TACFRS: PLUG-IN FLOW REVERSAL FOR HIGH-FREQUENCY TACTILE REVISION OF VLA MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models offer strong behavioral priors for robotic manipulation, but contact-rich tasks demand rapid feedback that infrequent visual policy updates cannot readily provide. We introduce Tactile Flow Reversal Steering (TacFRS), a lightweight framework that equips pretrained flow-based VLAs with high-frequency tactile action correction while keeping the base policy frozen. TacFRS uses finite-step flow reversal as an action encoding mechanism, mapping predicted action chunks into observation-conditioned structured noise. This representation connects a slow VLA to a fast tactile-conditioned decoder through an asymmetric encoding–decoding architecture. Using this representation, the latest tactile observations, and proprioception, the decoder revises each action between base-policy updates without requiring the VLA to learn a new latent action space. To preserve pretrained behavior while enabling contact-dependent adaptation, we introduce tactile-gated supervision that interpolates between base-policy actions and expert demonstrations according to tactile deviation from a contact-free reference, together with conditional flow matching and endpoint reconstruction objectives. Across four real-world single-arm and bimanual manipulation tasks, TacFRS improves every evaluated task for both backbones, raising average success by 12.50 percentage points for SmolVLA (28.75% to 41.25%) and by 13.75 points for $\pi_{0.5}$ (53.75% to 67.50%). Offline analyses further show that low tactile deviations leave base-policy actions nearly unchanged, whereas high deviations induce larger corrections that reduce ground-truth action error on average.

1 INTRODUCTION

Vision–language–action models (VLAs) provide robots with a general interface for generating actions from visual observations, but reliable physical manipulation also requires contact sensing. Tactile sensing provides complementary information about local contact, enabling robots to adjust actions after contact occurs (Calandra et al., 2018; Lee et al., 2019). However, integrating tactile into VLAs must satisfy three requirements: synchronized visual-tactile demonstrations are harder to collect at scale than vision-only ones (Jones et al., 2025; Higuera et al., 2025); vision and language should describe the global scene and task intent, while touch mainly reflects local contact constraints (Xue et al., 2025; Wu et al., 2026; Zhang et al., 2026); VLAs typically execute actions open-loop with low-frequency updates, whereas contact dynamics can change much faster and require higher-frequency closed-loop feedback (Xue et al., 2025; Li et al., 2026b; Wang et al., 2026a).

Existing works have explored these directions at different levels. N_0-VTLA, Sparsh, and UniT learn tactile priors and transferable representations through pretraining (NeoteAI Team & Fudan TEAI Team, 2026; Higuera et al., 2025; Xu et al., 2025); TouchGuide and DECO further integrate touch into action generation (Zhang et al., 2026; Li et al., 2026a); and VLA-Touch uses a separate controller to refine VLA actions (Bi et al., 2025). However, these methods have not jointly achieved high-frequency tactile closed-loop control independent of base-policy updates and lightweight adaptation using only small-scale data. Reactive Diffusion Policy (RDP) combines low-frequency diffusion over action latents learned by an asymmetric tokenizer with high-frequency tactile-conditioned decoding, enabling closed-loop feedback within action chunks (Xue et al., 2025). However, its slow policy must be trained to predict tokenizer-specific latents, limiting reuse of pretrained VLAs and their behavioral priors. This raises a central question: **Can we directly use a VLA, which is**

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

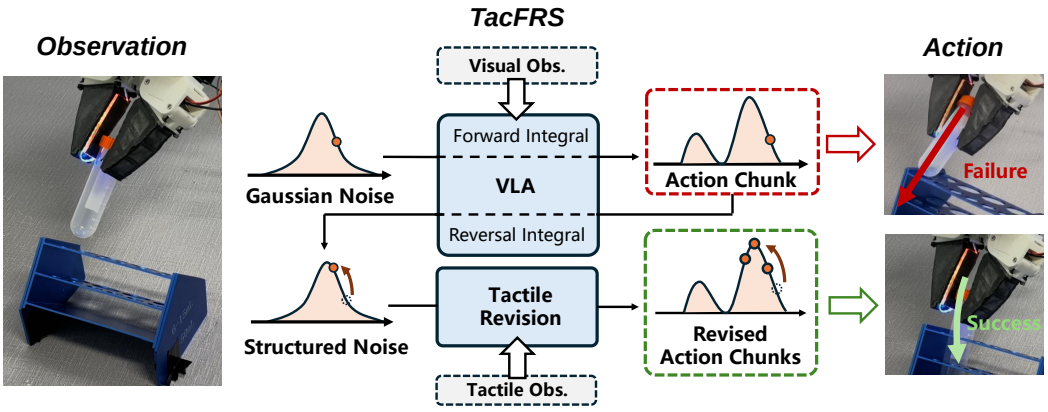


Figure 1: **Overview of TacFRS.** Flow reversal encodes a frozen VLA's predicted actions into structured noise, which a tactile-conditioned decoder uses for high-frequency action revision. The insertion example illustrates tactile-guided correction.

pretrained on large-scale vision-only demonstrations, to construct action representations and train an independent, lightweight tactile-conditioned action revision model that operates at high frequency without updating the VLA model?

One key to addressing this problem lies in designing an effective information interface between the VLA and the action revision network. Conditioning on vision, touch, and VLA-generated actions may let the revision network bypass touch by relying on visual or state cues (Jones et al., 2025; Huang et al., 2022), while using only touch and VLA-generated actions may provide insufficient task context for choosing the appropriate correction (Wu et al., 2026). Flow reversal offers inspiration for constructing such an interface (Rout et al., 2024; Wang et al., 2025). In robot manipulation, finite-step reverse flow integration has been shown to approximately preserve reference behavior in structured noise, which can then be used to steer outputs toward neighboring action modes of the base policy (Tang et al., 2026). Subsequent work has further used noise representations to guide behavior generation (Lu et al., 2026; Murray et al., 2026; Cao et al., 2026). These findings motivate us to use structured noise obtained through finite-step approximate flow reversal as an action representation.

Building on these ideas, **we introduce Tactile Flow Reversal Steering (TacFRS), a plug-and-play method for tactile-guided action revision that attaches to existing flow-based VLAs without modifying the base policy.** The VLA first generates preliminary actions from vision, language, and proprioceptive state, and then integrates them backward under the same observation conditions to obtain approximate structured noise. TacFRS treats this flow reversal as low-frequency action encoding and uses the resulting structured noise to steer the subsequent tactile revision network. Initialized with the structured noise, the tactile-conditioned revision network generates revised actions at high frequency, serving as an action decoder. Together, VLA flow reversal and the tactile revision network form an asymmetric conditional encoder-decoder pair, in which the structured noise serves as a high-level latent action representation. To guide the tactile revision network to learn appropriate correction directions under different tactile conditions, we introduce tactile-gated supervision that encourages reconstruction of VLA-generated actions when little correction is needed and assigns greater weight to expert-action supervision when contact requires adjustment.

Our main contributions are as follows:

1. We introduce TacFRS, which turns finite-step flow reversal of a frozen pretrained VLA into an action encoder and connects low-frequency policy inference with high-frequency tactile revision through asymmetric encoding and decoding. We also design tactile-gated supervision to balance preserving base behavior and learning contact-related corrections.

2. On four real-robot contact-rich tasks, TacFRS improves both evaluated VLAs on every task (average success gains of 13.75 and 12.50 percentage points for $\pi_{0.5}$ and SmolVLA); with $\pi_{0.5}$, it also outperforms the compared visuo-tactile fusion policies on all four tasks.

3. Through action reconstruction, observation-condition swapping, and comparative loss analysis, we provide evidence that the inverted representation retains base-action and observation-dependent information, and we distinguish the influence of touch on output behavior from its contribution to action prediction quality.

2 Related Work

Tactile-augmented policies. Tactile sensing complements vision with contact cues (Yuan et al., 2017; Lambeta et al., 2020). Visuo-tactile learning supports manipulation (Calandra et al., 2017; 2018; Lee et al., 2019; Li et al., 2022; Chen et al., 2023; Huang et al., 2024). Multimodal datasets support tactile learning (Yang et al., 2022; Gao et al., 2023; Fu et al., 2024; Wu et al., 2025). Pre-training learns reusable tactile representations (Guzey et al., 2023; Yang et al., 2024; Zhao et al., 2024; Higuera et al., 2025; Feng et al., 2025; Liu et al., 2025; Feng et al., 2026). Language grounding and world models enable reasoning and prediction (Yu et al., 2024; Higuera et al., 2026; Huang et al., 2026). VT-Refine fine-tunes policies; DECO adds adapters; VLA-Touch combines semantic feedback with diffusion-based action refinement (Huang et al., 2025; Li et al., 2026a; Bi et al., 2025). TacFRS instead links a frozen VLA to a tactile revision network through inversion, with tactile-adaptive supervision balancing behavior preservation and contact-dependent correction.

Slow–fast visuo-tactile control. Chunked and hierarchical policies model extended behavior (Chi et al., 2023; Zhao et al., 2023; Ze et al., 2024; Ma et al., 2024). Generalist VLAs supply behavioral priors (Kim et al., 2024; Black et al., 2024; Kim et al., 2025). Distillation and asynchronous execution reduce latency (Prasad et al., 2024; Black et al., 2025). Force-aware policies support reactive control (He et al., 2024; Chen et al., 2026; Zhuo et al., 2026; Wang et al., 2026b; Zhou et al., 2026). LIFT adds a force-conditioned action expert (Wang et al., 2026a); FAVLA adapts action-expert frequency to predicted force changes (Li et al., 2026b). Reactive Diffusion Policy combines tokenizer-based latent diffusion with fast tactile decoding (Xue et al., 2025). TacFRS uses approximate VLA flow inversion and a separate tactile decoder to revise actions between base-policy updates, preserving the original action-generation interface.

Flow inversion and policy steering. Diffusion and flow models connect data with noise (Ho et al., 2020; Song et al., 2021a;b; Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023). Image inversion and editing preserve reference structure (Meng et al., 2022; Hertz et al., 2022; Mokady et al., 2023; Wallace et al., 2023; Rout et al., 2024; Wang et al., 2025). Guidance and policy optimization steer generation and decision-making (Dhariwal & Nichol, 2021; Ho & Salimans, 2022; Janner et al., 2022; Ajay et al., 2023; Wagenmaker et al., 2025). Flow Reversal Steering inverts reference actions into noise, then decodes related action modes (Tang et al., 2026). TouchGuide and ViTaL apply tactile steering (Zhang et al., 2026; Wu et al., 2026). TacFRS learns supervised contact-dependent revisions from inverted representations using a separate tactile-conditioned flow decoder.

3 Preliminaries

Flow Matching and Reversal. Flow matching learns a velocity field that transports Gaussian noise to actions conditioned on observation $\boldsymbol{o}$ (Lipman et al., 2023):

$$\frac{\mathrm{d}\boldsymbol{a}^{(t)}}{\mathrm{d}t} = \boldsymbol{v}_\theta\Big(\boldsymbol{a}^{(t)}, t \mid \boldsymbol{o}\Big), \qquad \boldsymbol{a}^{(0)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{1}$$

where $t \in [0,1]$ is flow time. The exact flow map μ_θ gives $\boldsymbol{a}^{(1)} = \mu_\theta(\boldsymbol{a}^{(0)}, \boldsymbol{o})$. Under well-posed dynamics, integrating the same velocity field backward from $t = 1$ to 0 with $\boldsymbol{o}$ fixed yields $\hat{\boldsymbol{a}}^{(0)} \approx \mu_\theta^{-1}(\boldsymbol{a}^{(1)}, \boldsymbol{o})$ (Wang et al., 2025; Deng et al., 2024; Duan et al., 2026). Following Tang et al. (2026), we view the recovered latent as **structured noise** retaining the original action sequence's behavioral structure, and use it to represent the VLA's high-level task strategy.

Asymmetric Conditional Encoding. Latent-variable models can use different conditions for encoding and decoding:

$$\boldsymbol{z} = E(\boldsymbol{x}, \boldsymbol{c}_{\mathrm{enc}}), \qquad \hat{\boldsymbol{y}} = D(\boldsymbol{z}, \boldsymbol{c}_{\mathrm{dec}}), \tag{2}$$

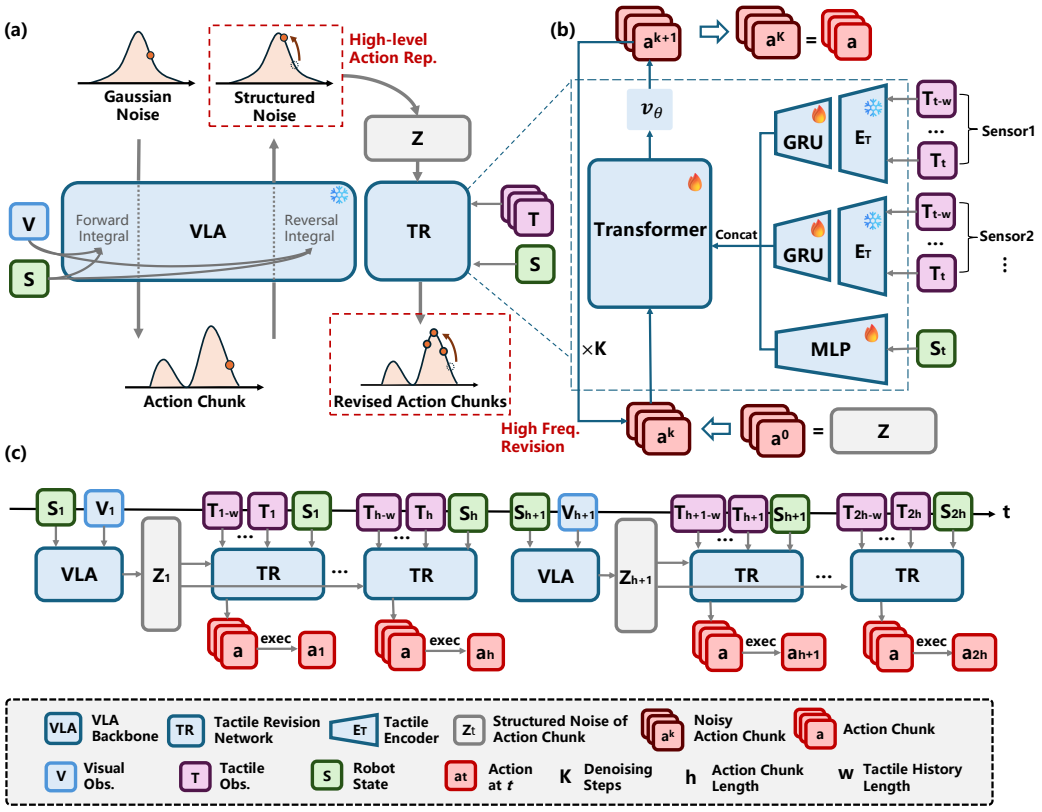


Figure 2: **TacFRS architecture and execution.** **(a)** VLA flow reversal produces structured noise for tactile revision. **(b)** A Transformer decoder combines tactile-history features and robot state to generate revised actions. **(c)** Between low-frequency VLA updates, TR performs high-frequency tactile correction by repeatedly decoding the same structured noise with the latest tactile feedback and robot state, executing only the current-step action.

where the conditions may differ in modality. This design recombines encoded information with decoding-time attributes, as used in conditional generation and disentangled representation learning (Lample et al., 2017; Zhu et al., 2017; Huang et al., 2018; Lee et al., 2018).

In latent plan and action models, high-level task intent can be encoded into a latent action representation based on the overall observation, and subsequently instantiated into concrete low-level actions by a decoder conditioned on task-relevant observations such as tactile feedback, visual inputs, or robot states (Lynch et al., 2020; Zhao et al., 2023; Xue et al., 2025).

4 Method

Tactile Flow Reversal Steering (TacFRS) enables high-frequency tactile correction of VLA-generated actions through a plug-in tactile revision (TR) network while preserving the capabilities of the underlying VLA. In Sec. 4.1, we detail the core components and policy architecture of TacFRS. In Sec. 4.2, we describe the loss functions designed to enable tactile correction while retaining the VLA's capabilities.

4.1 TacFRS Policy Architecture

Generating Structured Noise. Flow reversal produces a *structured noise* that retains the information required to reconstruct the VLA action while depending on the observation under which the action is generated (Appendix C.4). We therefore use it as the interface between the frozen VLA and the tactile revision (TR) network, preserving the VLA's high-level action strategy while allowing tactile feedback to modify its execution.

At each VLA update, the policy receives visual observations $\boldsymbol{V}_t$, proprioceptive states $\boldsymbol{S}_t$, a language instruction ℓ, and samples $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. It first generates an action chunk and then reverses the same observation-conditioned flow (Fig. 2(a)):

$$\begin{aligned} \boldsymbol{a}_t^{\mathrm{VLA}} &= \mu_\theta(\boldsymbol{\epsilon}_t, \boldsymbol{V}_t, \boldsymbol{S}_t, \ell), \\ \boldsymbol{z}_t &\approx \mu_\theta^{-1}(\boldsymbol{a}_t^{\mathrm{VLA}}, \boldsymbol{V}_t, \boldsymbol{S}_t, \ell), \end{aligned} \qquad \boldsymbol{a}_t^{\mathrm{VLA}}, \boldsymbol{z}_t \in \mathbb{R}^{H\times d}. \tag{3}$$

Here, μ_θ denotes the VLA flow map parameterized by θ, while μ_θ^{-1} denotes its numerical approximation obtained by reverse integration. H and d are the action-chunk length and action dimension, respectively, and $\boldsymbol{z}_t$ is the resulting structured noise.

Tactile Revision Network. The plug-in TR network is also a flow-matching model (Fig. 2(b)). Its role is to decode the VLA-derived structured noise under the latest tactile and proprioceptive observations, thereby producing contact-dependent corrections without re-running the VLA. With n tactile sensors and a history window of w frames, we denote the tactile history by $\boldsymbol{T}_t = \{\boldsymbol{T}_i^{(j)}\}_{i=t-w+1,\dots,t;\, j=1,\dots,n}$. The TR network maps $\boldsymbol{a}_t^{\mathrm{TR}} = \mu_\phi(\boldsymbol{z}_t, \boldsymbol{T}_t, \boldsymbol{S}_t)$, where μ_ϕ denotes the TR flow map parameterized by ϕ.

The network uses a Transformer backbone. A pretrained tactile encoder (Appendix E) maps the tactile images to normalized latent vectors $\{\bar{\boldsymbol{e}}_i^{(j)}\}_{i=t-w+1,\dots,t;\, j=1,\dots,n}$. These vectors are processed temporally by a GRU, retaining the final hidden state $\boldsymbol{h}_t^{(j)}$ for each sensor. An MLP encodes the proprioceptive state as $\boldsymbol{s}_t$. The sensor hidden states $\{\boldsymbol{h}_t^{(j)}\}_{j=1}^n$ and $\boldsymbol{s}_t$ are concatenated to condition the Transformer.

Execution Timing. TacFRS separates low-frequency VLA planning from high-frequency tactile revision. The VLA generates one structured-noise chunk through forward and reverse integration, and this chunk remains fixed during its execution. Before each action is sent to the robot, the TR network re-decodes the same structured noise using the latest tactile history and proprioceptive state, generates a revised action chunk, and executes only the action corresponding to the current time step. This enables high-frequency tactile correction without repeatedly querying the VLA.

4.2 Learning Objective and Optimization

Training protocol. We fine-tune the vision-only VLA, freeze it, and train the TR network on a smaller demonstration set. Only the tactile-history encoder, reference/state projections, and conditional velocity field are optimized; the base VLA and tactile image encoder remain frozen.

Tactile-adaptive supervision. The key idea of the TacFRS objective is tactile-adaptive supervision, which interpolates between frozen base-policy actions and demonstrations according to tactile deviation from a contact-free reference. We first present the single-arm formulation and then its bimanual extension. Let $\mathcal{S}$ denote the arm's tactile sensors, and $\boldsymbol{T}_t^{(j)}$, $\boldsymbol{T}_{\mathrm{ref}}^{(j)}$ the current and contact-free reference images. A frozen tactile encoder E_ψ produces

$$\bar{\boldsymbol{e}}_q^{(j)} = \frac{E_\psi(\boldsymbol{T}_q^{(j)})}{\max(\|E_\psi(\boldsymbol{T}_q^{(j)})\|_2, \epsilon)}, \qquad q \in \{t, \mathrm{ref}\}. \tag{4}$$

We define the tactile deviation and gate as

$$s(t) = \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}}\left(1 - \left(\bar{\boldsymbol{e}}_t^{(j)}\right)^\top \bar{\boldsymbol{e}}_{\mathrm{ref}}^{(j)}\right), \qquad \alpha(t) = \sigma\left(\frac{s(t)-\tau}{T_g}\right), \tag{5}$$

where τ and T_g control the gate midpoint and sharpness. The supervision endpoint is

$$\boldsymbol{Y}_t = \big(1-\alpha(t)\big)\boldsymbol{a}_t^{\mathrm{VLA}} + \alpha(t)\boldsymbol{a}_t^{\mathrm{GT}}, \tag{6}$$

where $\boldsymbol{a}_t^{\mathrm{GT}}$ is the aligned demonstration chunk.

For $(\boldsymbol{z}_t, \boldsymbol{Y}_t)$ and $u \sim \mathcal{U}(0,1)$, we use

$$\boldsymbol{x}_{t,u} = (1-u)\boldsymbol{z}_t + u\boldsymbol{Y}_t, \qquad \boldsymbol{v}_{t,u}^* = \boldsymbol{Y}_t - \boldsymbol{z}_t. \tag{7}$$

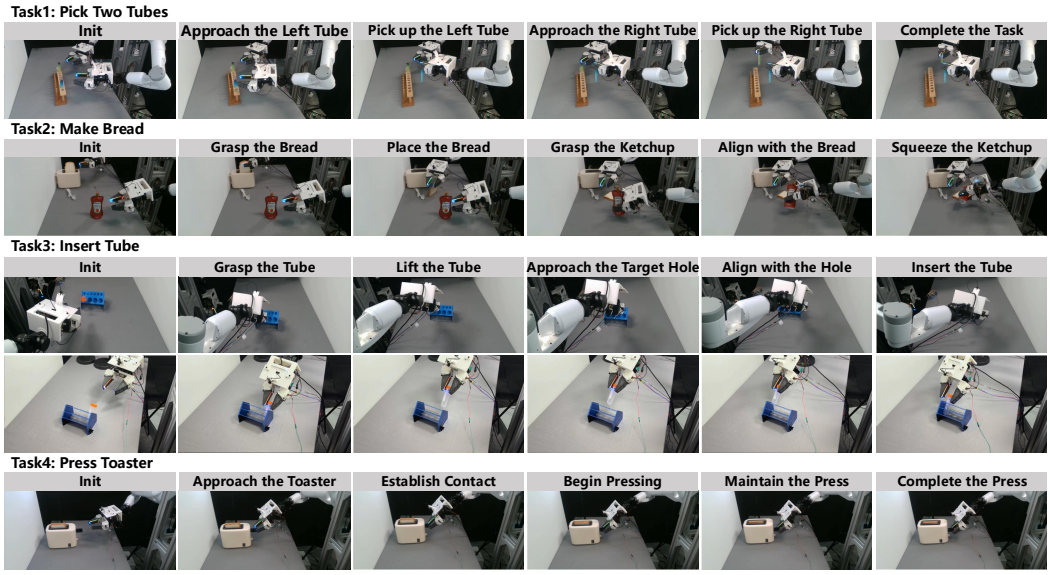


Figure 3: Four real-world tasks: bimanual Pick Two Tubes and Make Bread, and single-arm Insert Tube and Press Toaster.

With $\|\boldsymbol{E}\|_{\text{act}}^2 = \|\boldsymbol{E}\|_F^2/(Hd)$ and TR condition $\boldsymbol{C}_t$,

$$\mathcal{L}_{\text{FM}} = \mathbb{E}\left[\|\boldsymbol{v}_\phi(\boldsymbol{x}_{t,u}, u; \boldsymbol{C}_t) - (\boldsymbol{Y}_t - \boldsymbol{z}_t)\|_{\text{act}}^2\right]. \tag{8}$$

We additionally supervise the decoded endpoint because local velocity matching does not directly guarantee accurate recovery of the final action chunk after numerical integration. The endpoint loss therefore provides direct action-level supervision and penalizes accumulated integration errors:

$$\begin{aligned} \widehat{\boldsymbol{a}}_{\phi,t} &= \text{ODESolve}_K\big(\boldsymbol{v}_\phi, \boldsymbol{z}_t; \boldsymbol{C}_t, 0\to 1\big), \\ \mathcal{L}_{\text{end}} &= \mathbb{E}\left[\|\widehat{\boldsymbol{a}}_{\phi,t} - \boldsymbol{Y}_t\|_{\text{act}}^2\right]. \end{aligned} \tag{9}$$

Gradients propagate through the K integration steps. The final objective is

$$\mathcal{L}_{\text{TacFRS}} = \mathcal{L}_{\text{FM}} + \lambda_{\text{end}}\mathcal{L}_{\text{end}}. \tag{10}$$

For bimanual tasks, each arm uses its own tactile gate to mix base-policy and demonstration actions; the concatenated targets are jointly generated by one velocity field conditioned on both arms' tactile inputs and states.

5 Experiments

Our experiments address the following questions:

Q1: Does TacFRS outperform alternative tactile fusion methods?
Q2: Can TacFRS be applied to different VLA models and improve their performance?
Q3: Does TacFRS improve action accuracy with tactile input and preserve VLA predictions without it?
Q4: What are the contributions of the structured noise representation and the loss design?

To answer these questions, we design four contact-rich tasks as follows (Figure 3): **Pick Two Tubes** (bimanual sequential tube extraction), **Make Bread** (bimanual bread handling and squeezing a ketchup container above the bread), **Insert Tube** (single-arm tube extraction and insertion), and **Press Toaster** (single-arm sustained pressing). Detailed task descriptions and evaluation settings are provided in Appendix A.

5.1 Experimental Setup

Hardware platform. We use two seven-degree-of-freedom arms with gripper-mounted tactile sensors as in ViTaMin-B (Li et al., 2025) and wrist cameras as the sole source of visual observations (Figure 4).



Figure 4: Data collection and robot systems.

Data and baselines. For policy training, we collect 1,050/1,507/1,211/1,600 UMI demonstrations with tactile observations for Tasks 1–4, respectively. For TR training, we use 300 demonstrations for Tasks 1–4, respectively. We evaluate SmolVLA and $\pi_{0.5}$ with and without TacFRS, alongside RDP and DECO. To accommodate our markerless sensors, we replace RDP's marker-based deformation-field input with tactile representations.

Evaluation protocol. All policies execute actions at 30 Hz with a shared 80 Hz low-level controller. SmolVLA and $\pi_{0.5}$ use 50-step action chunks with a nominal update rate of 0.6 Hz. TacFRS with $\pi_{0.5}$ retains this chunk update rate while running TR inference at 30 Hz to revise each action before execution using the latest tactile feedback. Based on the flow inversion solver evaluation in Appendix C, we use 15-step SlerpFlow for TacFRS flow reversal. We report success rate and task progress over 20 trials per policy per task with varied object poses. Evaluation lighting and scene layouts differ substantially from data collection.

5.2 Main Results

5.2.1 Comparison with Alternative Tactile Fusion Methods

TacFRS outperforms the evaluated tactile fusion baselines (Q1). Under our evaluation conditions, $\pi_{0.5}$ with TacFRS achieves the highest observed success rate on all four tasks (Figure 5), averaging 67.50%, compared with 53.75% for both DECO and the base policy and 12.50% for RDP.

The RDP implementation does not use a large-scale pretrained VLA. We hypothesize that its low success partly reflects sensitivity to the visual domain shift, whereas TacFRS builds on pretrained VLA predictions and learns contact-related corrections. DECO couples tactile cross-attention to action-chunk generation, while TacFRS uses the latest tactile observations to correct actions between VLA updates.

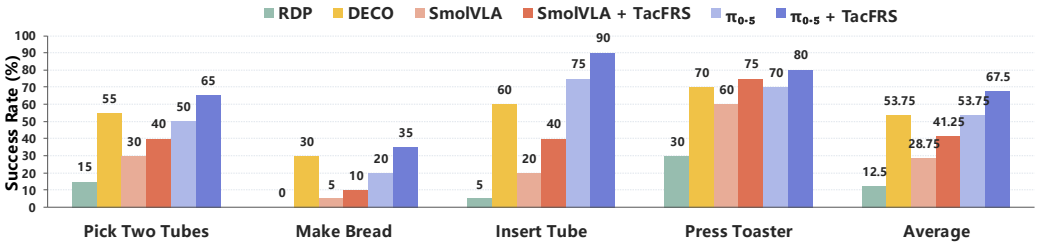


Figure 5: Real-world task success rates, with 20 trials per policy per task.

5.2.2 Cross-Backbone Applicability

TacFRS improves both evaluated VLA backbones (Q2). With separately trained TacFRS modules, both $\pi_{0.5}$ and SmolVLA improve on all four tasks (Table 1). TacFRS raises average success by 13.75 percentage points for $\pi_{0.5}$ (53.75% to 67.50%) and by 12.50 points for SmolVLA (28.75% to 41.25%).

Qualitative improvements include more stable grasping, pose adjustments after contact during insertion, and improved contact maintenance during pressing. Make Bread remains challenging (35% and 10% success for the two augmented backbones, respectively): local contact improvements alone do not ensure full-task completion. Task-wise failure and correction analyses appear in Appendix B.

Table 1: Real-robot success rates across VLA backbones, with 20 trials per task. Average is the unweighted mean across tasks.

Method	Pick Two Tubes	Make Bread	Insert Tube	Press Toaster	Average
$\pi_{0.5}$	50%	20%	75%	70%	53.75%
$\pi_{0.5}$ + TacFRS	**65%**	**35%**	**90%**	**80%**	**67.50%**
SmolVLA	30%	5%	20%	60%	28.75%
SmolVLA + TacFRS	**40%**	**10%**	**40%**	**75%**	**41.25%**

5.2.3 Training and Action Behavior on a Single-Arm Task

Gate-dependent action correction (Q3). On single-arm Task 4, TacFRS achieves lower normalized validation action MSE than the frozen VLA at every evaluated epoch, suggesting that it learns effective action corrections not captured by the frozen base policy (Figure 6).

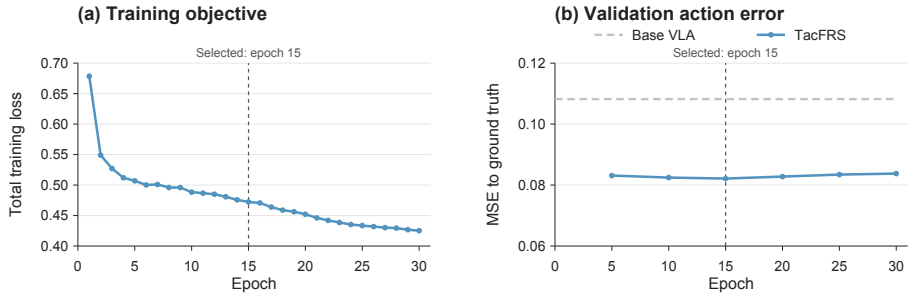


Figure 6: Single-arm Task 4: (a) training loss and (b) validation action MSE against ground truth. Dashed lines mark the selected epoch-15 checkpoint.

Gate-wise analysis at epoch 15 (Figure 7) shows negligible changes to base actions and nearly unchanged ground-truth MSE at low gate weights. At high weights, the mean squared action difference is approximately 0.0145, while ground-truth MSE drops 23%, from 0.124 to 0.096. These results suggest preservation of low-gate base actions and beneficial high-gate corrections on average.

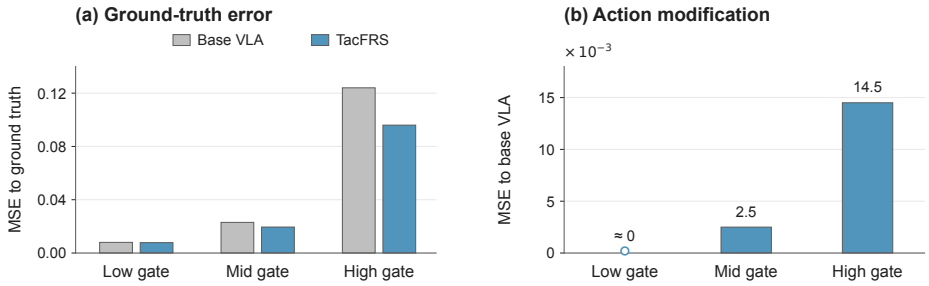


Figure 7: Task 4 validation results at epoch 15 by gate weight: (a) mean ground-truth MSE and (b) mean squared difference from base-VLA actions. Panel scales differ; (b) uses a 10^{-3} factor.

5.2.4 Ablation Studies

Both the structured-noise and loss ablation studies use the $\pi_{0.5}$ backbone.

Structured noise ablation (Q4). We compare TacFRS with a direct correction baseline that uses the same tactile revision network architecture as TacFRS. Instead of using structured noise and flow matching, this baseline directly predicts corrected actions from VLA-predicted actions and tactile embeddings. It is trained using only the endpoint loss $\mathcal{L}_{\text{end}}$ from TacFRS, with the same adaptive weighting scheme for action supervision.

This baseline achieves a 0% success rate across all four tasks, compared with 35–90% for TacFRS (Table 1); its actions deviate from normal task execution. For example, in Task 4, the model skips the toaster-localization step and immediately executes a downward pressing motion, following a steeper trajectory at a higher speed than in normal execution. A possible explanation is that these inputs provide insufficient context to distinguish the corrections required under different scene conditions. In contrast, TacFRS uses observation-conditioned flow inversion to obtain structured noise carrying both action- and vision-related information (Appendix C.4), providing the tactile-conditioned decoder with visual context for context-dependent corrections.

Loss Ablation (Q4). On Task 3, we compare the full objective with three ablations: (1) removing the endpoint reconstruction loss; (2) replacing tactile-adaptive supervision with ground-truth (GT) targets only; and (3) replacing the adaptive gate with a constant equal to its mean over the training set. The full objective yields the lowest MSE to base-VLA predictions at low gate weights and to GT actions at high weights (Figure 8), suggesting that endpoint reconstruction and tactile-adaptive supervision help balance base-policy preservation and tactile-dependent correction. The three ablations lower real-robot success from 90% to 40%, 50%, and 65%, respectively (Table 2).

Table 2: Real-robot success rates on Task 3 (Insert Tube) using the $\pi_{0.5}$ backbone, comparing the full TacFRS training objective with its loss ablations. Success denotes full-task completion.

Variant	Success rate (%)
TacFRS (full)	90%
w/o Endpoint Reconstruction	40%
w/o Tactile Gate (GT only)	50%
Constant Tactile Gate	65%

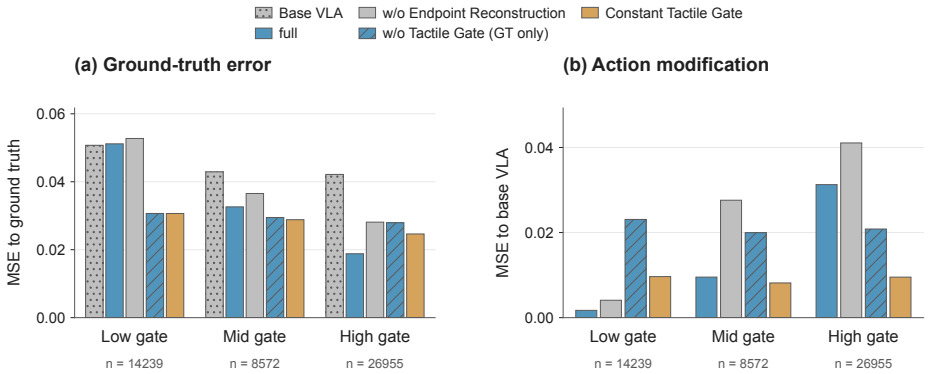


Figure 8: Loss ablations on Task 3: action MSE relative to GT and base-VLA predictions across low-, medium-, and high-gate groups.

6 Conclusion

We presented TacFRS, a plug-in framework for high-frequency tactile correction of frozen VLAs through finite-step flow reversal and independent tactile decoding of observation-conditioned structured noise. Tactile-gated supervision and endpoint reconstruction balance base-policy preservation and expert-guided correction. Across four real-world single-arm and bimanual tasks, TacFRS improves both backbones on every task, with average success gains of 13.75 and 12.50 percentage points for $\pi_{0.5}$ and SmolVLA, respectively.

Limitations. TacFRS requires flow-based backbones and adds reverse-integration overhead. Our evaluation covers four tasks on a single robot platform and tactile sensor type; transfer across tasks, embodiments, and sensors remains untested. The tactile gate measures contact deviation rather than correction necessity, and local corrections do not guarantee long-horizon success.

AI USE STATEMENT

In this work, we used generative AI tools to assist with research ideation and execution, and with proving mathematical claims. We also used these tools for retrieval and discovery of related literature, to draft sections of the paper, and to aid and polish the writing. The authors take full responsibility for the final content of this work, including all text, mathematical claims, proofs, and other research artifacts produced with the assistance of generative AI.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is Conditional Generative Modeling All You Need for Decision-Making? In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sP1fo2K9DFG.

Michael S. Albergo and Eric Vanden-Eijnden. Building Normalizing Flows with Stochastic Interpolants. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.15571.

Jianxin Bi, Kevin Yuchen Ma, Ce Hao, Mike Zheng Shou, and Harold Soh. Vla-touch: Enhancing vision-language-action models with dual-level tactile feedback, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies. *arXiv preprint arXiv:2506.07339*, 2025. URL https://arxiv.org/abs/2506.07339.

Roberto Calandra, Andrew Owens, Manu Upadhyaya, Wenzhen Yuan, Justin Lin, Edward H. Adelson, and Sergey Levine. The Feeling of Success: Does Touch Sensing Help Predict Grasp Outcomes? In *Proceedings of the 1st Annual Conference on Robot Learning*, 2017. URL https://proceedings.mlr.press/v78/calandra17a.html.

Roberto Calandra, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H. Adelson, and Sergey Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *IEEE Robotics and Automation Letters*, 3(4):3300–3307, 2018. doi: 10.1109/LRA.2018.2852779.

Haidong Cao, Wenjun Cao, Quanhao Li, Sicheng Xie, Zhiying Du, Jiaqi Leng, Zuxuan Wu, and Yu-Gang Jiang. Segdiff: Segmented trajectory diffusion for consistent and adaptive robot manipulation, 2026.

Wendi Chen, Han Xue, Yi Wang, Fangyuan Zhou, Jun Lv, Yang Jin, Shirun Tang, Chuan Wen, and Cewu Lu. ImplicitRDP: An end-to-end visual-force diffusion policy with structural slow-fast learning. *IEEE Robotics and Automation Letters*, 2026. URL https://implicit-rdp.github.io/.

Yizhou Chen, Andrea Sipos, Mark Van der Merwe, and Nima Fazeli. Visuo-tactile transformers for manipulation. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 2026–2040. PMLR, 2023. URL https://proceedings.mlr.press/v205/chen23d.html.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. *arXiv preprint arXiv:2303.04137*, 2023. URL https://arxiv.org/abs/2303.04137.

Yingying Deng, Xiangyu He, Changwang Mei, Peisong Wang, and Fan Tang. FireFlow: Fast inversion of rectified flow for image semantic editing. *arXiv preprint arXiv:2412.07517*, 2024. URL https://arxiv.org/abs/2412.07517.

Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. *arXiv preprint arXiv:2105.05233*, 2021. URL https://arxiv.org/abs/2105.05233.

Wenbin Duan, Yan Shu, Zhuoyuan Fu, Fangmin Zhao, Yan Li, Yaru Zhao, and Binyang Li. SlerpFlow: Spherical trajectory correction for rectified flow inversion. *arXiv preprint arXiv:2607.21326*, 2026. URL https://arxiv.org/abs/2607.21326.

Ruoxuan Feng, Jiangyu Hu, Wenke Xia, Tianci Gao, Ao Shen, Yuhao Sun, Bin Fang, and Di Hu. AnyTouch: Learning Unified Static-Dynamic Representation across Multiple Visuo-tactile Sensors. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.12191.

Ruoxuan Feng, Yuxuan Zhou, Siyu Mei, Dongzhan Zhou, Pengwei Wang, Shaowei Cui, Bin Fang, Guocai Yao, and Di Hu. AnyTouch 2: General Optical Tactile Representation Learning For Dynamic Tactile Perception. *arXiv preprint arXiv:2602.09617*, 2026. URL https://arxiv.org/abs/2602.09617.

Letian Fu, Gaurav Datta, Huang Huang, William Chung-Ho Panitch, Jaimyn Drake, Joseph Ortiz, Mustafa Mukadam, Mike Lambeta, Roberto Calandra, and Ken Goldberg. A Touch, Vision, and Language Dataset for Multimodal Alignment. *arXiv preprint arXiv:2402.13232*, 2024. URL https://arxiv.org/abs/2402.13232.

Ruohan Gao, Yiming Dou, Hao Li, Tanmay Agarwal, Jeannette Bohg, Yunzhu Li, Li Fei-Fei, and Jiajun Wu. The ObjectFolder Benchmark: Multisensory Learning with Neural and Real Objects. *arXiv preprint arXiv:2306.00956*, 2023. URL https://arxiv.org/abs/2306.00956.

Irmak Guzey, Ben Evans, Soumith Chintala, and Lerrel Pinto. Dexterity from Touch: Self-Supervised Pre-Training of Tactile Representations with Robotic Play. *arXiv preprint arXiv:2303.12076*, 2023. URL https://arxiv.org/abs/2303.12076.

Zihao He, Hongjie Fang, Jingjing Chen, Hao-Shu Fang, and Cewu Lu. FoAR: Force-aware reactive policy for contact-rich robotic manipulation. *arXiv preprint arXiv:2411.15753*, 2024. URL https://arxiv.org/abs/2411.15753.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-Prompt Image Editing with Cross Attention Control. *arXiv preprint arXiv:2208.01626*, 2022. URL https://arxiv.org/abs/2208.01626.

Carolina Higuera, Akash Sharma, Chaithanya Krishna Bodduluri, Taosha Fan, Patrick Lancaster, Mrinal Kalakrishnan, Michael Kaess, Byron Boots, Mike Lambeta, Tingfan Wu, and Mustafa Mukadam. Sparsh: Self-supervised touch representations for vision-based tactile sensing. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 885–915. PMLR, 2025.

Carolina Higuera, Sergio Arnaud, Byron Boots, Mustafa Mukadam, Francois Robert Hogan, and Franziska Meier. Visuo-Tactile World Models. *arXiv preprint arXiv:2602.06001*, 2026. URL https://arxiv.org/abs/2602.06001.

Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. *arXiv preprint arXiv:2207.12598*, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *arXiv preprint arXiv:2006.11239*, 2020. URL https://arxiv.org/abs/2006.11239.

Binghao Huang, Yixuan Wang, Xinyi Yang, Yiyue Luo, and Yunzhu Li. 3D-ViTac: Learning Fine-Grained Manipulation with Visuo-Tactile Sensing. *arXiv preprint arXiv:2410.24091*, 2024. URL https://arxiv.org/abs/2410.24091.

Binghao Huang, Jie Xu, Iretiayo Akinola, Wei Yang, Balakumar Sundaralingam, Rowland O'Flaherty, Dieter Fox, Xiaolong Wang, Arsalan Mousavian, Yu-Wei Chao, and Yunzhu Li. VT-Refine: Learning Bimanual Assembly with Visuo-Tactile Feedback via Simulation Fine-Tuning. *arXiv preprint arXiv:2510.14930*, 2025. URL https://arxiv.org/abs/2510.14930.

Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision*, 2018.

Yu Huang, Junyang Lin, Chang Zhou, Hongxia Yang, and Longbo Huang. Modality competition: What makes joint training of multi-modal network fail in deep learning? (provably). In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9226–9259. PMLR, 2022.

Yunao Huang, Shiyu Sang, Haotao Lu, Suting Ni, Shijie Wu, Ziyang Guo, Ye Shi, and Jingya Wang. ViTacWorld: Scaling Visuo-Tactile World Models for Contact-Rich Robot Manipulation. *arXiv preprint arXiv:2607.22530*, 2026. URL https://arxiv.org/abs/2607.22530.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with Diffusion for Flexible Behavior Synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Joshua Jones, Oier Mees, Carmelo Sferrazza, Kyle Stachowicz, Pieter Abbeel, and Sergey Levine. Beyond sight: Finetuning generalist robot policies with heterogeneous sensors via language grounding, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. *arXiv preprint arXiv:2502.19645*, 2025. URL https://arxiv.org/abs/2502.19645.

Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg Kammerer, Dinesh Jayaraman, and Roberto Calandra. DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor with Application to In-Hand Manipulation. *arXiv preprint arXiv:2005.14679*, 2020. URL https://arxiv.org/abs/2005.14679.

Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. Fader Networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European Conference on Computer Vision*, 2018.

Michelle A. Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019. doi: 10.1109/ICRA.2019.8793485.

Chuanyu Li, Chaoyi Liu, Daotan Wang, Shuyu Zhang, Lusong Li, Zecui Zeng, Fangchen Liu, Jing Xu, and Rui Chen. Vitamin-b: A reliable and efficient visuo-tactile bimanual manipulation interface. *arXiv preprint arXiv:2511.05858*, 2025.

Hao Li, Yizhi Zhang, Junzhe Zhu, Shaoxiong Wang, Michelle A. Lee, Huazhe Xu, Edward Adelson, Li Fei-Fei, Ruohan Gao, and Jiajun Wu. See, Hear, and Feel: Smart Sensory Fusion for Robotic Manipulation. *arXiv preprint arXiv:2212.03858*, 2022. URL https://arxiv.org/abs/2212.03858.

Xukun Li, Yu Sun, Lei Zhang, Bosheng Huang, Yibo Peng, Yuan Meng, Haojun Jiang, Shaoxuan Xie, Guacai Yao, Alois Knoll, Zhenshan Bing, Xinlong Wang, and Zhenguo Sun. Deco: Decoupled multimodal diffusion transformer for bimanual dexterous manipulation with a plugin tactile adapter, 2026a.

Yao Li, Peiyuan Tang, Wuyang Zhang, Chengyang Zhu, Yifan Duan, Weikai Shi, Xiaodong Zhang, Zijiang Yang, Jianmin Ji, and Yanyong Zhang. Favla: A force-adaptive fast-slow vla model for contact-rich robotic manipulation, 2026b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Fangchen Liu, Chuanyu Li, Yihua Qin, Jing Xu, Pieter Abbeel, and Rui Chen. ViTaMIn: Learning contact-rich tasks through robot-free visuo-tactile manipulation interface. *arXiv preprint arXiv:2504.06156*, 2025. URL https://arxiv.org/abs/2504.06156.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Junjie Lu, Xinyao Qin, Yuhua Jiang, Kaixin Wang, Chuheng Zhang, Bin Liang, Jun Yang, Min Xu, and Li Zhao. Unified noise steering for efficient human-guided vla adaptation, 2026.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132. PMLR, 2020. URL https://proceedings.mlr.press/v100/lynch20a.html.

Xiao Ma, Sumit Patidar, Iain Haughton, and Stephen James. Hierarchical Diffusion Policy for Kinematics-Aware Multi-Task Robotic Manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/papers/Ma_Hierarchical_Diffusion_Policy_for_Kinematics-Aware_Multi-Task_Robotic_Manipulation_CVPR_2024_paper.pdf.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022. URL https://sde-image-editing.github.io/.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text Inversion for Editing Real Images using Guided Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6038–6047, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Mokady_NULL-Text_Inversion_for_Editing_Real_Images_Using_Guided_Diffusion_Models_CVPR_2023_paper.html.

Michael Murray, Daphne Chen, Simran Bagaria, Dean Fortier, Tess Hellebrekers, Galen Mullins, Harshavardhan Gajarla, Oier Mees, Maya Cakmak, and Andrey Kolobov. Flowdagger: Human-in-the-loop adaptation of generative robot policies in latent space, 2026.

NeoteAI Team and Fudan TEAI Team. N_0-VTLA: Scaling Vision-Tactile-Language-Action Model with Latent Tactile Tokens, 2026.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency Policy: Accelerated Visuomotor Policies via Consistency Distillation. *arXiv preprint arXiv:2405.07503*, 2024. URL https://arxiv.org/abs/2405.07503.

Litu Rout, Yujia Chen, Nataniel Ruiz, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Semantic image inversion and editing using rectified stochastic differential equations, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021b. URL https://arxiv.org/abs/2011.13456.

Andy Tang, William Chen, Andrew Wagenmaker, Chelsea Finn, and Sergey Levine. Improving robotic generalist policies via flow reversal steering, 2026.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering Your Diffusion Policy with Latent Space Reinforcement Learning. *arXiv preprint arXiv:2506.15799*, 2025. URL https://arxiv.org/abs/2506.15799.

Bram Wallace, Akash Gokul, and Nikhil Naik. EDICT: Exact Diffusion Inversion via Coupled Transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22532–22541, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Wallace_EDICT_Exact_Diffusion_Inversion_via_Coupled_Transformations_CVPR_2023_paper.html.

Jiangshan Wang, Junfu Pu, Zhongang Qi, Jiayi Guo, Yue Ma, Nisha Huang, Yuxin Chen, Xiu Li, and Ying Shan. Taming rectified flow for inversion and editing. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 64044–64058. PMLR, 2025.

Yi Wang, Wendi Chen, Zimo Wen, Han Xue, Xueqi Li, Wenye Yu, Zhijie Chen, Hao Yang, Jun Lv, Chuan Wen, and Cewu Lu. Never too late for force: Accelerating vla post-training with reactive force injection, 2026a.

Zhenan Wang, Yanzhe Wang, Meixuan Ren, Peng Li, Yang Liu, Yifei Nie, Limin Long, Yun Ye, Xiaofeng Wang, Zhen Zhu, and Huixu Dong. TacMamba: A Tactile History Compression Adapter Bridging Fast Reflexes and Slow VLA Reasoning. *arXiv preprint arXiv:2603.01700*, 2026b. URL https://arxiv.org/abs/2603.01700.

Longyan Wu, Checheng Yu, Jieji Ren, Li Chen, Ran Huang, Guoying Gu, and Hongyang Li. FreeTacMan: Robot-free Visuo-Tactile Data Collection System for Contact-rich Manipulation. *arXiv preprint arXiv:2506.01941*, 2025. URL https://arxiv.org/abs/2506.01941.

Yilin Wu, Zilin Si, Zeynep Temel, Oliver Kroemer, and Andrea Bajcsy. Inference-time policy steering via vision and touch, 2026.

Zhengtong Xu, Raghava Uppuluri, Xinwei Zhang, Cael Fitch, Philip Glen Crandall, Wan Shou, Dongyi Wang, and Yu She. Unit: Data efficient tactile representation with generalization to unseen objects. *IEEE Robotics and Automation Letters*, 2025.

Han Xue, Jieji Ren, Wendi Chen, Gu Zhang, Yuan Fang, Guoying Gu, Huazhe Xu, and Cewu Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.052.

Fengyu Yang, Chenyang Ma, Jiacheng Zhang, Jing Zhu, Wenzhen Yuan, and Andrew Owens. Touch and Go: Learning from Human-Collected Vision and Touch. *arXiv preprint arXiv:2211.12498*, 2022. URL https://arxiv.org/abs/2211.12498.

Fengyu Yang, Chao Feng, Ziyang Chen, Hyoungseob Park, Daniel Wang, Yiming Dou, Ziyao Zeng, Xien Chen, Rit Gangopadhyay, Andrew Owens, and Alex Wong. Binding Touch to Everything: Learning Unified Multimodal Tactile Representations. *arXiv preprint arXiv:2401.18084*, 2024. URL https://arxiv.org/abs/2401.18084.

Samson Yu, Kelvin Lin, Anxing Xiao, Jiafei Duan, and Harold Soh. Octopi: Object Property Reasoning with Large Tactile-Language Models. *arXiv preprint arXiv:2405.02794*, 2024. URL https://arxiv.org/abs/2405.02794.

Wenzhen Yuan, Siyuan Dong, and Edward H. Adelson. GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force. *Sensors*, 2017. URL https://www.mdpi.com/1424-8220/17/12/2762.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D Diffusion Policy: Generalizable Visuomotor Policy Learning via Simple 3D Representations. *arXiv preprint arXiv:2403.03954*, 2024. URL https://arxiv.org/abs/2403.03954.

Zhemeng Zhang, Jiahua Ma, Xincheng Yang, Xin Wen, Yuzhi Zhang, Boyan Li, Yiran Qin, Jin Liu, Can Zhao, Li Kang, Haoqin Hong, Zhenfei Yin, Philip Torr, Hao Su, Ruimao Zhang, and Daolin Ma. Touchguide: Inference-time steering of visuomotor policies via touch guidance, 2026.

Jialiang Zhao, Yuxiang Ma, Lirui Wang, and Edward H. Adelson. Transferable Tactile Transformers for Representation Learning Across Diverse Sensors and Tasks. *arXiv preprint arXiv:2406.13640*, 2024. URL https://arxiv.org/abs/2406.13640.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://roboticsproceedings.org/rss19/p016.html.

Jianyi Zhou, Feiyang Hong, Yunhao Li, Yicheng Zhao, Yongjue Cen, Zirui Liu, Jiakang Huang, Zirui Chen, Ruiyang Zhang, Weizhuo Zhu, Xuhua Song, and Shuo Yang. TouchWorld: A Predictive and Reactive Tactile Foundation Model for Dexterous Manipulation. *arXiv preprint arXiv:2607.07287*, 2026. URL https://arxiv.org/abs/2607.07287.

Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Lifeng Zhuo, Wendi Chen, Han Xue, Shirun Tang, Jun Lv, Cewu Lu, and Chuan Wen. FA-RDP: A frequency-adaptive reactive diffusion policy for contact-rich manipulation. *arXiv preprint arXiv:2607.28596*, 2026. URL https://arxiv.org/abs/2607.28596.

Appendix A. We provide detailed descriptions of the four real-world manipulation tasks, including task procedures, contact-related challenges, and evaluation settings.

Appendix B. We provide a task-wise analysis of real-robot failure modes and the observed corrections with TacFRS.

Appendix C. We compare flow inversion solvers in terms of noise recovery consistency, offline computational cost, and downstream action reconstruction to motivate our inversion configuration.

Appendix D. We summarize the training hyperparameters and deployment configurations of the VLA backbones and the DECO and RDP baselines.

Appendix E. We describe our visuotactile representation pretraining framework, including temporal tactile encoding, contrastive learning extensions, the training protocol, and cross-modal retrieval results.

A Task Details and Evaluation Settings

We describe the task procedures, contact-related challenges, and evaluation settings for the four tasks shown in Figure 3.

Pick Two Tubes (bimanual). The left and right arms sequentially grasp and extract their respective target tubes from a tube rack. This task requires stable grasping and extraction within a confined space, testing the policy's response to changes in grasp and contact conditions during extraction. During evaluation, we vary the positions of the rack and target tubes to assess performance across spatial layouts.

Make Bread (bimanual). The right arm removes a slice of bread from a toaster and places it on the table. The left arm then grasps a ketchup container, positions it above the bread, and completes a squeezing motion. The task involves grasping a deformable object, manipulating a condiment container, and coordinating sequential subtasks across the two arms. During evaluation, we vary the positions of the ketchup container and bread.

Insert Tube (single-arm). The right arm extracts a target tube from a rack and inserts it into another designated slot. Successful execution requires aligning the tube with the target slot and adjusting actions in response to contact during extraction and insertion, including unintended contact caused by misalignment. During evaluation, we translate and rotate the rack to assess insertion performance across object poses.

Press Toaster (single-arm). The right arm locates the toaster's target pressing surface and applies sustained pressure. This task tests contact localization and the policy's response to slip or contact displacement during sustained pressing. During evaluation, we vary the toaster's position along the front–back direction.

B Stage-wise Real-Robot Performance and Failure Analysis

This appendix complements Section 5.2.2 with measured stage-wise performance for six methods and a task-wise analysis of representative $\pi_{0.5}$ rollouts. Section B.1 defines the cumulative stage-completion metric and reports the measured results. The subsequent case figures compare vision-only failures (top) with TacFRS rollouts (bottom), with frames ordered chronologically within each row. These selected examples illustrate observed behaviors rather than failure frequencies.

B.1 Stage-wise Success Rates

We organize the real-robot evaluation by the ordered task stages defined below. Tables 3–6 report measured cumulative stage-completion rates for RDP, DECO, SmolVLA, SmolVLA + TacFRS, $\pi_{0.5}$, and $\pi_{0.5}$ + TacFRS using 20 trials per method per task.

Metric and counting protocol. For method m and stage k, we report the cumulative stage-completion rate

$$C_{m,k} = \frac{n_{m,k}}{N_m} \times 100\%, \qquad N_m = 20,$$

where $n_{m,k}$ is the number of trials that complete all stages up to and including stage k. The denominator includes all 20 trials for the method on the task, including trials that never reach stage k. Thus, $C_{m,k}$ is non-increasing across stages and is not conditioned on reaching stage k. The final column reports the full-task success rate for each method on that task, as also reported in Figure 5 and, for the VLA methods, Table 1.

Pick Two Tubes. The stages are approach to the left tube, grasp and extraction of the left tube, approach to the right tube, and grasp and extraction of the right tube. Full-task success requires both target tubes to be extracted from the rack and retained in their respective grippers at the end of the trial.

Table 3: Pick Two Tubes: measured cumulative stage-completion rates (%) over 20 trials per method. The final column denotes full-task success.

Method	Approach left tube	Grasp and lift left tube	Approach right tube	Grasp and lift right tube
RDP	70%	55%	20%	15%
DECO	95%	90%	65%	55%
SmolVLA	75%	65%	50%	30%
SmolVLA + TacFRS	80%	80%	50%	40%
$\pi_{0.5}$	80%	65%	60%	50%
$\pi_{0.5}$ + TacFRS	90%	90%	70%	65%

Make Bread. The stages are grasping the bread, placing it, grasping the ketchup container, positioning it above the bread, and completing the squeezing motion. Grasping is assessed by secure object acquisition, rather than gripper closure alone. Full-task success requires the bread to be placed on the table and the ketchup container to be positioned above the bread, where the squeezing motion is completed. Actual ketchup deposition on the bread is not required.

Table 4: Make Bread: measured cumulative stage-completion rates (%) over 20 trials per method. The final column denotes full-task success.

Method	Grasp bread	Place bread	Grasp ketchup	Apply ketchup
RDP	25%	10%	0%	0%
DECO	100%	75%	30%	30%
SmolVLA	55%	40%	20%	5%
SmolVLA + TacFRS	70%	60%	25%	10%
$\pi_{0.5}$	95%	80%	35%	20%
$\pi_{0.5}$ + TacFRS	100%	100%	50%	35%

Insert Tube. The stages are grasping the tube, lifting it clear of the source rack, approaching the target hole, aligning the tube with the hole, and completing insertion. Alignment and insertion must use distinct criteria; contact with the hole entrance alone does not establish complete insertion.

Table 5: Insert Tube: measured cumulative stage-completion rates (%) over 20 trials per method. The final column denotes full-task success.

Method	Grasp tube	Lift tube	Approach target hole	Align with hole	Insert tube
RDP	35%	20%	10%	5%	5%
DECO	100%	100%	85%	80%	60%
SmolVLA	80%	70%	65%	50%	20%
SmolVLA + TacFRS	80%	75%	75%	55%	40%
$\pi_{0.5}$	100%	95%	95%	80%	75%
$\pi_{0.5}$ + TacFRS	100%	100%	100%	90%	90%

Press Toaster. The stages are approaching the toaster, establishing contact, beginning the press, maintaining effective contact during the pressing motion, and completing the press. Full-task success requires the target component to reach its fully depressed position.

Table 6: Press Toaster: measured cumulative stage-completion rates (%) over 20 trials per method. The final column denotes full-task success.

Method	Approach toaster	Establish contact	Press down	Complete the press
RDP	100%	55%	45%	30%
DECO	100%	90%	85%	70%
SmolVLA	100%	90%	70%	60%
SmolVLA + TacFRS	100%	95%	80%	75%
$\pi_{0.5}$	100%	95%	80%	70%
$\pi_{0.5}$ + TacFRS	100%	95%	85%	80%

B.2 Pick Two Tubes

Vision-only failures. Figure 9 shows a representative failure of the vision-only $\pi_{0.5}$ policy during left-hand tube extraction. The gripper reaches the tube's grasping position but does not close sufficiently to secure it. As the gripper moves upward, the tube slips from its grasp and remains in the rack. This case illustrates a failure in grasp establishment during the transition to extraction, despite successful initial positioning.

Observed changes with TacFRS. In the TacFRS rollout, the gripper closes around the tube and maintains its grasp as it moves upward, successfully extracting the tube from the rack. The contrast is consistent with tactile feedback helping establish and maintain a secure grasp during extraction. The sequence demonstrates successful extraction of the target tube, rather than completion of the entire two-tube task.

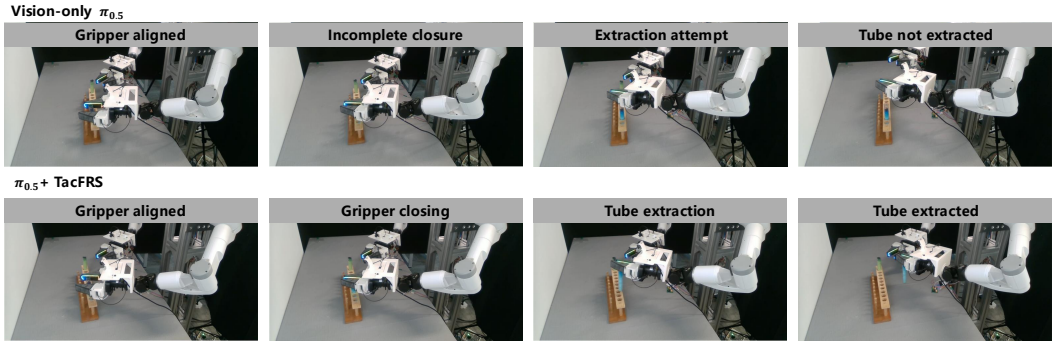


Figure 9: Tube extraction with $\pi_{0.5}$. Top: the vision-only gripper reaches the target but closes incompletely; the tube slips during the extraction attempt and remains in the rack. Bottom: with TacFRS, the gripper maintains its grasp and extracts the tube. Frames run chronologically from left to right within each rollout.

B.3 Make Bread

Vision-only failures. Figure 10 focuses on the left-hand grasp of the ketchup container in Make Bread. The vision-only $\pi_{0.5}$ policy reaches the container but closes the gripper incompletely. During the lifting attempt, it fails to maintain a secure grasp; the container slips and falls onto the table. The failure therefore concerns grasp retention during lifting, rather than simply reaching the object.

Observed changes with TacFRS. With TacFRS, the gripper closes around the container and retains it during lifting and subsequent movement toward the bread. This sequence illustrates improved grasp establishment and retention, but does not by itself demonstrate completion of the squeezing motion above the bread or of the full bimanual task. Full-task success with TacFRS remains 35% for $\pi_{0.5}$ and 10% for SmolVLA (Table 1); reliable execution of the remaining stages is still required.

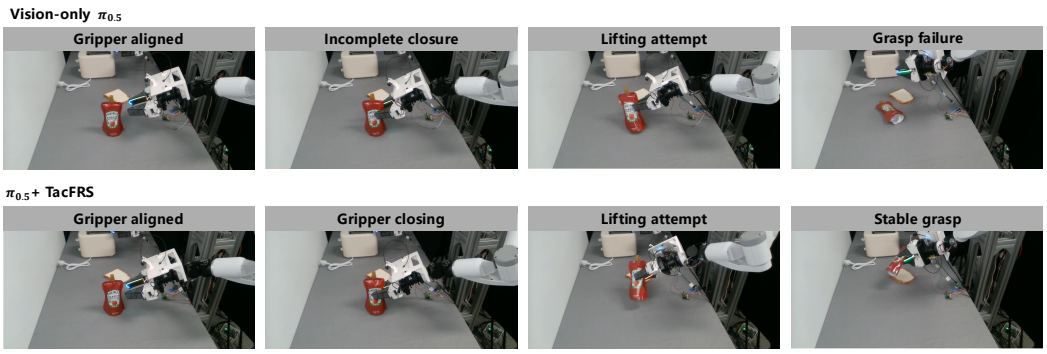


Figure 10: Ketchup-container grasping in Make Bread with $\pi_{0.5}$. Top: incomplete closure leads to loss of the container during lifting. Bottom: with TacFRS, the container is lifted and remains held during subsequent movement. The endpoint depicts local grasp success, not full-task completion. Frames progress from left to right.

B.4 Insert Tube

Vision-only failures. Figure 11 shows an insertion failure after contact with the rack. The vision-only $\pi_{0.5}$ policy brings the tube toward the target hole, but the tube contacts the rack and remains misaligned during the insertion attempt. The tube is not seated in the target hole and ends outside the rack. Unlike the grasping failures above, this case requires correcting the tube–hole configuration after contact.

Observed changes with TacFRS. The TacFRS sequence also includes rack contact, followed by an adjustment of the tube pose before insertion is completed. The contrast is consistent with tactile

feedback supporting correction of residual misalignment during contact-constrained motion. The frames illustrate the observed pose adjustment and insertion outcome; they do not establish which tactile feature triggered the correction.

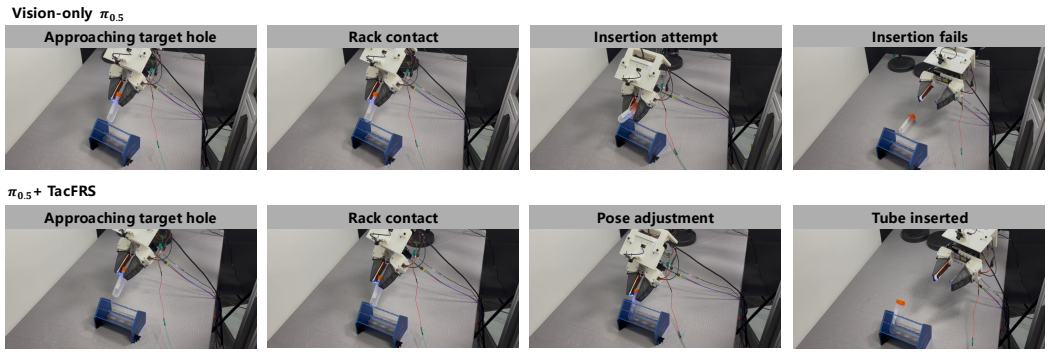


Figure 11: Tube insertion with $\pi_{0.5}$. Top: the tube contacts the rack and the subsequent insertion attempt fails, leaving it outside the rack. Bottom: with TacFRS, rack contact is followed by pose adjustment and successful insertion. Frames progress from left to right.

B.5 Press Toaster

Vision-only failures. Figure 12 illustrates a failure during pressing despite reaching the target component. The vision-only $\pi_{0.5}$ policy closes the gripper insufficiently, and contact slips during the downward motion. The press remains incomplete. Successful initial positioning is therefore insufficient when contact cannot be maintained throughout the pressing motion.

Observed changes with TacFRS. With TacFRS, the gripper closes and sustains contact during pressing, allowing the target component to reach its depressed position. The observed improvement concerns maintaining contact throughout the downward motion. It complements the grasp-retention and insertion-pose corrections illustrated by the preceding tasks.

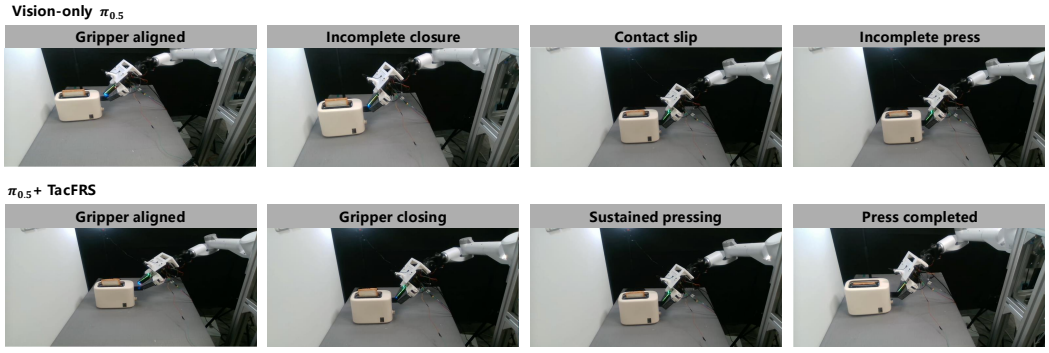


Figure 12: Toaster pressing with $\pi_{0.5}$. Top: incomplete gripper closure is followed by contact slip and an incomplete press. Bottom: with TacFRS, the gripper maintains contact during sustained pressing and completes the press. Frames progress from left to right.

C Selection and Evaluation of Flow Inversion Solvers

Supplementary question 1: Which flow inversion solver is best suited to our framework?

Reverse integration is a key step in constructing structured-noise action representations in our framework. We compare Euler, FireFlow, and SlerpFlow in terms of noise recovery consistency, offline computational efficiency, and downstream action reconstruction quality to select the default inversion configuration.

Table 7: Selected inversion step counts and offline computational cost. Timing includes only reverse integration. Lower time indicates faster inversion.

Solver	Steps	Time (s/sample)
Euler	120	0.3062
FireFlow	50	0.2775
SlerpFlow	**15**	**0.2527**

C.1 Evaluation Protocol

We evaluate flow reversal using noise recovery consistency and offline computational efficiency. These measurements guide the solver-specific inversion step selection in Sec. C.2.

Noise recovery consistency. Given an observation condition $\boldsymbol{c}$, we first sample standard Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and generate an action sequence $\boldsymbol{a}$ using 10-step Euler forward integration. Starting from this action sequence, we then perform N reverse integration steps with solver S to obtain the recovered noise $\hat{\boldsymbol{\epsilon}}_{S,N}$:

$$\boldsymbol{a} = \Phi^{0\to1}_{\text{Euler},10}(\boldsymbol{\epsilon}; \boldsymbol{c}), \qquad \hat{\boldsymbol{\epsilon}}_{S,N} = \Phi^{1\to0}_{S,N}(\boldsymbol{a}; \boldsymbol{c}). \tag{11}$$

Here, forward generation proceeds from noise at time $t = 0$ to actions at time $t = 1$, and reverse integration follows the opposite direction.

For sample i, the noise recovery error is defined as

$$\text{MSE}^{\text{noise}}_i(S, N) = \frac{1}{TA} \left\| \hat{\boldsymbol{\epsilon}}_{i,S,N} - \boldsymbol{\epsilon}_i \right\|^2_{\text{F}}, \tag{12}$$

where T is the action sequence length, A is the action dimension, and $\| \cdot \|_{\text{F}}$ denotes the Frobenius norm over the $T \times A$ entries. We report the mean error across test samples and use the same random seeds and observation samples for all configurations. This metric measures the round-trip consistency of forward action generation and reverse noise recovery, rather than the action reconstruction error of a trained decoder.

Offline computational efficiency. We measure the mean per-sample time required to reverse integrate an action sequence into structured noise. Timing includes only reverse integration and excludes data preprocessing and forward action generation.

C.2 Inversion Step Selection and Computational Efficiency

Inversion step selection. As shown in Figure 13, increasing the number of reverse integration steps does not consistently reduce noise recovery error. The errors of Euler and FireFlow generally increase before leveling off at higher values. SlerpFlow exhibits non-monotonic behavior at small step counts and then remains relatively stable at approximately 1.2 MSE.

Because forward generation uses a fixed, finite-step Euler integrator, these curves characterize the complete round-trip procedure and do not directly measure the numerical accuracy of the reverse solver in isolation.

For SlerpFlow, the mean noise recovery MSE is 1.2057, 1.1963, and 1.1915 at 15, 20, and 25 steps, respectively. The relatively small changes over this range motivate our choice of 15 inversion steps for SlerpFlow, avoiding additional integration cost once the error becomes relatively insensitive to the step count. Together with the noise recovery and step-sensitivity analysis for Euler and FireFlow, this analysis gives the three configurations used for comparison: 120-step Euler, 50-step FireFlow, and 15-step SlerpFlow. These are empirical operating configurations under the fixed forward integration protocol, rather than step counts that necessarily minimize noise recovery error.

Computational efficiency. Table 7 compares the offline inversion cost of the selected configurations. SlerpFlow with 15 steps requires 0.2527 s per sample, compared with 0.2775 s for 50-step FireFlow and 0.3062 s for 120-step Euler.

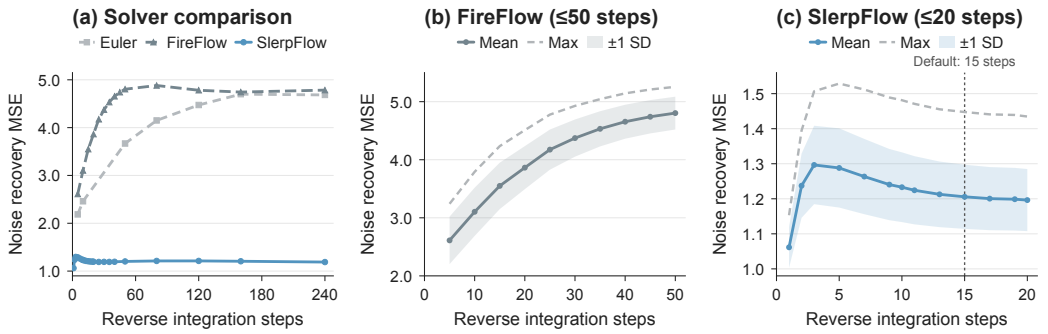


Figure 13: Noise recovery error and sensitivity to the number of reverse integration steps. (a) Mean noise recovery MSE for Euler, FireFlow, and SlerpFlow across step counts. (b) Detailed results for FireFlow with at most 50 steps. (c) Detailed results for SlerpFlow with at most 20 steps.

Among these configurations, 15-step SlerpFlow has the lowest offline inversion cost and requires the fewest reverse integration steps. Its noise recovery error also changes little with further increases to 20 or 25 steps, supporting this choice of inversion step count.

C.3 Unconditional Action Reconstruction from Structured Noise

Supplementary question 2: Does structured noise retain sufficient information for action reconstruction?

Experimental procedure. Using the three inversion configurations selected in Sec. C.2, we first generate source-policy actions by forward integration, invert them into structured noise, train an unconditional decoder on the resulting noise–action pairs, and evaluate action reconstruction accuracy.

Specifically, for each observation $\boldsymbol{c}_i$, we sample $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and construct a pair $(\boldsymbol{z}_i, \boldsymbol{a}_i)$ by

$$\boldsymbol{a}_i = \Phi^{0\to1}_{\text{Euler},10}(\boldsymbol{\epsilon}_i; \boldsymbol{c}_i), \qquad \boldsymbol{z}_i = \Phi^{1\to0}_{S,N_S}(\boldsymbol{a}_i; \boldsymbol{c}_i). \tag{13}$$

Here, N_S is the selected inversion step count for solver S, and $\boldsymbol{z}_i$ is its resulting structured noise. For each solver, we train a separate unconditional flow matching decoder on the training pairs and reconstruct the source action as

$$\hat{\boldsymbol{a}}_i = \Psi_{\theta,N_{\text{dec}}}(\boldsymbol{z}_i), \tag{14}$$

where N_{dec} is the number of decoding integration steps. The decoder receives only structured noise, without observation or tactile conditioning. We evaluate reconstruction against the VLA-generated source actions in the normalized action space:

$$\text{MSE}^{\text{action}} = \frac{1}{N_{\text{val}}TA} \sum_{i=1}^{N_{\text{val}}} \|\hat{\boldsymbol{a}}_i - \boldsymbol{a}_i\|_{\text{F}}^2, \tag{15}$$

where N_{val} is the number of validation samples. We report this reconstruction MSE and the validation flow matching loss $\mathcal{L}_{\text{flow}}$ to compare the structured-noise representations produced by the three selected configurations.

Control baselines. We introduce two baselines to contextualize the reconstruction error. The first predicts the mean of all VLA-generated source actions in the training set, computed separately at each time step and action dimension:

$$\bar{\boldsymbol{a}}_{\text{train}} = \frac{1}{N_{\text{train}}} \sum_{i\in\mathcal{D}_{\text{train}}} \boldsymbol{a}_i \in \mathbb{R}^{T\times A}, \tag{16}$$

where $N_{\text{train}} = |\mathcal{D}_{\text{train}}|$. Its validation error is

$$\text{MSE}^{\text{ref}} = \frac{1}{N_{\text{val}}TA} \sum_{i=1}^{N_{\text{val}}} \|\bar{\boldsymbol{a}}_{\text{train}} - \boldsymbol{a}_i\|_{\text{F}}^2. \tag{17}$$

Table 8: Unconditional action reconstruction on all 9,637 validation samples, comparing two control baselines with structured noise produced by the selected inversion configurations. Each noise-input method uses a separately trained decoder. Action MSE measures reconstruction of the VLA-generated source actions in the normalized action space. Lower values are better for both metrics. Dashes indicate quantities that are not applicable or were not reported.

Method	Reversal steps	Action MSE	$\mathcal{L}_{\text{flow}}$
Training-set mean	–	0.035	–
Original-noise decoder	–	0.034	–
Euler	120	0.0043	0.0037
FireFlow	50	0.0034	0.0029
SlerpFlow	**15**	**0.0024**	**0.0013**

This baseline quantifies the error of a constant prediction without sample-specific input.

The second baseline trains a separate unconditional decoder on $(\boldsymbol{\epsilon}_i, \boldsymbol{a}_i)$ pairs, using the original Gaussian noise sampled for VLA action generation as input and the corresponding generated action as supervision:

$$\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \qquad \boldsymbol{a}_i^{\text{rand}} = \Psi_{\theta_{\text{rand}}, N_{\text{dec}}}(\boldsymbol{\epsilon}_i), \tag{18}$$

$$\text{MSE}^{\text{rand}} = \frac{1}{N_{\text{val}} T A} \sum_{i=1}^{N_{\text{val}}} \left\| \boldsymbol{a}_i^{\text{rand}} - \boldsymbol{a}_i \right\|_{\text{F}}^2 . \tag{19}$$

Here, $\boldsymbol{\epsilon}_i$ is the stored noise used to generate the corresponding source action, and θ_{rand} denotes the separately trained decoder parameters. The decoder receives only this noise, without observation or tactile conditioning.

Data and episode split. We split 40 episodes into disjoint training and validation sets at an 80%/20% ratio: 32 training episodes with 38,251 samples and 8 validation episodes with 9,637 samples. Each sample is an action chunk with $T = 50$ steps. All three solvers use the same episode split. The validation set is used for checkpoint selection and the reported reconstruction evaluation.

Results. Table 8 compares the two control baselines with the three structured-noise decoders. The training-set mean yields an action MSE of 0.035, while the original-noise decoder achieves 0.034, providing little improvement over the constant prediction. In comparison, the structured-noise decoders achieve action MSEs between 0.0024 and 0.0043. SlerpFlow achieves the lowest action MSE (0.0024), approximately 93% lower than either baseline, followed by FireFlow (0.0034) and Euler (0.0043). It also achieves the lowest flow matching loss (0.0013) among the three structured-noise decoders.

These results support two conclusions. First, the substantially lower error than both control baselines provides empirical evidence that structured noise retains sufficient information for approximate reconstruction of the source-policy actions without observation or tactile conditioning under the evaluated protocol. The comparison with the original-noise decoder further supports using the flow-reversed representation for this reconstruction task. Second, SlerpFlow produces structured noise with the best action reconstruction performance among the three selected configurations, achieving the lowest action MSE and flow matching loss. This further supports our use of 15-step SlerpFlow as the default inversion configuration.

C.4 The Influence of Observations on Structured Noise.

Supplementary question 3: Does the observation affect the structured noise and its decoding?

Influence of Observation Conditions on Structured Noise. To examine whether the observation condition used during flow reversal affects the resulting structured noise, we sample 100 matched observation–action pairs,

$$\mathcal{D}_{\text{test}} = \{(\boldsymbol{o}_j, \boldsymbol{a}_j)\}_{j=1}^{100}.$$

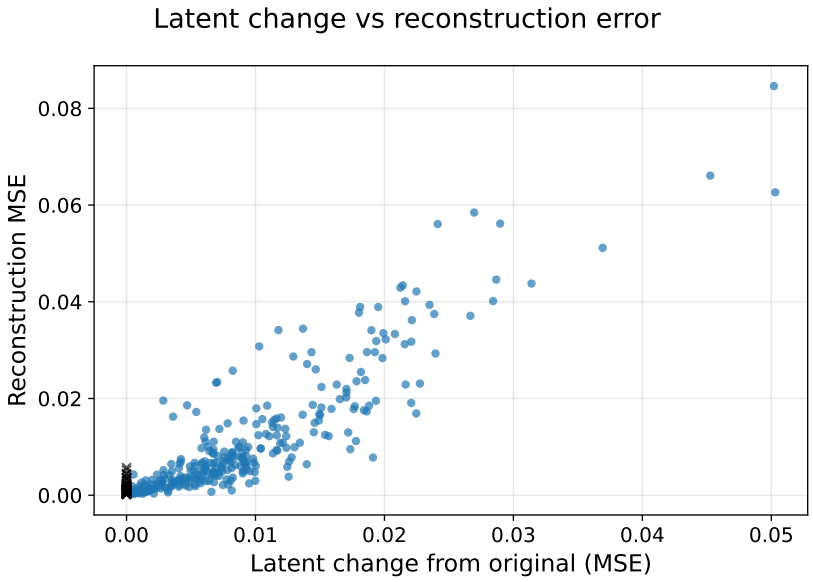


Figure 14: Observation-Induced Structured Noise Change vs. Action Reconstruction Error

We then randomly select four observation conditions, indexed by $\mathcal{I} = \{i_1, i_2, i_3, i_4\} \subset \{1, \ldots, 100\}$, and pair each selected observation with all 100 actions, yielding 400 combinations

$$\{(\boldsymbol{o}_{i_k}, \boldsymbol{a}_j)\}_{k=1,\ldots,4;\, j=1,\ldots,100}.$$

For each combination, we perform observation-conditioned flow reversal to obtain

$$\boldsymbol{z}_{k,j} = \mathcal{R}_\theta\left(\boldsymbol{a}_j \mid \boldsymbol{o}_{i_k}\right), \tag{20}$$

where $\mathcal{R}_\theta$ denotes the VLA flow-reversal operator. For reference, we denote the structured noise obtained from the original matched pair $(\boldsymbol{o}_j, \boldsymbol{a}_j)$ as

$$\boldsymbol{z}_j^\star = \mathcal{R}_\theta\left(\boldsymbol{a}_j \mid \boldsymbol{o}_j\right). \tag{21}$$

We decode each $\boldsymbol{z}_{k,j}$ using the unconditional decoder $\Psi_{\theta,N_{\text{dec}}}$ trained for the default 15-step SlerpFlow configuration in Sec. C.3:

$$\hat{\boldsymbol{a}}_{k,j} = \Psi_{\theta,N_{\text{dec}}}\left(\boldsymbol{z}_{k,j}\right). \tag{22}$$

Because the decoder receives no observation input, differences in $\hat{\boldsymbol{a}}_{k,j}$ induced by changing $\boldsymbol{o}_{i_k}$ must be transmitted through the structured noise representation.

For each mismatched observation–action pair, we measure the change in structured noise relative to the matched condition,

$$d_{ij}^z = \|\boldsymbol{z}_{jj} - \boldsymbol{z}_{ij}\|_2^2, \tag{23}$$

and the corresponding deviation of the decoded action from the original action,

$$d_{ij}^a = \|\boldsymbol{a}_j - \hat{\boldsymbol{a}}_{ij}\|_2^2. \tag{24}$$

Figure 14 plots d_{ij}^z against d_{ij}^a. We observe a positive correlation between changes in structured noise and deviations in the decoded actions. This result indicates that the observation condition affects not only the inverted representation itself, but also the action recovered from that representation, showing that observation-dependent information is propagated through structured noise to the decoding process.

D Training and Deployment Details

This appendix reports the experimental setup, baseline training, tactile revision (TR) network training, and baseline deployment configurations. VLA fine-tuning, two-stage baseline training, and

baseline deployment settings are summarized in Tables 9, 10, and 13, respectively. TR training settings are provided in Tables 11 and 12. Batch sizes are global. Visuotactile representation pretraining is described separately in Appendix E.

D.1 Experimental Setup

Computation. Model training uses NVIDIA H100 GPUs. Real-world deployment and evaluation are performed on a workstation equipped with an Intel Core Ultra 7 265K CPU and an NVIDIA RTX 5080 GPU.

D.2 Baseline Training Configurations

VLA backbones. The bimanual SmolVLA and $\pi_{0.5}$ backbones use 20-dimensional state and action vectors. Both use LoRA in the language component of the VLM; the action expert is fully fine-tuned in SmolVLA and LoRA-adapted in $\pi_{0.5}$. Both learning-rate schedules use warmup and cosine decay.

Table 9: **VLA fine-tuning.** Bimanual backbones without FRS.

Hyperparameter	**SmolVLA**	$\boldsymbol{\pi_{0.5}}$
Model adaptation		
Prediction horizon H_{train}	50	50
VLM LoRA (r, α)	$(16, 16)$	$(16, 16)$
Action-expert tuning	Full	LoRA
Expert LoRA (r, α)	—	$(32, 32)$
Optimization		
Global batch size	128	128
Maximum updates	30,000	40,000
Peak learning rate	5×10^{-5}	2.5×10^{-5}
Final LR target	2.5×10^{-6}	2.5×10^{-6}
Warmup updates	1,500	1,000
Decay end update	30,000	40,000

r: LoRA rank; α: LoRA scaling parameter. The final LR is the schedule target, not necessarily the LR at the selected checkpoint. —: not applicable.

DECO. Stage 1 trains the visual policy. Stage 2 freezes that policy and the tactile encoder, and trains sensor embeddings, tactile attention key/value projections, gates, and PI adapters. Both stages predict 32 actions. Their default RGB resolution is 256×256, and the default PI-adapter rank is 32. Stage 2 additionally receives four 224×224 tactile RGB images and retains the Stage 1 RGB and sampling configurations. AdamW uses $(\beta_1, \beta_2) = (0.95, 0.999)$. The Stage 1 visual-backbone LR decays from 10^{-5} to 5×10^{-7}; the policy/adapter LRs are listed in Table 10.

Table 10: **Two-stage baseline training.** Optimization settings for DECO and RDP.

	DECO		**RDP**	
Hyperparameter	Stage 1	Stage 2	AT	LDP
Optimizer	AdamW			
LR schedule	Cosine; warmup as listed below			
Main learning rate	10^{-4}	10^{-4}	10^{-3}	10^{-4}
Final LR target	5×10^{-6}	5×10^{-6}	—	—
Weight decay	10^{-6}	10^{-6}	10^{-4}	10^{-6}
Warmup updates	0	0	100	500
Global batch size	8	512	64	64
Maximum epochs	100	50	60	40

RDP. The asymmetric tokenizer (AT) is trained first and frozen during latent diffusion policy (LDP) training. The AT uses KL-regularized action reconstruction; the LDP predicts diffusion noise with an

Table 11: Network architecture and optimization settings for bimanual TR training. Epoch counts specify maximum training budgets.

Hyperparameter	**SmolVLA**	$\pi_{0.5}$
Architecture and inputs		
Base VLA	Frozen	Frozen
Tactile image encoder	Frozen	Frozen
Transformer width / layers	256 / 6	256 / 6
Attention heads / MLP ratio	4 / 4	4 / 4
GRU layers / hidden size	1 / 256	1 / 256
Tactile feature dimension	512	512
Tactile image resolution	224×224	224×224
Tactile sensor streams	4	4
Tactile history length	5	10
History stride (frames)	3	3
Training-frame stride	5	5
Action horizon	20	50
Physical action dimension	20	20
Optimization		
Optimizer	AdamW	AdamW
Peak learning rate	10^{-4}	10^{-4}
Weight decay	10^{-4}	10^{-4}
Gradient clipping norm	1.0	1.0
Batch size	64	128
Maximum epochs	30	50
Warmup epochs	1	2
Learning-rate schedule	Cosine	Cosine
Minimum / peak learning rate	0.1	0.1
State-token dropout	0.1	0.1
Validation interval (epochs)	2	5
Validation fraction	0.1	0.1
Random seed	42	42

MSE objective. The AT has 4 encoder-output channels, 16 Gaussian latent channels, and Conv/GRU hidden widths of 32/32. The LDP uses U-Net widths $(512, 1024, 2048)$ and 100 diffusion training timesteps. Neither stage accumulates gradients across minibatches; EMA is disabled for AT and enabled for LDP with maximum decay 0.9999.

The training window contains 32 steps. Two visual observations are sampled with stride two from a four-step raw observation window, yielding an effective action horizon of $32 - 4 + 1 = 29$. Tactile PCA features have 30 dimensions for bimanual tasks and 15 for right-arm tasks.

D.3 Tactile Revision Network Training

We train the TR network after vision-only VLA fine-tuning, using a smaller demonstration dataset than that used for VLA fine-tuning. The base VLA and tactile image encoder remain frozen. The trainable components are the tactile-history encoder, reference and state projections, and conditional velocity field. Tables 11 and 12 summarize the network, optimization, and integration settings for the two bimanual backbones.

Optimization. We use linear learning-rate warmup followed by cosine decay. State-token dropout is applied to the entire token during training and disabled during evaluation and inference.

The TR network conditions on tactile history, the contact-free reference, and proprioceptive state. The reference is fixed, and the same condition construction is used during training and inference, apart from training-only state-token dropout. For each backbone, both arms share the gate hyperparameters but compute their weights from their own tactile sensors. The sigmoid output is used directly as the interpolation weight, without additional thresholding or clipping.

Table 12: Supervision and numerical integration settings for bimanual TR training. The objective contains only flow matching and endpoint reconstruction.

Hyperparameter	**SmolVLA**	$\boldsymbol{\pi_{0.5}}$
Gate midpoint τ	0.4	0.65
Gate temperature T_g	0.1	0.15
Flow-matching weight	1	1
Endpoint weight λ_{end}	4	4
Flow-time sampling	$\mathcal{U}(0,1)$	$\mathcal{U}(0,1)$
VLA reversal solver	SlerpFlow	SlerpFlow
VLA reversal steps	15	15
TR forward solver	FireFlow	FireFlow
Endpoint integration steps K	10	10
Validation integration steps	10	10

For arm $r \in \{L, R\}$, the target is

$$Y_t^{(r)} = \big(1 - \alpha_r(t)\big) a_t^{\mathrm{VLA},(r)} + \alpha_r(t) a_t^{\mathrm{GT},(r)}. \tag{25}$$

We concatenate these targets along the action dimension and train one joint conditional velocity field. For an error tensor associated with arm r, we define $\|E^{(r)}\|^2_{\mathrm{act},r} = \|E^{(r)}\|^2_F/(Hd_r)$, where d_r is that arm's action dimension. The two bimanual losses are

$$\begin{aligned} \mathcal{L}^{\mathrm{bi}}_{\mathrm{FM}} &= \mathbb{E}\left[\frac{1}{2}\sum_{r\in\{L,R\}} \left\| v_\theta^{(r)}(x_{t,u}, u; C_t) - \big(Y_t^{(r)} - z_t^{(r)}\big) \right\|^2_{\mathrm{act},r}\right], \\ \mathcal{L}^{\mathrm{bi}}_{\mathrm{end}} &= \mathbb{E}\left[\frac{1}{2}\sum_{r\in\{L,R\}} \left\| \widehat{a}_{\theta,t}^{(r)} - Y_t^{(r)} \right\|^2_{\mathrm{act},r}\right]. \end{aligned} \tag{26}$$

Here, the expectations follow the training distribution and flow-time sampling in Sec. 4.2; flow-time sampling applies only to the flow-matching term. The final objective is $\mathcal{L}^{\mathrm{bi}}_{\mathrm{TacFRS}} = \mathcal{L}^{\mathrm{bi}}_{\mathrm{FM}} + 4\mathcal{L}^{\mathrm{bi}}_{\mathrm{end}}$. Both terms therefore preserve equal weighting between arms, without introducing additional objectives.

Endpoint reconstruction penalizes the mean squared error to the mixed target itself. We backpropagate through the complete K-step TR decoding trajectory. The frozen VLA's action predictions and reversed flow states are cached before TR optimization; its reverse integration is not part of the TR gradient graph. Training-frame subsampling controls the temporal density of training examples and does not specify the ratio of TR to VLA demonstrations.

D.4 Baseline Deployment Configurations

Let f_a denote the action time-grid frequency, H the prediction horizon, K the number of actions dispatched per scheduling cycle, and N_s the number of sampling iterations. For RDP, H is the effective horizon after accounting for observation history. The low-level controller frequency is $f_c = 80$ Hz in all listed configurations. Sampling iterations are distinct from H and K.

Table 13: **Deployment settings.** Frequencies are configured targets, not measured throughput.

Policy	f_a (Hz)	H	K	N_s	**Execution**
SmolVLA (w/o FRS)	30	50	50	10	Chunkwise
$\pi_{0.5}$ (w/o FRS)	30	50	50	10	Chunkwise
DECO Stage 1	30	32	32	5	Chunkwise
DECO Stage 2	30	32	32	5	Chunkwise
RDP	30	29	1	8	Asynchronous

VLA rows are bimanual.

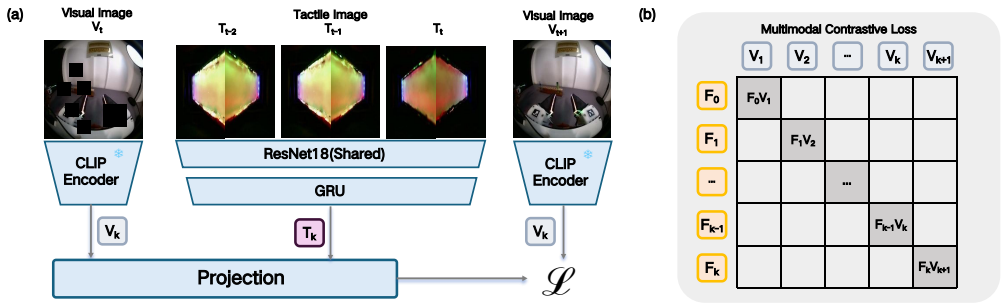


Figure 15: Visuotactile representation pretraining. A shared ResNet-18 and a GRU encode temporal tactile information, which is fused with masked visual features and aligned with the next-step unmasked visual embedding through contrastive learning.

Execution timing. The nominal duration covered by K action intervals is

$$\Delta t_{\text{nominal}} = K/f_a. \tag{27}$$

This gives 1.67 s for the VLA rows and approximately 1.07 s for DECO. These durations exclude inference and communication overhead and are not end-to-end latency measurements.

Asynchronous RDP. Background planning requests are capped at 6 Hz. New plans normally become eligible for adoption after 16 steps on the 30 Hz action grid ($16/30 \approx 0.53$ s), provided a valid completed plan is available. An earlier switch is allowed when the active plan approaches the end of its effective horizon. Meanwhile, the foreground loop uses the latest tactile observation to decode one action. Thus, $K = 1$ denotes single-action dispatch, not replanning at every step.

E Visuotactile Representation Pretraining

E.1 Pretraining Framework

We follow the visuotactile contrastive pretraining approach of ViTaMIn (Liu et al., 2025): masked visual observations at the current timestep are fused with tactile information, and the resulting representation is aligned with the embedding of the unmasked visual observation at the next timestep. We extend this framework with a temporal tactile encoder and a contrastive learning mechanism tailored to bimanual sequential data.

E.2 Temporal Tactile Encoding

As illustrated in Figure 15(a), we combine a shared ResNet-18 backbone with a shared gated recurrent unit (GRU) to encode tactile histories. For each tactile stream, ResNet-18 extracts features from the current and two preceding sampled frames, and the GRU aggregates them into a temporal representation. This design incorporates the evolution of contact states rather than relying solely on single-frame tactile appearance.

For each wrist, the temporal features from its two tactile streams are concatenated with the masked current visual features extracted by a frozen CLIP ViT-B/16 encoder. The concatenated features are projected into the 512-dimensional CLIP embedding space and aligned with the embedding of the next unmasked visual observation, extracted by the same frozen visual encoder. We use symmetric InfoNCE as the base alignment objective, with the contrastive training extensions described below.

E.3 Extensions to Contrastive Learning

We adapt the contrastive training procedure in three ways.

Memory queue and hard negative sampling. We maintain a queue of target visual embeddings from previous iterations to expand the negative candidate pool. For each query, we select the top-K eligible negatives with the highest embedding similarity.

Wrist-specific contrastive sampling. We restrict contrastive comparisons to samples from the same wrist, constructing the left- and right-wrist comparisons separately.

Multiple temporal positives. Samples from the same trajectory and wrist whose temporal indices fall within a $\pm w$ neighborhood are treated as additional positives. We aggregate positive logits using log-sum-exp rather than retaining only a single positive. This is intended to reduce the ambiguity associated with treating temporally neighboring observations as negatives.

E.4 Training and Evaluation

We pretrain on a bimanual UMI dataset containing 1,019 trajectories and approximately 145,000 training samples. Training uses the AdamW optimizer with a cosine annealing learning-rate schedule, a batch size of 64, and a contrastive temperature of 0.07. We mask 30% of the current visual image.

We evaluate the learned representations through cross-modal retrieval. Each visuotactile embedding serves as a query to retrieve its matching future visual observation from a candidate pool containing one matching future image and 31 randomly sampled negative images from the same wrist. We report Recall@1, defined as the fraction of queries for which the matching future image is ranked first.

E.5 Pretraining Results

The best validation checkpoint, obtained at epoch 6, achieves an overall Recall@1 of 97.9%, with 97.6% for the left wrist and 98.3% for the right wrist. These results indicate accurate retrieval of the paired next-step visual observations under the 32-candidate evaluation protocol. Because each query contains both visual and tactile information, this metric evaluates joint visuotactile alignment rather than the contribution of tactile input alone.

After pretraining, we freeze the tactile encoder and use it as the feature extraction frontend for the downstream tactile-conditioned flow decoder.