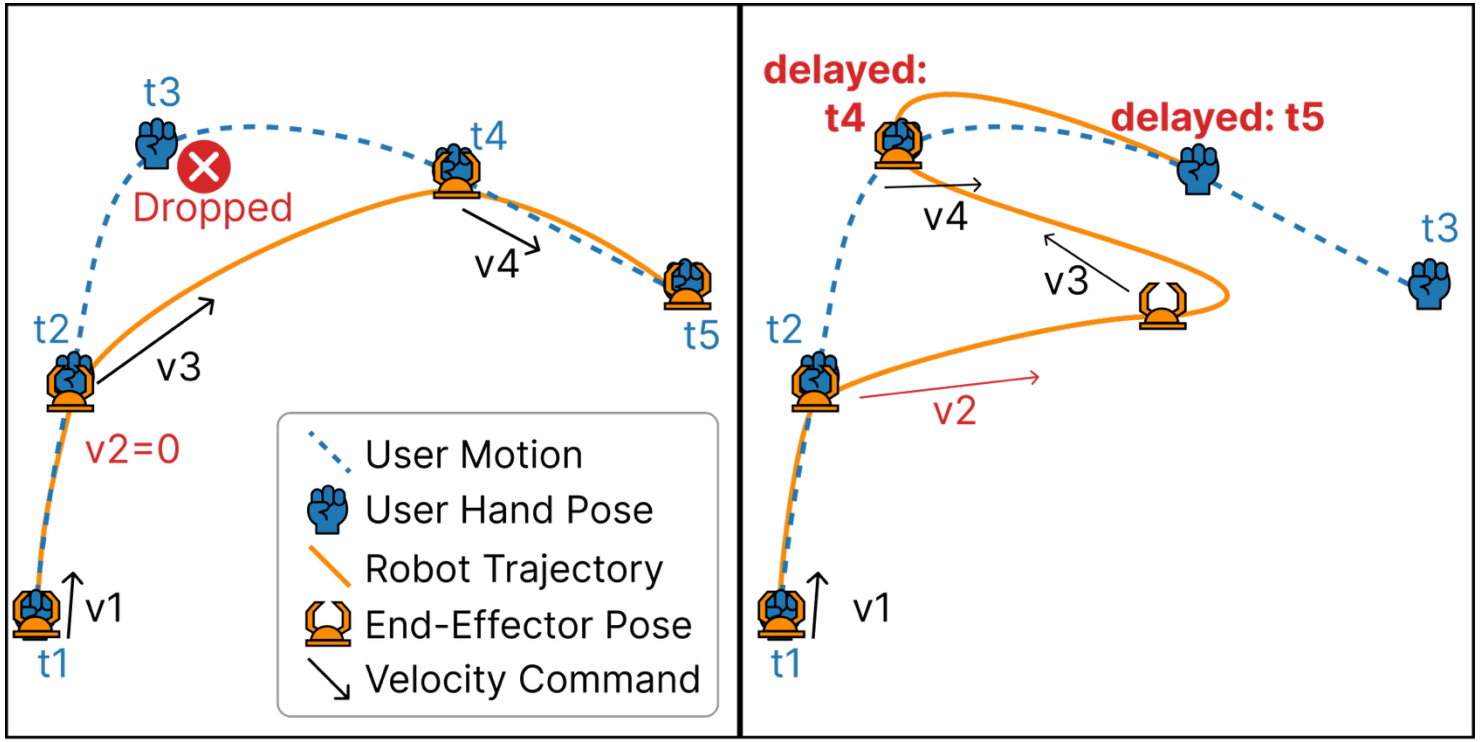


(A) C1: Pose drop effect

However, it is hard to solve the problem…

C1: Network-dependent Teleoperation Performance

- Pose Drop: robot skips t3 and produces irrecoverable deviation
- Out-of-order arrival: robot prematurely executes t3 before receiving the correct t4



(A) C1: Pose drop effect

(B) C1: Out-of-order effect

However, it is hard to solve the problem…

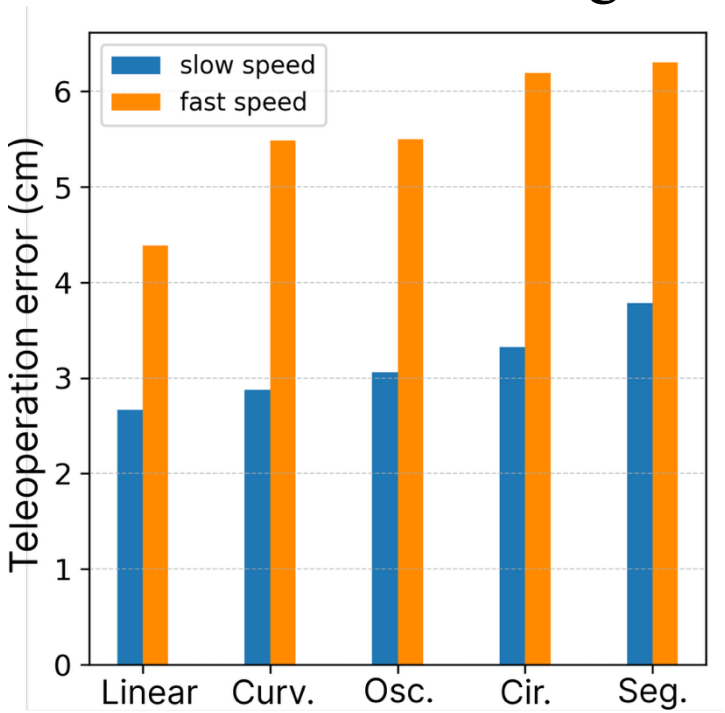
C2: Motion-driven Teleoperation Accuracy

- Higher speeds increase Tele. Error: distance between the displayed and remote robot naturally grows when user speed increases

However, it is hard to solve the problem…

C2: Motion-driven Teleoperation Accuracy

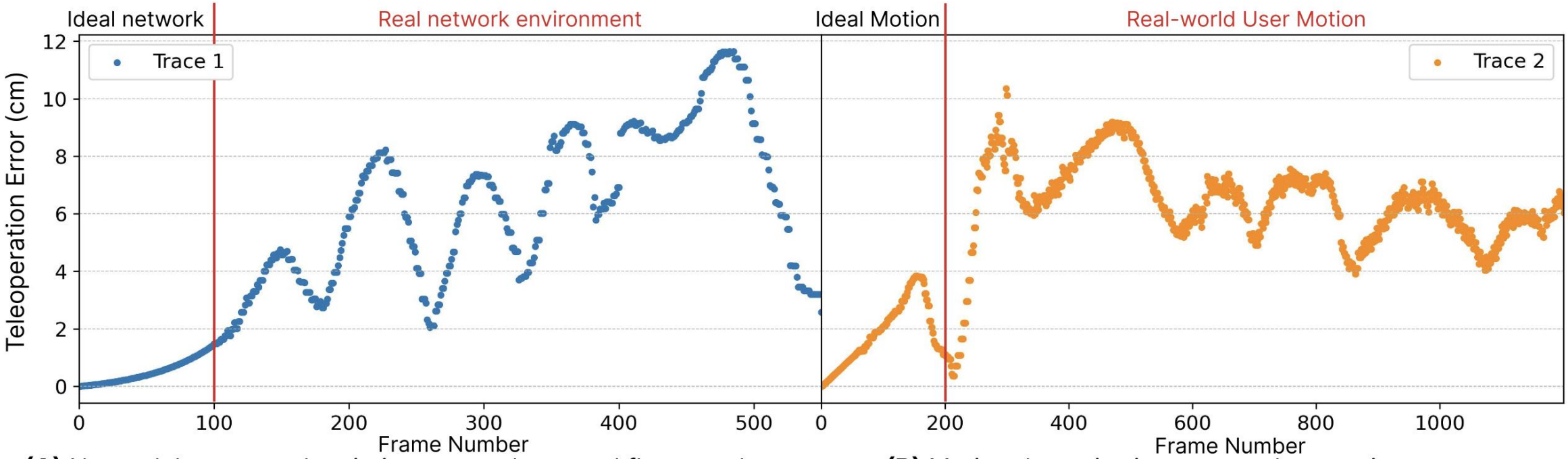
- Higher speeds increase Tele. Error: distance between the displayed and remote robot naturally grows when user speed increases
- High-curvature motions increase Tele. Error: higher-curvature motions amplify the impact of delay



(C) C2: Motion effect on MATER

Therefore, in real-world deployment

- Error increases to over 10 cm due to real network fluctuation and motion dynamics



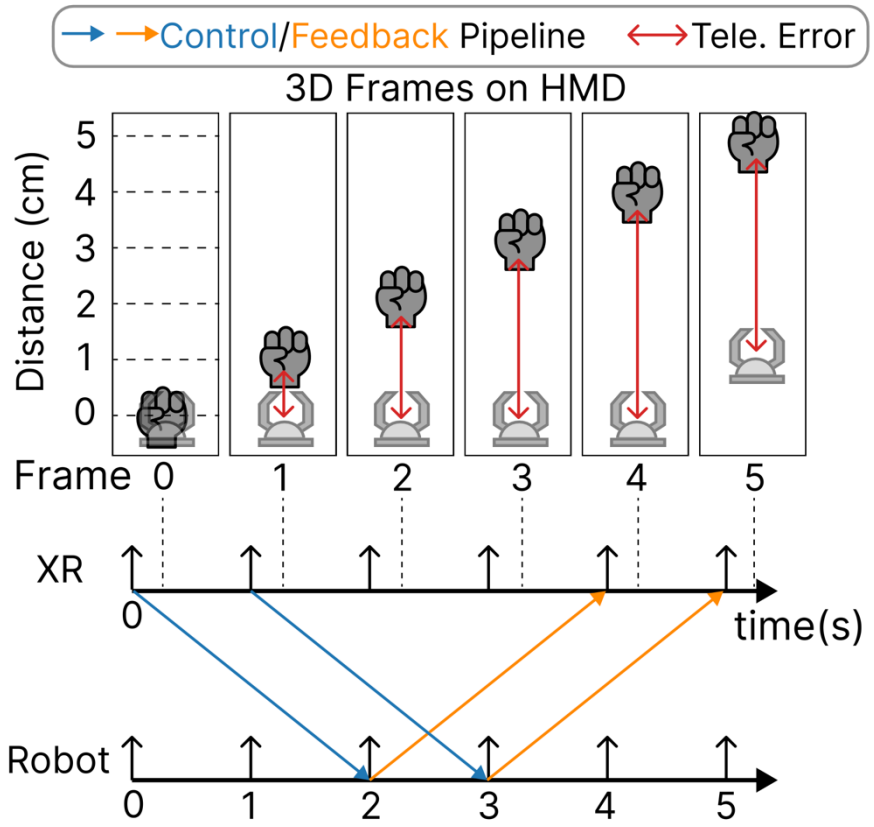
(A) Network latency and variation causes large and fluctuated error

(B) Motion dynamics increases teleoperation error

Our Solution: MATER

Root cause: Synchronous execution architecture

- Both XR and robot must wait for network updates to maintain state consistency
  → Vulnerable to network latency (C1) and motion dynamics (C2)



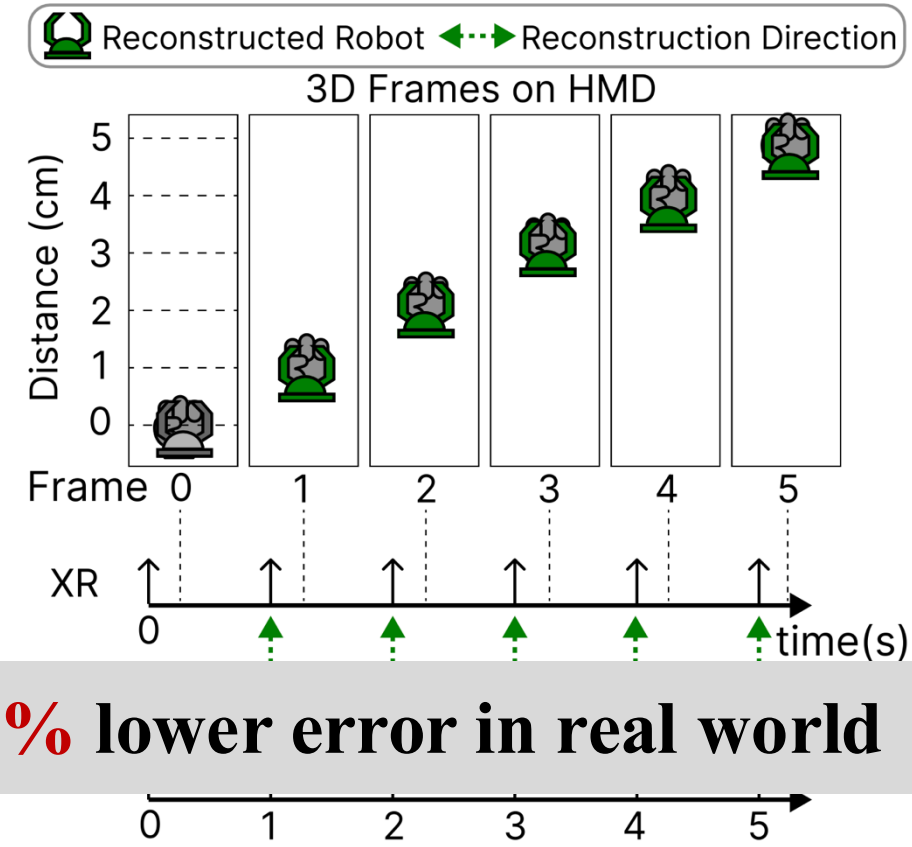
Our Solution: MATER

Root cause: Synchronous execution architecture

- Both XR and robot must wait for network updates to maintain state consistency
  → Vulnerable to network latency (C1) and motion dynamics (C2)

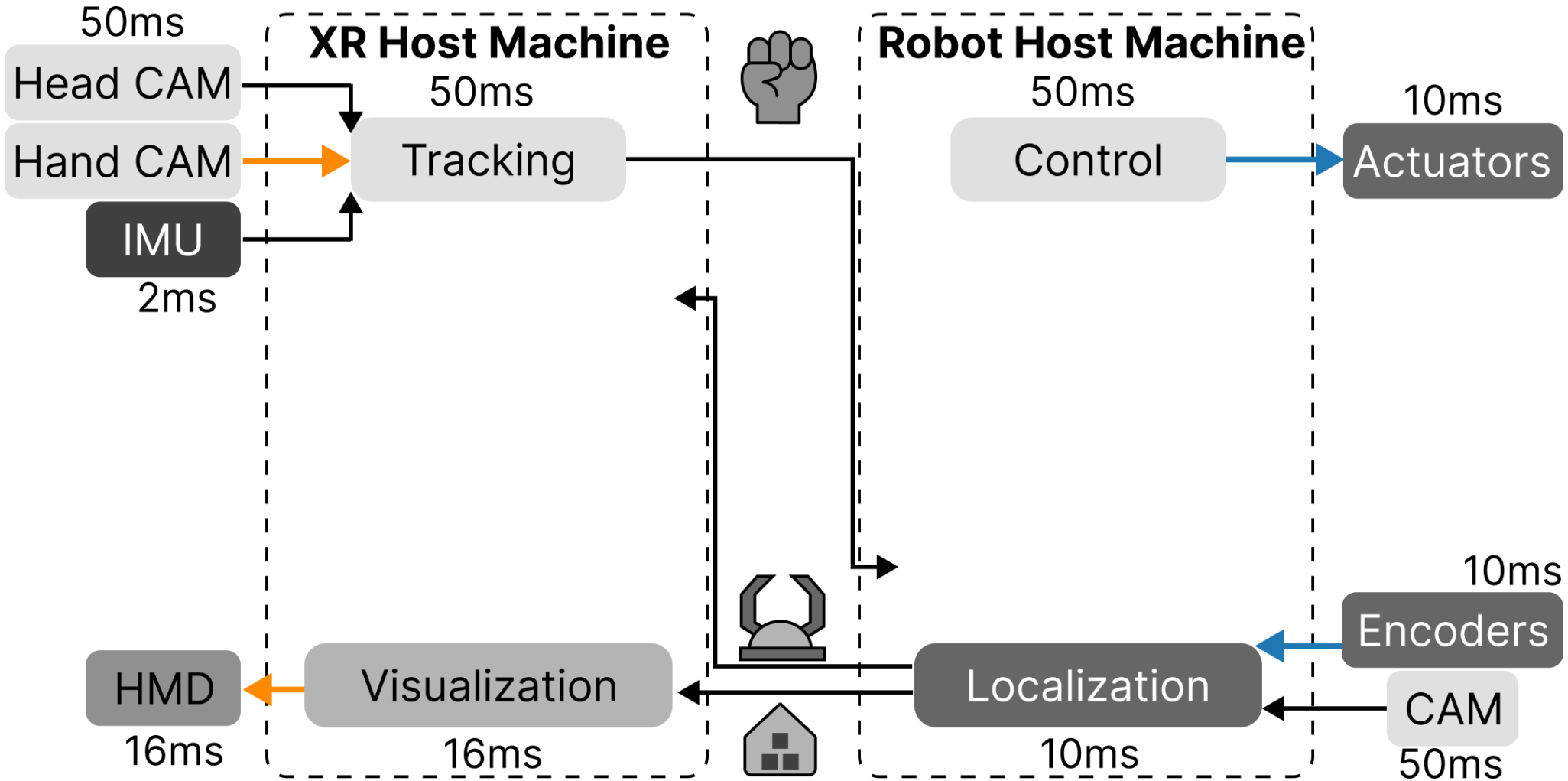
Key insight: Mutually-aware paradigm

- Decouple execution from network dependency by locally reconstructing its counterpart's states
- Three architectural shifts
  - Bidirectional Reconstruction of Time-Shifted States
  - Asynchronous Global & Synchronous Local Execution
  - Control/Data Plane Separation

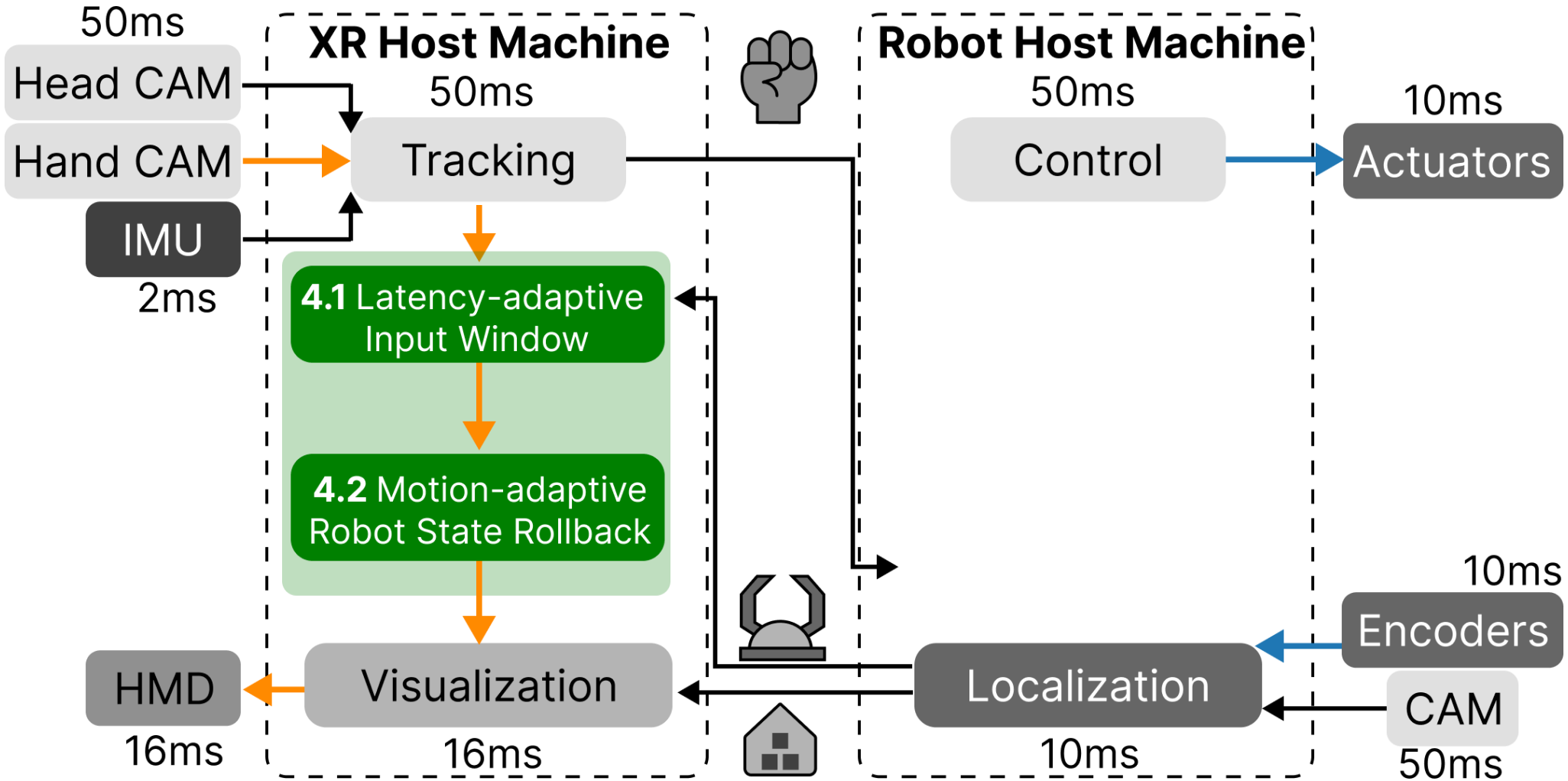


Up to 37.7% faster mission completion & 57.2% lower error in real world

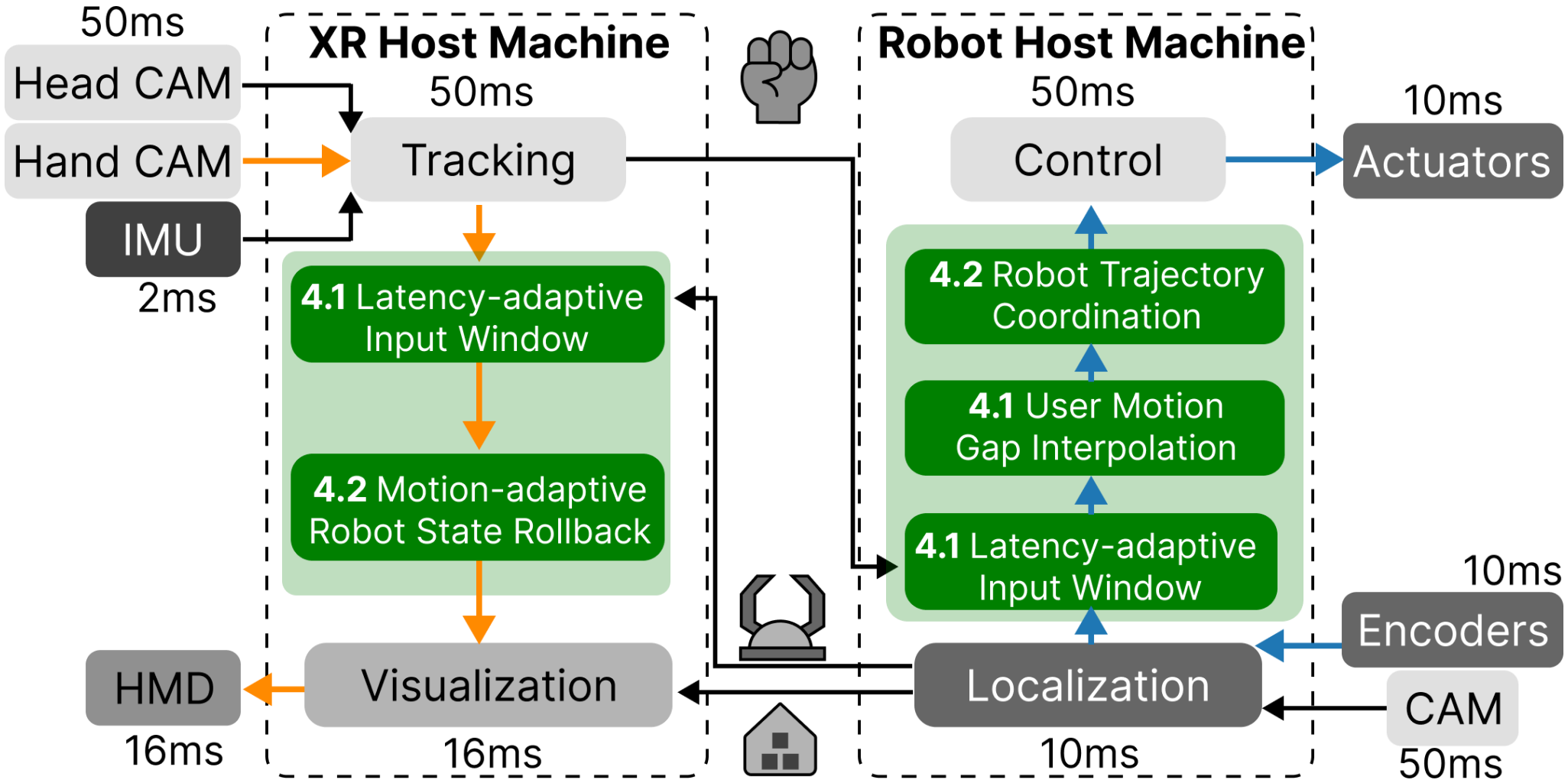
MATER Framework Overview



MATER Framework Overview

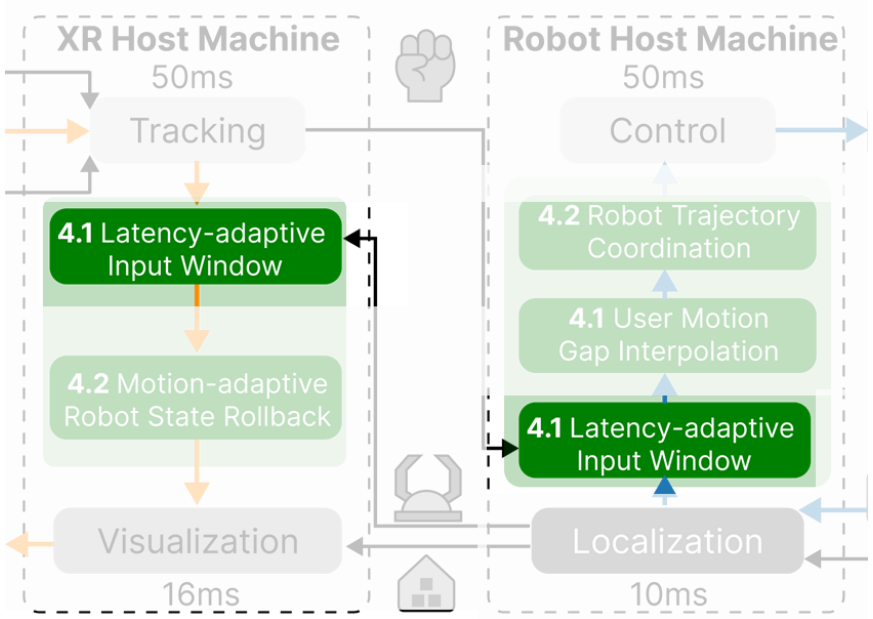


MATER Framework Overview



Bidirectional Reconstruction: Network Fluctuations (C1)

Problem: Reconstruction relies on poses sent over the network → latency, drops, and out-of-order arrivals



Latency-adaptive Input Window

- Adapts input window to the most recently round-trip latency and the robot encoder period

Round-trip latency

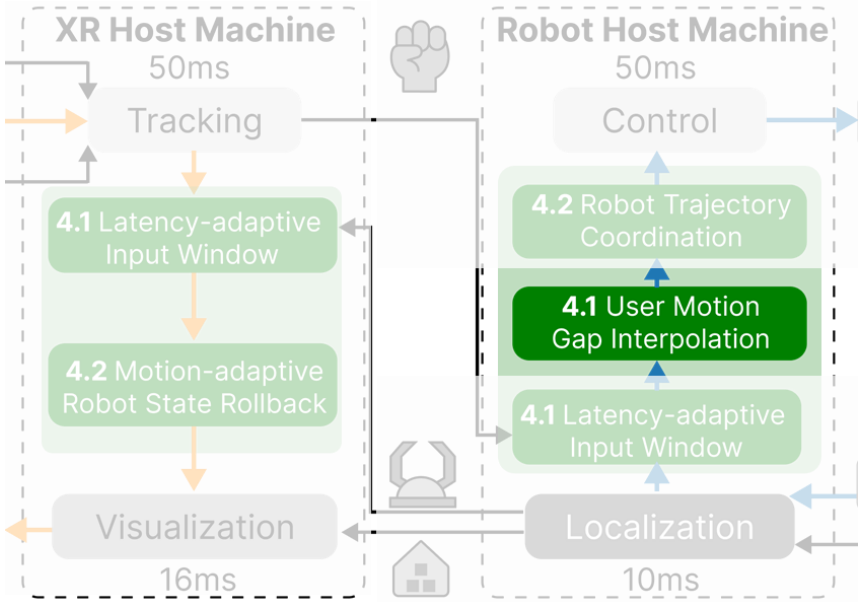
User selected range

$$k_t^\star = \left\lceil \frac{\Delta t_t}{\tau} \right\rceil, \quad k_t = \min\left\{ \max\{k_{\min}, k_t^\star\}, k_{\max} \right\}$$

Robot encoder period

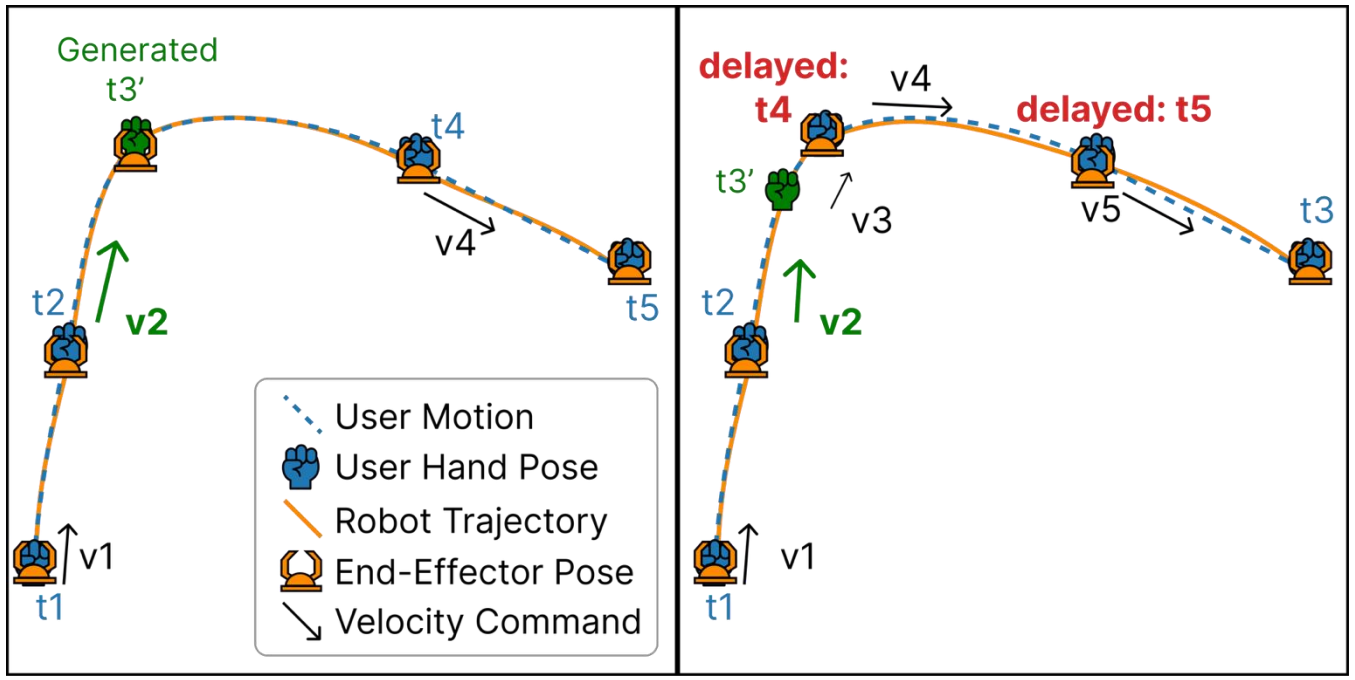
Bidirectional Reconstruction: Network Fluctuations (C1)

Problem: Robot still needs a continuous target when user poses drop or arrive late



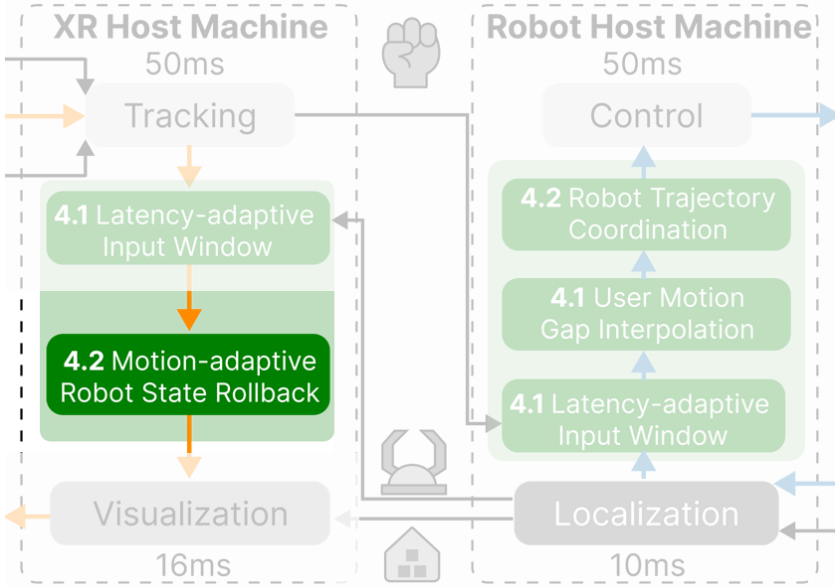
User Motion Gap Interpolation

- Detects gap & interpolates waypoints toward next pose



Bidirectional Reconstruction: Motion Dynamics (C2)

Problem: Reconstruction accuracy drops under fast/high-curvature motion, and a wrong robot state can go unnoticed by the user



Motion-adaptive Robot State Rollback (XR side)

- When motion is too fast/abrupt to reconstruct reliably, roll back to an earlier reconstructed state
  → Nudges the user to slow down

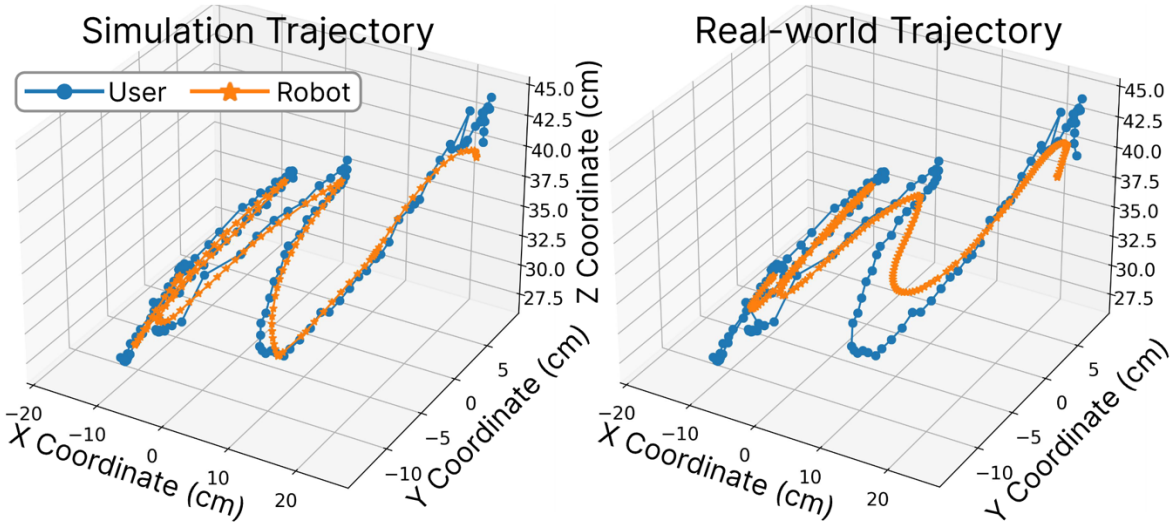
Decision tree policy

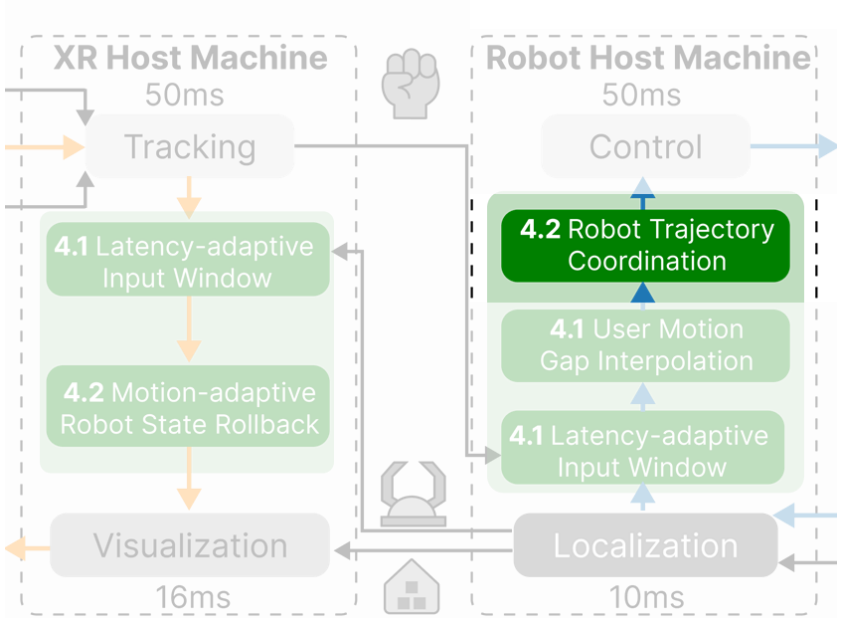
$$j_t = \pi_\psi(z_t), \qquad j_t \in \{0, 1, \ldots, J_{\max}\}$$

recent reconstructed state, user motion and network

Bidirectional Reconstruction: Motion Dynamics (C2)

Problem: Two views (robot local state vs. what user sees) are unaligned. Error is much higher in real robots.





Robot Trajectory Coordination (robot side)

- Fuses both views into one robot motion goal, weighted by reconstruction confidence α_t and link quality $\Pi_\Omega(\tilde{r}_{t+\Delta t})$.

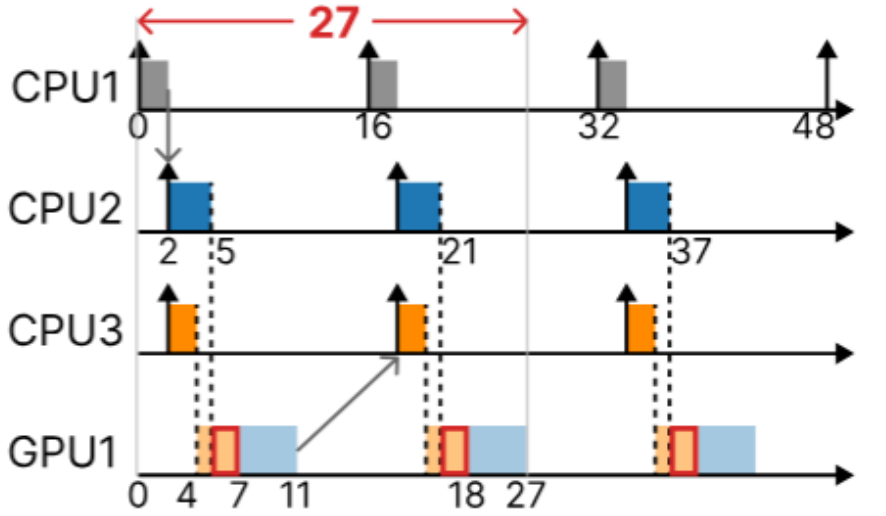
$$g_t = (1-\alpha_t)\,u^\star + \alpha_t\,\Pi_\Omega(\tilde{r}_{t+\Delta t})$$

Async Global and Sync Local Execution

Asynchronous Global: Bidirectional reconstruction lets XR visualization and robot control run on independent timelines (sync only when network messages arrive)

How about Local component execution?

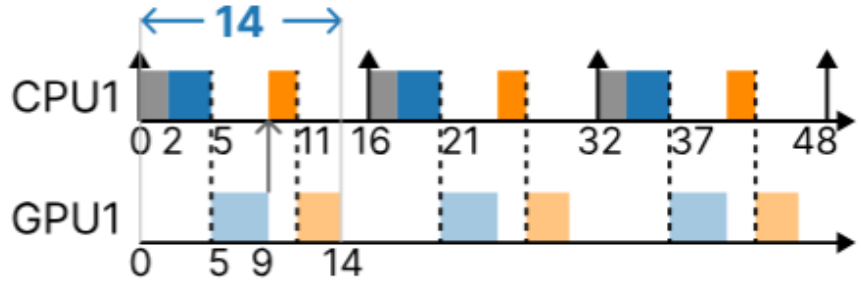
- Existing frameworks : asynchronous local execution with pub-sub mechanisms (DDS)
- Problem: each side has contention on shared resources → jitters & skipped executions



(A) Rendering and Reprojection contention

Our approach: Synchronous Local Execution

- Locally enforcing consistent ordering and cycle boundaries



(B) MATER Runtime Synchronization

Control/Data Plane Separation

Why separate?

- Existing frameworks send everything over DDS (ROS-compatible)
- Fine for small pose data, but not for large point clouds (needed for situation awareness)

Plane split in MATER

- Control plane (pose): DDS low overhead on small payloads
- Data plane (point cloud): UDP -- droppable without hurting control accuracy

Bandwidth-adaptive Point Cloud Scaling

- Scale down point-cloud data in object interior to reduce data transmitted while keeping structural integrity

Evaluation Setup

Hardware Platforms

- *Simulation*: PC hosted Gazebo* simulation with Kinova Gen3 manipulator
- *PC-to-PC*: Both XR and real robot are powered by PC with RTX 3060 GPU
- *Xavier-to-Nano*: XR powered by AGX Xaiver and robot powered by Orin Nano

Motion

- Motion dataset: Roboset** with 9,500 teleoperation traces
- Motion pattern with increasing complexity: *Linear, Curved, Oscillatory, Circular,* and *Segmented*

Network

- WLAN 5GHz and WLAN 2.4GHz, Cellular 5G and Cellular 4G
- XR and robot communicate through a cloud VPN server to control distance

* Nathan Koenig, and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)
** Vikash Kumar, Rutav Shah, Gaoyue Zhou, Vincent Moens, Vittorio Caggiano, Abhishek Gupta, and Aravind Rajeswaran. Robohive: A unified framework for robot learning. Advances in Neural Information Processing Systems, 36, 2024

Evaluation Baselines

- **ROS Reality***: state-of-the-art XR teleoperation framework adopted in industry project
- **MATER-NoOp**: only the robot-side and XR-side reconstruction components
- **MATER: full MATER** system with all components and optimization

Ablation study is conducted by comparing MATER-NoOp and MATER

* David Whitney, Eric Rosen, Daniel Ullman, Elizabeth Phillips, and Stefanie Tellex. Ros reality: A virtual reality framework using consumer-grade hardware for ros-enabled robots. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1–9. IEEE, 2018.

MATER system overhead

- MATER incurs **6.7%** overhead on *PC-to-PC* configuration and only **3.6%** on *Xavier-to-Nano* configuration
- XR-side visualization on the *Xavier-to-Nano* is reduced by **13.4%** because of the local synchronization

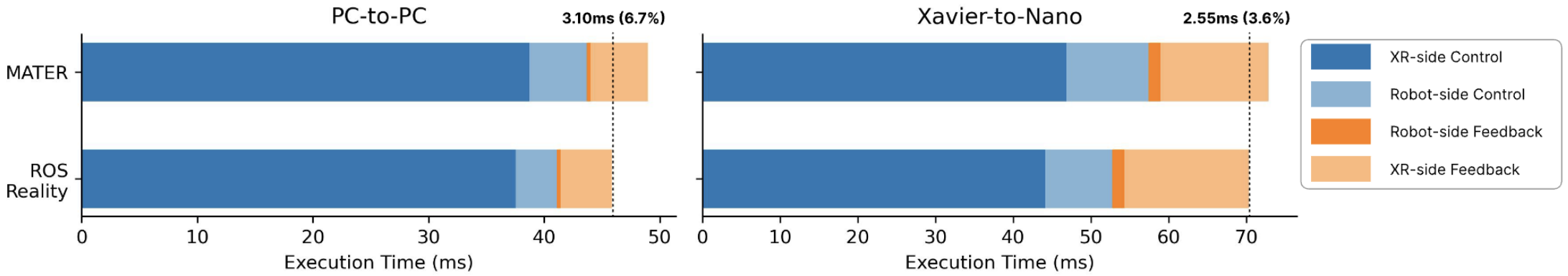


Fig. 9. MATER System Overhead.

Teleoperation Error

- MATER-NoOp: reduced **50.8%** under WLAN and **54.5%** under cellular networks
- MATER: reduced up to **69.8%** in WLAN and **73.1%** in cellular networks

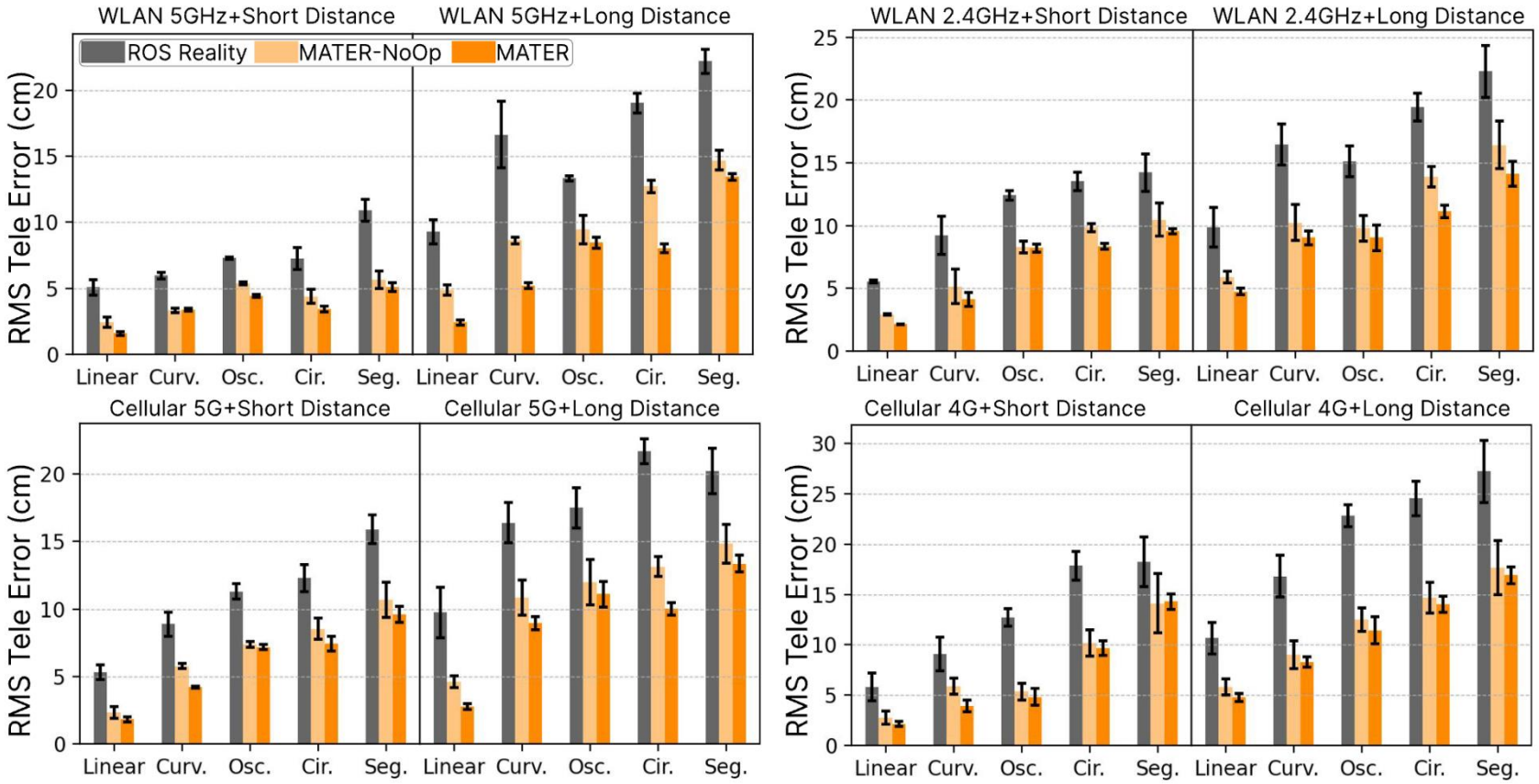
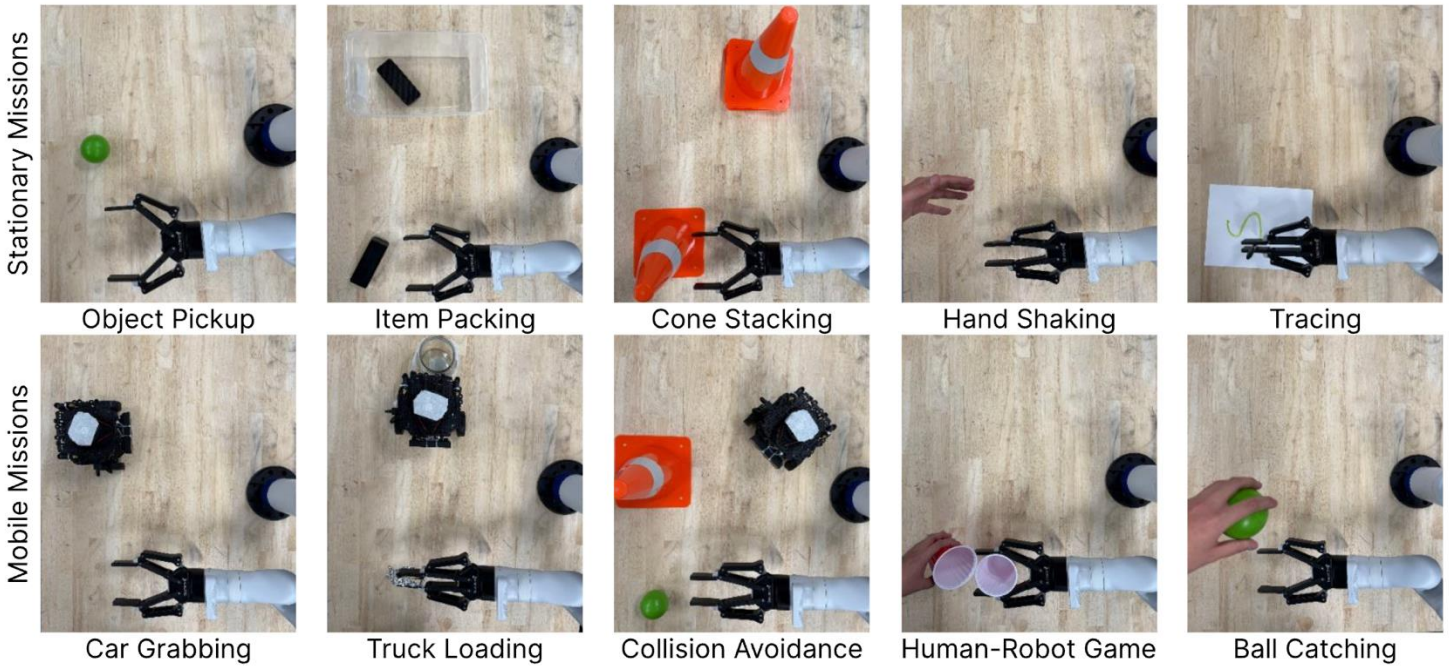
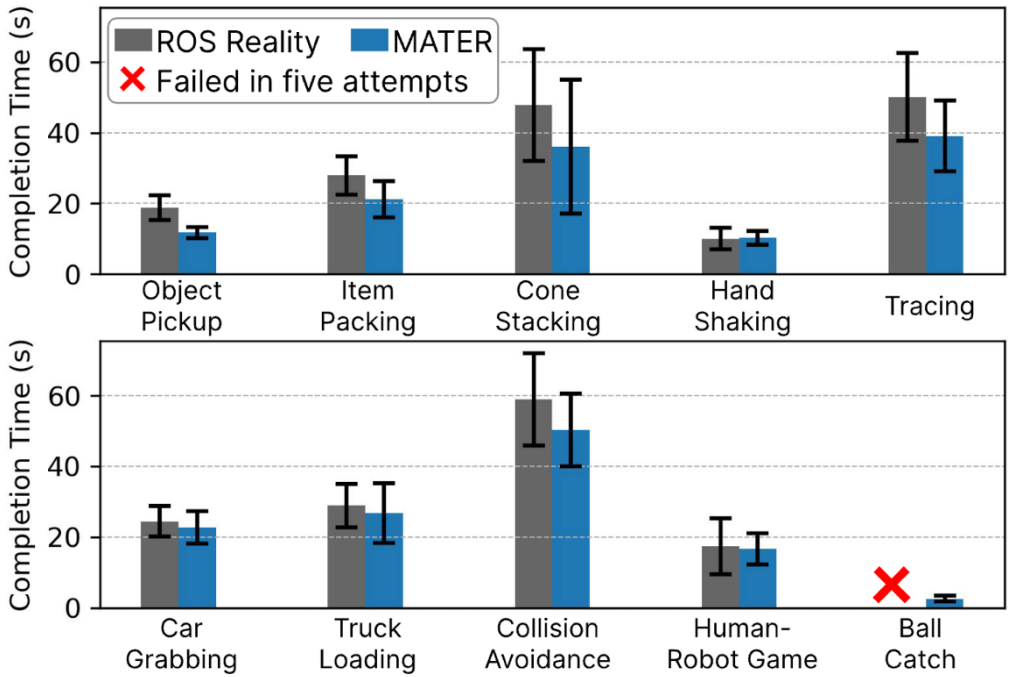


Fig. 11. Teleoperation Error comparison under PC-to-PC

Real-world Evaluation

- Real-world mission setup: both stationary and mobile missions
- Completion Time reduced by up to **37.7%** and teleoperation error by up to **57.2%**
- MATER can **accomplish ball catch**





Thanks for your attention!

MATER GitHub

Acknowledgment: This work was supported by the National Science Foundation (NSF) grants 1943265, 2312395, 2300525, 2312397, and 2343653.