RELATIVE FLOW MATCHING FOR TIME-CRITICAL CONTROL IN HARDWARE-LIMITED FLIGHT

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative policies are a natural fit for multimodal robot control, but generative sampling can be too expensive for time-critical control on hardware-limited platforms. We introduce *Relative Flow Matching* (RFM), a reparameterization of flow matching that moves the fixed distribution source towards an action-dependent anchor. This is particularly useful in robotics, where information on prior actions can substantially reduce the transport required between source and consecutive action. By defining the flow in relative coordinates, RFM retains a standard Gaussian source without loss of generality, allowing existing flow-matching objectives and fast samplers to be applied directly. We evaluate RFM on collision-free flight of a morphing-wing drone, where RFM reaches $95.2 \pm 1.8\%$ success in simulation while requiring 4.7 ms per inference on a Raspberry Pi Zero 2 W, and transfers to the physical aircraft with 14/15 successful flights. We show that RFM outperforms conventional Flow Matching and other action conditioned methods in terms of performance and speed. Across standard Robomimic and D3IL simulated manipulation benchmarks, RFM improves one-step generation in four of five settings and remains competitive with iterative flow matching at substantially lower inference cost. This work, establishes a continuous connection between standard fixed-source flow matching, stochastic action anchoring, and the deterministic behaviour-cloning boundary. Our results show that RFM's affine reparameterization materially improves very-low-NFE generative control because it changes the conditional source/transport geometry.

1 INTRODUCTION

In robotics, overactuation, redundancy in control, and multiple task solutions often allow for multiple valid action sequences (i.e., multimodal optimal control). For example, a multimodal action space arises in obstacle avoidance scenarios, where multiple action sequences can allow a robot to navigate around an obstacle under similar optimization cost Ravichandar et al. (2020). Diffusion policies are able to imitate such multimodal action distributions and avoid detrimental mode-averaging artifacts of standard behavior cloning methods (Ho et al., 2020; Chi et al., 2023; Lipman et al., 2023). However, the generative process in diffusion policies can be too expensive for time-critical control of robotic platforms with limited computational resources.

Recent methods (Song et al., 2021; 2023; Prasad et al., 2024; Geng et al., 2025) reduce diffusion-time cost with faster solvers, distillation, or models trained directly for one-step generation from a fixed source, i.e., isotropic Gaussian noise ($\mathcal{N}(\mathbf{0}, \boldsymbol{I})$). MeanFlow, for example, predicts the average velocity over a finite interval and can generate a sample with one network evaluation (Geng et al., 2025). In a robotic context, fixed-source action generation can be wasteful, as actions are generated in sequence and hence temporally correlated. Furthermore, fixed-source action generation increases the chances of mode switching between consecutive actions, leading to excessive noise in the action sequence that can destabilize a real robotic system. Previous-action anchoring such as Action-to-Action Flow Matching (Jia et al., 2026) and Streaming Flow Policy (Jiang et al., 2025) instead tie generation to recent actions, leveraging information between subsequent actions. This reduces the action-space transport between consecutive samples (i.e., the diffusion sample trajectory), by removing the assumption of a fixed-source Gaussian and shrinking its variance towards zero which can reduce the ability to express multimodal action distributions (Jiang et al., 2025; Jia

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.

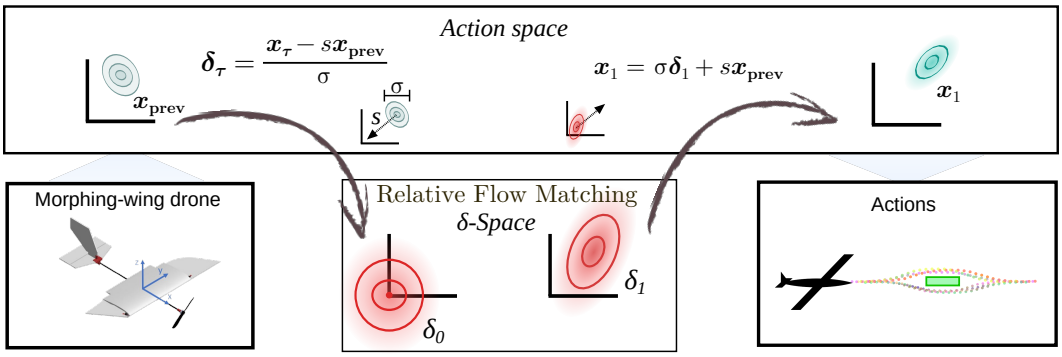


Figure 1: Relative Flow Matching on a morphing wing drone. RFM redefines flow matching relative to a previous generated action ($\boldsymbol{\delta}$-space), instead of absolute action space. After conventional flow matching from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ in residual coordinates, the generated action is transformed back to action space.

et al., 2026). As a consequence, action anchoring removes generality with standard diffusion which makes integrating fast methods not straightforward.

In this work, we propose Relative Flow Matching (RFM) to leverage previously generated actions to anchor the source without loss of generality. Instead of action anchoring in absolute action space, RFM generates actions in residual coordinates ($\boldsymbol{\delta}$-space, Fig. 1), around a contraction ($s\boldsymbol{x}_{\text{prev}}$) of the previous action chunk with a normalised residual noise $\sigma\boldsymbol{\epsilon}$. This standardises the flow around this anchor back into a fixed-source Gaussian ($\mathcal{N}(\mathbf{0}, \boldsymbol{I})$), resulting in a well-defined standard flow-matching formulation. The anchor strength s sets how much of the previous action is retained and the residual scale σ how much stochastic variation remains.

We validate RFM on a miniature morphing-wing drone with tight payload constraints, which require computationally efficient controllers running on a Raspberry Pi Zero 2 W microcontroller that runs at 50 Hz. The morphing-wing drone is overactuated (i.e., multiple control actions result in the same manoeuvres) and is required to fly forward while avoiding an obstacle by flying above or below it, more details in the experiment section. Combined with MeanFlow, RFM achieves expert-level success rates with a single network evaluation taking 4.7 ms on the onboard microcontroller, thus outperforming state-of-the-art fixed-source one-step models (Prasad et al., 2024; Geng et al., 2025) and action-conditioned (Jiang et al., 2025; Jia et al., 2026) methods in both speed and performance. Furthermore, RFM shows at least equal performance at faster inference times compared to other state-of-the-art methods on Robomimic (Mandlekar et al., 2021) and D3IL (Jia et al., 2024) manipulator tasks using state- and vision-based control. We analyse the effects of our parameterisation on the diffusion transport distance, varying the anchor strength, noise scale, and testing previous action quality. This paper contributes:

- **Relative Flow Matching.** We present Flow Matching in residual coordinates around a previous-action anchor that retains a standard Gaussian source, and show its competitiveness in a time-critical task during hardware-limited drone flight. The results show that this affine reparameterization can materially improve very-low-NFE generative control because it changes the conditional source/transport geometry.
- **A generalised source parameterisation.** We show that, by construction, our parameterisation (s, σ) formalises a continuous interpolation between fixed-source flow matching, stochastic action anchoring, and the deterministic one-step behaviour-cloning boundary, deriving the roles of anchor strength and source stochasticity explicitly.
- **Extended application and analysis.** In addition to real drone flight, we further validate RFM on five manipulator tasks using Robomimic and D3IL using different sensor modalities, and analysis of anchoring on transport, mode consistency and decision making; followed by a fullsweep on anchor-strength and residual-scale.

2 Related Work

Generative policies in robotics. Diffusion and flow-based policies have been used for manipulation, motion generation, planning, and general robot control (Chi et al., 2023; Janner et al., 2022; Carvalho et al., 2023; Black et al., 2024). In aerial robotics, diffusion has been used to model uncertain quadrotor dynamics before applying a conventional controller (Das et al., 2025), while other generative methods operate at the level of longer-horizon trajectories or high-level Model Predictive Controller (Janner et al., 2022; Carvalho et al., 2023; Marquez Julbe et al., 2025). In these cases the generative model is not itself part of a fast lower-level actuator-control loop; our flight task runs the learned policy directly at 50 Hz to replace the MPC.

Fast sampling. Fast generative methods reduce inference cost with deterministic solvers, consistency distillation, or direct prediction of finite-time transport (Song et al., 2021; 2023; Prasad et al., 2024; Geng et al., 2025). MeanFlow learns the average velocity over an interval and supports one-step generation (Geng et al., 2025). These methods shorten the sampler but leave the source-to-target transport itself unchanged. RFM changes where the transport starts, and does so in residual coordinates so that the Gaussian formulation these samplers rely on is preserved.

Action-anchoring methods. Autoregressive and action-conditioned policies use recent actions to exploit temporal dependence (Zhao et al., 2023; Jia et al., 2026; Jiang et al., 2025; Zhou et al., 2024). Conditioning the source has shown to benefit generative models more broadly, e.g., Stochastic interpolants and rectified flow (Albergo et al., 2023; Liu et al., 2023); minibatch optimal transport couplings (Tong et al., 2024; Pooladian et al., 2023); and PriorGrad (Lee et al., 2022). Most closely related to our work, Action-to-Action Flow Matching and Streaming Flow Policy condition generation directly on the previous action by using it to centre the initial distribution. However, as a result, they lose the fixed gaussian assumption and possibly their multimodal expressivity of their actions (Jia et al., 2026; Jiang et al., 2025). Alternatively, a diffusion bridge, which conditions the source at a known point to generalise generation to arbitrary endpoint distributions, could be formalised on previous actions but would require its own training objective and its own few-step sampler, which is precisely the cost that RFM avoids. (Zhou et al., 2024). RFM instead, writes this dependence as an affine source transformation with anchor strength s and residual scale σ, which ranges from a fixed Gaussian source to an almost deterministic anchor allowing us to keep standard flow matching assumptions.

3 Methods

We learn a conditional generative policy $p_\theta(\boldsymbol{x}_1 \mid \boldsymbol{c})$ over action chunks $\boldsymbol{x}_1 = [\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+H-1}]$, with future horizon length (H) while conditioned on current observations $\boldsymbol{c} = \{\boldsymbol{o}_t\}$. Flow matching (Lipman et al., 2023) learns a velocity field that transports a isotropic source distribution to the action distribution. Given a target sample $\boldsymbol{x}_1 \sim q(\boldsymbol{x} \mid \boldsymbol{c})$ and a source sample $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, the linear probability path

$$\boldsymbol{x}_\tau = (1-\tau)\,\boldsymbol{x}_0 + \tau\,\boldsymbol{x}_1, \qquad \tau \sim \mathcal{U}(0,1), \tag{1}$$

has target velocity $\boldsymbol{u}_\tau = \boldsymbol{x}_1 - \boldsymbol{x}_0$, and a network $\boldsymbol{v}_\theta(\boldsymbol{x}_\tau, \tau, \boldsymbol{c})$ is trained with the regression loss

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{\tau, \boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{c}}\Big[\big\|\boldsymbol{v}_\theta(\boldsymbol{x}_\tau, \tau, \boldsymbol{c}) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\big\|_2^2\Big]. \tag{2}$$

At inference, an action sample is generated by integrating the learned field from $\tau{=}0$ to $\tau{=}1$.

3.1 Relative Flow Matching

In standard conditional flow matching actions are sampled from a fixed-source $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. RFM replaces this fixed source with the following

$$\boldsymbol{x}_0 = \boldsymbol{\mu} + \sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \qquad \boldsymbol{\mu} = s\boldsymbol{x}_{\mathrm{prev}}, \qquad s \in [0,1],\ \sigma > 0, \tag{3}$$

where $\boldsymbol{x}_{\mathrm{prev}}$ is the previous action chunk and is included in the conditioning $\boldsymbol{c}$. The anchor strength s determines how much of the previous action is retained in the source, and the residual scale σ determines the stochastic variation around it.

Following Eq. 1, the linear path will be defined as $\boldsymbol{x}_\tau = (1-\tau)(\boldsymbol{\mu} + \sigma\boldsymbol{\epsilon}) + \tau\boldsymbol{x}_1$, which can be described in residual coordinates

$$\boldsymbol{\delta}_\tau = \frac{\boldsymbol{x}_\tau - \boldsymbol{\mu}}{\sigma} = (1-\tau)\boldsymbol{\delta}_0 + \tau\boldsymbol{\delta}_1, \qquad \boldsymbol{\delta}_0 = \boldsymbol{\epsilon}, \qquad \boldsymbol{\delta}_1 = \frac{\boldsymbol{x}_1 - \boldsymbol{\mu}}{\sigma}. \tag{4}$$

The source is now $\boldsymbol{\delta}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ for every anchor. We train the residual velocity $\boldsymbol{u}^{\boldsymbol{\delta}}_\tau = \boldsymbol{\delta}_1 - \boldsymbol{\delta}_0$ using

$$\mathcal{L}_{\mathrm{RFM}} = \mathbb{E}_{\tau,\boldsymbol{\epsilon},\boldsymbol{x}_1,\boldsymbol{c}}\left[\left\|\boldsymbol{v}_\theta(\boldsymbol{\delta}_\tau, \tau, \boldsymbol{c}) - \boldsymbol{u}^{\boldsymbol{\delta}}_\tau\right\|_2^2\right], \tag{5}$$

and recover the action after sampling with

$$\boldsymbol{x} = \sigma\boldsymbol{\delta} + \boldsymbol{\mu}. \tag{6}$$

For $\sigma > 0$ this is an invertible affine transformation. Appendix A shows that the source, path, objective, and sampler correspond exactly to standard fixed-Gaussian flow matching in $\boldsymbol{\delta}$-space (Proposition 1). When consecutive actions are close, RFM only predicts the remaining correction, thus reducing the total transport between consecutive actions.

The RFM source parameters (s, σ) play different roles. Increasing the anchor strength s shortens the immediate transport and for $s = 1$, the initial distribution is centred on the previous action. The residual scale σ determines the stochastic variation around the anchor and can tune the amount of mode consistency. Interestingly, Eq. 3 defines a continuous interpolation of state-of-art method at its limits. At s=0, σ=1 the construction is conventional flow matching, and as $s \to 1$, $\sigma \to 0$ it approaches deterministic action anchoring (Jia et al., 2026; Jiang et al., 2025). Furthermore, at σ=0 the source collapses to a single point, resulting in a deterministic sampler whose expected behaviour $\mathbb{E}[\boldsymbol{x}_1 \mid \boldsymbol{c}]$ is defined by the L2 Behaviour-Cloning estimator (Proposition 2).

3.2 One-step RFM with MeanFlow

RFM is able to integrate action anchoring with well-defined fixed source methods. This allows us to define a one-step previous action conditioned policy using MeanFlow (Geng et al., 2025), denoted RFM-MF. MeanFlow predicts the average velocity over a finite interval instead of integrating many local velocities over multiple steps. This is done by redefining the flow matching loss as an average flow field over τ. Because RFM starts from the same $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ source in $\boldsymbol{\delta}$-space, we apply the MeanFlow objective without modification (Corollary 1). RFM-MF thus still requires only a single network evaluation, but predicts a correction applied to the previous action rather than starting from Gaussian noise; the action is recovered with Eq. 6.

4 Experiments

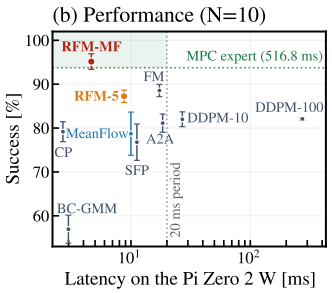
4.1 Experimental setup

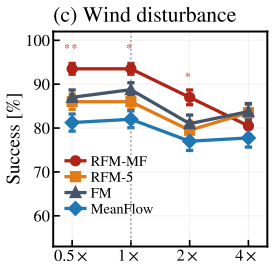
We validate RFM in a time-critical flight control experiment where a morphing-wing drone must fly towards a distant goal and avoid an oncoming obstacle on its longitudinal trajectory. This scenario offers two action choices: the drone must choose to fly either above or under the obstacle. The chosen 130 g drone is furthermore payload limited, demonstrating the necessity to constrain the available compute resources to a minimal Raspberry Pi Zero 2 W. The aircraft is launched towards a goal with a horizontal beam on the direct path and must pass above or below it within approximately one second. The observation is 21-dimensional (position, attitude, linear and angular velocity, measured actuator positions) and the action contains five actuator commands (normalised in $[-1, 1]$: left wing sweep, right wing sweep, throttle, elevator, and rudder). Policies predict four action chunks ($H = 4$) and only execute the first action for closed-loop robustness at 50 Hz, thus requiring action generation to complete within 20 ms. We provide the same 41-dimensional input for all models: 21 drone state; with the previous generated action chunk ($4 \times 5 = 20$)

Demonstrations are generated by a nonlinear model-predictive-control expert (NMPC). In addition, we compare BC-GMM, DDPM with 100 and 10 denoising steps (Ho et al., 2020; Chi et al., 2023), ,Consistency Policy (Prasad et al., 2024) distilled from DDPM-100, standard flow matching (FM) (Lipman et al., 2023), MeanFlow (Geng et al., 2025), Action-to-Action Flow Matching (A2A) (Jia et al., 2026), and Streaming Flow Policy (SFP) (Jiang et al., 2025). RFM-5 integrates the

(a) Model cost [M]

Method	NFE	Params
DDPM-100	100	0.39
DDPM-10	10	0.39
Consistency Policy	1	0.39
A2A	6	0.94
Flow Matching	10	0.57
SFP	4	0.56
MeanFlow	1	2.89
BC-GMM	1	1.28
RFM-5	**5**	**0.57**
RFM-MF, small	**1**	**0.96**
RFM-MF, big	**1**	**2.89**





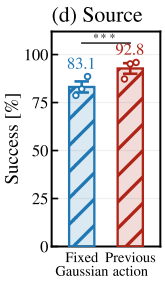


Figure 2: Morphing-wing flight in simulation, error bars denote ±95% confidence interval, * denote significant differences with highest performance. Top row, model comparison in simulation (with matched input): (a) Number of function evaluations and parameter count of trained models. (b) Success rates and inference speed (tested on a Raspberry Pi Zero 2 W). The dotted lines mark the 20 ms and NMPC expert boundaries. (c) Success rate under scaled wind disturbance. (d) Aggregate source comparison (action anchoring outperforms fixed source, $N = 30$ with Bonferroni correction)

RFM field with five Euler steps and RFM-MF uses MeanFlow in residual coordinates. The 2.89M MeanFlow and RFM-MF models share the same trunk, action representation, NFE, and measured latency, so the source is the only difference between them. We optimised our model size and number of function evaluation NFE to maximize performance within our 20 ms budget (Fig. 2a). We evaluate each drone policy on 400 held-out initial conditions with N=10 independently trained policies, nominally and with wind disturbance, and measure inference on the deployment computer; all drone results are means with 95% confidence intervals over seeds (Appendix B gives the data-generation, training, hardware, and evaluation details).

4.2 Main Results

Morphing-wing simulation results. RFM-MF is the only trained policy that reaches expert-level control (in both network sizes), far within the 20 ms control period (Fig. 2b). The deployed 0.96M policy reaches $95.2 \pm 1.8\%$ success with one network evaluation in 4.7 ms, whereas the MPC expert reaches 93.8% in 516.8 ms per solve in our evaluation setup (RFM-MF big reaches the same success rate at twice the inference time). Conventional FM approaches this performance only with ten evaluations ($88.5 \pm 1.4\%$ in 17.3 ms), close to the 20 ms deadline, and the remaining baselines are lower. The full details of each method and corresponding 95% confidence intervals are reported in Appendix C.1.

When equalising flow-matching methods and network architecture, we find that RFM's previous action anchoring provides additional benefits in both speed (RFM-5 reaches $87.2 \pm 1.4\%$ in 8.8 ms, close to ten-step FM at approximately half its latency), and success rate (conventional MeanFlow reaches $78.7 \pm 4.9\%$ while RFM-MF big reaches $95.9 \pm 1.6\%$). The source parameterisation is therefore the controlled difference between these models. Across all methods, previous-action-based policies also outperform the fixed-source group in aggregate (Fig. 2d).

In the single-run disturbance sweep, RFM-MF remains above FM through twice the training wind and only falls slightly below FM and RFM-5 at four times that scale (Fig. 2c, Appendix C.2). We suspect this happens when the required response changes abruptly resulting in the previous action misplacing the anchor source resulting in a worse performance.

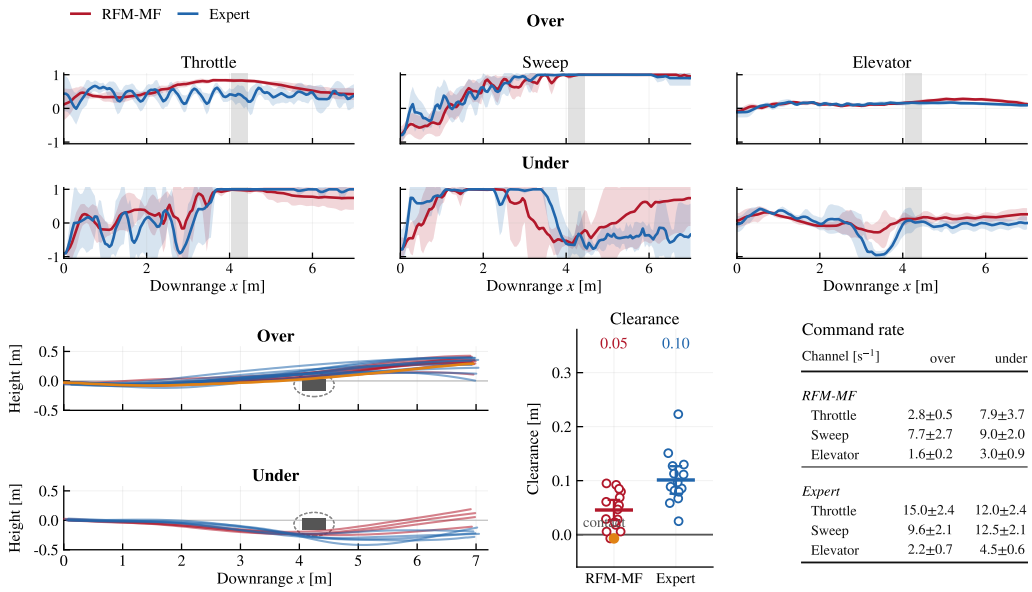
Real hardware validation. We deploy the 0.96M RFM-MF policy on the physical drone in a $10 \times 10 \times 8$ m arena (see Fig. 3) with a virtual obstacle defined in the motion-capture frame. A detailed description of the real hardware setup can be found in Appendix B. It clears the beam on 14 of 15 flights, with one contact at an over attempt, and flies both the over and the under route (10 over, 4 under), while the MPC expert clears all 15 reference flights (Fig. 3). Over/under distribution is expected to differ as the real world experiment have less initial state variation compared to simulation. The trained policy passes the beam considerably closer to the expert, with a mean minimum clearance of 0.05 ± 0.02 m against 0.10 ± 0.03 m.

Flight summary

	RFM-MF	Expert
Trials	15	15
Success [%]	93	100
Route ↑ /-/ ↓	10/1/4	10/0/5
Clearance [m]	0.05±0.02	0.10±0.03
Action MAE [10^{-3}]	32±15	38±18

Flight composite





Command rate

Channel [s^{-1}]	over	under
RFM-MF		
Throttle	2.8±0.5	7.9±3.7
Sweep	7.7±2.7	9.0±2.0
Elevator	1.6±0.2	3.0±0.9
Expert		
Throttle	15.0±2.4	12.0±2.4
Sweep	9.6±2.1	12.5±2.1
Elevator	2.2±0.7	4.5±0.6

Figure 3: Real world flights of both expert (blue) and RFM-MF (red), with error bars denoting 95% confidence intervals where shown. Top row: flight summary and a composite of the trajectories from the arena video. Middle rows: mean throttle, sweep, and elevator commands separated by obstacle route (over: RFM-MF n=11, expert n=10; under: n=4 and 5). Bottom row: corresponding trajectories, minimum obstacle clearance per flight, and command rate per channel (total variation per second).

A second observation from the physical flights is the smoothness of the generated commands with respect to the expert (Fig. 3 command rate). RFM-MF reproduces the route-specific sweep and elevator changes of the expert, but its throttle varies one and a half to five times more slowly over the flight; sweep and elevator variation are also lower on both routes. The learned policy therefore reproduces the expert's route-specific morphing without copying its high-frequency throttle changes. We suspect that generating each action as a correction around the previous one contributes to this smoothing, although the simulation comparison in Appendix C.4 shows that it is not a universal advantage over iterative FM.

4.3 Additional validation and analysis

Manipulator benchmark. We test the efficacy of our method across several manipulator benchmark using different sensor information, using ToolHang and Square from Robomimic (Mandlekar et al., 2021) and Stacking from D3IL (Jia et al., 2024), with state observations on all three tasks and image observations on ToolHang and Square. Each task is trained on the same training seeds for a total of N=30 repetitions per cell. These tasks have slower dynamics and no strict compute budget or control deadline. Model and evaluation details are given in Appendix D.

The results on the additional benchmarks show that RFM-MF improves over fixed-source MeanFlow in four of the five settings with non-overlapping 95% confidence intervals, while Stacking shows no clear difference (Table 1). The comparison with iterative FM is closer as RFM-MF is higher on

Method	State observations						Image observations			
	ToolHang		Square		Stacking		Square		ToolHang	
DDPM-100	66.2 ±4.1	13.56	40.2 ±3.4	13.64	73.0 ±6.0	26.51	48.3 ±4.3	31.27	26.7 ±4.5	31.05
DDPM-10	66.5 ±3.6	1.41	39.3 ±3.5	1.41	**79.1** ±2.7	2.74	50.7 ±4.3	3.20	30.2 ±2.8	3.16
FM-10	63.0 ±2.7	1.25	**79.0** ±1.6	1.26	70.1 ±9.0	2.41	**72.8** ±2.0	3.04	30.3 ±4.1	2.99
BC-GMM	*29.3 ±0.9	0.21	52.4 ±2.7	0.21	*24.8 ±5.5	0.20	46.8 ±2.5	1.82	*20.3 ±3.9	1.83
MF	53.6 ±6.1	0.16	63.8 ±2.7	0.32	68.5 ±8.4	0.30	45.6 ±2.7	0.36	22.1 ±3.5	0.35
RFM-5 (ours)	31.1 ±1.1	0.48	48.1 ±2.7	0.46	**78.1** ±2.8	0.56	61.1 ±0.1	0.66	12.6 ±2.5	0.66
RFM-MF (ours)	**74.4** ±2.1	0.16	**77.8** ±2.6	0.32	67.6 ±3.5	0.30	**73.5** ±2.5	0.37	**44.8** ±3.7	0.36

Table 1: Success rate (black mean values ±95CI, N=30 repetitions) and inference time (gray in ms per executed action on an RTX 4070) on five manipulator tasks: ToolHang and Square from state and from images, and Stacking from state. Cells marked * are the published numbers of the benchmark paper, used where our reproduction of that baseline stayed below 5% success.

both ToolHang settings, while Square (both sensor modalities) and Stacking are not statistically significantly different. Nevertheless, RFM-MF inference time remains lower at 0.16–0.56 ms per executed action, compared with 1.25–3.04 ms for FM and 13.6–31.3 ms for DDPM-100 (tested on an RTX 4070).

Transport reduction through anchoring. Moving away from fixed-source flow matching allows us to position the prior closer to the target action chunk. Thus defining the anchor on the previously generated actions shortens the transport the network has to predict, and this is the quantity RFM changes before any model is trained. On the drone, the fixed Gaussian source is on average 4.92 normalised action units from the target, while the anchored source at s=0.5 is 1.16 away. Changing between fixed Gaussian source and RFM reduces expected transport by 5.4× on Square and 6.7× on ToolHang, showing that the transport reduction is not specific to the drone (Appendix C.5). Besides reducing total transport, σ rescales the residual coordinates as well, and the sweep below shows that the source scale moves closed-loop success on its own, without any anchor. We therefore read the transport numbers as characterising the total source geometry defined by RFM, which influences the anchor and the residual scale, not only transport.

Mode consistency. Fig. 2b shows a significant underperformance of behavioural-cloning compared to sampling based methods. When we highlight some more detailed results of the drone trajectory, presented in Appendix C.1, we can see a high hit-rate (32%) for BC indicating mode-averaging behaviour due to the policies deterministic nature. Nevertheless, mode-switching between actions (switch between over/under strategy during flight) can also cause a high hit-rate, shown by the anchored one-step policies stay at the expert's level (0–3% against 1%), while every fixed-source sampler sits at 8–16%. This can explain the significant outperformance of action-anchoring (Fig. 2d) over fixed-source diffusion methods that are required regenerate the full transport in action space and are therefore more sensitive to temporal mode inconsistency.

Route information. The lower through-beam rate of RFM does not by itself establish that fixed-source policies switch between modes. We therefore test whether the previous action supplies route information that is absent from the current observation. A five-fold logistic regression predicts the over/under route from the state alone with 0.94 accuracy by control step 5 and 0.99 by step 10; adding the previous action improves only step 5, by 0.05 (Appendix C.5). The anchors therefore mainly contributes a shorter, better-scaled source rather than route information the observation lacks.

Parameter sweep. RFM defines flow matching in $\boldsymbol{\delta}$-space, an action-anchored transformation based on two parameters (s, σ). In Fig. 4 we show the effect of varying these parameters. Fig. 4a shows that increasing s continuously consistently reduces the transport, from 1.75 at s=0 to 0.90 at s=1. Interestingly, this monotonic reduction in transport does not correspond with consistent performance increase, as RFM drone success rate peaks at s=0.5 and falls at $s \geq 0.8$ (Fig. 4b). We suspect that at full anchoring, errors in the generated action are transferred directly into the next source, which may explain the drop in closed-loop performance. The best anchor is therefore a balance between information propagation $s > 0$ and reducing errors $s < 1$.

When keeping s=0.5 fixed, success drops when σ falls well below its calibrated value σ^*, defined as the residual RMS measured on the training set at that anchor strength. At well-calibrated anchoring ($s = 0.5$) noise variation does not seem to vary performance, while, interestingly, at $s = 1$ lower

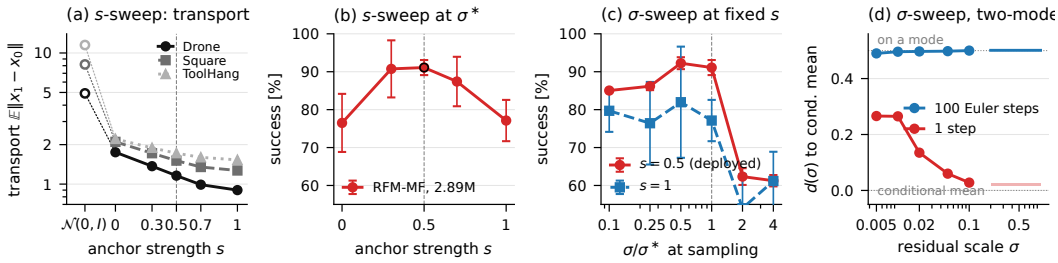


Figure 4: Parameter sweep. (a) Mean source-to-target transport $\mathbb{E}\|\boldsymbol{x}_1 - \boldsymbol{x}_0\|$ per training window against anchor strength s, in normalised action units, for the fixed $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ source (hollow) and the anchored source (filled). (b) Closed-loop drone success of the 2.89M RFM-MF policy against s, each at its calibrated residual scale σ^*. (c) Success of the same policy against σ/σ^* at s=0.5 and s=1, rescaled at sampling over the deployed checkpoints. (d) Modal collapse, when noise levels decreases mean distance to the conditional mean collapses $d(\sigma)$, for the same field sampled with 100 Euler steps and with one; the band marks the range of the $\sigma \geq 0.2$ cells.

noise levels seem to help performance $0.1\sigma^*$ (Fig. 4c). This supports the idea that at $s = 1$ error propagate as noise through the anchor, and add to the importance of anchoring strength as a transport reduction mechanism. Finally, we see an unbalanced $\sigma \neq 1$ is more damaging at higher values, at fixed s. At a wider source this can be explained by the increased transport, while on smaller scales a loss of stochasticity reduces expresiveness (i.e., loss of multimodality), which is the part of the effect the anchor does not explain. Table 5 provides the full sweep values behind Fig. 4b,c.

As laid out in Proposition 2, the deterministic boundary occurs at $\sigma = 0$. At this boundary, a one-step squared-error prediction returns the conditional mean, matching the L2 behaviour-cloning estimator. The controlled two-mode experiment probes the corresponding finite-σ regime (Fig. 4d and Appendix C.6). The field sampled with 100 Euler steps stays on a mode across the whole sweep, while the one-step sample approaches the conditional mean for $\sigma \geq 0.1$. In this controlled setting, source noise alone does not recover the two modes under one-step sampling.

5 Discussion

In this paper, we introduce RFM to reduce the amount of transport required for generative control when only one or a few network evaluations are available. For $\sigma > 0$, RFM is an invertible reparameterisation of standard flow matching in residual coordinates, and therefore does not increase the expressiveness of the underlying model. Nevertheless, changing the source has a large effect when inference is limited. In our matched one-step experiment, conventional MeanFlow reaches $68.5 \pm 3.4\%$ success while RFM-MF reaches $95.9 \pm 1.6\%$ with the same network architecture, inference time, and number of evaluations. This indicates that the source becomes increasingly important when the learned flow has to be approximated with very few inference steps.

Our analysis shows that the main effect of RFM is the reduction in transport between source and target. At the deployed anchor strength, the average transport on the drone decreases from 4.92 to 1.16 normalised action units. A possible explanation could be that the previous action contains information on whether the drone should pass above or below the obstacle. We test this directly and find that the current state already predicts the selected route with 0.94 accuracy at control step 5 and 0.99 at step 10, while adding the previous action only improves prediction at step 5. The previous action therefore mainly provides a better starting point for generating the next action, rather than information that is missing from the observation.

Increasing the anchor strength reduces source-to-target transport, but does not monotonically improve closed-loop performance. At the default residual scale, $s = 0.5$ substantially outperforms both the fixed source ($s = 0, \sigma = 1$) and full anchoring ($s = 1$). The sampling-only σ sweep further shows that this effect depends on source stochasticity: moderate noise is beneficial, whereas excessive noise strongly degrades performance. Full anchoring is particularly sensitive to this choice,

suggesting an interaction between inherited action error and source noise rather than transport length alone.

The residual scale σ controls a different property of the source. At $\sigma = 0$, the source becomes deterministic and the one-step squared-error solution reduces to the conditional mean. For finite σ, this collapse is not inevitable. In the controlled two-mode experiment, the same learned field remains on either mode when integrated with 100 Euler steps, while one-step inference approaches the conditional mean. This shows that source stochasticity alone is not sufficient to maintain multimodality when the complete flow is compressed into one network evaluation. RFM therefore separates two effects that are otherwise coupled in previous-action methods: 1) how much information is retained from the previous action through s, and 2) how much stochastic variation remains through σ.

The manipulation experiments show that the one-step advantage of RFM also extends beyond the drone. RFM-MF improves over fixed-source MeanFlow in four of five settings, while the comparison with iterative flow matching is smaller. Due to the extend of experiments, we adapted published training settings task-specific architectures, action representations, and horizons, to compare with our own optimised implementation. Interestingly, temporal correlation does not predict the size of the improvement. Square and ToolHang have almost identical consecutive-action correlations, but show different gains from RFM. This suggests that the usefulness of the anchor also depends on the action representation, horizon, model, and closed-loop dynamics of the task.

The physical flights furthermore show that RFM-MF generates smoother throttle commands than the NMPC expert while maintaining route-specific sweep and elevator behaviour. We did not explicitly optimise for smoothness in the loss, and in simulation RFM-MF does not appear smoother than other iterative methods. Interestingly, this suggests that relative action mapping can implicitly regulate high-frequency command variation through its re-centering mechanism and mode consistency. As a possible downside, re-activity to quickly change action output might have limited obstacle clearance with respect to the expert (0.05 versus $0.10\,\mathrm{m}$). These results indicate a trade-off between maintaining temporal consistency and reacting strongly to changes in the state.

For future work, we would like to extend the application of RFM on more complex flight task. For this we should improve our methods and increase our physical flight time (here we validated only 15 trajectories). For example, a likely improvement could be gained by optimising the anchor strength s and residual scale σ at test-time, e.g., as Table 5 shows ad-hoc tuning can increase performance. In the same light, policy fine-tuning after training our diffusion method could be interesting as well. Finally, our experiments use MeanFlow as the main one-step method. Since RFM recovers a standard Gaussian source in residual coordinates, other fixed-source flow-matching and distillation methods can be applied in the same way.

6 Conclusion

We introduced Relative Flow Matching (RFM), which moves the flow source towards the previous action and learns the remaining transport in standard Gaussian residual coordinates. When used to imitate an expert controller for a morphing-wing drone in a multimodal obstacle avoidance task, our method reduces a long fixed-source transport to a local correction and allows one-step MeanFlow to reach expert-level performance in experiments with a physical drone. The computational load remains within the compute budget of an embedded computer. The same trend on manipulation benchmarks suggests that the flow source is an important part of fast generative control whenever only few network evaluations are possible.

AI use statement

Generative AI tools were used for writing, experimental data management, literature search, and drafting and editing manuscript text. The authors checked the final claims, mathematical statements, reported numbers, and citations against the implementation, underlying experiments, and cited sources, and take responsibility for the final submission.

Ethics statement

This work concerns control policies for small aerial robots, evaluated in simulation, on standard manipulation benchmarks, and on a physical aircraft inside a netted indoor arena. No human subjects or personal data are involved.

Reproducibility statement

Appendix A gives the formal assumptions and proofs. Appendix B specifies the drone task, data generation, model configurations, and evaluation protocol. Appendix C reports the additional drone analyses, and Appendix D documents the manipulation configurations and held-out evaluation sets. Seeded success rates use 95% confidence intervals over seeds unless stated otherwise. Code and data will be made available after acceptance.

References

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Joao Carvalho, An T. Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

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. In *Robotics: Science and Systems*, 2023.

Avirup Das, Rishabh Dev Yadav, Sihao Sun, Mingfei Sun, Samuel Kaski, and Wei Pan. DroneDiffusion: Robust quadrotor dynamics learning with diffusion models. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1604–1610, 2025.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915, 2022.

Jindou Jia, Gen Li, Xiangyu Chen, Tuo An, Yuxuan Hu, Jingliang Li, Xinying Guo, and Jianfei Yang. Action-to-action flow matching. In *Robotics: Science and Systems*, 2026.

Xiaogang Jia, Denis Blessing, Xinkai Jiang, Moritz Reuss, Atalay Donat, Rudolf Lioutikov, and Gerhard Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *International Conference on Learning Representations*, 2024.

Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Siddharth Ancha. Streaming flow policy: Simplifying diffusion/flow-matching policies by treating action trajectories as flow trajectories. In *Conference on Robot Learning*, 2025.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Sang-gil Lee, Heeseung Kim, Chaehun Shin, Xu Tan, Chang Liu, Qi Meng, Tao Qin, Wei Chen, Sungroh Yoon, and Tie-Yan Liu. PriorGrad: Improving conditional denoising diffusion models with data-dependent adaptive prior. In *International Conference on Learning Representations*, 2022.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, 2021.

Pau Marquez Julbe, Julian Nubert, Henrik Hose, Sebastian Trimpe, and Katherine J. Kuchenbecker. Diffusion-based approximate MPC: Fast and consistent imitation of multi-modal action distributions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *International Conference on Machine Learning*, 2023.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems*, 2024.

Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:297–330, 2020. doi: 10.1146/annurev-control-100819-063206.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. In *Transactions on Machine Learning Research*, 2024.

Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14:147–183, 2022.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

Linqi Zhou, Aaron Lou, Samar Khanna, and Stefano Ermon. Denoising diffusion bridge models. In *International Conference on Learning Representations*, 2024.

A PROOFS

We give the two formal results used in the main text. Throughout, $\boldsymbol{\mu} = s\boldsymbol{x}_{\mathrm{prev}}$ is a deterministic function of the conditioning $\boldsymbol{c}$ with $\boldsymbol{x}_{\mathrm{prev}} \in \boldsymbol{c}$, residual coordinates require $\sigma > 0$, and the source noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is independent of $(\boldsymbol{x}_1, \boldsymbol{c})$.

A.1 AFFINE EQUIVALENCE

RFM is standard conditional flow matching after the affine change of coordinates of Section 3.1. We show this for the source, the path, the loss, and the sampler in turn.

Proposition 1 (Affine equivalence)**.** *Let $\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{c})$ be a deterministic function of the conditioning $\boldsymbol{c}$, and let $\sigma > 0$. Define $T_{\boldsymbol{c}}(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu})/\sigma$. Then applying $T_{\boldsymbol{c}}$ to the anchored source $\boldsymbol{x}_0 = \boldsymbol{\mu} + \sigma\boldsymbol{\epsilon}$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and target $\boldsymbol{x}_1 \sim q(\cdot \mid \boldsymbol{c})$ yields standard conditional flow matching between $\boldsymbol{\delta}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\boldsymbol{\delta}_1 = T_{\boldsymbol{c}}(\boldsymbol{x}_1)$. The transformed probability path and RFM regression objective are exactly the standard fixed-Gaussian flow-matching path and objective in $\boldsymbol{\delta}$-coordinates, and the corresponding action-space flow is recovered by the inverse affine map.*

We use the change of coordinates

$$T_{\boldsymbol{c}}(\boldsymbol{x}) = \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\sigma}, \tag{7}$$

with inverse $g(\boldsymbol{\delta}) = \sigma\boldsymbol{\delta} + \boldsymbol{\mu}$. The map is invertible for $\sigma > 0$, and $\boldsymbol{\mu}$ is known from the conditioning.

Source. $T_{\boldsymbol{c}}(\boldsymbol{x}_0) = (\boldsymbol{\mu} + \sigma\boldsymbol{\epsilon} - \boldsymbol{\mu})/\sigma = \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, independently of $\boldsymbol{c}$. The $\boldsymbol{\delta}$-space source is therefore $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ for every conditioning and anchor.

Path. Because $T_{\boldsymbol{c}}$ is affine and the interpolation is linear,

$$T_{\boldsymbol{c}}(\boldsymbol{x}_\tau) = \frac{(1-\tau)\boldsymbol{x}_0 + \tau\boldsymbol{x}_1 - \boldsymbol{\mu}}{\sigma} = (1-\tau)\,\frac{\boldsymbol{x}_0 - \boldsymbol{\mu}}{\sigma} + \tau\,\frac{\boldsymbol{x}_1 - \boldsymbol{\mu}}{\sigma} = (1-\tau)\,\boldsymbol{\delta}_0 + \tau\,\boldsymbol{\delta}_1 = \boldsymbol{\delta}_\tau, \tag{8}$$

using $(1-\tau)\boldsymbol{\mu} + \tau\boldsymbol{\mu} = \boldsymbol{\mu}$. The action-space path therefore maps exactly to the standard linear flow-matching path from $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to $\boldsymbol{\delta}_1 = (\boldsymbol{x}_1 - \boldsymbol{\mu})/\sigma$.

Velocities and losses. Differentiating, $\boldsymbol{u}_\tau = \boldsymbol{x}_1 - \boldsymbol{x}_0 = \sigma(\boldsymbol{\delta}_1 - \boldsymbol{\delta}_0) = \sigma\,\boldsymbol{u}^{\boldsymbol{\delta}}_\tau$. Because $\boldsymbol{x}_\tau$ and $\boldsymbol{\delta}_\tau$ determine each other given $\boldsymbol{c}$, conditioning on one is the same as conditioning on the other, so the conditional expectations that define the optimal velocity fields correspond:

$$\mathbb{E}[\boldsymbol{x}_1 - \boldsymbol{x}_0 \mid \boldsymbol{x}_\tau, \boldsymbol{c}] = \sigma\,\mathbb{E}[\boldsymbol{\delta}_1 - \boldsymbol{\delta}_0 \mid \boldsymbol{\delta}_\tau, \boldsymbol{c}]\,. \tag{9}$$

The minimizer of the standard flow-matching regression in $\boldsymbol{\delta}$-space is $\boldsymbol{v}^*(\boldsymbol{\delta}, \tau, \boldsymbol{c}) = \mathbb{E}[\boldsymbol{\delta}_1 - \boldsymbol{\delta}_0 \mid \boldsymbol{\delta}_\tau = \boldsymbol{\delta}, \boldsymbol{c}]$. Equation 9 then gives the corresponding action-space field $\boldsymbol{v}^*_{\boldsymbol{x}}(\boldsymbol{x}, \tau, \boldsymbol{c}) = \sigma\,\boldsymbol{v}^*(T_{\boldsymbol{c}}(\boldsymbol{x}), \tau, \boldsymbol{c})$. Note that the condition $\boldsymbol{x}_{\text{prev}} \in \boldsymbol{c}$ is needed here because $\boldsymbol{\delta}_1$ depends on the anchor to break state symmetry for relative flows.

Flows. Let $\boldsymbol{\delta}(\tau)$ solve $\dot{\boldsymbol{\delta}} = \boldsymbol{v}^*(\boldsymbol{\delta}, \tau, \boldsymbol{c})$ with $\boldsymbol{\delta}(0) = \boldsymbol{\epsilon}$, assuming the field is locally Lipschitz so the solution is unique. Then $\boldsymbol{x}(\tau) = \sigma\boldsymbol{\delta}(\tau) + \boldsymbol{\mu}$ satisfies $\dot{\boldsymbol{x}} = \sigma\dot{\boldsymbol{\delta}} = \boldsymbol{v}^*_{\boldsymbol{x}}(\boldsymbol{x}, \tau, \boldsymbol{c})$ with $\boldsymbol{x}(0) = \boldsymbol{\mu} + \sigma\boldsymbol{\epsilon}$. The two flows therefore correspond at every time through the affine transformation, and sampling in $\boldsymbol{\delta}$-space and transforming the result back gives the same action-space samples. □

A.2 TRANSFER OF FIXED-SOURCE RECIPES

Because RFM is standard flow matching in residual coordinates, fixed-source training and sampling recipes transfer directly.

Corollary 1. *In $\boldsymbol{\delta}$-coordinates, (a) MeanFlow, consistency training and distillation recipes derived for a $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ source apply without modification, and (b) the noise component of the network's inputs and regression targets has unit scale for every (s, σ).*

(a) Proposition 1 leaves the $\boldsymbol{\delta}$-space problem as standard conditional flow matching with a Gaussian source and linear path, so fixed-source training and sampling procedures apply in these coordinates. For MeanFlow, the average velocity $\bar{\boldsymbol{v}}(\boldsymbol{\delta}, r, \tau) = \frac{1}{\tau - r}\int_r^\tau \boldsymbol{v}(\boldsymbol{\delta}_s, s)\,ds$ and its identity follow from the $\boldsymbol{\delta}$-space ODE (Geng et al., 2025). The result holds for any anchor that is a fixed function of the conditioning.

(b) At $\tau = 0$, the network input is $\boldsymbol{\delta}_0 = \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ for every (s, σ), and the stochastic part of $\boldsymbol{u}^{\boldsymbol{\delta}}_\tau = \boldsymbol{\delta}_1 - \boldsymbol{\epsilon}$ has unit scale. In action space the source is $\mathcal{N}(\boldsymbol{\mu}, \sigma^2\boldsymbol{I})$ and its width shrinks with σ. The target component $\boldsymbol{\delta}_1 = (\boldsymbol{x}_1 - \boldsymbol{\mu})/\sigma$ is still task-dependent and grows as $\sigma \to 0$, which is why we test the finite-σ regime in Appendix C.6. □

A.3 THE DETERMINISTIC-SOURCE BOUNDARY

At $\sigma = 0$, a deterministic one-step source cannot represent more than one output for fixed conditioning, and squared-error training returns the conditional mean.

Proposition 2 (Deterministic-source degeneracy)**.** *Let the conditioning $\boldsymbol{c}$ be fixed and let $\sigma = 0$, so every generation starts from the single source point $\boldsymbol{x}_0 = \boldsymbol{\mu}(\boldsymbol{c})$. Any deterministic sampler with a unique trajectory from its initial condition then returns a single output for that conditioning and therefore cannot represent a non-degenerate multimodal conditional distribution. For a one-step squared-error velocity prediction at $\tau = 0$ on the linear path, the optimal displacement is $\mathbb{E}[\boldsymbol{x}_1 - \boldsymbol{\mu} \mid \boldsymbol{c}]$, so the one-step output is $\mathbb{E}[\boldsymbol{x}_1 \mid \boldsymbol{c}]$.*

Proof. At $\sigma = 0$, the source distribution is the Dirac measure at $\boldsymbol{\mu}(\boldsymbol{c})$. A deterministic sampler initialized at that point has no random degree of freedom; under the usual uniqueness condition on the sampled dynamics, the endpoint is therefore a deterministic function of $\boldsymbol{c}$, so the sampler cannot realise more than one action for the same conditioning.

For the second statement, evaluate the linear flow-matching target at the start of the path. Since $\boldsymbol{x}_0 = \boldsymbol{\mu}$, the target velocity at $\tau = 0$ is $\boldsymbol{x}_1 - \boldsymbol{\mu}$. Squared-error regression is minimised by the conditional expectation,

$$\boldsymbol{v}^*(\boldsymbol{\mu}, 0, \boldsymbol{c}) = \mathbb{E}[\boldsymbol{x}_1 - \boldsymbol{\mu} \mid \boldsymbol{c}] = \mathbb{E}[\boldsymbol{x}_1 \mid \boldsymbol{c}] - \boldsymbol{\mu}. \tag{10}$$

A single Euler-like update over the full interval therefore gives

$$\boldsymbol{x}_{\text{out}} = \boldsymbol{\mu} + \boldsymbol{v}^*(\boldsymbol{\mu}, 0, \boldsymbol{c}) = \mathbb{E}[\boldsymbol{x}_1 \mid \boldsymbol{c}]. \tag{11}$$

The conditional-mean result is specific to the deterministic source with a one-step squared-error prediction. A finite-σ multi-step flow need not collapse to the mean, as Appendix C.6 shows. □

B EXPERIMENTAL DETAILS: MORPHING-WING DRONE

Task and simulator. The drone must pass above or below a beam placed directly between its launch position and goal. We simulate a $20 \times 10 \times 12\,\text{m}$ arena with a $0.4 \times 2 \times 0.2\,\text{m}$ beam centred at $(4.25, 0, -0.1)$. The observation has 21 dimensions (position, quaternion, body velocities, and internal actuator states) and the action has five normalised actuator commands (left and right wing sweep, thrust, elevator, rudder). Policies predict $H{=}4$ actions. Executing only the first action runs the policy at 50 Hz; executing the full chunk replans at 12.5 Hz. Table 2 gives the initial-condition distribution.

Table 2: Initial conditions per episode, sampled uniformly over the stated ranges. Quantities with "–" are held at their nominal value.

Quantity	Nominal	Initial range
Position x [m]	0	$U(0,\ 0.5)$
Position y	0	–
Position z	0	$U(-0.5, 0.5)$
Pitch	level	$U(-15^\circ, 15^\circ)$
Roll, yaw	0	–
Body velocity u	6 m/s	$U(-1, 1)$
Body velocity v	0	–
Body velocity w	0	$U(-2, 2)$
Pitch rate q	0	$U(-0.2, 0.2)$
Roll, yaw rate	0	–
Motor speed	0.545915	$U(-0.2, 0.2)$
Elevator	0	$U(-0.2, 0.2)$
Sweep	-0.9915741	–
Sweep velocity	-4.71	–
Rudder, twist	0	–

Demonstrations. Demonstrations are successful trajectories from a two-dimensional nonlinear model-predictive controller (NMPC) implemented using `acados` (Verschueren et al., 2022). We

generate them without disturbances and retain trajectories that reach a target placed 10 m ahead of the starting location within a radius of 1 m. The expert NMPC is implemented using a model of the longitudinal dynamics of the morphing-wing drone, under the constraint that the flight path should not intersect an elliptical boundary that circumscribes the rectangular obstacle. The elliptical boundary is defined such that its semi-axis is 0.1 m larger than the respective height and width of the rectangular obstacle. The nonlinear optimization problem is initialized using a random control input sequence and solved numerically using an interior point method. This randomization of the initial control sequence can split the computed optimal trajectory above or below the obstacle, leading to the multi-modal behavior observed in the expert. For generating the demonstrations, the NMPC operates at 50 Hz, and with a prediction horizon of 58 control steps. The expert consequently provides control inputs for a normalized elevator rate command, a wing-sweep command, and a throttle command to the student policy.

We sample 10,000 training, 1,000 validation, and 1,000 evaluation initial conditions from Table 2. The training split used for the route analysis contains 9,555 retained trajectories. All actions are normalised to $[-1, 1]$. The anchor is taken from the same demonstration: for a chunk starting at step i the target is actions$[i{:}i{+}H]$ and the anchor is the single action actions$[i{-}1]$, padded with the fixed initial action at the start of a trajectory. Within the chunk the anchors chain,

$$\boldsymbol{\delta}_1^{(k)} = \frac{\boldsymbol{x}_1^{(k)} - \boldsymbol{a}^{(k-1)}}{\sigma}, \qquad \boldsymbol{a}^{(-1)} = s\boldsymbol{x}_{\text{prev}}, \qquad \boldsymbol{a}^{(k)} = s\boldsymbol{x}_{\text{prev}} + \sigma \sum_{j \le k} \boldsymbol{\delta}^{(j)}, \tag{12}$$

so slot $k > 0$ is anchored on the demonstrated action actions$[i{+}k{-}1]$ during training and on its own generated predecessor at sampling. The conditioning vector additionally contains a four-step history of executed actions, but only its most recent entry acts as the anchor.

Due to the random initial control sequence seed in combination with the chosen obstacle dimensions, the expert demonstration dataset has a split of 46.2% successful trajectories that go over the obstacle, while the remaining go underneath.

Real-hardware setup. During control deployment on the real morphing-wing drone, a motion capture system provides control feedback on position and orientation. Angular velocity and flight speed feedback in drone frame is derived with a butterworth filter. For all trials, the drone is launched at a speed of 6 m/s using an automated launch rail with initial control inputs trimmed for steady flight. All trials are flown across a forward distance of 7 m. While the longitudinal motion is controlled with the NMPC or RFM-MF controllers, the lateral attitude is stabilized using a cascaded PID that provides corrective rudder and asymmetric wing-sweep commands to maintain a forward heading and zero roll angle. The NMPC requires a compute budget that would make it undeployable on a lightweight onboard microcontroller at an update rate of 50 Hz (Fig. 2b). To keep systems comparable we control NMPC and RFM-MF on the same offboard compute, using a laptop with an Intel Core Ultra 7 356H CPU and a Nvidia GeForce RTX 5070 GPU. The NMPC control frequency and prediction horizon were further downgraded to 30 Hz and $H = 30$ steps, to ensure reliable execution. We evaluate obstacle clearance to a virtual obstacle that is identical in size and shape to the obstacle used in the simulated results. We report the mean and 95% confidence interval of the minimum distance between the obstacle and the closest extremity of the drone airframe. 15 RFM-MF flights and 15 expert NMPC flights are evaluated.

Architecture and training. The deployed RFM-MF policy is a 0.96M-parameter FiLM-conditioned MLP with hidden width 256 and four blocks. We train it for 480 epochs with cosine annealing and no weight averaging, and select the checkpoint by rollout on conditions disjoint from the reported test set. The policy uses s=0.5, σ=0.276, chunk 4, and one network evaluation per chunk. A wider hidden-512 version has 2.89M parameters and measures 10.0 ms; the MeanFlow baseline uses the same trunk. RFM-5 instead uses a 0.57M concat-MLP.

The RFM-MF trunk uses SELU activations (Klambauer et al., 2017) and AdamW (Loshchilov & Hutter, 2019; Kingma & Ba, 2015). Its inputs are $\boldsymbol{\delta}_\tau$, the flow time (or (r, τ) for MeanFlow), and the conditioning vector containing the observation, goal, and previous action chunk. The output is the velocity field in $\boldsymbol{\delta}$-coordinates and is transformed back with $g(\boldsymbol{\delta}) = \sigma\boldsymbol{\delta} + \boldsymbol{\mu}$. Baselines use their reference architectures with the same input and output as RFM.: FM uses a concat MLP $[256, 512, 512, 256]$, A2A a latent-128 AdaLN MLP (0.94M), BC-GMM a five-mode GMM MLP (1.28M), and DDPM a 0.39M diffusion-policy convolutional U-Net at chunk 16 / execute 8.

Table 3: Beam-avoidance flight on the morphing-wing drone, extending Figure 2: clean success (goal reached with no beam contact) under nominal and disturbed wind, and the route taken at the beam. N=10 seeds per row, 400 held-out conditions per seed; mean with a 95% confidence interval over seeds. The route triple is over/through/under and sums to 100; the training data is 46/0/54. Latency is one policy call on the aircraft's Raspberry Pi Zero 2 W.

Method	**Params**	**NFE**	**Latency**	**SR nom.**	**SR wind**	↑/–/↓
MPC expert	n/a	n/a	516.8 ms	93.8	91.8	39/1/60
DDPM-100	0.39M	100	273.0 ms	82.1 ± 0.3	84.5 ± 0.5	40/15/44
DDPM-10	0.39M	10	26.9 ms	82.0 ± 1.7	83.8 ± 1.5	32/16/53
Consistency Policy	0.39M	1	2.7 ms	79.2 ± 2.3	80.1 ± 2.8	39/15/46
A2A (Jia et al., 2026)	0.94M	6	18.4 ms	81.1 ± 2.1	81.9 ± 2.7	39/16/45
SFP (Jiang et al., 2025)	0.56M	4	11.2 ms	76.8 ± 4.1	75.0 ± 4.2	32/16/52
MeanFlow (Geng et al., 2025)	2.89M	1	10.0 ms	78.7 ± 4.9	77.7 ± 4.9	42/10/48
BC-GMM	1.28M	1	3.0 ms	57.0 ± 3.2	56.7 ± 3.5	31/32/37
RFM-5 (ours), 5 steps	0.57M	5	8.8 ms	87.2 ± 1.4	86.5 ± 0.6	44/12/44
RFM-MF (ours), 0.96M	0.96M	1	4.7 ms	95.2 ± 1.8	93.8 ± 2.2	54/3/43
RFM-MF (ours), 2.89M	2.89M	1	10.0 ms	$\mathbf{95.9 \pm 1.6}$	$\mathbf{95.2 \pm 1.9}$	52/0/48

Evaluation protocol. All drone policies are reported on the same 400 held-out conditions. Checkpoint selection uses a separate set of 100 conditions, so selection and reporting never share an initial condition. An episode is successful only if it reaches the finish line without contacting the beam, leaving the lateral corridor ($|y| > 1.0$ m), diverging ($|z| > 2.0$ m), or failing to reach the beam. Data generation and evaluation use the same success predicate. Latency is measured for one complete policy call on the Raspberry Pi Zero 2 W and includes observation construction. Drone results use N=10 training seeds and cross-task results use N=30; confidence intervals are $t_{n-1}s/\sqrt{n}$. For the wind experiment, we resample a zero-mean Gaussian curl disturbance every step with standard deviation $[0.5, 0, 0.5, 0, 0.2, 0]$ over the six force and moment axes. This matches the per-step disturbance scale used during training.

C Additional drone results

C.1 Full comparison

Table 3 gives the complete drone comparison behind Fig. 2, adding NFE, parameter count, disturbed-wind success, and the route taken at the beam. The two RFM-MF sizes have the lowest through-beam share among the learned policies and are the only learned policies within a few percentage points of the expert on this failure measure.

C.2 Disturbance sweep

We test how far the anchor advantage extends beyond the training disturbance by scaling the wind disturbance from $0.5\times$ to $4\times$ its training value. RFM-MF remains highest through $2\times$, but its advantage disappears at $4\times$, where FM and RFM-5 are slightly higher (Table 4). Each cell is one evaluation over 400 episodes.

Table 4: Disturbance sweep on the drone task. Success rate (%) at multiples of the training disturbance, using one run of 400 episodes per cell.

	$0.5\times$	$1\times$	$2\times$	$4\times$
RFM-MF (ours), 0.96M	**93.5**	**93.5**	**87.0**	80.5
RFM-5 (ours)	86.0	86.0	79.5	83.5
Flow matching, 10 steps	87.0	88.8	81.0	**83.8**
MeanFlow	61.8	63.5	58.0	73.5

C.3 Anchoring strength and residual scale

Table 5 gives the exact values behind Fig. 4b,c. Both blocks are single changes off the deployed 2.89M RFM-MF policy, so every cell is comparable with the MeanFlow and RFM-MF in the drone table. In the top block the anchor strength s is varied and σ is recalibrated at that s as the residual RMS measured on the training set; at $s{=}0$, $\sigma{=}1$ our method defines conventional MeanFlow. In the bottom block σ is scaled at *sampling* over the deployed checkpoints, which trains nothing and so measures the reported policy itself over the same $N{=}10$ seeds.

Table 5: Anchoring strength s and residual scale σ on the drone task, as one grid: clean success (%) over 400 held-out conditions, single-run sensitivity. Every cell is the same configuration – the deployed 2.89M recipe, 480 epochs, no EMA – so rows and columns are comparable. σ^* is the residual RMS measured over the chunk window at that s; the columns scale it.

		SR nominal at σ/σ^*					
0.0	0.497	84.6 ± 0.3	84.3 ± 0.6	86.9 ± 2.9	76.5 ± 7.7	60.5 ± 7.6	45.5 ± 25.9
0.3	0.360	84.8 ± 1.3	84.9 ± 1.4	91.1 ± 6.9	90.8 ± 7.5	63.2 ± 6.8	70.8 ± 6.5
0.5	0.276	85.0 ± 0.3	86.2 ± 1.0	92.2 ± 1.6	91.1 ± 2.0	62.3 ± 2.2	61.2 ± 1.5
0.7	0.206	84.9 ± 0.9	86.2 ± 1.2	$\mathbf{94.0} \pm 4.3$	87.4 ± 6.5	61.9 ± 4.8	57.9 ± 11.1
1.0	0.166	79.7 ± 5.6	76.4 ± 10.9	82.0 ± 14.7	77.1 ± 5.4	54.0 ± 9.0	61.2 ± 7.7

C.4 Command smoothness

In simulation, RFM-MF reduces the command variation of one-step generation about five-fold relative to MeanFlow (1.206 against 5.777 TV/s, Table 6). A2A and SFP are smoother still, showing that low command variation is not specific to RFM-MF. RFM-MF is also not smoother than ten-step FM (0.923), and smoothness does not track success. Fig. 3 compares the same quantity per channel against the expert on the physical aircraft.

Table 6: Command smoothness. Lower total variation is smoother.

Method	**TV/s (nominal)**	**TV/s (wind)**	**mean peak/s**
A2A	**0.433**	1.987	38.0
SFP	0.462	**1.600**	32.2
RFM-5 (ours)	0.864	2.211	50.2
Flow matching	0.923	2.166	63.6
BC-GMM	1.122	2.194	68.8
RFM-MF (ours)	1.206	2.382	66.7
DDPM-10	2.119	2.930	29.5
DDPM-100	2.347	2.821	40.5
MeanFlow	5.777	5.405	58.9

C.5 Transport, route predictability, and training error

Shorter transport, not missing route information, is the main explanation for the one-step gain. Table 7 gives the full transport sweep behind Fig. 4a. State-only route accuracy is 0.69, 0.94, 0.99, and approximately 1.00 at control steps 0, 5, 10, and 15+; adding the previous action changes only step 5, by +0.05.

C.6 A controlled study of the one-step mean collapse

We test whether a finite stochastic source makes a well-fitted flow collapse to the conditional mean. It does not for the multi-step sampler; the one-step predictor, however, does approach the mean as σ increases in the same controlled problem (Fig. 4d, Table 8).

Table 7: Mean source-to-target transport $\mathbb{E}\|\boldsymbol{x}_1 - \boldsymbol{x}_0\|$ per training window, in normalised action units, against anchoring strength s.

	Fixed	Anchored, by s				
Task (chunk × dim)	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$	0	0.3	0.5	0.7	1.0
Drone (4×5)	4.92	1.75	1.37	1.16	0.99	0.90
Square (8×7)	8.18	2.10	1.72	1.51	1.35	1.27
ToolHang (16×7)	11.50	2.21	1.89	1.72	1.60	1.53

The problem uses $c \sim \mathcal{U}[-1, 1]$ and draws the branch independently of c, with target $\boldsymbol{x}_1 = (c, \pm s/2) + 0.05\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ at separation s=1. The conditional mean is therefore exactly $(c, 0)$ and the anchor lies on the symmetry axis, so neither carries branch information. For each $\sigma \in [0.005, 1]$ and three seeds, we train one field and evaluate the same field with 100 Euler steps and with one. We report $d(\sigma) = \mathbb{E}\|\text{output} - \mathbb{E}[\boldsymbol{x}_1 \mid c]\|$, where d=0.5 corresponds to a mode and d=0 to the conditional mean.

Table 8: Distance $d(\sigma)$ from the policy output to the conditional mean on the two-dimensional problem, for one field per σ graded at 100 Euler steps and at one. d=0.5 is a mode and d=0 is the mean; three seeds per cell.

σ	0.005	0.01	0.02	0.05	0.1	≥ 0.2
d, 100 steps	0.490	0.496	0.497	0.498	0.500	0.501
d, 1 step	0.266	0.265	0.135	0.060	0.028	0.017–0.026

The 100-step policy remains on a mode throughout the sweep, including at σ=0.005, while the one-step policy approaches the conditional mean for $\sigma \geq 0.1$. At $\sigma \leq 0.01$, the larger one-step distance from the mean comes from a poorly fitted field whose samples spread across the gap, not from recovered mode coverage. This separates the exact σ=0 result of Proposition 2 from the behaviour of a finite-σ multi-step flow.

D MANIPULATION BENCHMARK DETAILS

The manipulation experiments test whether the one-step RFM effect transfers beyond the drone; they do not provide a fully matched source comparison, because several RFM cells use task-specific horizons or trunks. The matched MeanFlow experiment on the drone therefore remains the controlled source comparison. Table 1 in the main text gives the full results.

Shared evaluation protocol. ToolHang and Square are Robomimic tasks (Mandlekar et al., 2021), evaluated from state and image observations, and Stacking is the D3IL task (Jia et al., 2024), evaluated from state. ToolHang uses a 34.5M FiLM-UNet with chunk 16, execute 8, and observation horizon 2; Square uses a 33.2M FiLM-UNet with the same horizons; Stacking uses a 16.4M FiLM-UNet with chunk 4, execute 4, and observation horizon 5. ToolHang and Square select checkpoints on 50 seed-0 conditions and confirm on 200 disjoint conditions. Stacking selects on contexts 0–59 and reports the held-out contexts 60–99. Inference time is divided by the number of executed actions.

For Square from state, the DDPM and BC-GMM rows use delta actions, following the convention of their reference implementations (Mandlekar et al., 2021; Chi et al., 2023). Three BC-GMM cells marked * in Table 1 use the published benchmark values (Mandlekar et al., 2021; Jia et al., 2024). We were unable to reproduce their results (we found 0.0 on ToolHang, 1.8 ± 1.1 on Stacking, and 0.3 ± 0.3 on ToolHang from images). Since these reproductions remained below 5%, we report the published numbers instead.

We report each method at its best deployment step count because this choice materially affects performance. On ToolHang, the same FM weights lose about 15 points when sampled at one step

instead of ten on the same 200 conditions. The RFM-5 row is also non-monotone in step count, which is why its reported NFE differs between columns.