TORL-VLA: Tactile Guided Online Reinforcement Learning for Contact-Rich Manipulation

A Appendix

This appendix provides implementation and experimental details omitted from the main text due to space constraints. Appendix A.1 describes the latch-box manipulation benchmarks, the success and failure criteria, and the evaluation protocol used for autonomous evaluation and online adaptation. Appendix A.2 summarizes the robot system, tactile-to-wrench processing, demonstrations, and online replay. Appendix A.3 details the baseline adaptation and the implementation of the wrench-aware reference model and stage-specific online actor-critic refiners. Appendix A.4 presents additional diagnostics complementing the main results, including wrench and contact-state analysis and intervention-censored critic diagnostics. Appendix A.5 reports deployment inference latency. All task names, method names, and ablation names follow the notation in the main paper.

A.1 Task and Evaluation Protocol

A.1.1 Latch-Box Manipulation Tasks

The real-robot evaluation is conducted on a latch-box manipulation benchmark with three contact-rich subtasks: **Coffee Cup**, **Latch**, and **Egg**. Each subtask contains one or more local contact-rich bottleneck stages, where the stage estimator can activate the corresponding stage-specific online refiner during execution. The workspace structure and subtask order are shared across methods, and all trials follow the same task-specific reset protocol, with object initial poses allowed to vary within predefined start regions. The three subtasks are selected to cover different contact bottlenecks. Coffee Cup evaluates tight insertion into a narrow holder under partial occlusion. Latch evaluates contact-dependent mechanical engagement, where the robot must grip, flip, and lock a latch. Egg evaluates fragile-object grasping and placement, where both insufficient and excessive contact can cause failure.

For full-task evaluation, the robot starts from the same latch-box initial configuration and executes the task sequence in one autonomous rollout. A full-task rollout is counted as successful only if the robot completes Coffee Cup, Latch, and Egg sequentially without human intervention, environment reset, safety stop, object drop, or visible object damage.

A.1.2 Success and Failure Criteria

The subtasks are designed to expose contact states that may appear visually close to success while differing in physical interaction, such as partial insertion, incomplete latch engagement, or excessive grasping force. All trials are reset according to the same task-specific reset protocol before evaluation.

The observable success and failure criteria are defined at the task-outcome level rather than by internal model signals or intermediate wrench traces, ensuring consistent evaluation across methods. Table 3 summarizes these criteria for all subtasks and the full-task evaluation. If any listed failure event occurs, the trial is marked as a failure even if human intervention could have recovered the task afterward.

A.1.3 Evaluation Metrics and Protocol

For both subtask and full-task success rates, each method is evaluated over 30 autonomous trials. The same task objects, workspace structure, reset procedure, action representation, control frequency, and receding-horizon execution protocol are used for all methods. During final autonomous evaluation, no human intervention, environment reset, safety-stop recovery, or online parameter update is allowed. Online parameter updates are performed only during the online data-collection

Table 3: **Observable success and failure criteria for autonomous evaluation.**

Task	Success condition	Failure condition
Coffee Cup	Pick cup; insert into holder; release; cup remains upright and stable.	Holder miss; rim hang-up; partial insertion; large tilt; drop; unsafe contact.
Latch	Grip latch; flip toward locking direction; press into locked state; latch remains secured after release.	Missed latch; slip during flipping; missed locking edge; incomplete lock; rebound; unsafe force.
Egg	Grasp egg; transfer to holder; release; egg remains stable without visible damage.	Slip; drop; holder collision; rolling out after release; excessive grasping force; visible damage.
Full Task	Complete Coffee Cup, Latch, and Egg sequentially in one autonomous rollout.	Any subtask failure; human intervention; reset; safety stop; object drop; object damage.

phase and are not performed while reporting the final success numbers. For reference-model ablations, variants are evaluated directly on the three contact-rich subtasks. For online-adaptation ablations, all variants use the same frozen wrench-aware reference model, checkpoint schedule, and matched online adaptation duration and interaction budget.

A.2 System, Sensing, and Data

A.2.1 Robot and Sensor Setup

The robot platform consists of a Piper robotic arm equipped with a Pika gripper from Agile Robots. The visual input consists of three camera views: a global view captured by an Intel RealSense D435i camera, a fisheye view captured by the camera integrated in the Pika gripper, and a wrist view captured by an Intel RealSense D405 camera mounted on the Pika gripper. The proprioceptive state includes the robot joint positions and the gripper state. Tactile sensing is provided by two Tacta fingertip tactile sensors from EnsuringTech, mounted on the inner contact surfaces of the gripper fingers. Each sensor has a compact size of 21 mm $\times$ 18.5 mm $\times$ 4.5 mm, and each tactile pad outputs a raw 6×8 tactile array.

Policy observations are queried and logged at 20 Hz. All modalities are timestamp-aligned during data collection. For higher-rate tactile and proprioceptive streams, the latest measurement before the query timestamp is used. For visual streams, the most recent synchronized image triplet is used. All methods that use tactile-derived wrench feedback share the same tactile preprocessing, normalization statistics, and temporal window.

A.2.2 Tactile-to-Wrench Processing

The main paper defines the tactile-derived wrench observation as $\mathbf{w}_t = [\mathbf{w}_t^L, \mathbf{w}_t^R] \in \mathbb{R}^{12}$. Here we clarify how this signal is obtained from the tactile sensor interface and how temporal wrench sequences are represented in implementation. The vendor SDK exposes two levels of tactile readout. The low-level interface returns 48 tactile points arranged on a 6×8 pad, with a 3D force vector at each point. The high-level SDK interface provides a calibrated 6-DOF fingertip wrench estimate $[f_x, f_y, f_z, \tau_x, \tau_y, \tau_z]$. We denote this SDK-estimated fingertip wrench as $\tilde{\mathbf{w}}_t^i \in \mathbb{R}^6$ for each finger $i \in \{L, R\}$. The left and right fingertip outputs are concatenated in a fixed order to form the 12D wrench $\tilde{\mathbf{w}}_t = [\tilde{\mathbf{w}}_t^L, \tilde{\mathbf{w}}_t^R] \in \mathbb{R}^{12}$.

This vendor-provided wrench abstraction decouples the policy input from the raw tactile array layout. It also provides a compact force–torque representation that serves as a unified downstream interface, rather than requiring the policy to process sensor-specific raw tactile layouts.

For the current measured wrench used in the online refinement context, we use the SDK-estimated 12D wrench, i.e., $\mathbf{w}_t = \tilde{\mathbf{w}}_t$. For temporal contact reasoning, the reference model uses $J = 10$

wrench samples over the most recent 2 seconds, corresponding to the recent wrench sequence $\mathbf{W}_t = \{\mathbf{w}_{t-J+1}, \ldots, \mathbf{w}_t\}$ in the main paper. In implementation, before feeding this sequence into the wrench-history encoder, we apply a current-wrench-centered transform to the SDK-estimated wrench samples. Specifically, the encoder input is formed as $\{\tilde{\mathbf{w}}_{t-J+1} - \tilde{\mathbf{w}}_t, \ldots, \tilde{\mathbf{w}}_{t-1} - \tilde{\mathbf{w}}_t, \mathbf{0}\}$. Similarly, the future wrench target used in joint action–wrench prediction is represented by residuals $\tilde{\mathbf{w}}_{t+k} - \tilde{\mathbf{w}}_t$. This implementation keeps the temporal structure described in the main paper while representing wrench sequences by relative wrench changes around the current contact state.

The same tactile-to-wrench preprocessing is used for all methods that consume wrench feedback, so differences between baselines and TORL-VLA come from the policy architecture and online adaptation mechanism rather than from different tactile preprocessing.

A.2.3 Training Data for Reference Model and Online Adaptation

Table 4 summarizes the main real-robot data budget used for the wrench-aware reference model and online actor-critic refinement. Full-task demonstrations are used to train the wrench-aware VLA reference model, which predicts reference action chunks and future wrench sequences. Main online interaction data are collected around contact-rich bottleneck stages for stage-specific online adaptation. Warmup settings are reported in Table 5.

Table 4: **Real-robot data budget for reference-model training and online adaptation.** Main online interaction time denotes accumulated robot execution time during online data collection, excluding human reset, object repositioning, and environment preparation. The Egg subtask contains two contact-rich bottleneck stages: grasping and placement.

Data source	Stage / Scope	Scale
Demonstrations	Full task	539 trajectories
Main online interaction	Coffee Cup	23.5 min
	Latch	40.3 min
	Egg: grasp	29.2 min
	Egg: place	27.8 min

A.3 Implementation Details

A.3.1 Baseline Adaptation

We compare all methods under the same robot platform, action representation, training demonstrations, execution protocol, and evaluation procedure. We do not compare against numbers reported in prior work, since robot embodiments, tactile sensors, task definitions, and data distributions differ across papers. Instead, we adapt the baselines to the same $\pi_{0.5}$ backbone and, where applicable, the same dual-fingertip tactile-derived wrench interface used by TORL-VLA.

The $\pi_{0.5}$ baseline directly executes the pretrained VLA policy without tactile-derived wrench input. TA-VLA and ForceVLA are reimplemented on the same $\pi_{0.5}$ backbone and adapted to the dual-fingertip wrench observation. This keeps the comparison focused on whether tactile- or force-aware offline VLA designs can improve contact-rich manipulation under the same hardware and data setting. TORL-VLA w/o RL directly executes the learned wrench-aware reference model, without online RL refinement. RLT uses reference-guided online refinement, but it does not incorporate wrench input during online refinement and does not include the intervention-censored critic. TORL-VLA uses the wrench-aware reference model together with wrench-conditioned online refinement and intervention-censored value learning.

All online-adaptation variants use the same frozen wrench-aware reference model and the same online adaptation duration, while each variant collects its own online robot data. This ensures that the comparison between RLT, TORL-VLA, and online ablations focuses on the effect of wrench-context conditioning and intervention-censored value learning under a matched online interaction budget.

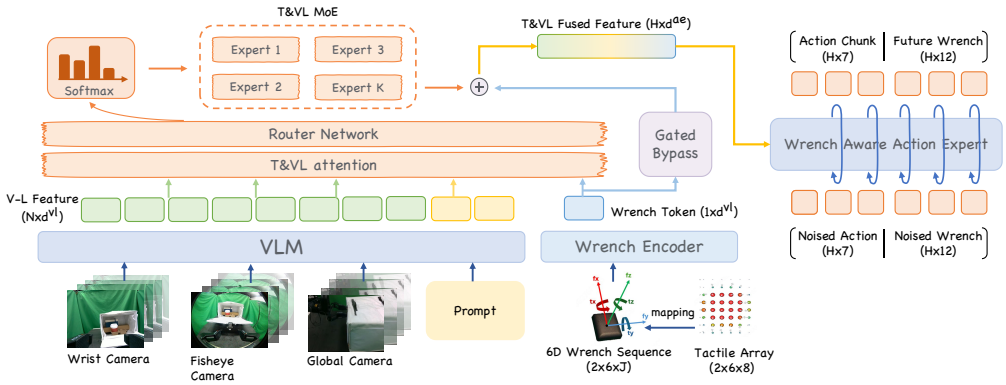


Figure 7: **Implementation details of the wrench-aware VLA reference model.**

A.3.2 Wrench-Aware Reference Model

This subsection specifies the implementation details of Stage I, which are omitted from the main text for space. At each timestep t, the wrench-aware VLA reference model takes as input three camera views (wrist, fisheye, and global), a language instruction, robot proprioception, and tactile readings from two piezoelectric pads mounted on the inner surfaces of the gripper fingers. As described in Appendix A.2.2, the tactile readings are converted into the current measured wrench $\mathbf{w}_t$, and the recent wrench window is represented in a current-wrench-centered form before being fed into the wrench-history encoder. The resulting history sequence is compressed into a fixed-size wrench token $\mathbf{z}_t^{\text{tac}} \in \mathbb{R}^{1\times d_{\text{vl}}}$ by a lightweight MLP encoder. In parallel, the three camera images, language instruction, and proprioception are encoded by the VLM backbone into visual-language prefix hidden states $\mathbf{P}_t \in \mathbb{R}^{N_p\times d_{\text{vl}}}$.

Rather than injecting the wrench token early into the VLM, we adopt a late fusion strategy. The wrench token is appended to $\mathbf{P}_t$ after visual-language encoding, and a self-attention layer is applied to the concatenated sequence $[\mathbf{P}_t; \mathbf{z}_t^{\text{tac}}]$ to produce the fused multimodal representation $\mathbf{H}_t \in \mathbb{R}^{(N_p+1)\times d_{\text{vl}}}$. This preserves the pretrained visual-semantic features while enabling cross-modal interaction with tactile-derived wrench feedback. The fused representation is then passed through a soft-routing Mixture-of-Experts module, where a router network computes per-expert weights $\boldsymbol{\rho}_t = \text{Softmax}(R_{\text{tac}}(\mathbf{H}_t))$, and the routed feature is obtained as $\mathbf{F}_t = \sum_{m=1}^{M} \boldsymbol{\rho}_{t,m} \odot E_m(\mathbf{H}_t)$. This allows the model to dynamically route features according to the current contact phase.

To retain fine-grained physical information that may be diluted by the MoE transformations, we add a zero-initialized wrench-token bypass. The routed feature and the original wrench token are projected to the action horizon and combined as $\mathbf{G}_t = \Pi_{\text{tac}}(\mathbf{F}_t) + \alpha P_{\text{tac}}^H(\mathbf{z}_t^{\text{tac}}) \in \mathbb{R}^{H\times d_{\text{ae}}}$, where Π_{tac} maps the routed feature to the action horizon, P_{tac}^H projects and expands the wrench token to horizon-aligned guidance tokens, and α is a learnable scalar initialized to zero. Thus, the bypass contributes nothing at the start of training and grows adaptively as the model learns to use wrench feedback.

Within the flow-matching action expert, the noisy joint generation variable $\mathbf{x}_{t,s}^{\text{fm}} \in \mathbb{R}^{H\times(d_a+12)}$ is processed at interpolation time $s \in [0,1]$ to yield action-side hidden states $\mathbf{S}_{t,s}$. The wrench-conditioned guidance signal is added as a residual, producing $\tilde{\mathbf{S}}_{t,s} = \mathbf{S}_{t,s} + \mathbf{G}_t$. A joint output head then predicts a single velocity field over the concatenated action–wrench target, which is split into action and wrench dimensions as $\hat{\mathbf{u}}_{t,s} = [\hat{\mathbf{u}}_{t,s}^a, \hat{\mathbf{u}}_{t,s}^w]$. Solving the learned flow from $s = 0$ to $s = 1$ yields the joint prediction: a reference action chunk $\hat{\mathbf{A}}_{t:t+H-1} \in \mathbb{R}^{H\times d_a}$ that serves as the motion prior for the online actor, and a future wrench sequence $\hat{\mathbf{W}}_{t:t+H-1} \in \mathbb{R}^{H\times 12}$ that provides the expected contact cue for wrench-conditioned action refinement in Stage II. Following the preprocessing described in Appendix A.2.2, the future wrench target used in training is represented by current-centered residuals, i.e., $\tilde{\mathbf{w}}_{t+k} - \tilde{\mathbf{w}}_t$.

A.3.3 Online Reference Adaptation

This subsection adds implementation details that are not fully specified in the main text. During Stage II adaptation, the wrench-aware VLA is kept frozen and only the lightweight stage-specific actor-critic refiners are updated.

Reference interface. The online refiner uses the same executable reference interface and context definition as in the main text. In implementation, the frozen wrench-aware VLA predicts a 50-step action sequence and a 50-step future wrench sequence, and the first $K = 10$ steps are used for online refinement and execution. At 20 Hz, this corresponds to a 0.5 s replanning interval.

To obtain the compact VLA token z_t^{vla}, we append a learned special token to the final hidden-state sequence of the frozen wrench-aware VLA and pass the augmented sequence through a lightweight token encoder. The output at the special-token position is used as z_t^{vla}. The token encoder is trained before online adaptation and is frozen during online actor-critic updates.

Stage estimator routing. As shown in Figure 8, the stage estimator is a lightweight chunk-level prediction module rather than an action policy. Its purpose is to decide how each VLA reference chunk should be executed during Stage II adaptation. At each VLA query, the frozen wrench-aware VLA first produces the reference action chunk and the policy-side features used by the online module. The stage estimator reuses these features, including the compact VLA token context, proprioception context, and wrench context, to predict a discrete stage label and its confidence. The predicted stage label is then mapped to a runtime route before executing the next K-step action chunk.

The route mapping separates full-task execution into base execution and local contact-window refinement. Labels outside predefined contact windows are mapped to the base route, where the frozen VLA reference chunk is executed directly. Only labels corresponding to contact-critical windows activate the associated stage-specific actor-critic refiner. Therefore, the stage estimator does not assign an entire subtask to an online refiner. Instead, it only selects short windows in which contact-aware refinement is needed. For example, approach and inter-stage transport are usually handled by the frozen VLA, while online refinement is activated around cup insertion, latch engagement/locking, egg grasping, and egg placement.

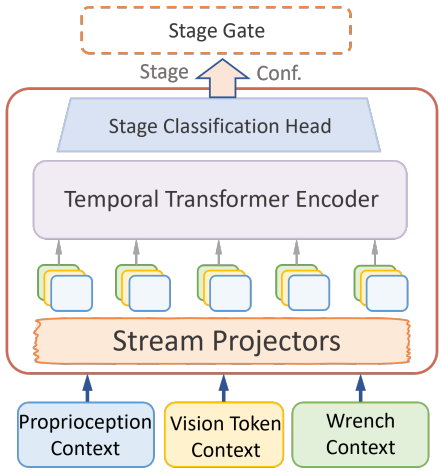


Figure 8: **Stage estimator architecture.** Policy-side features are projected, temporally aggregated, and used to predict a stage label and confidence for route filtering.

To avoid unstable switching caused by isolated prediction noise, we apply a simple route gate before execution. A predicted route is accepted only when the stage confidence exceeds 0.9 and the same mapped route is predicted for two consecutive chunks. Otherwise, the system keeps the previous stable route or falls back to the base route. Because the stage estimator reuses features already produced at the VLA query, this routing step does not require an additional VLA forward pass.

Figure 9 compares the estimated stages with ground-truth annotations over a full-task rollout. The stage estimator achieves an absolute semantic accuracy of 95.5%. The remaining mismatches are concentrated near the continuous boundaries between adjacent stages. This is expected because the physical transition between two neighboring phases is gradual rather than instantaneous, and manual stage annotations at such boundaries can vary slightly. Direct inspection of the aligned camera observations shows that the physical manipulation states corresponding to the ground truth (GT) and estimation results (ER) are visually and functionally similar across stable intervals. Thus, these boundary-level discrepancies mainly reflect annotation ambiguity around phase transitions rather

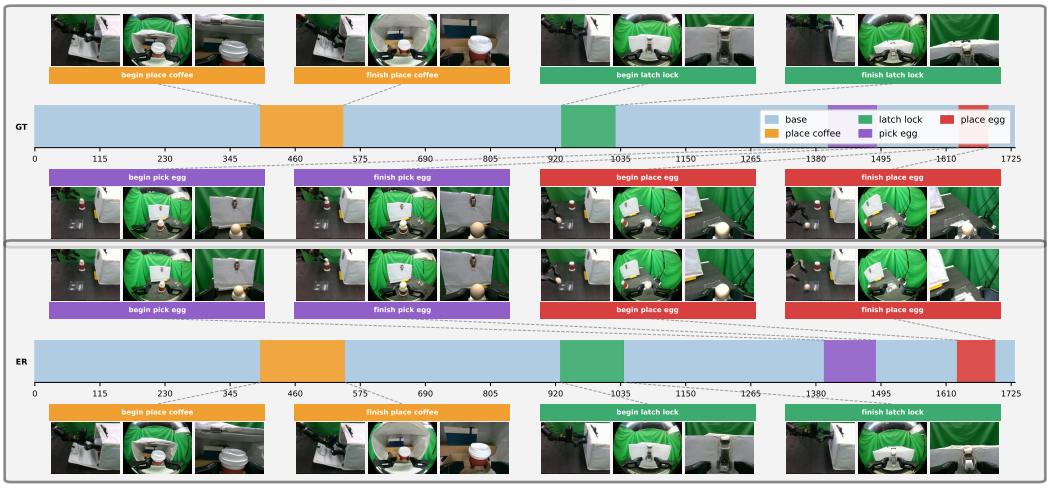


Figure 9: **Detailed alignment of predicted stages against ground truth over full-task execution.** The visualization trace shows chunk-level stage estimates aligned with sequential camera observations. Minor discrepancies mainly appear at continuous phase boundaries, while the physical manipulation states in GT and ER remain consistent across stable intervals.

than incorrect recognition of stable task stages. In deployment, the confidence and temporal-stability gate further reduces the chance that such boundary fluctuations lead to unstable actor switching. Moreover, the stage-specific online actors operate on continuous wrench-conditioned contexts, making the refinement robust to small timing shifts around the beginning or end of a contact window.

When a chunk is routed to a contact window, the corresponding stage-specific actor-critic refiner is used. The actor embeds the input groups in c_t, concatenates the features, and directly outputs the refined executable chunk:

$$A_t^{\phi} = \pi_{\phi_{g_t}}(c_t). \tag{5}$$

The actor does not predict an explicit residual added to A_t^{ref}. Action chunks are normalized internally for training and converted back to executable robot commands before deployment.

Replay buffer and control-source labels. Before main online adaptation, each stage-specific refiner is initialized with warmup data, where execution mainly follows the VLA reference and human intervention is applied near failure or unsafe contact. After warmup, the corresponding actor snapshot is used in closed-loop execution.

Online data are stored at the executed K-step chunk level. Each replay entry contains A_t^{ref}, the executed chunk A_t^{data}, robot proprioception, the current measured wrench, recent measured wrench history, the aligned predicted wrench cue $\hat{W}_t^{\mathrm{ref}}$, rewards, terminal flag, next context, and per-step control-source labels. For each low-level step j, $\sigma_{t,j} \in \{\mathrm{policy}, \mathrm{human}\}$ records whether the action was generated by the learned policy or by human intervention. These labels are not actor observations; they are used only for behavior regularization, intervention-boundary target construction, and diagnostics.

A.3.4 Training Details and Hyperparameters

Table 5 summarizes the key hyperparameters used for reference generation and online adaptation. We report the settings that define the reference interface, execution schedule, online update cadence, intervention-boundary construction, and stage-specific actor-critic refiners. Since the Egg subtask contains two contact-rich bottleneck stages, we use separate actor-critic refiners for egg grasping and egg placement.

Table 5: **Key hyperparameters for reference generation and online adaptation.**

Reference / execution	Value
Reference horizon H	50
Executed horizon K	10
Action / wrench dimension	7 / 12
Robot command representation	delta chunk
Control frequency	20 Hz
Replanning interval	0.5 s
Wrench history length	10
Wrench history window	2.0 s
Future wrench horizon	50
Future wrench loss λ_w	0.3
VLA token dimension z_t^{vla}	2048

Online learning / AC	Value
Discount factor γ	0.99
Update-to-data ratio	5
Critic update frequency	every step
Actor update frequency	every 2 steps
Target-network EMA τ	0.005
Actor snapshot interval	100 steps
Checkpoint interval	1000 steps
Intervention cost c_{int}	1.0
Human-control threshold ρ_{int}	0.5
Warmup replay chunks	300
Warmup gradient updates	5k
Hidden dimension	256
Coffee Cup AC layers	3 / 3
Egg-grasp AC layers	3 / 3
Egg-place AC layers	3 / 3
Latch AC layers	4 / 4

Online learning uses replay-buffer updates. For each newly added replay transition, the learner performs five gradient updates on average; the critic is updated at every learner step, while the actor is updated every two learner steps. Checkpoints are saved every 1000 learner steps.

These settings specify the TORL-VLA implementation used in our online adaptation experiments, including the execution horizon, control frequency, warmup schedule, online update cadence, and stage-specific actor-critic architecture.

A.4 Additional Diagnostics

A.4.1 Wrench and Contact-State Analysis

This section analyzes tactile-derived wrench signals during contact-rich execution. The goal is not to introduce an additional benchmark metric, but to show how measured wrench and predicted future wrench behave in the task phases visualized below.

Measured contact evolution. Figures 12 and 10 show representative real-execution segments from the latch-locking and cup-insertion tasks. The wrench curves shown in this analysis are measured from the right-fingertip tactile sensor. The latch sequence in Figures 12 covers the process from the initial latch grasp to latch flipping. During this process, the commanded action changes smoothly, while the measured fingertip wrench exhibits clear temporal variation. In the cup task shown in Figures 10, the gripper must maintain a normal force f_z to stably hold the cup, while also applying an appropriate downward force f_y during insertion to place the cup into the holder. These examples show that the contact state in our tasks is not a simple binary contact/no-contact variable, but evolves continuously with the interaction among the robot, the manipulated object, and the environment. Beyond characterizing normal contact evolution, the wrench signal also provides cues for identifying deviations from the expected execution. For example, grasping the latch too tightly may introduce excessive contact forces that prevent the latch from being flipped. Similarly, during cup insertion, a slight positional offset may be difficult to detect from visual observations alone, but it can be revealed by abnormal variations in f_y.

Future-wrench prediction. Figure 13 evaluates whether the reference model can predict future contact trends. Each panel pairs the current visual observation and the observation after 50 action steps with the predicted and measured fingertip wrench norms. The examples cover latch locking, cup insertion, and egg grasping, thereby testing the prediction head under mechanism contact, environmental contact, and delicate object contact. These curves suggest that the reference

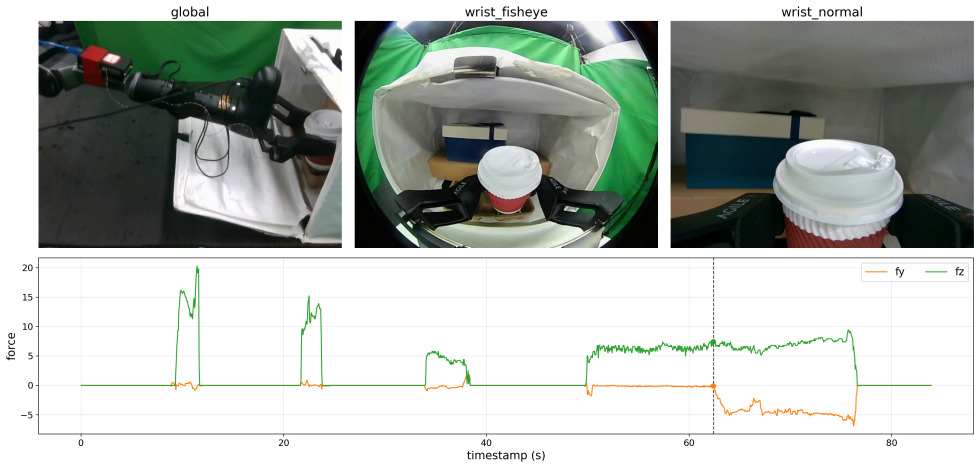


Figure 10: Measured contact evolution in the cup-insertion task. The wrench curves are measured from the right-fingertip tactile sensor. The gripper maintains the normal force f_z to stably hold the cup while applying an appropriate downward force f_y during insertion.

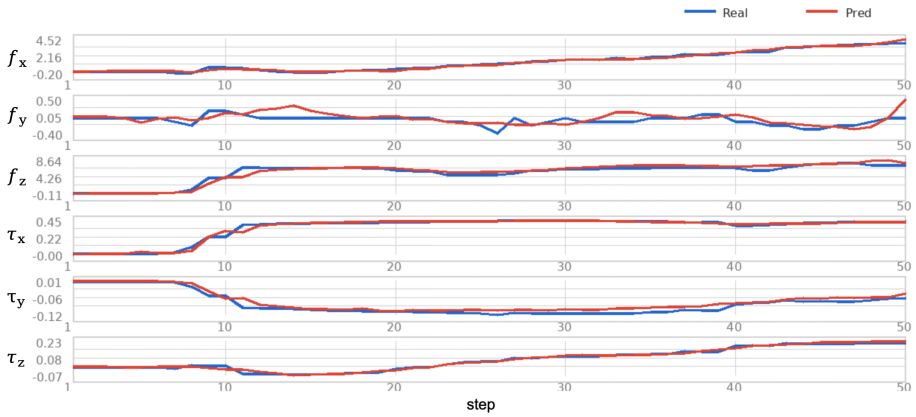


Figure 11: Detailed six-dimensional future-wrench prediction on a latch-locking segment. The figure compares the predicted and measured six-dimensional wrench trajectories for the left-fingertip tactile sensor, including the three force components and three torque components. This view complements the norm-based summaries in Figure 13 by showing whether the model captures the component-wise contact evolution.

model anticipates the contact consequences of its proposed future action. Figure 11 further expands one latch-locking segment into the left-fingertip components, showing the predicted and measured force–torque trajectories along all six dimensions.

Beyond history extrapolation. A potential concern is that the future-wrench head may simply extrapolate the trend of the input wrench history. To probe this, we examine the examples in subpanels (a1), (b1), and (c1) of Figure 13. In these cases, the input wrench histories are equal to zero before substantial contact is established. A history-copying predictor would therefore remain near zero or continue the previous flat trend. Instead, the model predicts a future rise or change in the fingertip wrench while generating the corresponding reference action. This suggests that the reference model infers upcoming contact dynamics from the visual state and the shared action–wrench generation context, rather than merely extrapolating short-term wrench history.

A.4.2 Intervention-Censored Critic Diagnostics

This section provides process-level diagnostics for the intervention-censored critic on the Latch-Lock stage, i.e., the locking contact window within the Latch subtask. We use this stage because

it contains sufficient policy-to-human boundary contexts for stable analysis. These diagnostics are not additional final evaluation metrics; they are used only to inspect whether Q_{ic} captures the intended intervention-risk signal and whether the learned actor moves away from actions with lower intervention-censored value. Unless otherwise specified, all diagnostics are computed from the final Latch-Lock online replay of the full model, and critic values are evaluated using the final critic checkpoint.

Boundary contexts and diagnostic scores. For diagnostics, we define a clean policy-generated chunk as a mostly policy-generated chunk that is not followed by an intervention-dominated chunk. A policy-to-human boundary chunk is a mostly policy-generated chunk immediately preceding an intervention-dominated chunk, using the same threshold as the intervention-boundary mask in the main method. A boundary context denotes the actor context c_t associated with such a boundary chunk. These diagnostic labels are used only for offline analysis and are not provided to the actor as observations.

All critic values are reported as the minimum over the double-Q pair. For AUC-based boundary-risk ranking, we use $-Q(c, A)$ as the score: lower critic values indicate lower predicted value, and for Q_{ic} this is intended to correspond to higher intervention risk. Thus, a larger AUC means that a critic more reliably ranks boundary chunks above clean policy-generated chunks under this diagnostic score.

Boundary-context distribution and actor behavior. Table 6 reports two diagnostics on the final Latch-Lock replay. The left table shows how policy-to-human boundary contexts are distributed across online episodes, indicating that they are recurring contact-risk events rather than a few isolated outliers. The right table compares the logged pre-intervention action, the VLA reference action, and the final actor action on the same 165 boundary contexts. The final actor achieves higher average Q_{ic} value than both the logged action and the VLA reference, suggesting that the learned actor moves away from lower-IC-value actions on these boundary contexts. We call a boundary context *IC-hinge-active* when the reference-relative IC hinge in the actor loss is active, i.e., $\min_i Q_{\mathrm{ic}}^{(i)}(c, A^{\phi}) < \min_i Q_{\mathrm{ic}}^{(i)}(c, A^{\mathrm{ref}})$.

Table 6: **Boundary-context distribution and actor behavior.** Left: episode-level distribution of policy-to-human boundary contexts in the final Latch-Lock replay. Right: action comparison on the same 165 boundary contexts using the final IC critic.

Metric	Value
Episodes with replay	162
Boundary contexts	165
Episodes $\geq$ 1 boundary	125 / 162 = 77.2%
Episodes $\geq$ 2 boundaries	32 / 162 = 19.8%
Episodes $\geq$ 3 boundaries	8 / 162 = 4.9%
Mean per episode	1.02
Median per episode	1
90th percentile	2
Maximum per episode	3

Actor diagnostic	Value
Logged $Q_{\mathrm{ic}}(c, A^{\mathrm{data}})$	-1.003
Reference $Q_{\mathrm{ic}}(c, A^{\mathrm{ref}})$	-0.573
Final actor $Q_{\mathrm{ic}}(c, A^{\phi})$	-0.271
ΔQ_{ic}: actor vs. reference	+0.301
ΔQ_{ic}: actor vs. logged data	+0.731
IC-hinge-active boundary contexts	21 / 165 = 12.73%

Boundary-risk separation. Table 7 compares Q_{ic} and Q_{task} on clean policy-generated chunks and policy-to-human boundary chunks. Clean Mean and Boundary Mean denote the average critic values on the two groups, and Clean–Boundary is their difference. AUC is computed using $-Q(c, A)$ as the boundary-risk score. The IC critic separates boundary chunks from clean chunks much more clearly than the task critic, supporting the separation between task-return estimation and intervention-risk modeling.

Actor checkpoint comparison. Table 8 compares actor checkpoints saved at learner global steps 10k, 20k, and 33k. To avoid comparing values from drifting critic checkpoints, we use the final Q_{ic} as a fixed diagnostic scorer for all actor checkpoints. The comparison is retrospective: all checkpoints are evaluated on the same final boundary-context set, although not all of these contexts

Table 7: **Boundary-risk separation by task and IC critics.** AUC is computed using $-Q(c, A)$ as the boundary-risk score; for Q_{ic}, lower values are intended to indicate higher intervention risk.

Critic / Action	Clean Mean	Boundary Mean	Clean–Boundary	AUC ↑
$Q_{\mathrm{ic}}(c, A^{\mathrm{data}})$	0.073	-1.003	1.076	0.999
$Q_{\mathrm{ic}}(c, A^{\mathrm{ref}})$	-0.007	-0.573	0.565	0.913
$Q_{\mathrm{ic}}(c, A^{\phi})$	0.091	-0.271	0.363	0.806
$Q_{\mathrm{task}}(c, A^{\mathrm{data}})$	0.333	0.247	0.086	0.621
$Q_{\mathrm{task}}(c, A^{\mathrm{ref}})$	0.316	0.241	0.074	0.626
$Q_{\mathrm{task}}(c, A^{\phi})$	0.351	0.272	0.079	0.618

were present in replay at earlier checkpoints. A boundary context is counted as IC-hinge-active at a checkpoint if the actor action at that checkpoint has lower IC value than the VLA reference under the same context. A 10k IC-hinge-active context is considered resolved if it is no longer IC-hinge-active under the final actor. The number of IC-hinge-active boundary contexts decreases from $89/165$ at 10k to $21/165$ at the final 33k checkpoint, indicating that the actor progressively reduces boundary contexts where its action has lower IC value than the VLA reference.

Table 8: **Actor checkpoint comparison on boundary contexts.** The final IC critic is used as a fixed scorer on the same final boundary-context set. Checkpoints are indexed by learner global step. Margin statistics are computed only on the 89 contexts that are IC-hinge-active at the 10k checkpoint.

Retrospective actor diagnostic	Value
IC-hinge-active contexts at 10k global-step checkpoint	89 / 165 = 53.94%
IC-hinge-active contexts at 20k global-step checkpoint	66 / 165 = 40.00%
IC-hinge-active contexts at final 33k global-step checkpoint	21 / 165 = 12.73%
10k IC-hinge-active contexts resolved by final actor	71 / 89 = 79.8%
Mean margin on 10k IC-hinge-active contexts	-0.207
Mean margin on the same contexts at final 33k checkpoint	+0.213
Mean margin improvement on the same contexts	+0.421

Online intervention behavior. Table 9 reports rollout-level process diagnostics during late online training. These statistics are not used as final autonomous evaluation metrics. Human-Controlled Step Ratio is computed at the low-level control-step level as the fraction of steps involving human control, and Avg. Human Interventions counts human takeover events per episode. Compared with the variant without the IC critic under the same late-training window, the full model shows lower human-intervention reliance and more intervention-free episodes during late online training. This is consistent with the intended role of Q_{ic}: reducing policy behaviors that tend to lead into human correction.

Table 9: **Latch-Lock online intervention diagnostics.** Statistics are computed over late online training rollouts and are not used as final autonomous evaluation metrics. Human-Controlled Step Ratio is computed at the low-level control-step level, and Avg. Human Interventions counts human takeover events per episode.

Variant / Window	Human-Controlled Step Ratio ↓	Avg. Human Interventions ↓	Intervention-Free Episodes ↑
w/o IC Critic, last 100 ep.	27.78%	13.40	11.0%
Full model, last 100 ep.	25.32%	10.68	29.0%
w/o IC Critic, last 50 ep.	25.42%	13.10	12.0%
Full model, last 50 ep.	18.12%	7.48	40.0%

A.5 Deployment Efficiency

A.5.1 Deployment Inference Latency

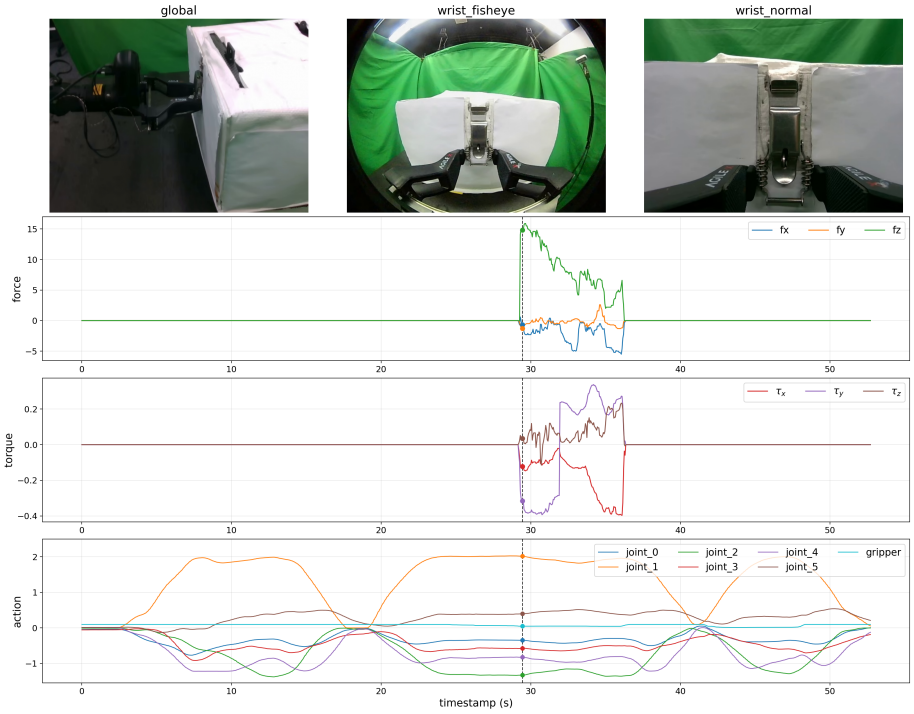
Table 10 reports deployment-time inference latency measured on a single NVIDIA RTX 4090 GPU with batch size one. We measure online policy computation only, excluding robot communication, observation acquisition, low-level controller execution, and environment preparation. All policy-query latencies are averaged over 30 timed warm-call queries after excluding the first JIT/initialization call and using five warmup queries.

During a TORL-VLA policy query, the frozen wrench-aware VLA generates the reference action sequence and predicted future wrench sequence. The stage estimator performs chunk-level routing using policy-query features and does not require an additional VLA forward pass. When the current chunk is routed to a contact-critical stage, the corresponding stage-specific online actor refines the reference action chunk. The reported TORL-VLA total is measured end-to-end for this actor-refinement route, rather than obtained by summing separately measured module times. The base $\pi_{0.5}$ row is included only as a reference-policy comparison and is not part of the TORL-VLA total.

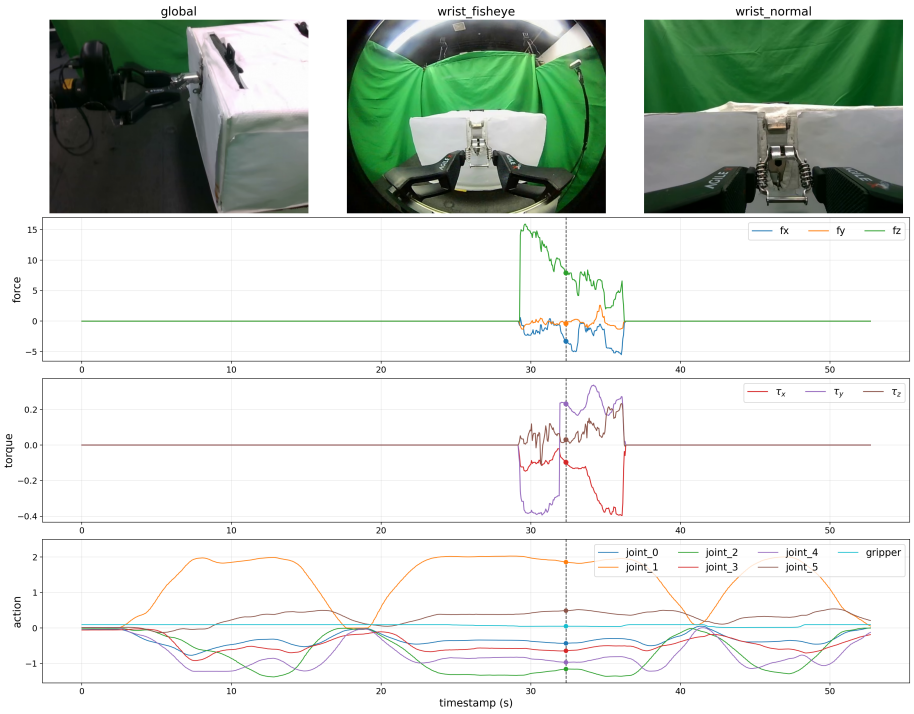
Table 10: **Deployment inference latency.** Mean warm-call latency is measured on a single NVIDIA RTX 4090 GPU with batch size one.

Module / Query	Timed Calls	Mean Latency
Base $\pi_{0.5}$ policy query (comparison)	30	94.7 ms
TORL-VLA wrench-aware reference query	30	100.2 ms
Stage estimator routing	30	1.7 ms
Stage-specific online actor	30	1.1 ms
End-to-end TORL-VLA query with actor refinement	30	103.0 ms

Compared with the base $\pi_{0.5}$ policy query, the full TORL-VLA actor-refinement route increases mean policy-query latency by 8.3 ms, corresponding to an 8.8% relative overhead. The stage estimator and online actor add only a few milliseconds of latency, indicating that TORL-VLA introduces lightweight deployment overhead while retaining the same real-time policy-query regime.

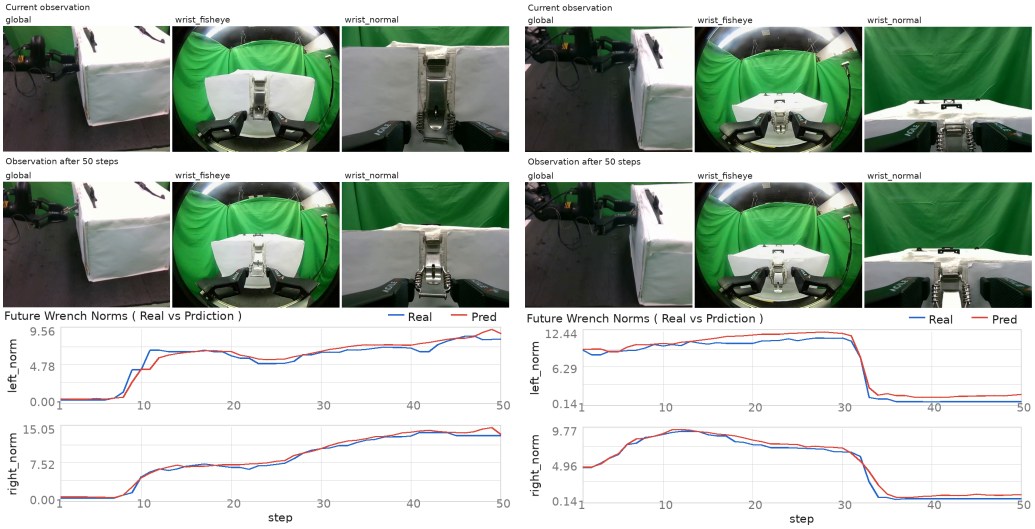


(a) Initial latch-grasping phase



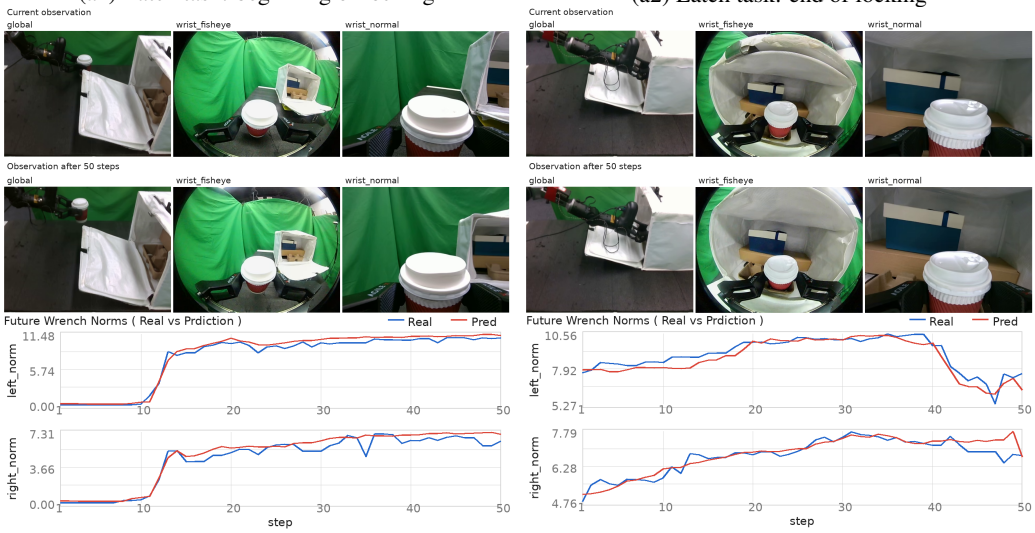
(b) Latch-flipping phase

Figure 12: Measured contact evolution in the latch-locking task. Each panel shows synchronized camera views, the measured wrench $(f_x, f_y, f_z, \tau_x, \tau_y, \tau_z)$ from the right-fingertip tactile sensor, and action traces. The sequence covers the process from initially grasping the latch to flipping it.



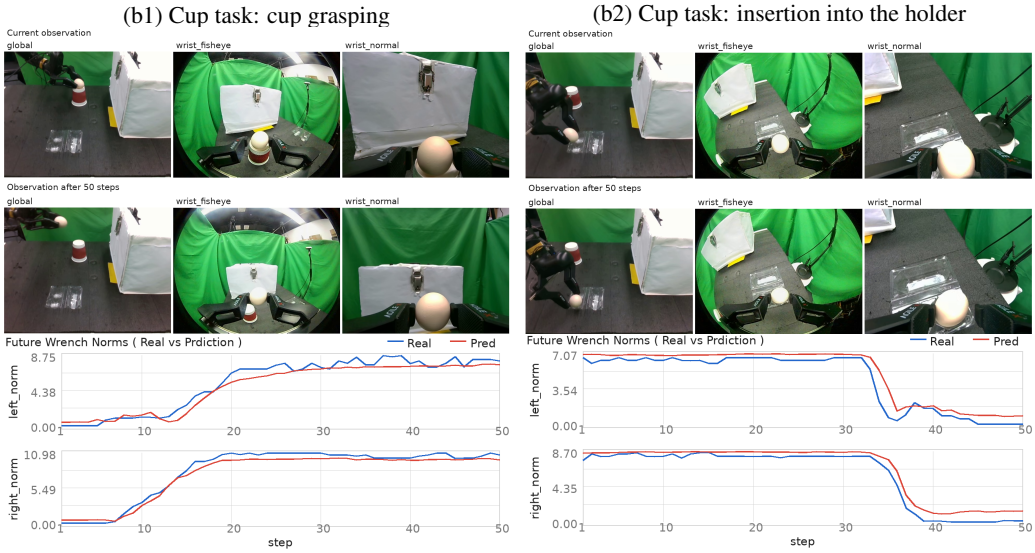
(a1) Latch task: beginning of locking

(a2) Latch task: end of locking



(b1) Cup task: cup grasping

(b2) Cup task: insertion into the holder



(c1) Egg task: beginning of grasping

(c2) Egg task: end of grasping

Figure 13: Future-wrench prediction examples across the latch, cup, and egg tasks. Each panel shows the current and post-50-step observations above, and compares the predicted future wrench norms with the measured wrench norms from the left and right fingertip tactile sensors below.