

# Reactive Diffusion Policy:

## Slow-Fast Visual-Tactile Policy Learning for Contact-Rich Manipulation

Han Xue<sup>1\*</sup> Jieji Ren<sup>1\*</sup> Wendi Chen<sup>1\*</sup>

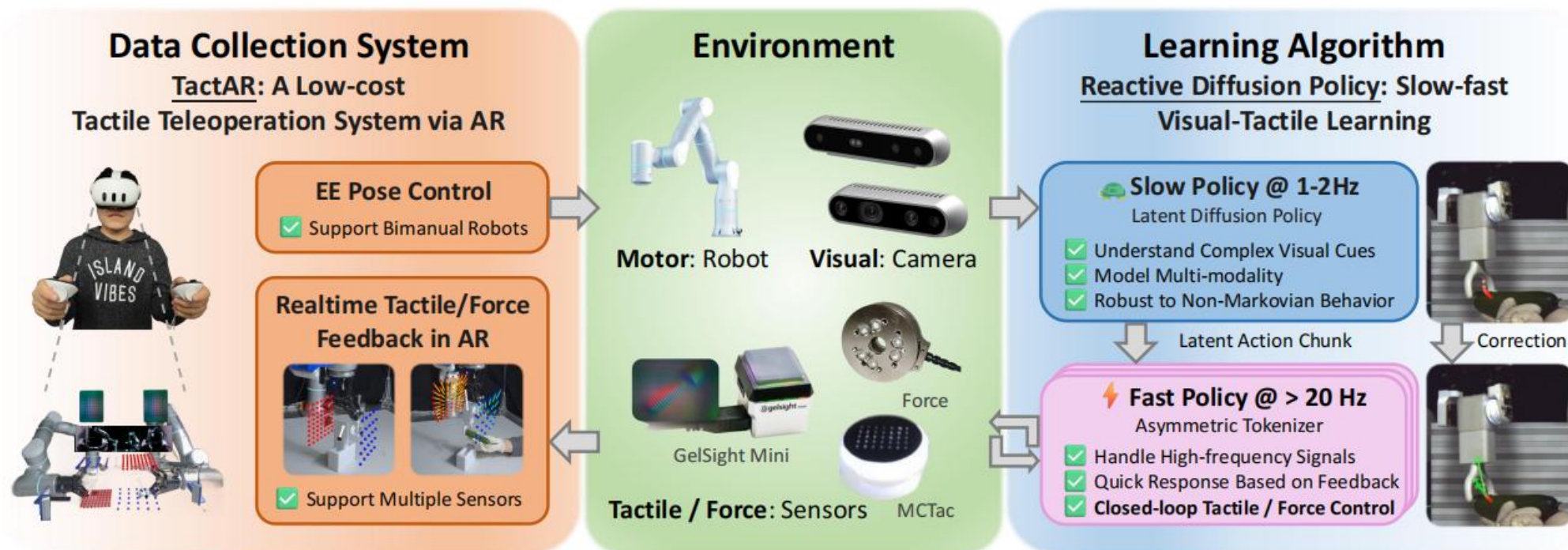
Gu Zhang<sup>234†</sup> Yuan Fang<sup>1†</sup> Guoying Gu<sup>1</sup> Huazhe Xu<sup>234‡</sup> Cewu Lu<sup>15‡</sup>

<sup>1</sup>Shanghai Jiao Tong University <sup>2</sup>Tsinghua University, IIS <sup>3</sup>Shanghai Qi Zhi Institute

<sup>4</sup>Shanghai AI Laboratory <sup>5</sup>Shanghai Innovation Institute

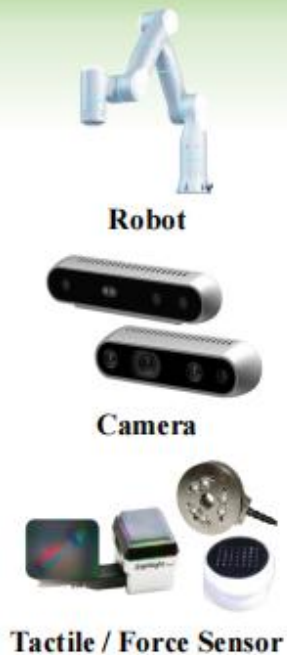
\*Equal contribution †Equal contribution ‡Equal advising

[reactive-diffusion-policy.github.io](https://reactive-diffusion-policy.github.io)





## Environment



Prior perception  
@ 120 Hz

Force / Torque Stream  
@ 120 Hz (opt.)

Image Stream  
@ 30 Hz

Tactile Stream  
@ 25-30 Hz (opt.)

Action Command  
@ 90 Hz



## Workstation

**Feature 1**  
Deformation Field Extraction

**Feature 2**  
Image Stream Processing

**Feature 3**  
Time Synchronization

TCP Pose  
@ 120 Hz

Image Stream  
@ 25-30 Hz (opt.)

3D Deformation Field  
@ 25-30 Hz (opt.)

Action Command  
@ 90 Hz

## Teleoperation System

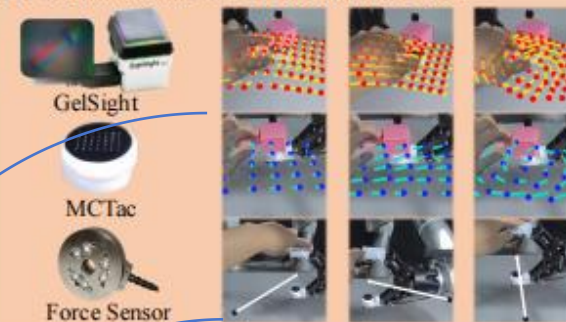


## VR Headset and Controller

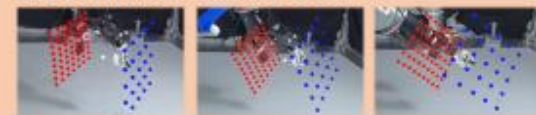
**Feature 1** Hand Pose Tracking for EE Pose Control

**Feature 2** Realtime Tactile Feedback in AR

2.1 Cross-sensor 3D Deformation Field



2.2 Attachment to End Effector



2.3 Camera Streaming



用 OpenCV 提取每个 marker 的位置  $D_t$   
计算初始帧  $D_0$  到当前帧  $D_t$  的光流:

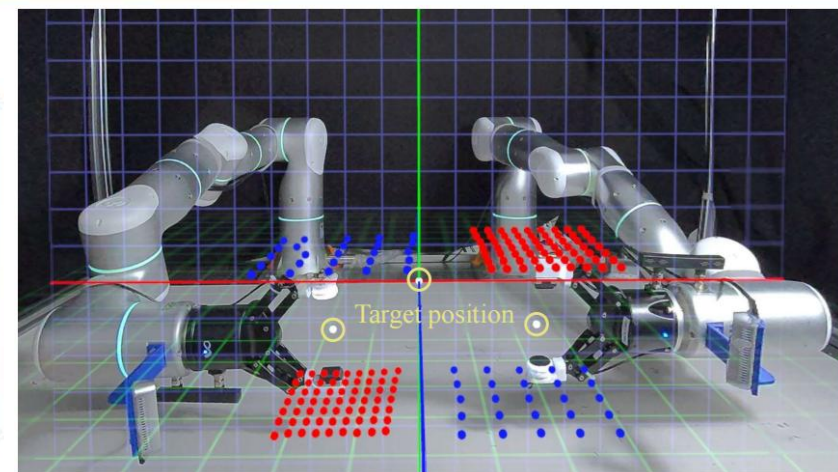
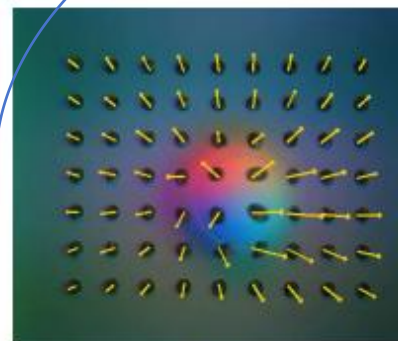
$$F_t = [dx, dy] = \text{Flow}(D_0, D_t)$$

再给每个 marker 加上一个法向位移  $o_z$ , 形成 3D 形变场:

$$V_t = [dx, dy, o_z]$$

直接把 3D 力 / 力矩  $[f_x, f_y, f_z]$  当成 3D 向量:

$$V_t = [f_x, f_y, f_z]$$



After calibration

最终转换成AR 世界坐标系下的 3D 向量场

$$V_{AR} = {}^{\text{AR}}T_{\text{Robot}} \cdot {}^{\text{Robot}}T_{\text{TCP}} \cdot {}^{\text{TCP}}T_{\text{Sensor}} \cdot V_{\text{sensor}}$$

# RDP 算法

力传感器：直接把 3D 力 + 3D 力矩拼进观测向量

## 1. Tactile / Force Representation

视触觉传感器：

每帧图像中，每个 marker 都会产生一个二维的切向位移 [dx,dy]，一帧触觉数据可以表示为一个高维向量  $F_t \in \mathbb{R}^{2n}$

GelSight Mini 有  $n=7 \times 9=63$  个 marker，每帧触觉数据是一个 126 维向量。

高维数据冗余，不同维度之间高度相关  $\rightarrow$  PCA（主成分分析）

构造触觉数据矩阵、求协方差、做特征分解，选取前  $d$  个最大特征值对应的特征向量构成投影矩阵  $T_{\text{proj}}$

$$D_{\text{tactile}} = \{F_1, F_2, \dots, F_m\}, \quad F_t \in \mathbb{R}^{2n}$$

$$F_{\text{concat}} = \begin{bmatrix} F_1^T \\ F_2^T \\ \vdots \\ F_m^T \end{bmatrix} \quad F_{\text{concat}} \in \mathbb{R}^{m \times 2n}$$

$$\tilde{F} = F_{\text{concat}} - \mu, \quad \mu = \frac{1}{m} \sum_{t=1}^m F_t$$

$$C = \frac{1}{m-1} \tilde{F}^T \tilde{F} \quad \text{描述各 marker 位移维度之间的相关性}$$

$$C = U \Lambda U^T \quad T_{\text{proj}} = U_{[:, 1:d]}$$

$U$ ：特征向量矩阵（每列是一个主方向）

$\Lambda$ ：特征值（表示方差大小）

对于新的触觉帧，先做去中心化，再乘以投影矩阵，

$$\tilde{F}_{t'} = F_{t'} - \mu \quad f_{t'}^{\text{PCA}} = T_{\text{proj}}^T \tilde{F}_{t'} \quad f_{t'}^{\text{PCA}} \in \mathbb{R}^d$$

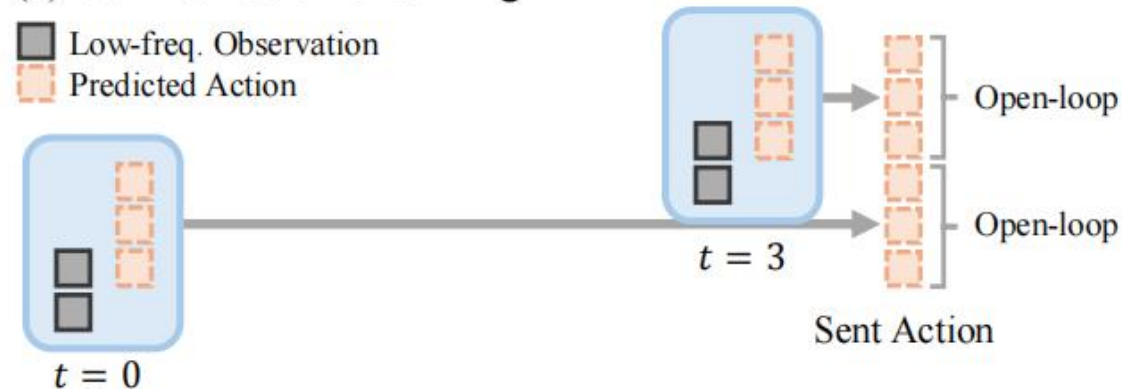
得到  $d$  维触觉 embedding ( $d = 15$ )

15维已经能覆盖 > 95% 的触觉方差信息

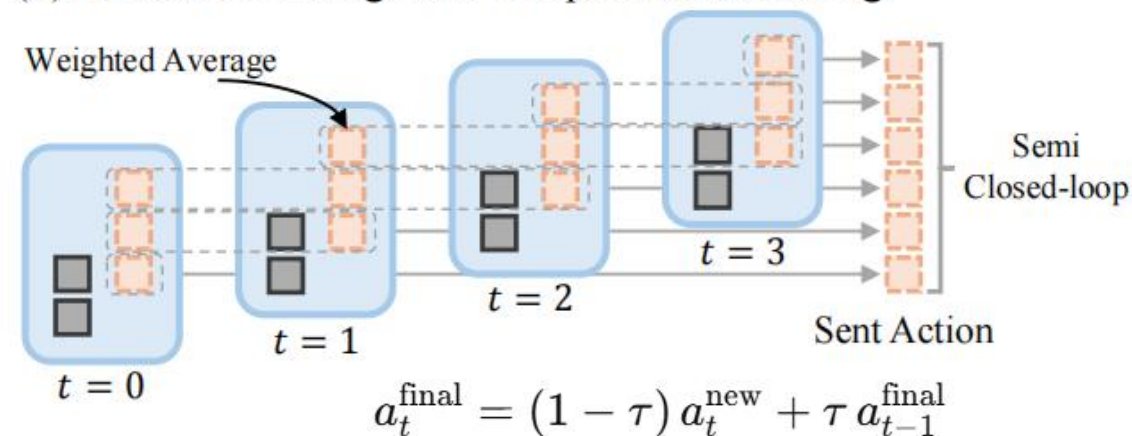
# RDP 算法

## 2.Slow-Fast Policy Learning

(a) Vanilla Action Chunking



(b) Action Chunking with Temporal Ensembling

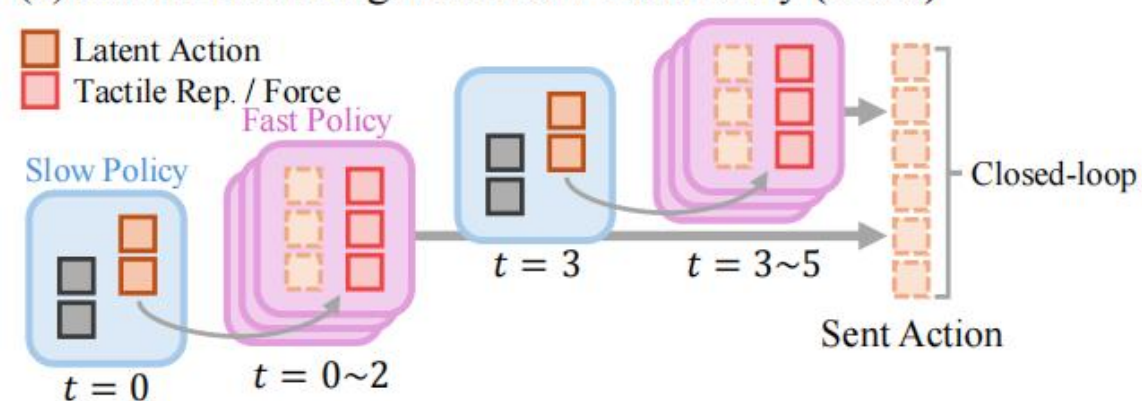


多个时间步对同一时刻的预测做加权平均

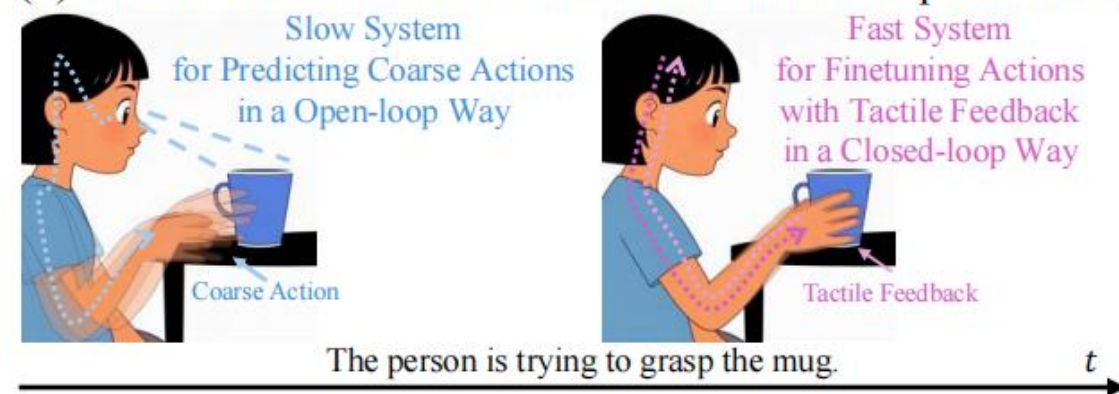
慢策略:低频上预测latent action chunk, “大脑/计划层”

快策略:更高频率读取最新的触觉/力反馈, “反射/小脑”  
对这段 latent 做帧级别的闭环细调,

(c) Action Chunking with Slow-Fast Policy (Ours)



(d) Human with Predictive Action and Closed-loop Finetuning

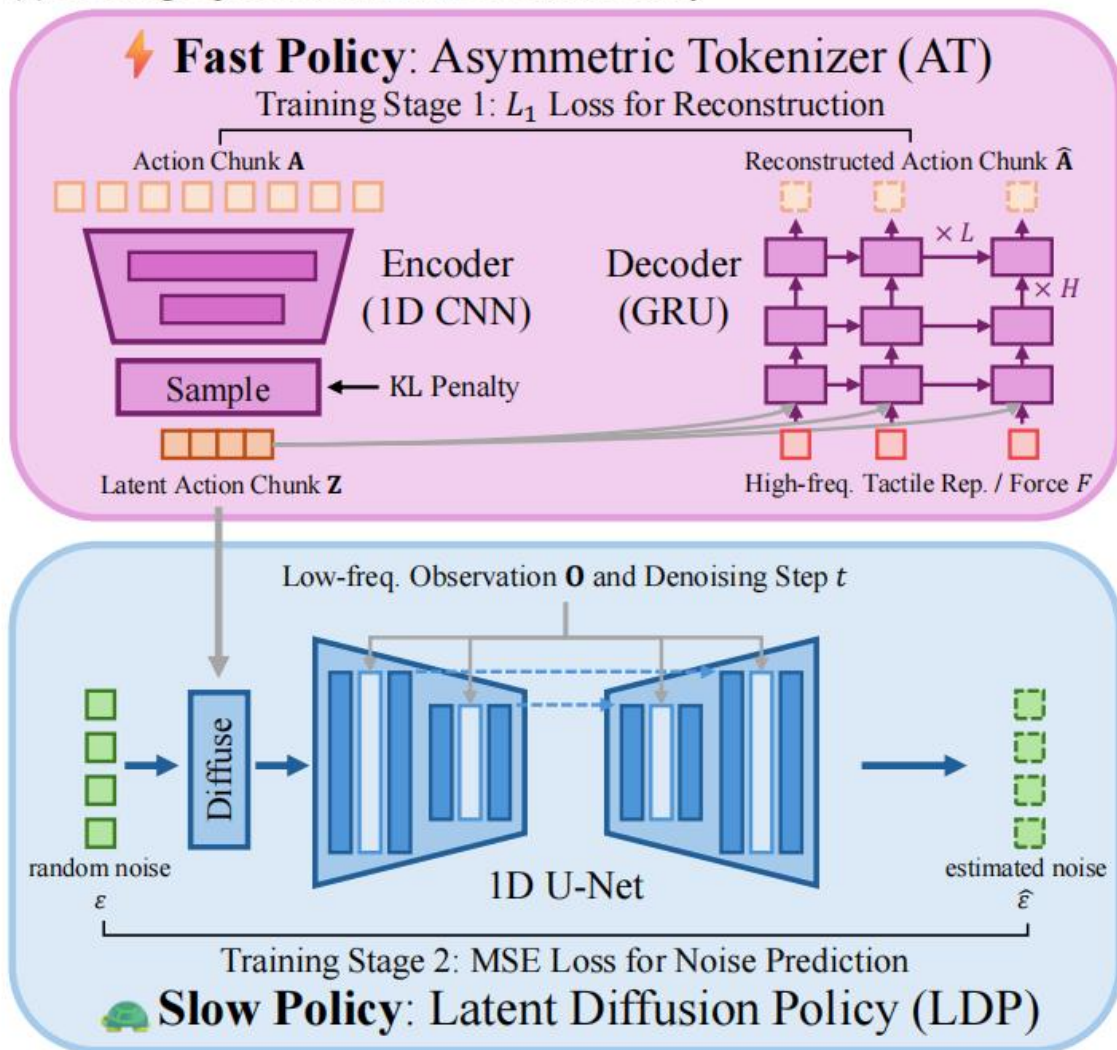




# RDP 算法

## 2.Slow-Fast Policy Learning

(a) Training Pipeline of Reactive Diffusion Policy



Stage 1: 把 action chunk 编成 latent 表示 (AT 的 encoder), 并学一个带触觉输入的 decoder 来重建动作, 快策略

$$A \in \mathbb{R}^{T \times D} \quad h_t^{(1)} = \text{Conv1D}(A_{t-k:t+k})$$

$$Z = E(A) \in \mathbb{R}^{t \times d}, \quad t < T \quad \text{只包含高层策略 / 轨迹结构}$$

$$\hat{A} = D(\text{concat}[Z, F^{\text{reduced}}]) \quad F^{\text{reduced}} \in \mathbb{R}^{T \times 15}$$

$$\text{训练目标: } \mathcal{L}_{\text{AT}} = \mathbb{E}_{(A, F^{\text{reduced}}) \sim \mathcal{D}_{\text{policy}}} [\|A - \hat{A}\|_1 + \lambda_{\text{KL}} \text{KL}]$$

$$\text{KL}(q(Z|A) \parallel \mathcal{N}(0, I))$$

高频闭环推理: AT decoder 每一帧都会拿最新的触觉 / 力 embedding + latent, 输出下一步动作, 跑到 >300 Hz

Stage 2: 在 latent 空间上训练 Diffusion Policy (LDP), 预测 latent action chunk, 慢策略

$$Z_0 = E(A_0) \quad \text{在 latent 上加噪声: } Z_k = Z_0 + \epsilon_k$$

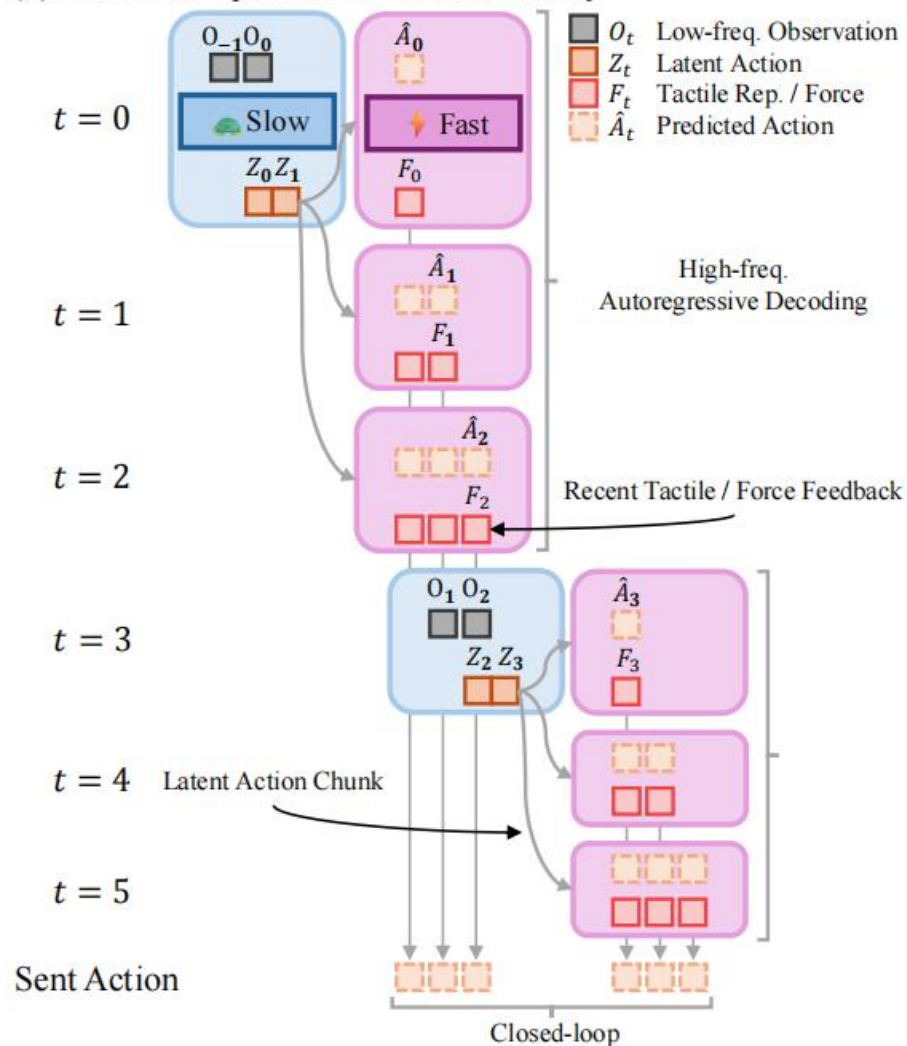
给定观测, 学习一个噪声预测器, 训练目标:

$$\mathcal{L}_{\text{LDP}} = \mathbb{E}_{(O, A_0), k, \epsilon_k} \|\epsilon_k - \epsilon_{\theta}(O, Z_0 + \epsilon_k, k)\|_2^2$$

# RDP 算法

## 2.Slow-Fast Policy Learning

(b) Inference Pipeline of Slow-Fast Policy



慢策略 (LDP) 低频 (1-2 Hz) 预测一个 latent action chunk  $Z$ ;

快策略 (AT decoder) 高频 (24 Hz) 读取当前触觉 / 力 embedding , 在 latent chunk 基础上逐步生成每一帧动作, 并闭环调整。

LDP  $\sim 100\text{ms}$ , AT  $< 1\text{ms}$  (在 4090 上)



# 实验

Q1: 只加触觉信息能帮多少?

Q2: RDP 的能力

Q3: 传感器泛化

Q4: 即时扰动响应

Q5: ablation: 为什么不能缩短 chunk?

用 temporal ensemble?

Q6 & Q7: TactAR 提升数据质量 →  
进一步提升策略性能

Diffusion Policy (纯视觉)

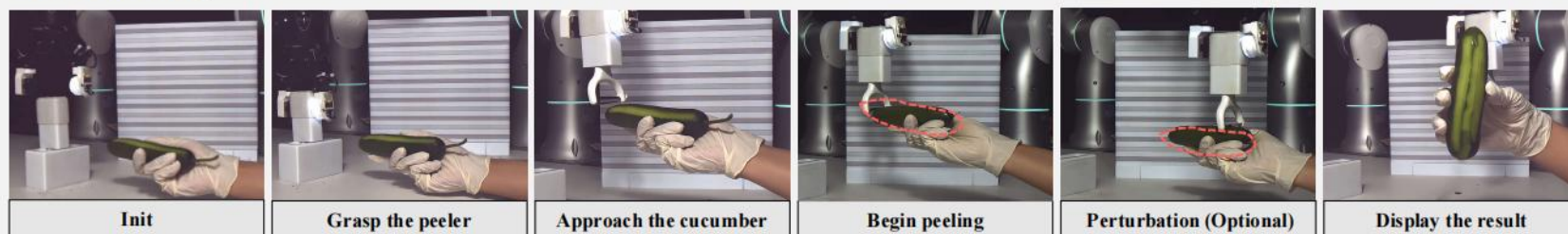
Diffusion Policy + 触觉图像

Diffusion Policy + 触觉 embedding

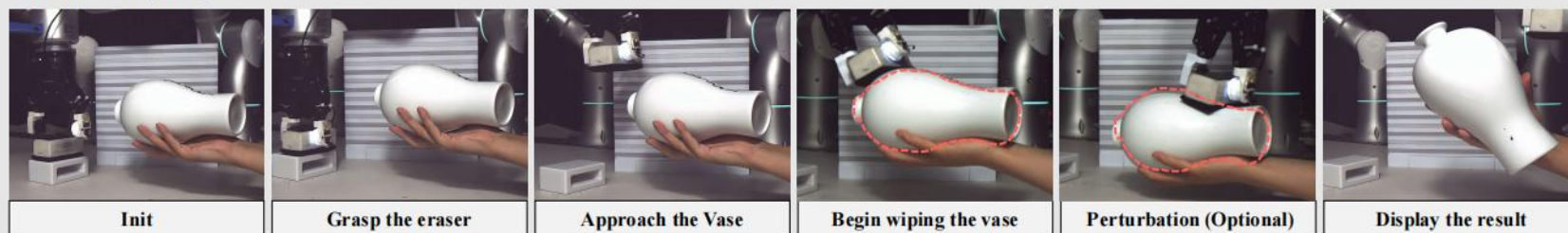
RDP (tactile embedding)

RDP (force)

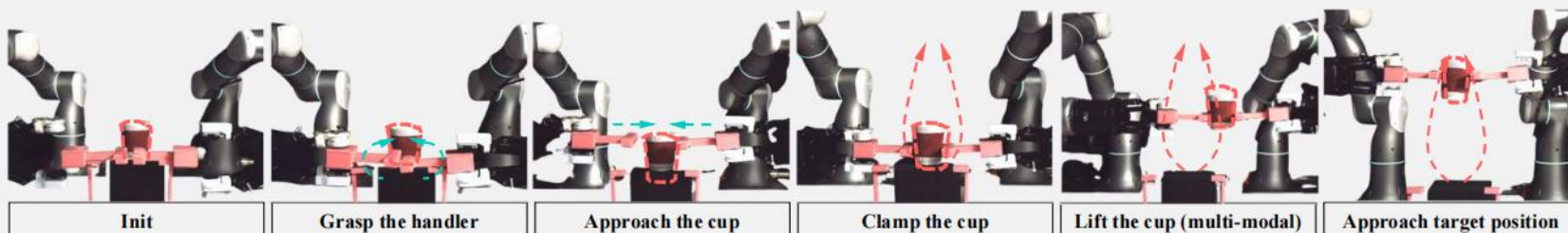
Task1: Peeling



Task2: Wiping



Task3: Bimanual Lifting



# 局限性与未来工作

- TactAR 虽然能在 AR 中展现触觉 / 力，但远不如直接用人手操作直观：未来可以从进一步降低延迟等方向优化；目前系统主要针对两指夹持：扩展到多指灵巧手 + 高维触觉是一个重要方向
- Fast policy 现阶段只处理高频触觉 / 力：之后可以考虑高频视觉（如高速相机）也接入 fast 通路；当前 RDP 只做单任务：未来可以把 RDP 嵌进 VLA（vision-language-action）模型，比如把原来的 tokenizer 换成类似 asymmetric tokenizer，使大模型策略也具备触觉驱动的 reactive 能力