SPECTRA: FORCE-GATED SPECTRAL DIFFUSION FOR ROBOT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot manipulation requires smooth movements as well as precise adjustments. Force and tactile feedback can help a robot understand its interaction with an object, especially when visual observations alone are insufficient. However, this feedback is not equally useful throughout a task. Similarly, predicting finer motion details can help the robot make necessary corrections, but can also reproduce noise in demonstrations. An effective policy should therefore learn when force feedback and finer motion are useful. We introduce SPECTRA, a force-aware spectral diffusion policy that makes this choice during execution. It combines a coarse policy for smooth, low-frequency motion with a fine policy that uses additional force/tactile feedback and can predict the full range of motion frequencies. A learned gate uses recent force history to select the policy that generates the next action chunk. Experiments on simulated and real-robot manipulation tasks show that SPECTRA improves real-robot task success by up to 20 percentage points over the strongest evaluated baseline on the corresponding task. Comparisons with individual policies further show that the benefit of force feedback depends on action bandwidth, supporting its selective use with fine motion prediction. https://iclr202780.github.io/spectra-policy/

1 INTRODUCTION

Reliable robot manipulation requires both smooth movements and precise adjustments when grasping and handling objects. Visual observations help a robot locate objects and plan its motion, but they do not fully describe the interaction between the gripper and the object. Small misalignments, unexpected resistance, or changes in grip may be difficult to detect from images, especially when the gripper obscures the interaction. Force and tactile feedback provide complementary information that can help the robot detect these conditions and adjust its actions. This motivates force-aware manipulation policies that combine visual observations with feedback from physical interaction to achieve more reliable execution.

However, adding force and tactile inputs does not by itself ensure better actions. Their usefulness changes throughout execution: they may reveal an important interaction at one moment and provide little information beyond visual observations at another. Even when this feedback is informative, the policy must be able to translate it into suitable corrections. Restricting predictions to smooth motion can limit rapid adjustments, while allowing finer motion detail can also reproduce unwanted variation in demonstrations. These challenges connect the use of sensory information with the actions a policy can express. An effective approach should therefore consider both when force feedback is useful and when additional motion detail is needed.

We use the frequency content of action sequences to control motion detail: low frequencies describe smooth motion, while higher frequencies allow faster changes that may support corrections. Our key idea is to adapt this detail together with force feedback, using coarse motion when sufficient and enabling force-conditioned, full-spectrum prediction when it improves action prediction. Learning this choice from data accommodates changing task demands.

We introduce SPECTRA, a force-aware spectral diffusion framework with independently trained coarse and fine policies. The coarse policy predicts low-frequency motion; the fine policy uses force/tactile memory and predicts the full action spectrum. Coefficient-wise normalization balances spectral signal-to-noise ratios during training. A gate uses recent force history to select one policy

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.

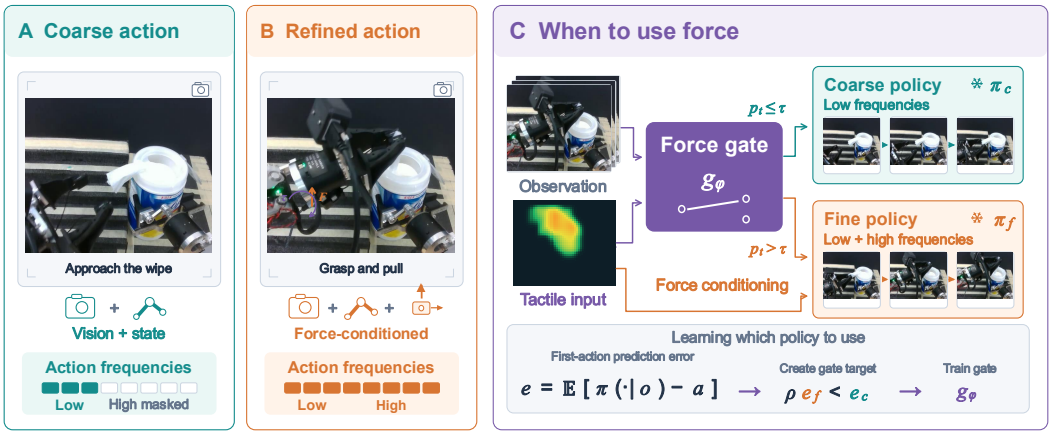


Figure 1: **Match sensing and action bandwidth to the current interaction.** A force-blind student predicts continuous actions in a low-frequency subspace; a force-aware teacher predicts the full spectrum. A gate uses recent force and torque to choose an expert, supervised by their relative action-prediction errors on calibration examples. The scenes and traces are conceptual illustrations, not measurements or observed routing outcomes.

per action chunk, learning from their relative prediction errors on held-out demonstrations without manually labeled contact phases. Simulation and real-robot experiments demonstrate gains across multiple tasks, including real-robot improvements of up to 20 percentage points over the strongest baseline per task. Ablations support both spectral balancing and the connection between force benefits and high-frequency action support. Our main contributions are as follows:

- We propose SPECTRA, a force-aware spectral diffusion framework that combines a coarse policy for low-frequency motion with a force-conditioned fine policy that predicts the full action spectrum.
- We introduce a learned gate that uses recent force history to select between the two policies. Supervision comes from their relative prediction errors on held-out examples, without requiring manually labeled contact phases.
- We validate SPECTRA in simulation and on real robots, showing gains over force-blind and force-aware baselines. Ablations support jointly selecting force conditioning and action bandwidth.

2 Related Work

Force- and tactile-conditioned policies. Visual–haptic representation learning establishes the value of interaction feedback (Lee et al., 2019), while Diffusion Policy provides a generative framework for action-sequence prediction (Chi et al., 2023). ForceMimic learns hybrid force–motion behaviors (Liu et al., 2025); Reactive Diffusion Policy introduces fast tactile feedback within action chunks (Xue et al., 2025); and ManipForce fuses asynchronous visual and force observations (Lee et al., 2025). Tactile-VLA and ForceVLA2 incorporate physical feedback into vision-language-action models and hybrid force-position control (Huang et al., 2025; Li et al., 2025). SPECTRA complements these approaches by selecting when force-aware, full-spectrum prediction is preferable to force-blind coarse motion. Its action frequencies describe within-chunk variation, not sensor sampling or feedback rates.

Spectral action representations. FAST compresses action sequences using DCT-based tokenization (Pertsch et al., 2025); and CSP conditions fine action generation on a realized coarse trajectory (Cao et al., 2026). SPECTRA instead trains independent low-frequency and full-spectrum policies, each generating a complete chunk. To address unequal spectral variances, we scale DCT coefficients before diffusion, inspired by the EqualSNR principle of balancing frequency-dependent corruption (Falck et al., 2025). Our focus is jointly selecting spectral support and force conditioning, rather than token compression or conditional coarse-to-fine generation.

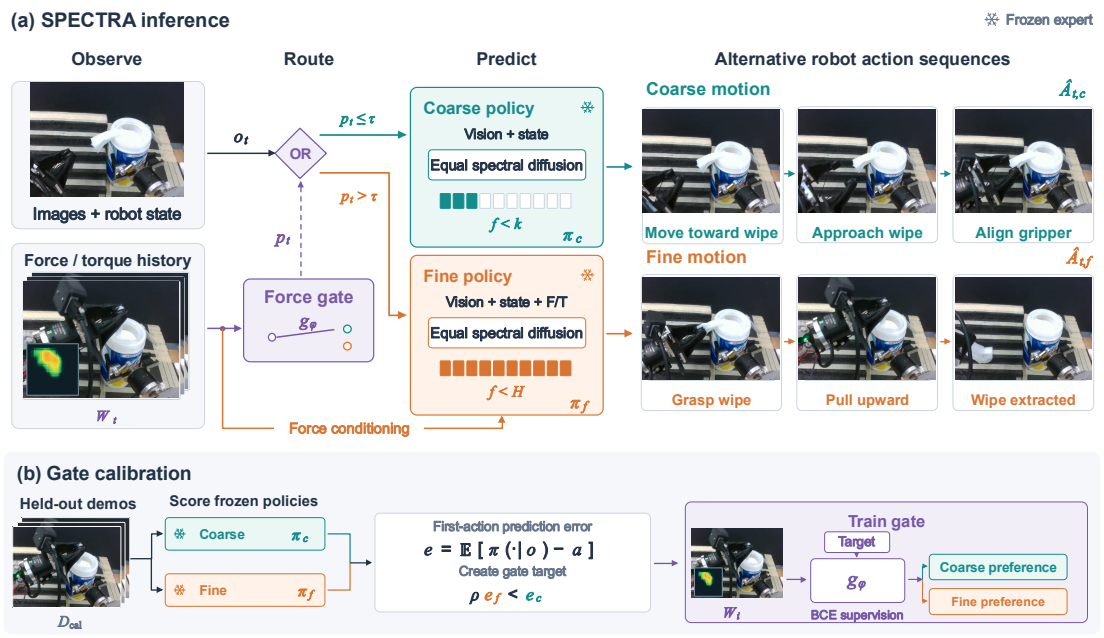


Figure 2: **Inference and gate calibration in SPECTRA. (a)** Recent force–torque history guides selection between a force-blind coarse policy with low-frequency support and a force-conditioned fine policy with full-spectrum support. Only the selected policy generates the next action chunk; the robot sequences illustrate the two alternatives. **(b)** Relative prediction errors of frozen policies on held-out demonstrations provide margin-based supervision for the gate. Colored bins indicate spectral support.

Selective conditioning and policy routing. FoAR uses predicted contact to modulate force-feature fusion (He et al., 2024), while ForceVLA routes multimodal features through a force-aware mixture of experts (Yu et al., 2026). Gated Memory Policy calibrates memory use from validation-set prediction errors, then freezes the gate and retrains the policy (Gao et al., 2026). We adopt this error-based supervision but retain two frozen action policies at deployment. A force-history gate selects one policy for the entire next chunk, jointly determining force access and spectral support. The calibration target therefore measures relative expert utility, rather than the isolated value of force.

3 Method

3.1 Problem Setup and Overview

We study imitation learning for contact-rich manipulation, where force feedback reveals visually ambiguous constraints but varies in usefulness and may introduce spurious correlations under limited demonstrations. We therefore jointly learn action prediction and when to use force-aware control. At time t, let o_t contain camera observations, proprioception, and task context, and let $w_t \in \mathbb{R}^F$ contain force, torque, and optional contact-force measurements. Their recent history is $W_t = [w_{t-L+1}, \ldots, w_t] \in \mathbb{R}^{L \times F}$. The policy $\pi_\theta(A_t \mid o_t, W_t)$ predicts an H-step chunk $A_t \in \mathbb{R}^{H \times (d_c + d_g)}$, with continuous-motion and gripper dimensions d_c and d_g.

SPECTRA combines a force-blind coarse policy $\pi_{\mathrm{c}}(A_t \mid o_t)$, restricted to low-frequency DCT coefficients, with a force-aware fine policy $\pi_{\mathrm{f}}(A_t \mid o_t, W_t)$ that models the full spectrum. Both preserve full temporal resolution for gripper commands. As shown in Figure 2, we train the policies independently on $\mathcal{D}_{\mathrm{fit}}$, freeze them, and compare their prediction errors on held-out calibration data $\mathcal{D}_{\mathrm{cal}}$. These comparisons supervise a gate $g_\phi(W_t) \in [0, 1]$ that selects one policy per new action chunk, jointly determining force conditioning and spectral support.

3.2 Spectral Action Representation

To control the motion detail available to the coarse and fine policies, we use the discrete cosine transform (DCT) to express each action sequence as a weighted sum of cosine curves. Slowly varying curves capture the overall motion, while faster-varying curves capture sharper changes. Let $X_t \in \mathbb{R}^{H \times d_c}$ contain the normalized continuous actions and $G_t \in \mathbb{R}^{H \times d_g}$ contain the normalized gripper commands. The orthonormal DCT-II computes the frequency coefficients as

$$C_t[n] = \gamma_n \sum_{j=0}^{H-1} X_t[j] \cos\left(\frac{\pi n(j+\frac{1}{2})}{H}\right). \tag{1}$$

Here $X_t[j]$ is the continuous-action vector at step j within the chunk, and $C_t[n]$ contains its coefficients at frequency index $n \in \{0, \ldots, H-1\}$. The normalization factors are $\gamma_0 = 1/\sqrt{H}$ and $\gamma_n = \sqrt{2/H}$ for $n > 0$. The coarse policy keeps only the lowest-frequency coefficients, producing smooth action chunks, while the fine policy can use all coefficients to capture faster changes. Gripper commands remain in the time domain, preserving a prediction at every step.

Inspired by EqualSNR Falck et al. (2025), we rescale the frequency coefficients so that diffusion noise affects them more evenly. Some coefficients vary much less than others across demonstrations, so the same amount of noise can hide their information sooner. We therefore put the coefficients on a common scale before adding noise:

$$Z_t = C_t / \hat{\sigma}, \tag{2}$$

where division is element-wise and $\hat{\sigma}$ contains the standard deviation of each frequency coefficient in each action dimension. Each policy estimates these values from its first training batch and keeps them fixed. We set a minimum value of 10^{-6} to avoid division by zero. This scaling gives the coefficients approximately equal variance, making the added noise comparable relative to their variation. We then train the diffusion model on $Y_t = [Z_t, G_t]$, which combines the rescaled motion coefficients with the normalized gripper commands. After sampling, we multiply the predicted coefficients by $\hat{\sigma}$ to restore their original scales, then apply the inverse DCT to recover the continuous actions. A more detailed theoretical analysis is provided in Appendix D.

3.3 Coarse and Fine Diffusion Policies

Using the representation Y_t, we train the coarse and fine policies independently on $\mathcal{D}_{\text{fit}}$. Both receive the current observations o_t—camera images, proprioception, and task context—while the fine policy additionally receives the force/tactile memory W_t. At a sampled diffusion step s, we construct the noisy input $Y_t^{(s)} = \sqrt{\bar{\alpha}_s}\, Y_t + \sqrt{1-\bar{\alpha}_s}\, \epsilon$, where $\bar{\alpha}_s$ controls the signal–noise balance and ϵ is independent standard Gaussian noise. The policies predict this noise as

$$\hat{\epsilon}_{\text{c}} = \epsilon_{\theta_{\text{c}}}(Y_t^{(s)}, s, o_t), \qquad \hat{\epsilon}_{\text{f}} = \epsilon_{\theta_{\text{f}}}(Y_t^{(s)}, s, o_t, W_t). \tag{3}$$

with respective parameters θ_{c} and θ_{f}. Each predictor comprises an observation encoder and a conditional one-dimensional U-Net. The coarse policy uses the observation embedding directly; the fine policy augments it with information retrieved from W_t through cross-attention before conditioning the U-Net.

We control motion detail through a binary loss mask M_m, shaped like Y_t, for each policy $m \in \{\text{c}, \text{f}\}$. The coarse mask retains only the lowest k frequencies of each continuous-action dimension and all gripper entries; the fine mask retains every entry. Both policies minimize

$$\mathcal{L}_m = \mathbb{E}\big[\text{mean}\big(M_m \odot (\hat{\epsilon}_m - \epsilon)^2\big)\big], \tag{4}$$

where $\odot$ denotes element-wise multiplication, the mean covers the full action tensor, and the expectation averages over demonstrations, diffusion steps, and noise. Both policies generate complete action chunks. Before decoding, we zero the coarse policy's excluded coefficients, restricting its continuous predictions to low frequencies, while retaining the fine policy's full spectrum. Further details are provided in Appendix D.

3.4 Learning the Force Gate

After training, we freeze both policies and use their relative prediction errors to supervise the gate. For each calibration example in $\mathcal{D}_{\text{cal}}$, we sample a chunk $\hat{A}_{t,m}$ from each policy $m \in \{\text{c}, \text{f}\}$ and compare it with the demonstrated chunk A_t in the same action coordinates and scale:

$$e_{t,m} = \frac{\|\hat{A}_{t,m} - A_t\|_F^2}{H(d_c + d_g)}, \qquad y_t = \mathbf{1}[\rho e_{t,\text{f}} < e_{t,\text{c}}]\,. \tag{5}$$

Here $e_{t,m}$ is the mean squared prediction error, and $y_t = 1$ favors the fine policy. The parameter $\rho > 0$ controls the selection margin; with $\rho > 1$, the fine policy must improve prediction accuracy sufficiently to be preferred.

The gate maps the flattened force/tactile memory W_t to $p_t = g_\phi(W_t) \in [0, 1]$ using a small multilayer perceptron with a sigmoid output, where ϕ denotes its parameters. We train it with binary cross-entropy:

$$\mathcal{L}_{\text{gate}} = -\mathbb{E}[y_t \log p_t + (1 - y_t) \log(1 - p_t)]\,. \tag{6}$$

We split the calibration examples into gate-training and gate-validation subsets, using the former for the expectation above and the latter for checkpoint selection. Both action policies remain frozen. At deployment, the gate requires only W_t to predict which policy to select.

3.5 Gated Action Generation

At each replanning step, the gate computes $p_t = g_\phi(W_t)$ and selects the policy that generates the next action chunk:

$$\hat{A}_t \sim \begin{cases} \pi_{\text{f}}(\cdot \mid o_t, W_t), & p_t > \tau, \\ \pi_{\text{c}}(\cdot \mid o_t), & p_t \le \tau, \end{cases} \tag{7}$$

where $\tau \in [0, 1]$ is the gate threshold. The selected policy generates the entire chunk, including gripper commands. Its continuous-action coefficients are rescaled and decoded through the inverse DCT. The controller then executes a prefix of the decoded chunk before replanning, while updating the force/tactile memory at every control step. This allows policy selection to change as the interaction unfolds.

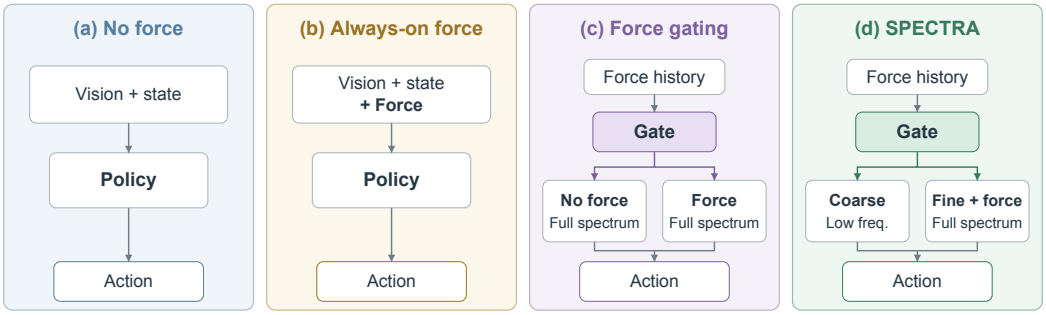
4 Experiments

We evaluate how force feedback, action frequency, and learned policy selection affect manipulation performance through four hypotheses:

- **H1 (the value of force):** Force feedback improves task success on some tasks but can reduce it on others.
- **H2 (balanced spectral learning):** Balancing signal-to-noise ratios across action frequencies improves task success.
- **H3 (force and motion detail):** Force feedback provides greater benefit when a policy can predict the full action spectrum than when it is restricted to low-frequency motion.
- **H4 (learned policy selection):** A gate trained from relative prediction errors improves task success over fixed or contact-triggered selection between the same coarse and fine policies.

4.1 Simulation Experiments

We evaluate SPECTRA on four ManiSkill Mu et al. (2021) tasks (PegInsertionSide, PickCube, PlugCharger, and StackCube) and two MimicGen Mandlekar et al. (2023) tasks (Coffee and Threading). Each task uses 100 demonstration episodes: 50 for policy training and 50 for gate calibration. Both suites run in MuJoCo, with ManiSkill tasks adapted to this simulator. Force-aware policies receive three-axis force and torque measurements from a simulated sensor between the arm and parallel-jaw gripper; MimicGen measurements are recovered by restoring recorded simulator states. ManiSkill additionally sums the 3D force magnitudes over all gripper contacts using `mj_contactForce`, giving seven force-related channels versus six for MimicGen. We report task success rates on independently generated scene configurations.



Policy	Space	ManiSkill				MimicGen	
		Peg Insertion	Pick Cube	Plug Charger	Stack Cube	Coffee	Threading
(a) No force							
DP	Action	30.0	55.0	10.0	60.0	77.5	7.5
DP	DCT	25.0	50.0	22.5	75.0	67.5	60.0
FM	Action	27.5	42.5	7.5	30.0	65.0	25.0
FM	DCT	**45.0**	42.5	10.0	52.5	40.0	35.0
(b) Always-on force							
DP	Action	35.0	57.5	12.5	55.0	62.5	50.0
DP	DCT	27.5	62.5	20.0	65.0	75.0	67.5
FM	Action	32.5	32.5	12.5	40.0	50.0	52.5
FM	DCT	42.5	55.0	10.0	55.0	32.5	45.0
(c) Force gating							
DP	Action	37.5	57.5	**32.5**	52.5	57.5	62.5
DP	DCT	27.5	62.5	25.0	50.0	57.5	77.5
FM	Action	30.0	40.0	0.0	30.0	60.0	62.5
FM	DCT	37.5	60.0	12.5	30.0	37.5	22.5
(d) SPECTRA	DCT	40.0	**77.5**	**32.5**	**82.5**	**82.5**	**80.0**

Table 1: Policy architectures and simulation success rates (%). Matching colors and labels connect each architecture to its results. DP: diffusion policy; FM: flow matching. Best results are bold.

4.1.1 Baselines and Ablations

We compare Diffusion Policy (DP) and flow matching (FM) in action and DCT space (Table 1). For each configuration, *No Force* uses vision and proprioception, *Always-on Force* adds force throughout execution, and *Force Gating* selects between force-blind and force-aware policies with full action support. SPECTRA jointly selects sensing and motion detail through a force-blind low-frequency policy and a force-conditioned full-spectrum policy. Ablations remove coefficient-wise spectral normalization and cross force conditioning with low-frequency and full-spectrum prediction to assess normalization and bandwidth-dependent force benefits. We also compare learned routing with fixed, random, force-threshold, and contact-triggered selection, holding the coarse and fine policies fixed.

4.1.2 Results and Analysis

Table 1 shows that SPECTRA outperforms force-blind action-space DP and FM on all six tasks. It achieves the highest success rates on PickCube, StackCube, Coffee, and Threading, and ties the best result on PlugCharger. Its mean success rate is 65.8%, compared with 52.9% for always using the fine policy and 50.0% for the strongest generic force-gating baseline, corresponding to gains of 12.9 and 15.8 percentage points, respectively. PegInsertionSide remains an exception: force-blind DCT-space FM achieves 45.0%, compared with 40.0% for SPECTRA.

Matched comparisons reveal benefits from force feedback, which validate **H1 (the value of force)**. On Threading, adding force raises action-space DP success from 7.5% to 50.0% and FM success from 25.0% to 52.5%. However, the same change reduces Coffee success from 77.5% to 62.5% for DP

Table 2: **Effect of spectral signal-to-noise balancing.** Task success (%) for diffusion baselines and SPECTRA variants. Spectral normalization improves SPECTRA on four tasks and preserves PlugCharger success. Bold denotes the highest success in each column.

Method	PegInsertion	PickCube	PlugCharger	StackCube	Coffee
DP (action, original)	30.0	55.0	10.0	60.0	77.5
DP (action, w/o EqualSNR)	30.0	32.5	0.0	55.0	72.5
DP (DCT, w/o EqualSNR)	20.0	45.0	20.0	67.5	60.0
SPECTRA w/o EqualSNR	37.5	70.0	**32.5**	67.5	75.0
SPECTRA	**40.0**	**77.5**	**32.5**	**82.5**	**82.5**

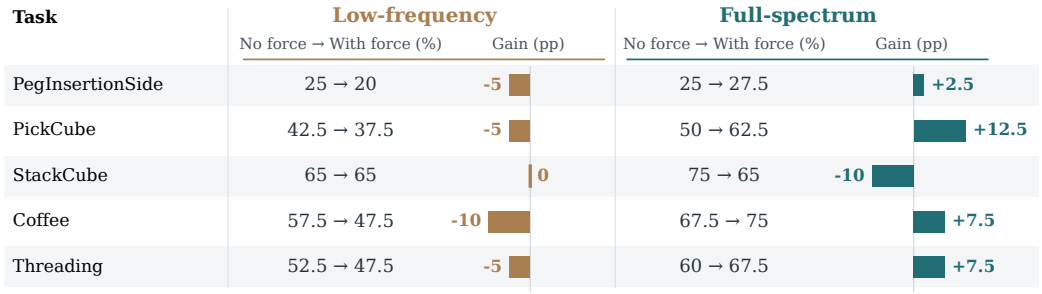


Figure 3: **Force benefits depend on action bandwidth.** Entries report success (%): without force → with force. Bars show the corresponding change in percentage points on a shared scale. Force improves full-spectrum success on four tasks but provides no gain under low-frequency restriction. All policies are evaluated without expert routing.

and from 65.0% to 50.0% for FM, motivating selective use of force. The improvement over always selecting the same fine policy supports **H4 (learned policy selection)** against the always-fine baseline: learned selection achieves higher success than using force-conditioned, full-spectrum prediction throughout execution. Generic force gating nevertheless provides mixed benefits: for DCT-space DP, it improves Threading success over always-on force from 67.5% to 77.5%, but reduces Coffee success from 75.0% to 57.5%. These results support the combined design of SPECTRA, which adapts force conditioning and action bandwidth together.

4.1.3 Effect of Spectral Signal-to-Noise Balancing

We compare SPECTRA with and without coefficient-wise spectral normalization, alongside action-space and DCT-space diffusion baselines. Table 2 shows that normalization improves success on four of the five tasks and leaves PlugCharger performance unchanged. The largest gain is on StackCube, from 67.5% to 82.5%, followed by gains of 7.5 percentage points on both PickCube and Coffee. The complete method also outperforms the listed diffusion baselines on every task. These comparisons demonstrate consistent benefits to task success, which validate the task-success component of **H2 (spectral noise balancing)**. Frequency-resolved prediction errors are still needed to establish whether normalization improves fine-action learning while preserving coarse-motion accuracy.

4.1.4 Interaction Between Force and Action Bandwidth

We compare low-frequency and full-spectrum policies with and without force conditioning, keeping expert selection disabled to isolate the interaction between force and action bandwidth. Figure 3 shows that adding force provides no gain under low-frequency restriction, but improves full-spectrum success on four of five tasks. On PickCube, the effect of force changes from a 5-percentage-point decline to a 12.5-percentage-point gain when the full spectrum is available. StackCube is the exception: force leaves low-frequency performance unchanged and reduces full-spectrum success by 10 percentage points. These results validate **H3 (force and action bandwidth)** on four tasks, showing

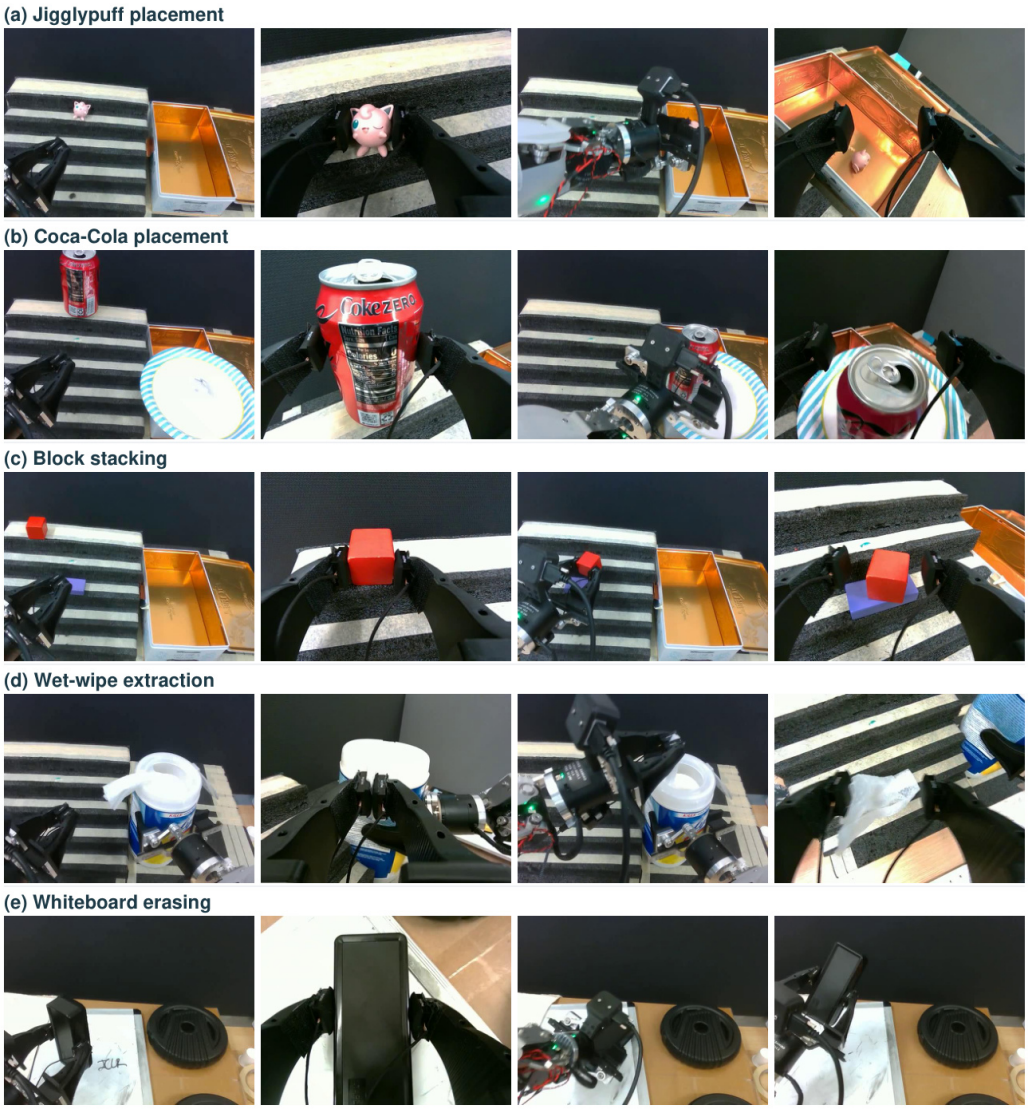


Figure 4: **Real-robot tasks.** Four chronological stages of each task, observed from fixed and wrist-mounted cameras.

that access to finer motion detail can increase the benefit of force feedback. They motivate adapting force conditioning and action bandwidth together, while highlighting that full-spectrum prediction does not guarantee a benefit from force or establish its relevance exclusively to high-frequency motion.

4.2 Real-Robot Experiments

We evaluate SPECTRA on an OpenArm robot equipped with a fixed camera and a wrist-mounted camera. Tasks include Jigglypuff placement, Coca-Cola placement, block stacking, wet-wipe extraction, and whiteboard erasing (Figure 4). Coca-Cola placement requires placing the can without making an audible sound, making gentle contact part of the success criterion. Whiteboard erasing measures the percentage of text removed; the other tasks are evaluated by task success. We compare against force-blind, action-space Diffusion Policy (DP) and flow matching (FM), with additional force-conditioning and representation comparisons reported in the appendix.

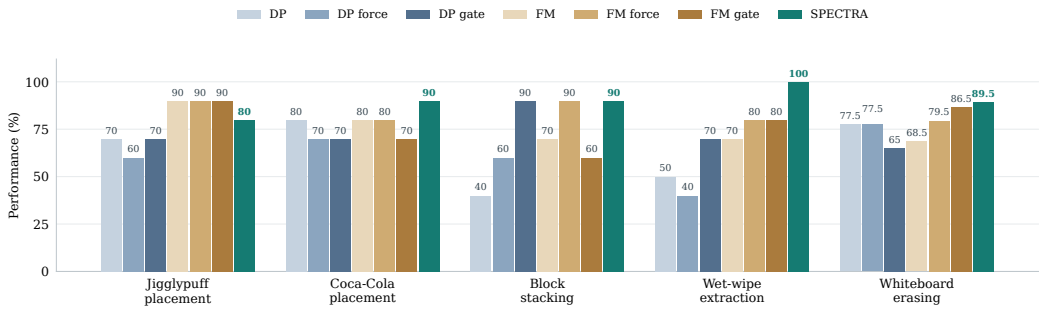


Figure 5: **Real-robot performance.** (a) Task success, including silent placement for Coca-Cola. (b) Percentage of text erased from the whiteboard. DP and FM predict actions without force conditioning. Bars show reported point estimates.

4.2.1 Experimental Setup

We evaluate SPECTRA on an OpenArm robot equipped with a fixed camera and a wrist-mounted camera across five tasks: Jigglypuff placement, Coca-Cola placement, block stacking, wet-wipe extraction, and whiteboard erasing (Figure 4). For each task, we collect 80 demonstration episodes, using 40 for policy training and 40 for gate calibration. Coca-Cola placement additionally requires placing the can without an audible contact sound, making gentle placement part of the success criterion. We evaluate whiteboard erasing by the percentage of text removed and the remaining tasks by success rate. We compare against force-blind, action-space Diffusion Policy (DP) and flow matching (FM), with additional comparisons of force conditioning and action representation provided in the appendix.

4.2.2 Results and Analysis

The five tasks test distinct demands on force-aware control. Jigglypuff placement and block stacking require accurate positioning and controlled release, while Coca-Cola placement requires gentle contact without audible sound. For wet-wipe extraction, a drop in resistance can signal complete removal. Whiteboard erasing requires sustained contact and consistent pressure against the board.

Figure 5 shows that SPECTRA outperforms force-blind DP and FM on four of five tasks. Wet-wipe extraction reaches 100% success, compared with 50% for DP and 70% for FM; quiet Coca-Cola placement reaches 90%, versus 80% for both baselines. Whiteboard erasing removes 89.5% of the text, compared with 77.5% for DP and 68.5% for FM. Jigglypuff placement is the exception: SPECTRA achieves 80% success, exceeding DP but below FM's 90%. Together with matched simulation comparisons, these results support **H1 (the value of force)**. The physical experiments evaluate the complete framework, while simulation ablations isolate the contributions of force conditioning, action bandwidth, and spectral normalization.

5 Conclusion

SPECTRA jointly adapts force conditioning and action bandwidth by selecting between coarse force-blind and full-spectrum force-conditioned policies. Spectral normalization balances signal-to-noise ratios, while held-out prediction errors supervise selection without manually labeled contact phases. Simulation and real-robot experiments demonstrate gains across multiple tasks. Ablations support spectral normalization and show greater force benefits with full-spectrum prediction on four of five tasks.

Limitations and Future Work. Performance remains task dependent, and offline prediction errors may not fully reflect execution success. The gate also selects between two policies with fixed spectral support. Future work will explore outcome-based gate supervision, adaptive action bandwidth, and generalization to new objects, sensing conditions, and robot platforms.

AI USE STATEMENT

We use generative AI tools, including ChatGPT and Codex, to assist with language polishing, code development, and figure design, layout, and formatting. The authors are responsible for the research ideas, methodology, experimental design, and interpretation of results, and review and revise AI-assisted material before inclusion. Reported experimental results come from policy evaluations rather than AI-generated estimates. Generative AI is not a component of the deployed SPECTRA policy. The authors take full responsibility for the final text, technical claims, code, and figures.

REPRODUCIBILITY STATEMENT

Section 3 specifies the spectral action representation, policy architectures, and supervised gate objective. Section 4 describes the simulation and real-robot experiments and evaluation metrics. Appendix A details demonstration allocation, force-measurement provenance, calibration, and evaluation protocols. Appendix B describes the architecture, configuration defaults, and inference procedure. Appendices C and D provide the theoretical analyses, while Appendices E and F present complete real-robot results and simulation task descriptions. The reported experiments use demonstration-disjoint policy training and calibration, consistently scaled gate-error comparisons, and expert switching without stale action or sensing state.

REFERENCES

Shuxin Cao, Liquan Wang, Walker Byrnes, Yiye Chen, Yilun Du, and Animesh Garg. Hierarchical policy learning via spectral decomposition. *arXiv preprint arXiv:2606.29570*, 2026. URL https://arxiv.org/abs/2606.29570.

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.

Fabian Falck, Teodora Pandeva, Kiarash Zahirnia, Rachel Lawrence, Richard Turner, Edward Meeds, Javier Zazo, and Sushrut Karmalkar. A fourier space perspective on diffusion models. *arXiv preprint arXiv:2505.11278*, 2025.

Yihuai Gao, Jeff Jinyun Liu, Shuang Li, and Shuran Song. Gated memory policy: In-context memorization and adaptation. *arXiv preprint arXiv:2604.18933*, 2026. URL https://arxiv.org/abs/2604.18933.

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.

Jialei Huang, Shuo Wang, Fanqi Lin, Yihang Hu, Chuan Wen, and Yang Gao. Tactile-vla: Unlocking vision-language-action model's physical knowledge for tactile generalization. *arXiv preprint arXiv:2507.09160*, 2025. URL https://arxiv.org/abs/2507.09160.

Geonhyup Lee, Yeongjin Lee, Kangmin Kim, Seongju Lee, Sangjun Noh, Seunghyeok Back, and Kyoobin Lee. Manipforce: Force-guided policy learning with frequency-aware representation for contact-rich manipulation. *arXiv preprint arXiv:2509.19047*, 2025.

Michelle A Lee, Yuke Zhu, Peter Zachares, Matthew Tan, Krishnan Srinivasan, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Learning multimodal representations for contact-rich tasks. *arXiv preprint arXiv:1907.13098*, 2019.

Yang Li, Zhaxizhouma, Hongru Jiang, Junjie Xia, Hongquan Zhang, Jinda Du, Yunsong Zhou, Jia Zeng, Ce Hao, Jieji Ren Qiaojun Yu, Cewu Lu, Yu Qiao, and Jiangmiao Pang. Forcevla2: Unleashing hybrid force-position control with force awareness for contact-rich manipulation. *arXiv preprint arXiv:2603.15169*, 2025. URL https://arxiv.org/abs/2603.15169.

Wenhai Liu, Junbo Wang, Yiming Wang, Weiming Wang, and Cewu Lu. Forcemimic: Force-centric imitation learning with force-motion capture system for contact-rich manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1105–1112. IEEE, 2025.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023. URL https://arxiv.org/abs/2310.17596.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations, 2021. URL https://arxiv.org/abs/2107.14483.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. URL https://arxiv.org/abs/2501.09747.

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 *Robotics: Science and Systems*, 2025. URL https://arxiv.org/abs/2503.02881.

Jiawen Yu, Hairuo Liu, Qiaojun Yu, Jieji Ren, Ce Hao, Haitong Ding, Guangyu Huang, Guofan Huang, Yan Song, Panpan Cai, et al. Forcevla: Enhancing vla models with a force-aware moe for contact-rich manipulation. *Advances in Neural Information Processing Systems*, 38:93409–93439, 2026.

A Data and Evaluation Protocol

Tasks and demonstration budgets. Our simulation experiments cover PegInsertionSide, PickCube, PlugCharger, and StackCube in custom MuJoCo ports of ManiSkill, and `coffee_d1` and `threading_d0` from MimicGen (Mandlekar et al., 2023). As specified in Section 4.1, each task provides 100 demonstrations, allocated equally between policy training and gate calibration. Real-robot experiments use OpenArm with a fixed camera and a wrist-mounted camera on Jigglypuff placement, Coca-Cola placement, block stacking, wet-wipe extraction, and whiteboard erasing. Each real-robot task provides 80 demonstrations, with 40 allocated to policy training and 40 to gate calibration. These budgets include calibration demonstrations.

Simulation force measurements. The ManiSkill MuJoCo wrappers read three end-effector force channels and three torque channels from the robot interface. A seventh channel summarizes gripper contact by summing the magnitudes of three-dimensional contact forces involving gripper geometries, obtained using `mj_contactForce`. This is a continuous contact-force signal rather than a binary contact flag, and the measurements originate from MuJoCo rather than a conversion of PhysX forces.

For MimicGen Coffee, the force-augmentation script restores each recorded simulator state, runs forward computation, and reads the robot's force and torque outputs. It appends the resulting six channels while preserving the original demonstration observations and actions. These annotations are reconstructed simulation signals; the procedure does not by itself establish exact recovery of the forces present during original data collection.

Policy fitting and gate calibration. Policy training and gate calibration use disjoint demonstration IDs, with trajectories partitioned before constructing temporal windows. Both experts use the same policy-training subset and remain frozen during gate calibration. For each calibration example, their predicted action chunks and the demonstration target are expressed in the same action coordinates and scale before computing the mean squared prediction errors. The fine-policy target is $y_t = \mathbf{1}[\rho e_{t,\mathrm{f}} < e_{t,\mathrm{c}}]$. Expert errors are cached separately, allowing the margin ρ to be varied without repeating policy sampling.

Evaluation scenes and metrics. Simulation performance is measured by task success on independently generated scene configurations. The reset-based ManiSkill evaluation utility generates a bank of 300 initial states per task for reuse across methods. The size of this bank is distinct from the number of rollouts completed by an individual configuration. MimicGen evaluation loads saved initialization files. Because the ManiSkill tasks are custom MuJoCo ports, their results are not directly interchangeable with those from the original PhysX environments.

Real-robot placement, stacking, and extraction are evaluated by task success. Coca-Cola placement additionally requires placing the can without audible contact sound. Whiteboard erasing is evaluated by the percentage of text removed rather than binary task success. The tables and figures report point estimates.

Comparison structure. The main comparisons cross generative backbone (DP or FM), action representation (action or DCT space), and force use (none, always-on, or generic gating). Generic gating selects between force-blind and force-aware policies with full action support. SPECTRA instead selects between a force-blind low-frequency policy and a force-conditioned full-spectrum policy. The force–bandwidth ablation evaluates individual policies without routing, while the spectral-balancing ablation compares SPECTRA with and without coefficient-wise scaling.

Policy fitting and gate calibration. Policy training and gate calibration use disjoint demonstration IDs, with trajectories partitioned before constructing temporal windows. Both experts use the same policy-training subset and remain frozen during gate calibration. For each calibration example, their predicted action chunks and the demonstration target are expressed in the same action coordinates and scale before computing the mean squared prediction errors. The fine-policy target is $y_t = \mathbf{1}[\rho e_{t,\mathrm{f}} < e_{t,\mathrm{c}}]$. Expert errors are cached separately, allowing the margin ρ to be varied without repeating policy sampling.

B Implementation Details

We describe the policy architectures, spectral representation, force conditioning, and gate-training procedure below. Execution horizon and temporal aggregation are configured per task rather than fixed globally.

Observation encoding. The default RGB encoder uses a pretrained, trainable ResNet-18-based backbone shared across camera streams. For diffusion policies, batch normalization is replaced with group normalization. Visual features are projected to 256 dimensions, and proprioceptive inputs use a separate MLP projection. The default observation configuration uses one observation step. Image augmentation combines color jitter with translation augmentation.

Spectral action representation. The action normalizer maps action dimensions to the range $[-1, 1]$ using fitted limits. Continuous actions are transformed using an orthonormal DCT-II along the temporal axis of a chunk of length $H = 16$. Gripper commands remain in the time domain. For each continuous-action dimension and DCT frequency, the implementation estimates the standard deviation across the first training batch, clamps it below at 10^{-6}, and stores it for subsequent training and inference. DCT coefficients are divided by these scales without additional DCT-space mean subtraction.

The coarse-policy configuration retains the lowest $k = 3$ frequencies by default; separate configurations provide $k \in \{4, 6, 8\}$. Its loss supervises these coefficients and all gripper entries. In the supplied implementation, excluded coefficients remain in the noisy denoiser input during training but receive no direct loss supervision. They are zeroed after sampling, before restoring spectral scales and applying the inverse DCT. The fine policy supervises every coefficient and predicts the complete action chunk independently.

Diffusion model and optimization. Both experts use a conditional one-dimensional U-Net with channel widths $[256, 512, 1024]$. The reference configuration uses the `squaredcos_cap_v2` noise schedule, 100 training diffusion indices, and 10 DDIM sampling steps. Policies minimize a noise-prediction loss, with the coarse policy applying the frequency mask described above.

The default optimizer is AdamW with learning rate 10^{-4}, weight decay 10^{-6}, $\beta_1 = 0.95$, and $\beta_2 = 0.999$. The batch size is 64, and the learning-rate schedule uses cosine decay with 1,000 warmup steps. The EMA decay ceiling is 0.9999. In the supplied implementation, EMA covers the observation encoder, observation projection, and denoiser; the separately attached force-fusion module is outside this parameter set.

Force-history conditioning. The fine policy receives an eight-step history of force-related observations. Each history entry is projected into an embedding, processed with channel LayerNorm and sinusoidal recency embeddings, and integrated into the observation conditioning through four-head cross-attention. A residual feed-forward block further processes the conditioning features. The fine-policy configuration enables this fusion unconditionally; the coarse policy has no force input. The ManiSkill wrapper provides seven channels. The Coffee configuration uses six force–torque channels.

Gate architecture and training. The gate flattens the force history and applies an MLP:

$$LF \to 128 \to 64 \to 32 \to 1. \tag{8}$$

The hidden layers use ReLU activations and dropout with probability 0.05. Training uses binary cross-entropy with logits; a sigmoid produces the selection probability at inference. The default optimizer is Adam with learning rate 10^{-3} and batch size 256. The training script defaults to 200 epochs, while the provided launch helpers request 300 epochs. Checkpoint selection minimizes validation loss. The default preference margin and decision threshold are $\rho = 1$ and $\tau = 0.5$, respectively; both can be overridden.

Action decoding and expert switching. At each replanning step, the gate selects the fine policy when $g_\phi(W_t) > \tau$ and the coarse policy otherwise. The selected expert generates an action chunk from the current observations, with the fine expert additionally conditioned on recent force history.

Table 3: **Reference configuration in the supplied source.** Values below are configuration defaults, not a reconstruction of every reported run.

Component	Reference setting
Action chunk	$H = 16$
Execution	Task-dependent action horizon and temporal aggregation
Observation history	One observation step
Visual encoder	Pretrained, trainable ResNet-18-based encoder; shared across cameras
Diffusion denoiser	Conditional 1-D U-Net; widths $[256, 512, 1024]$
Diffusion process	`squaredcos_cap_v2`; 100 training indices; 10 DDIM sampling steps
Spectral scaling	Per-frequency, per-dimension standard deviation; first training batch; floor 10^{-6}
Coarse spectral cutoff	$k = 3$ by default; configurations for $k = 4, 6, 8$ also provided
Force conditioning	Eight-step history; four-head cross-attention; recency embeddings and residual FFN
Gate network	$8F \to 128 \to 64 \to 32 \to 1$; ReLU; dropout 0.05
Gate decision	Default $\rho = 1, \tau = 0.5$
Policy optimizer	AdamW; learning rate 10^{-4}; weight decay 10^{-6}; batch size 64
Policy learning-rate schedule	Cosine decay; 1,000 warmup steps
Gate optimizer	Adam; learning rate 10^{-3}; batch size 256
Gate training duration	Script default: 200 epochs; launch helpers: 300 epochs
Gate validation	10% of cached examples; minimum validation-loss checkpoint
EMA	Decay ceiling 0.9999; force-fusion module excluded

Continuous-action coefficients are restored to their original spectral scales, decoded through the inverse DCT, and converted to controller action coordinates.

Force history is kept current throughout execution. Expert switching does not resume stale action buffers or sensing histories from an inactive policy. The execution horizon and use of temporal aggregation vary by task. These settings govern how predicted chunks are executed and combined, while policy selection occurs when a new chunk is requested.

Flow-matching reference implementation. The supplied FM implementation trains a velocity predictor along the conditional path

$$x_t = \big[1 - (1 - \sigma_{\min})t\big]x_0 + tx_1, \qquad x_0 \sim \mathcal{N}(0, I), \tag{9}$$

where x_1 is the normalized action target and $\sigma_{\min} = 0.001$. The target velocity is $x_1 - (1 - \sigma_{\min})x_0$. The default time sampler draws $z \sim \mathrm{Beta}(1.5, 1)$ and sets $t = (1 - \sigma_{\min})(1 - z)$. Sampling starts from Gaussian noise and uses 10 uniform explicit Euler steps. The final action prediction is clamped to $[-1, 1]$ before the ordinary action-decoding procedure.

C Torque Variation and Action Frequencies

We provide a local analysis connecting changes in contact torque to the frequency content of corrective actions. Consider one rotational axis under impedance control. Let $\delta m(t)$ denote an external torque disturbance and $\delta u(t)$ the rotational reference correction needed to compensate for it while preserving the intended motion. For small corrections and approximately constant effective rotational stiffness $k_{\mathrm{rot}} > 0$, the local compensation relationship is

$$\delta u(t) = -\frac{\delta m(t)}{k_{\mathrm{rot}}}. \tag{10}$$

Here $\delta m(t)$ represents an external contact-torque disturbance, rather than a commanded motor torque. Differentiating Equation (10) gives

$$\frac{d\,\delta u(t)}{dt} = -\frac{1}{k_{\mathrm{rot}}}\frac{d\,\delta m(t)}{dt}. \tag{11}$$

Thus, under this model, rapidly changing torque requires rapidly changing action corrections. A large but constant torque disturbance, however, requires only a constant correction.

To connect this relationship to our action representation, let $u_j = \delta u(j\Delta t)$ and $m_j = \delta m(j\Delta t)$ be sampled correction and torque sequences of length H, where $\Delta t > 0$ is the sampling interval. Let c_n denote the orthonormal DCT-II coefficient of u at frequency index n. The energy of the adjacent-sample differences satisfies

$$\sum_{j=0}^{H-2} \left| \frac{u_{j+1} - u_j}{\Delta t} \right|^2 = \sum_{n=0}^{H-1} \lambda_n |c_n|^2, \qquad \lambda_n = \frac{4}{\Delta t^2} \sin^2\Big(\frac{\pi n}{2H}\Big). \tag{12}$$

This identity follows by substituting the inverse DCT into the sample differences and using the orthogonality of the resulting sine functions. Since λ_n increases with n, higher-frequency coefficients contribute more strongly to rapid changes.

The difference energy also depends on signal amplitude. To separate temporal variation from overall scale, define, for a nonzero correction chunk,

$$R(u) = \frac{\sum_{j=0}^{H-2} |(u_{j+1} - u_j)/\Delta t|^2}{\sum_{j=0}^{H-1} |u_j|^2} = \frac{\sum_{n=0}^{H-1} \lambda_n |c_n|^2}{\sum_{n=0}^{H-1} |c_n|^2}. \tag{13}$$

The second equality uses the energy-preserving property of the orthonormal DCT. Thus, $R(u)$ is a weighted average of the frequency-dependent values λ_n and is unchanged by multiplying the entire signal by a nonzero constant.

A coarse correction restricted to the lowest k DCT coefficients, with $1 \le k < H$, necessarily satisfies $R(u_{\text{coarse}}) \le \lambda_{k-1}$. Moreover, Equation (10) implies $R(u) = R(m)$ because the constant stiffness scaling cancels. Consequently,

$$\boxed{R(m) > \lambda_{k-1} \quad \Longrightarrow \quad \sum_{n=k}^{H-1} |c_n|^2 > 0.} \tag{14}$$

This gives a sufficient condition under which the required compensation cannot be represented using only the coarse policy's frequency support.

The result motivates pairing force/tactile memory with access to higher-frequency action coefficients when that memory helps predict rapid contact-induced corrections. Its scope is local: the compensation model assumes approximately constant gains, small corrections, and disturbances within the controller's response bandwidth. The analysis does not imply that large force or torque magnitudes alone require high-frequency actions, or that past measurements always predict future disturbances.

D Theoretical Motivation for Spectral Signal-to-Noise Balancing

We analyze why coefficient-wise scaling is useful for diffusion over spectral actions. The analysis establishes three properties: it balances the variance-based signal-to-noise ratio across coefficients, makes denoising sensitive to relative spectral errors, and preserves the original action representation through an invertible transformation. These properties motivate the design; improved manipulation performance remains an empirical finding.

D.1 Setup

Let $X_t \in \mathbb{R}^{H \times d_c}$ denote the normalized continuous-action chunk and let $D \in \mathbb{R}^{H \times H}$ be the orthonormal DCT-II matrix. Its spectral representation is

$$C_t = DX_t, \qquad D^\top D = I. \tag{15}$$

For frequency n and action dimension d, define the training-distribution moments

$$\mu_{n,d} = \mathbb{E}[C_t[n,d]], \qquad \sigma^2_{n,d} = \mathrm{Var}(C_t[n,d]). \tag{16}$$

We initially assume $\sigma_{n,d} > 0$ and exact knowledge of these variances. Finite-sample estimates and the numerical variance floor are considered below. No Gaussianity or independence assumption on the clean coefficients is required for the SNR and loss identities.

The analysis concerns continuous-action coefficients. Gripper commands remain in the time domain, as in the main method. We use s for diffusion time and t for control time.

D.2 Unequal Spectral Variances Produce Unequal SNRs

Consider diffusion directly on an unscaled coefficient:

$$C_t^{(s)}[n,d] = \sqrt{\bar{\alpha}_s}\, C_t[n,d] + \sqrt{1-\bar{\alpha}_s}\, \epsilon[n,d], \qquad \epsilon[n,d] \sim \mathcal{N}(0,1), \tag{17}$$

where the noise is independent of the clean action. For $0 < \bar{\alpha}_s < 1$, define the variance-based SNR as the variance of the clean contribution divided by the noise variance:

$$\mathrm{SNR}^{\mathrm{raw}}_{s,n,d} = \frac{\mathrm{Var}\big(\sqrt{\bar{\alpha}_s}\, C_t[n,d]\big)}{\mathrm{Var}\big(\sqrt{1-\bar{\alpha}_s}\, \epsilon[n,d]\big)} = \frac{\bar{\alpha}_s}{1-\bar{\alpha}_s}\sigma^2_{n,d}. \tag{18}$$

Consequently, for any two coefficients with positive variance,

$$\frac{\mathrm{SNR}^{\mathrm{raw}}_{s,n,d}}{\mathrm{SNR}^{\mathrm{raw}}_{s,m,e}} = \frac{\sigma^2_{n,d}}{\sigma^2_{m,e}}. \tag{19}$$

Thus, a shared noise schedule does not imply comparable relative corruption. Whenever fine-scale coefficients have smaller variance than coarse coefficients, they have lower SNR at the same diffusion step. This is a property of the coefficient distribution, rather than an assumption that all high-frequency coefficients necessarily have small variance.

Exact balancing. Scale each coefficient by its standard deviation:

$$Z_t[n,d] = \frac{C_t[n,d]}{\sigma_{n,d}}, \qquad Z_t^{(s)}[n,d] = \sqrt{\bar{\alpha}_s}\, Z_t[n,d] + \sqrt{1-\bar{\alpha}_s}\, \epsilon[n,d]. \tag{20}$$

Since $\mathrm{Var}(Z_t[n,d]) = 1$, we obtain

$$\boxed{\mathrm{SNR}^{\mathrm{scaled}}_{s,n,d} = \frac{\bar{\alpha}_s}{1-\bar{\alpha}_s}.} \tag{21}$$

The variance-based SNR is therefore identical across all positive-variance coefficients at every interior diffusion step.

The scaling does not subtract the coefficient mean. Accordingly, Equation (21) concerns variance, not the uncentered second moment $\mathbb{E}[Z_t[n,d]^2]$. It also does not require the coefficients to become statistically independent.

D.3 Estimated Statistics and Approximate Balancing

In the implementation, each policy estimates coefficient standard deviations from its first training batch and fixes them thereafter. Write

$$\widehat{\sigma}_{n,d} = \max(\widetilde{\sigma}_{n,d}, \delta), \tag{22}$$

where $\widetilde{\sigma}_{n,d}$ is the batch estimate and $\delta > 0$ is the numerical floor. The actual scaling is $Z_t[n,d] = C_t[n,d]/\widehat{\sigma}_{n,d}$, giving

$$\mathrm{SNR}^{\mathrm{scaled}}_{s,n,d} = \frac{\bar{\alpha}_s}{1-\bar{\alpha}_s}\frac{\sigma^2_{n,d}}{\widehat{\sigma}^2_{n,d}}. \tag{23}$$

This identity precisely characterizes the remaining imbalance.

For coefficients whose estimates satisfy

$$(1-\eta)\sigma_{n,d} \le \widehat{\sigma}_{n,d} \le (1+\eta)\sigma_{n,d}, \qquad 0 \le \eta < 1, \tag{24}$$

Equation (23) implies

$$\frac{1}{(1+\eta)^2} \le \frac{\mathrm{SNR}^{\mathrm{scaled}}_{s,n,d}}{\bar{\alpha}_s/(1-\bar{\alpha}_s)} \le \frac{1}{(1-\eta)^2}. \tag{25}$$

For any collection satisfying this condition, the maximum SNR divided by the minimum SNR is bounded by

$$\frac{\max_{n,d} \mathrm{SNR}^{\mathrm{scaled}}_{s,n,d}}{\min_{n,d} \mathrm{SNR}^{\mathrm{scaled}}_{s,n,d}} \le \left(\frac{1+\eta}{1-\eta}\right)^2. \tag{26}$$

Hence, accurate estimates yield approximately balanced SNRs. A single training batch does not guarantee a small η; the bound states how estimation accuracy controls balancing. Coefficients affected by the floor, including constant coefficients, need not attain unit variance.

D.4 Effect on the Denoising Objective

The scaling also changes how prediction errors are measured. Consider one diffusion step and suppress coefficient indices. Given a predicted noise $\widehat{\epsilon}$, the associated clean-coefficient estimate is

$$\widehat{Z}_t = \frac{Z_t^{(s)} - \sqrt{1-\bar{\alpha}_s}\,\widehat{\epsilon}}{\sqrt{\bar{\alpha}_s}}, \qquad \widehat{C}_t = \widehat{\sigma} \odot \widehat{Z}_t. \tag{27}$$

Substituting the forward process gives

$$\widehat{Z}_t - Z_t = \sqrt{\frac{1-\bar{\alpha}_s}{\bar{\alpha}_s}}\,(\epsilon - \widehat{\epsilon})\,. \tag{28}$$

Therefore, coefficient-wise,

$$(\widehat{\epsilon}[n,d] - \epsilon[n,d])^2 = \frac{\bar{\alpha}_s}{1-\bar{\alpha}_s} \frac{\left(\widehat{C}_t[n,d] - C_t[n,d]\right)^2}{\widehat{\sigma}^2_{n,d}}. \tag{29}$$

For the continuous-action portion of the policy loss, let $\mathcal{I}_m$ denote the supervised coefficients: the low-frequency subset for the coarse policy and all coefficients for the fine policy. Up to the fixed averaging factor used in the implementation,

$$\mathcal{L}^{\mathrm{cont}}_m = \mathbb{E}\left[\frac{\bar{\alpha}_s}{1-\bar{\alpha}_s} \sum_{(n,d)\in\mathcal{I}_m} \frac{\left(\widehat{C}_{t,m}[n,d] - C_t[n,d]\right)^2}{\widehat{\sigma}^2_{n,d}}\right]. \tag{30}$$

Thus, noise prediction in the scaled representation is equivalent to a diffusion-step-weighted clean reconstruction objective that measures errors relative to coefficient variance. With exact statistics, errors equal to the same fraction of a coefficient's standard deviation receive equal weight, regardless of its absolute scale.

This matters when useful corrections occupy low-variance coefficients: their small amplitude alone no longer makes their relative reconstruction errors inexpensive. The identity describes the training-time clean estimate; it does not directly characterize errors after a complete iterative sampling trajectory.

D.5 Information Preservation and Scope

For any fixed positive scale estimates, encoding and decoding are inverse operations:

$$Z_t = C_t \oslash \widehat{\sigma}, \qquad X_t = D^\top (\widehat{\sigma} \odot Z_t)\,. \tag{31}$$

Spectral scaling therefore preserves the information in the action chunk. It does not itself remove fine motion or smooth the trajectory. The coarse policy's frequency mask is a separate restriction.

Orthonormality also gives

$$\|\widehat{X}_t - X_t\|_F^2 = \|\widehat{C}_t - C_t\|_F^2 = \sum_{n,d} \widehat{\sigma}^2_{n,d} \left(\widehat{Z}_t[n,d] - Z_t[n,d]\right)^2. \tag{32}$$

Consequently, equal weighting in scaled coordinates is not the same as equal weighting of physical action errors. Balancing changes the corruption process and optimization geometry; it does not guarantee lower action-space error for every coefficient.

The guarantees are also marginal over the training distribution. For a conditioning input h_t, the conditional SNR is

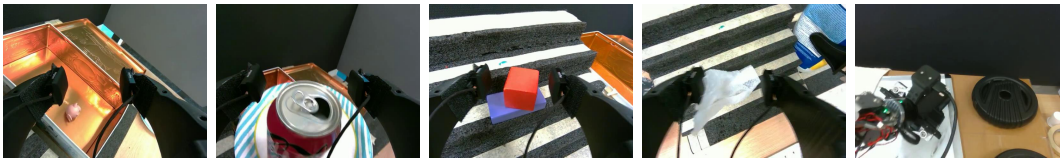
$$\mathrm{SNR}_{s,n,d|h_t} = \frac{\bar{\alpha}_s}{1-\bar{\alpha}_s} \frac{\mathrm{Var}(C_t[n,d] \mid h_t)}{\widehat{\sigma}^2_{n,d}}, \tag{33}$$

which need not be equal across coefficients. Moreover, diagonal variance scaling does not eliminate cross-coefficient correlations and cannot distinguish useful fine corrections from demonstration noise.

These results justify spectral balancing as a way to remove marginal scale disparities from diffusion training. They do not establish equal prediction difficulty, improved closed-loop success, or preservation of coarse-motion accuracy. Those claims require the corresponding empirical ablations and frequency-resolved prediction measurements.

E Complete Real-Robot Results

We report all real-robot results across diffusion policy (DP) and flow matching (FM), action-space and DCT-space prediction, and three force-conditioning strategies: no force, always-on force, and generic force gating. Figure 6 illustrates the five tasks, performed using OpenArm with fixed and wrist cameras. The first four tasks are evaluated by success rate, with Coca-Cola placement requiring placement without audible sound. Whiteboard erasing is evaluated by the percentage of text erased.



Jigglypuff placement | Coca-Cola placement | Block stacking | Wet-wipe extraction | Whiteboard erasing

Figure 6: **Real-robot tasks.** Representative frames, ordered to match Table 4.

Table 4: **Complete real-robot results (%).** The first four task columns report success rates; Whiteboard reports text-erasure percentage. Bold indicates the highest reported value per task, including ties.

Model	Space	Force	Jigglypuff	Coca-Cola	Blocks	Wipes	Whiteboard
DP	Action	None	70.0	80.0	40.0	50.0	77.5
DP	Action	Always-on	60.0	70.0	60.0	40.0	77.5
DP	Action	Generic gate	70.0	70.0	**90.0**	70.0	65.0
DP	DCT	None	60.0	60.0	40.0	60.0	75.5
DP	DCT	Always-on	70.0	50.0	30.0	40.0	66.5
DP	DCT	Generic gate	60.0	60.0	60.0	60.0	72.5
FM	Action	None	**90.0**	80.0	70.0	70.0	68.5
FM	Action	Always-on	**90.0**	80.0	**90.0**	80.0	79.5
FM	Action	Generic gate	**90.0**	70.0	60.0	80.0	86.5
FM	DCT	None	80.0	60.0	70.0	70.0	48.0
FM	DCT	Always-on	**90.0**	70.0	**90.0**	70.0	63.5
FM	DCT	Generic gate	70.0	**90.0**	80.0	70.0	78.0
SPECTRA (ours)			80.0	**90.0**	**90.0**	**100.0**	**89.5**

Table 4 shows that SPECTRA achieves the highest reported performance on wet-wipe extraction and whiteboard erasing, while matching the best baseline on Coca-Cola placement and block stacking. Extraction success reaches 100%, exceeding the strongest baseline by 20 percentage points; text erasure reaches 89.5%, a 3-point improvement. Jigglypuff placement remains an exception, with 80%

success compared with the best baseline's 90%. Across baselines, neither always-on conditioning nor generic gating yields uniform improvements, reinforcing the task-dependent value of force feedback and policy selection.

F Simulation Tasks

We evaluate SPECTRA on six tasks spanning object transport, stacking, and precise insertion. PegInsertionSide, PickCube, PlugCharger, and StackCube use custom MuJoCo ports of ManiSkill tasks; Coffee and Threading use the MimicGen `coffee_d1` and `threading_d0` variants, respectively. Figure 7 illustrates their objectives.

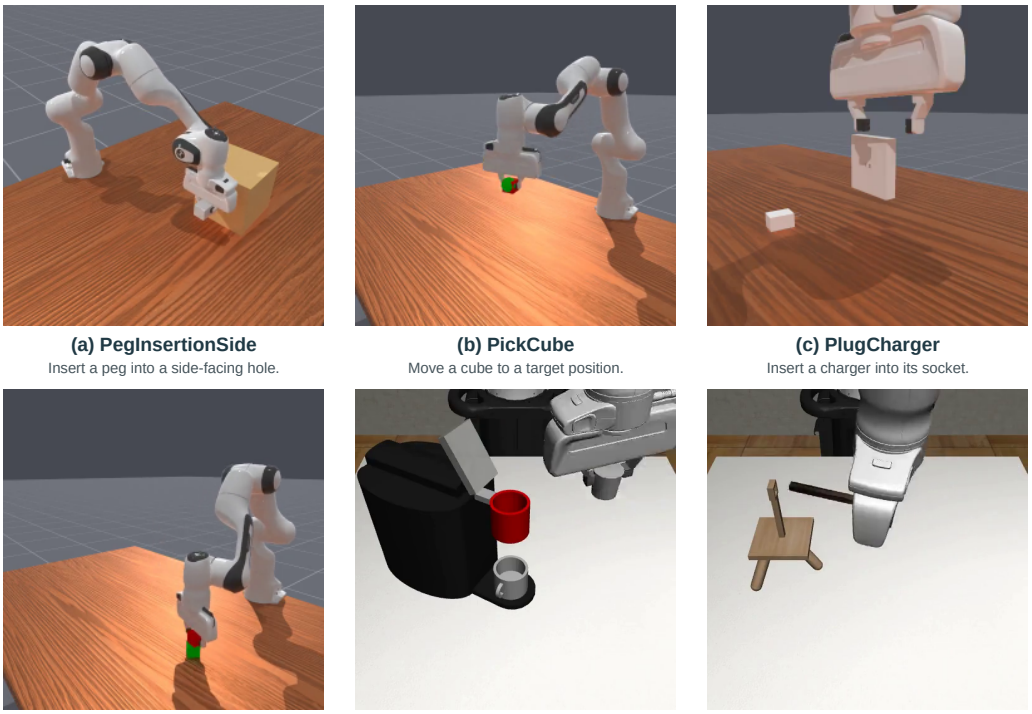


Figure 7: **Simulation task overview.** Reference images from the official ManiSkill documentation and MimicGen website illustrate the task objectives. These are benchmark illustrations, not our policy rollouts. Our ManiSkill experiments use custom MuJoCo ports; the MimicGen images illustrate the task families rather than specific reset distributions.

ManiSkill task ports. **PegInsertionSide** requires grasping a peg, aligning it with a side-facing hole, and inserting it. **PickCube** requires grasping a cube and moving it to a target position, providing a comparison centered on object transport. **PlugCharger** requires aligning a charger's prongs with a socket and inserting them. **StackCube** requires placing one cube on another and releasing it without disrupting the stack. Together, these tasks contrast broad positioning with alignment-sensitive insertion and controlled release. Because our environments are MuJoCo ports, their results should not be treated as directly interchangeable with those from the original ManiSkill simulator.

MimicGen tasks. **Coffee** requires picking up a coffee pod, inserting it into the machine's holder, and closing the lid. **Threading** requires grasping a needle and inserting it through a ring mounted on a stand. Both combine transport with precise alignment, providing settings in which interaction feedback may complement visual observations. We use the D1 reset distribution for Coffee and D0 for Threading.