Learning High-Frequency Continuous Action Chunks in Latent Space

with Reuse-then-Refine for Smooth Real-Time Robot Control

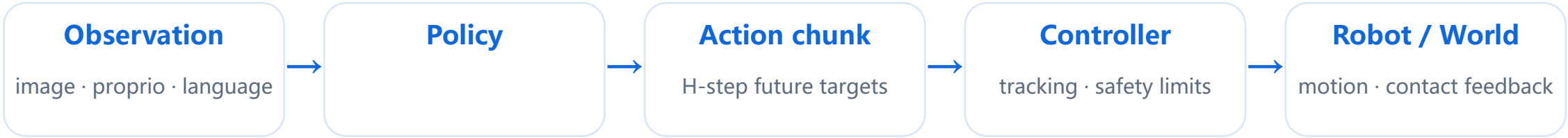
Kunyun Wang, Yuhang Zheng, Yupeng Zheng, Jieru Zhao, Wenchao Ding

Github: https://github.com/tars-robotics/RTR

ICML 2026

From Policy to Robot: pipeline & action chunking

Policies predict short action chunks; inference latency limits real-time execution.



Action chunking

Predict an H-step chunk per inference: $o_t \rightarrow \pi_\theta \rightarrow [a_t, a_{t+1}, \ldots, a_{t+H-1}]$

- Captures local motion trends; mitigates non-Markovian noise (e.g., tremors, brief pauses).
- Each chunk covers a short horizon — long-horizon tasks require repeated inference.
- Observations aren't used again until the next inference call.

Synchronous execution: robot waits for inference

Observe → Infer → Execute → Repeat

- Inference latency directly becomes idle time.
- Worsens with large VLAs.
- Asynchronous inference overlaps inference with execution — but creates chunk-boundary discontinuities.

Robot Action: what does a policy output?

Robot action = arm control + gripper command. Arm control can be low-level (torque) or high-level position (absolute or relative).

Robot Action

Arm control

Low-level control

torque / current

Motor torque/current; high bandwidth but platform-coupled — hard to merge across robots.

High-level — Position control

Absolute

TCP xyz + rpy absolute pose target

⇅

Relative

Δxyz + Δrpy increment from current pose

Absolute ⇌ Relative are interconvertible

Gripper command

open / close / width

Open / close / width; combined with arm action to form the full execution command.

Action in our setting

We use absolute end-effector pose (TCP xyz + rpy) + gripper width. An action chunk stacks H of these.

Motivation I: action frequency shapes execution

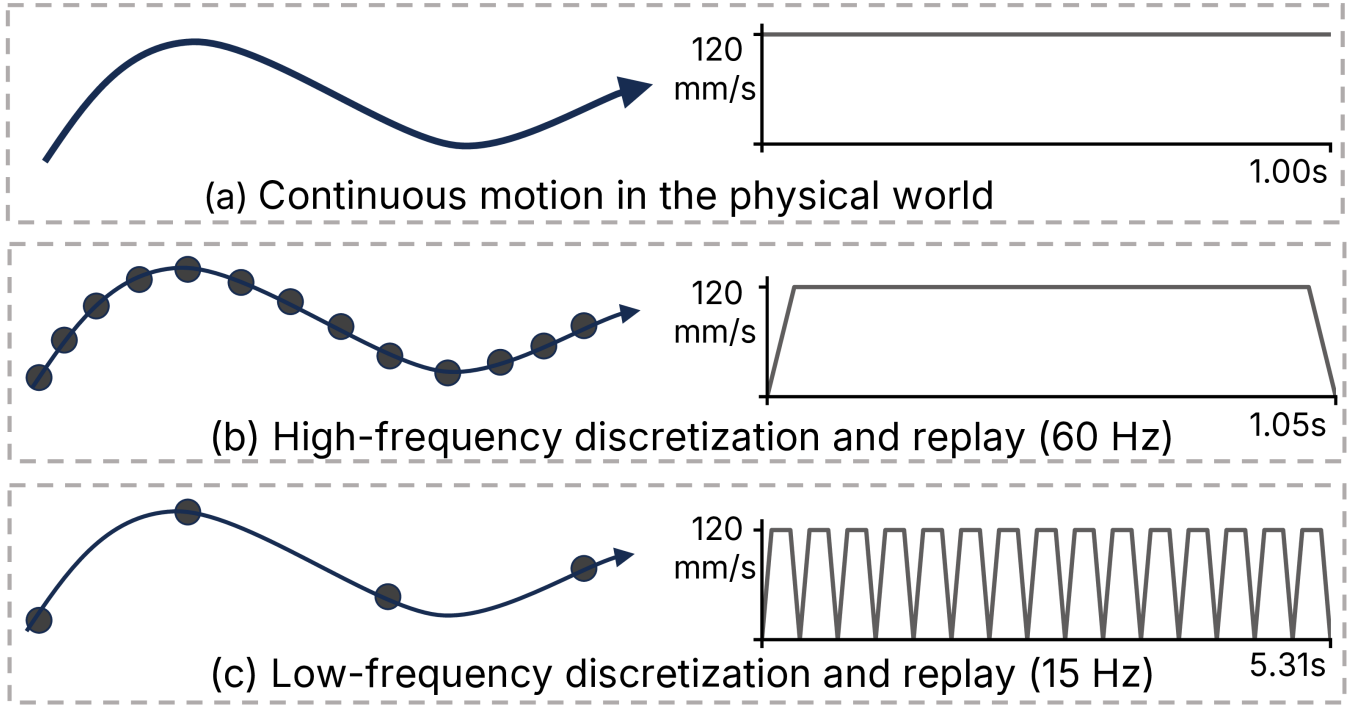
Low-frequency actions create distant targets → stop-and-go; high-frequency actions enable continuous, servo-like control.

Low-freq (15 Hz)

Large strides between targets — velocity drops at each step, motion becomes piecewise.

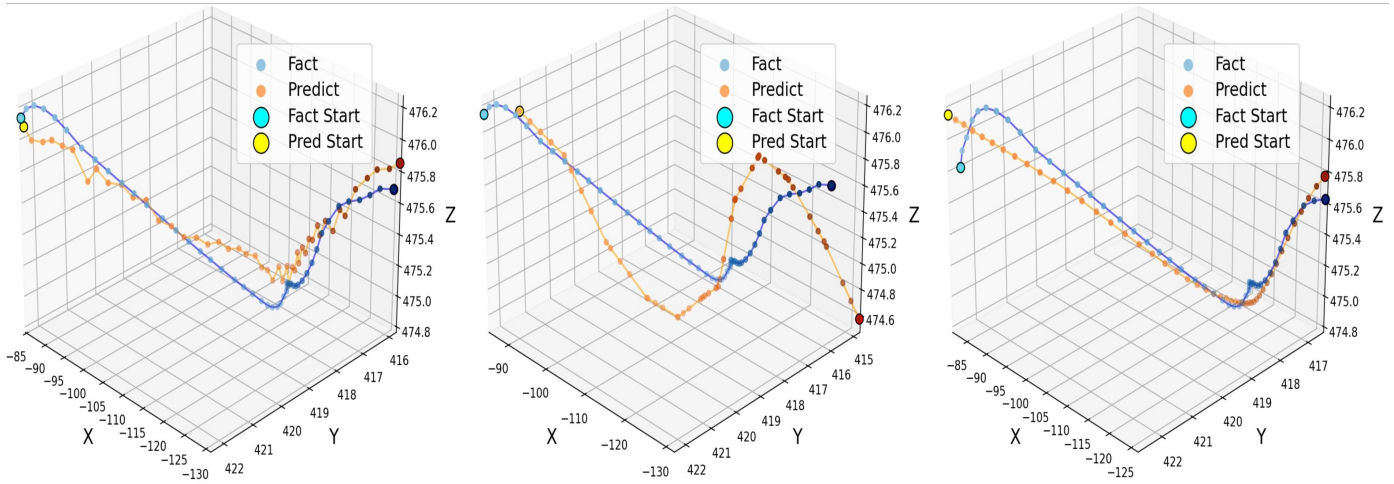
High-freq (60 Hz)

Dense targets preserve velocity — continuous, servo-like motion.

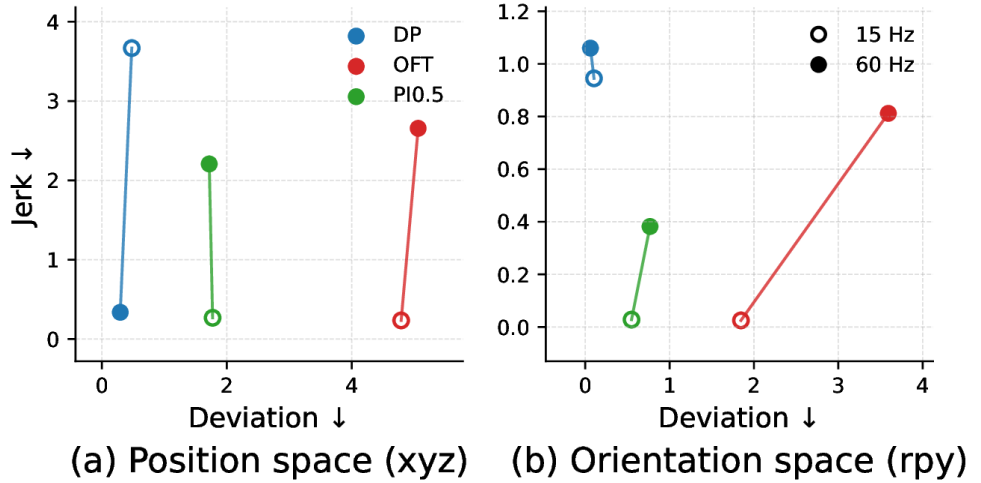


Motivation II: high-frequency actions are harder to learn

Fine temporal and spatial detail amplify quantization errors and jitter when learning directly in action space.



(a) Original (b) Interpolate (c) Latent



(a) Position space (xyz) (b) Orientation space (rpy)

Deviation vs jerk: 15 Hz (hollow) vs 60 Hz (solid). High-frequency helps DP, but substantially hurts OFT and PI0.5.

Original

Learn high-freq directly: low precision, visible jitter.

Interpolate

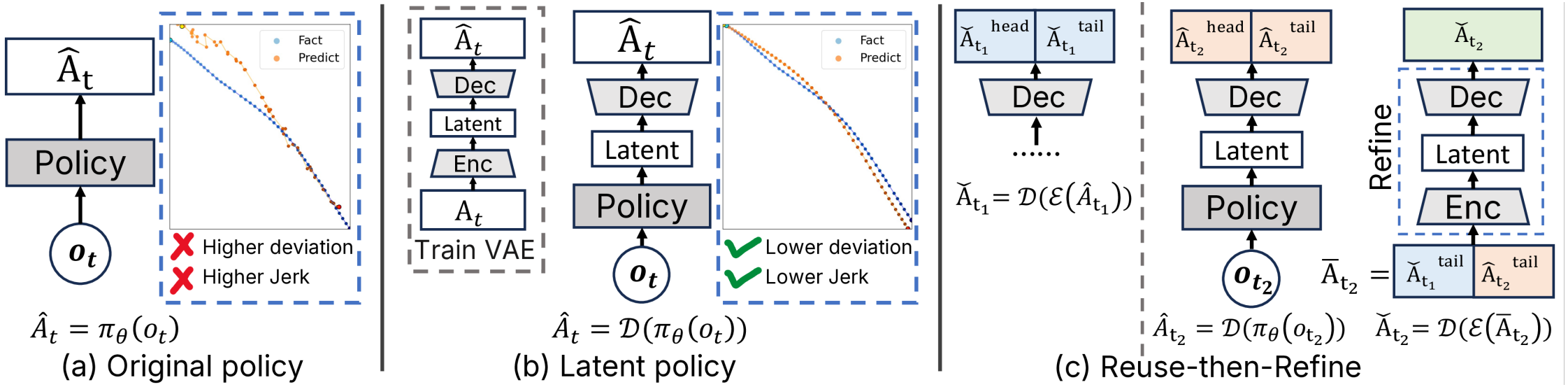
Predict low-freq then upsample: smoother, but imprecise.

Latent

Learn in a continuous latent space, decode back to actions.

Method I: Learning high-frequency actions in latent space

A VAE shifts policy learning to a continuous latent space — predict motion patterns instead of every fine-grained command.



Command-level variation → **Temporal compression** → **Smooth manifold** → **High-frequency decode**

Each action is fine-grained; noise and quantization stand out.

One latent step absorbs fluctuations across multiple adjacent actions.

KL regularization yields a continuous, smooth manifold.

Decoder restores the fine-grained trajectory.

Method II: Reuse-then-Refine for chunk-level continuity

RTR is a training-free strategy: reuse recently executed actions, then refine the concatenated chunk via the VAE.

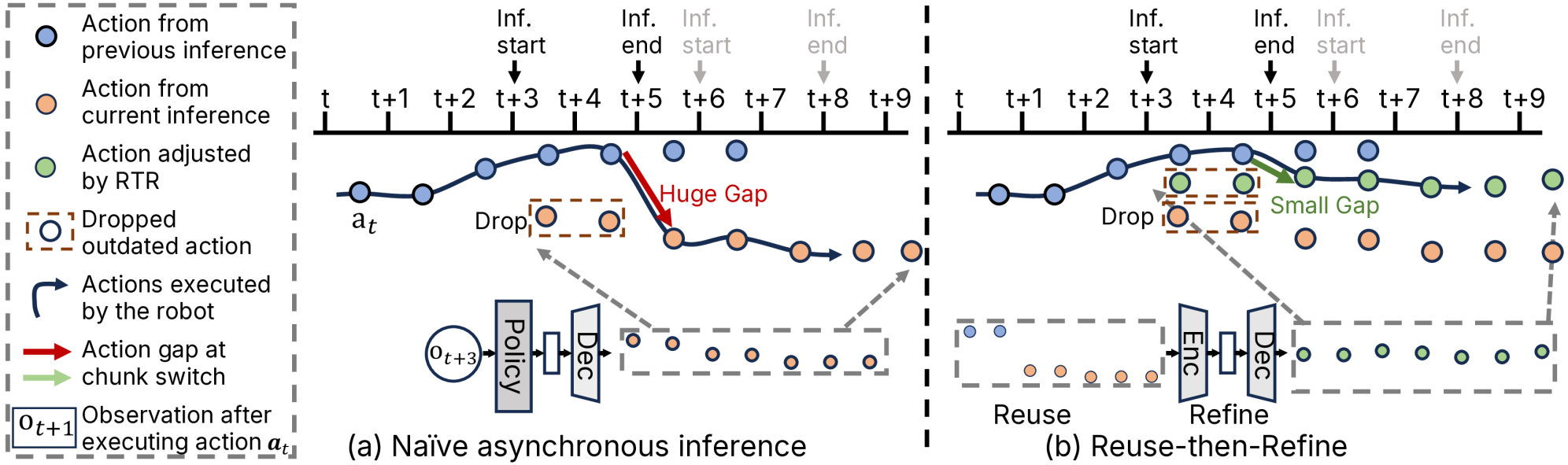
Reuse → **Concatenate** → **Refine** → **Execute**

Reuse actions executed during the inference window.

Combine with non-outdated actions from the new chunk.

VAE encode–decode smooths local inconsistencies (~2 ms).

Seamless transition into the next chunk.

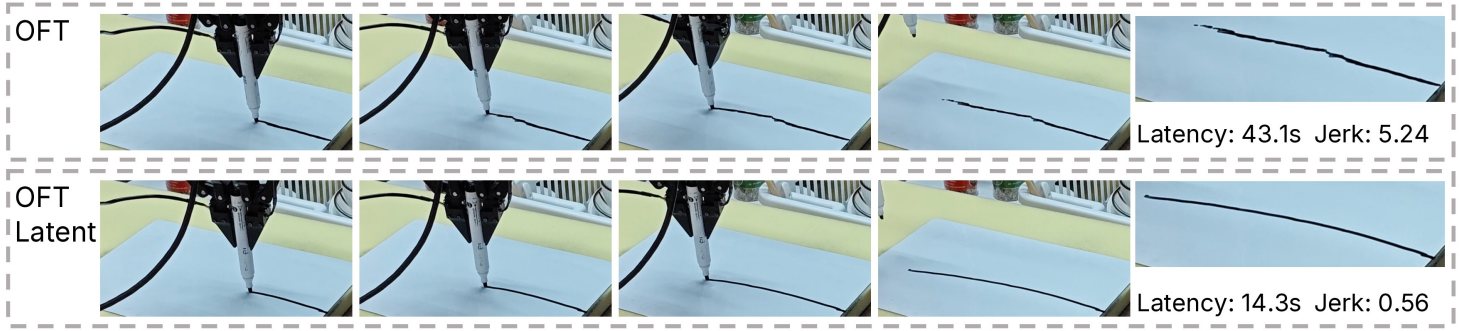


Asynchronous inference hides latency by overlapping computation with execution; RTR closes the boundary gap that asynchronous switching introduces.

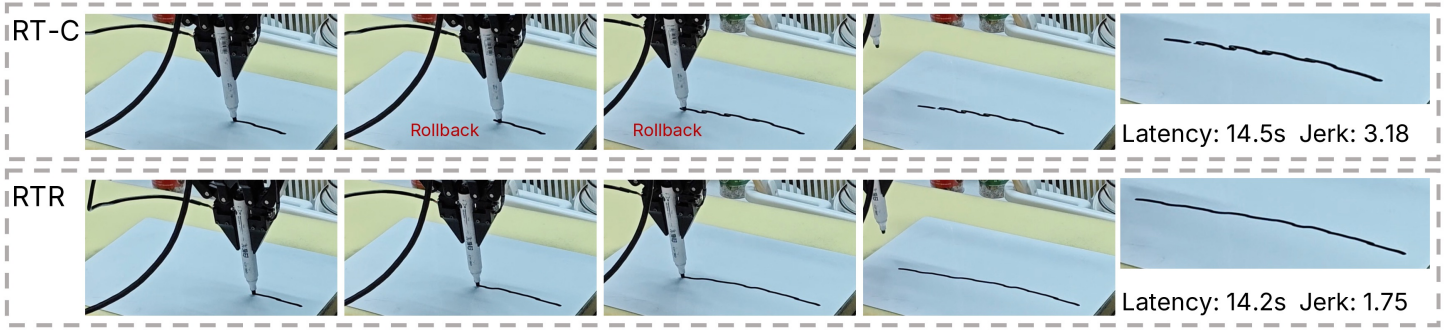
Experiments: smoother motion, fewer stalls, lower latency

Beyond success rate, we measure jerk, exceed count, chunk continuity, and end-to-end latency.

Latent space smooths motion within each chunk.



RTR smooths transitions across chunks under asynchronous inference.



Synchronous

OFT Write Board jerk **5.24** → **0.56**
OFT success **74%** → **100%**

Asynchronous

PI0.5 Write Board jerk
RT-C **3.18** → Latent+RTR **1.75**

End-to-end latency

Peel Cucumber latency (DP)
Original **20.4 s** → Latent+RTR **14.6 s**

Combined: stable, continuous robot execution with fewer stalls and lower end-to-end latency.

Demos I: high-frequency + latent for smoother control

Synchronous execution • same task (Wipe Vase) • same policy (DP); four action representations.

DP 15 Hz

Low-frequency, stop-and-go.

DP interpolated 60 Hz

Upsampled to 60 Hz; still jittery.

DP original 60 Hz

Direct high-freq learning; visible jitter.

DP latent 60 Hz (ours)

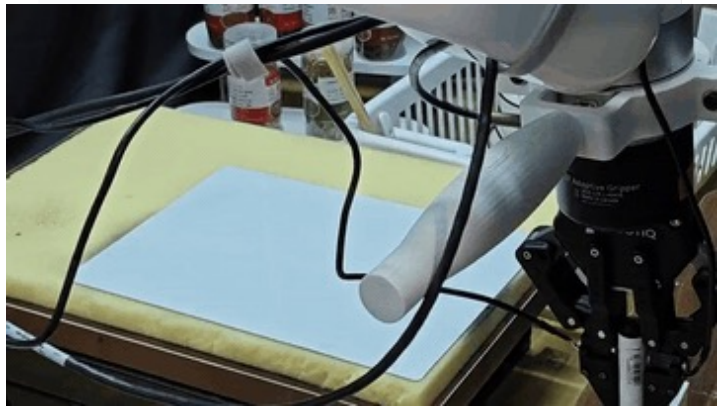
High-freq in latent space; smooth and stable.

Takeaway: smooth real-robot control needs high-frequency actions, but learning them directly is jittery. Latent space achieves both.

Demos II: RTR removes chunk jumps under async inference

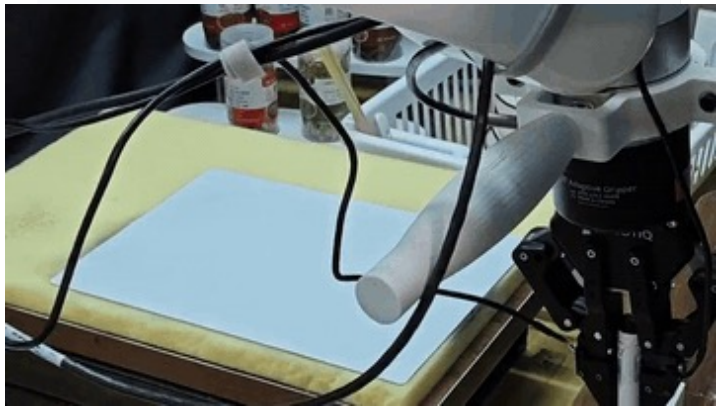
Asynchronous execution • same task (Write Board) • same policy (OpenVLA-OFT); two representations × two stitching strategies.

OFT original (async)



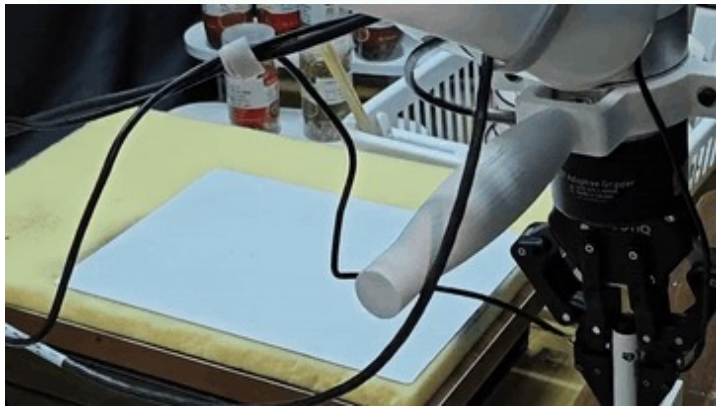
Large jumps at chunk boundaries; frequent stalls.

OFT latent (async)



Smoother within chunks, but boundaries still jump.

OFT latent + RTR (async, ours)



VAE refines boundaries; continuous, no stalls.

Takeaway: latent solves chunk-internal smoothness; RTR solves chunk-to-chunk continuity. Together: real-time smooth control.

TARS 它石智航

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Takeaways

Smooth real-robot control needs the right action representation AND the right execution strategy.

01 High-frequency actions enable smooth execution but are hard to learn directly — latent space makes them learnable.

02 Asynchronous inference hides latency, but introduces chunk-boundary misalignment.

03 Latent policy ensures chunk-internal smoothness; Reuse-then-Refine ensures chunk-to-chunk continuity.

Thank you!