Learning-Assisted Control under Unknown Payloads

Correcting the Model, Monitoring the Policy, and Handing Over

Shengwei Li
The University of Sydney, NSW, Australia
Project technical report, second version, November 2026

***Abstract*—When a robot arm holds a payload of unknown mass, the model inside its controller is wrong, and an imitation policy that was trained on the empty arm no longer reaches its targets. This report studies three questions in controlled simulation. First, how should the controller's model be corrected: by estimating the payload mass, by learning the model error with a neural network, or both? On a two-link arm with a sampling-based model predictive controller (MPC), the amount of data decides: with 9 s of motion the mass estimate is better, with 30 s or more physics plus a learned residual is best, but outside the region of its data the MPC exploits the learned model's errors. Second, when a diffusion policy controls the arm, which model should a monitor use to detect its failure, and when should it hand control to the MPC? Residuals of a model of the empty arm detect the payload, and a learned residual does so where the physics model has a structural error; only the consistency of the policy's own actions detects that the policy has left its training region. Third, the same experiments on a Franka Panda with joint impedance control at two stiffnesses confirm the monitoring results, while the impedance loop makes a learned model almost useless for control: its value moves from control to monitoring. For Parts II and III, all 26 expected outcomes were written down before the test runs; 16 held, 8 held in part, and 2 failed.**

I. Introduction

A model predictive controller (MPC) selects each action by predicting where candidate actions lead, so its decisions are only as good as its model. An imitation policy has no explicit model; it copies the actions of an expert on the states it was shown. When a manipulator holds an object of unknown mass, both fail: the MPC's predictions are wrong, and the policy acts on an arm that no longer responds as in its demonstrations.

This report follows the problem through three parts, each building on the one before:

- **Part I, correct the model.** A sampling-based MPC on a planar two-link arm. The model is corrected from recorded motion by estimating the payload mass [1], by training a neural network on the residual of the physics model [2], [3], or both. The question is when a learned correction helps and when it hurts.
- **Part II, monitor the policy.** A closed-form diffusion policy (CFDP) [4] controls the same arm. Monitors look for its failure; on an alarm, the MPC with an online mass estimate takes over, as in a Simplex architecture [5]. The question is which model a monitor should use: a coarse physics model of the empty arm or an accurate learned one.
- **Part III, change the platform.** The same experiments on a 7-joint Franka Panda with joint impedance control, at two joint stiffnesses.

The setting is chosen for control over every source of error, not for realism. In each platform, the controller's model reproduces the simulator step for step when it is told the truth, so an exact reference exists and every remaining error comes from a mismatch that we set. The main observations are:

1) The amount of data decides the best correction. With little data, the mass estimate is better; with 30 s or more, physics plus learning is best inside the training region.
2) Outside the training region, the MPC exploits learned-model errors. Its expected outcome is optimistic by half to all of the model's error, for every wrong model; learned models outside their data have larger errors to exploit. An ensemble of five networks reduces the worst failures.
3) Monitors that use a model detect the payload; the consistency of the policy's actions detects that the policy has left its region. A learned residual helps a monitor where the physics model has a structural error (unmodelled friction).
4) On the Panda, the joint impedance loop absorbs most model error. A learned model can improve the final error by at most 3.5 mm, but it brings the alarm forward from about 1.6 s to 0.2 s when friction hides a payload.

No new method is proposed. The contribution is a controlled comparison of known ingredients, with the expected results fixed before the tests.

II. Related Work

Model correction. Rigid-body dynamics are linear in the inertial parameters, so a payload can be estimated by least squares [1]. A learned residual on a nominal model has been used for drone landing [2] and, as a Gaussian process inside MPC, for a robot arm [3]; [6] reviews learning-based MPC. Neural dynamics are combined with sampling-based MPC in [7], and ensembles represent model uncertainty in [8]. Path-integral control [9] has been extended to uncertain physical parameters [10].

Diffusion policies. Diffusion Policy [11] generates chunks of future actions by denoising; CFDP [4] computes the score in closed form from the demonstrations and needs no training. We use CFDP throughout and Diffusion Policy as a second policy in Part II.

Runtime monitoring. Sentinel [12] monitors the temporal consistency of a generative policy's actions and its task progress.

FIPER [13] and FAIL-Detect [14] combine policy-derived uncertainty scores with conformal calibration [15] on successful runs. Model-based monitors predict outcomes with a learned dynamics model, in a latent space [16] or with latent innovations conditioned on action chunks [17]. We accumulate signals with CUSUM [18].

Fallback. The Simplex architecture [5] lets a monitor switch from a high-performance controller to a simple, trusted one. Here the learned policy is the first and an MPC with an online mass estimate is the second.

Each of these ingredients exists. We did not find a study of diffusion policies under payload changes, or one that compares which model a monitor should use, but our search was not exhaustive.

III. Setup

A. Platforms and model errors

Two-link arm (Parts I and II). MuJoCo [19], 2 ms step; two 0.3 m, 1 kg links in a vertical plane under gravity; torque limits of 20 and 10 N m (Fig. 1). The controller's model is written by hand and checked against the simulator's inertia matrix and bias forces. The payload is 1 kg.

Franka Panda (Part III). The Menagerie model [20] with a 0.1 m tool. The arm is driven as the real robot is: a joint impedance loop with the gains of the manufacturer's examples tracks joint setpoints, with gravity compensation for the load that the controller *believes* the arm holds. The controller's model is a second MuJoCo copy; with the true load it matches the simulator to 4×10^{-16}. A soft setting uses a quarter of the stiffness (damping halved, same damping ratio). Payloads are 0, 1 and 2.5 kg.

Model errors. On each platform, three worlds differ from the model of the empty arm in increasing ways: the payload mass only (W1); its centre of mass offset by a few centimetres (W2); and W2 plus joint friction that the model lacks (W3). In W1 a correct mass makes the model exact; in W2 and W3 no single mass does.

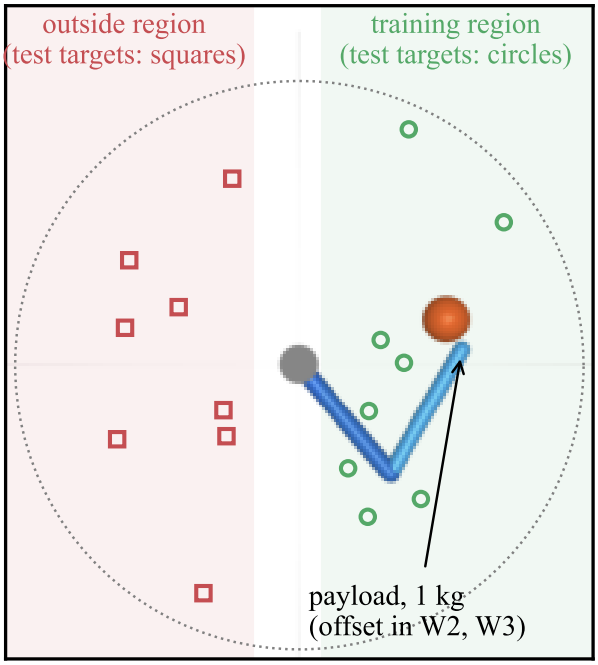


Fig. 1. The two-link arm with its payload, the region where training data was collected (right) and the region it never aimed at (left), with the targets of the 8 test tasks in each.

TABLE I
Methods. All model-based methods use the same MPC.

C0	physics model of the empty arm
C1	physics model with the payload mass estimated from data
C2, C3	C0's or C1's model plus a residual network
C2-E, C3-E	as C2, C3, averaging five networks
Reference	physics told the true payload and world (a limit, not a method)
Policy	CFDP imitating the reference MPC on the empty arm
Monitors	physics, learned or oracle residual; action consistency
Handover	policy until the first alarm, then MPC with online C1

B. Controllers

All model-based methods share one sampling-based MPC with path-integral weighting [9]; only the predictor changes. On the two-link arm, every 20 ms it blends 256 perturbed torque plans over 0.2 s; on the Panda, 128 plans of joint setpoints over 0.3 s. The cost is the squared tip-to-target distance plus a speed penalty. The settings were tuned once on development tasks and then fixed.

Table I lists the predictors. C1 fits the payload mass by least squares; the online version used for handover re-estimates it every 10 periods from the run so far. The residual networks (two hidden layers of 64 tanh units, Adam, early stopping on held-out episodes) predict the physics model's one-period error from joint angles, speeds and commands.

C. Policy, monitors and handover

Policy. For each world, the expert (the reference MPC on the empty arm) records 200 reaching and 200 corrective episodes in the training region. CFDP observes joint angles, speeds and the target relative to the tip, and outputs a chunk of 8 future periods, of which 4 are executed. On the two-link arm the chunk is joint angles tracked by a PD loop; on the Panda it is the expert's joint setpoints, because the impedance loop lags its setpoint and a policy that sent the demonstrated angles stopped short (1.5 cm instead of 0.19 cm, unloaded). Gravity compensation always assumes no load, as on a real robot whose load is unknown.

Monitors. A residual monitor predicts each period's end speeds from the measured start state and the command, with a model of the empty arm, and uses the mean prediction error since the last check. The *physics* monitor uses the model without friction; the *learned* monitor adds a residual network trained on the unloaded demonstrations; the *oracle* knows the friction (a limit). The *consistency* monitor needs no model: it compares the overlapping parts of consecutive action chunks [12]. Monitors read measurements with noise (1×10^{-4} rad, 0.01 rad/s). Each signal is standardised on 20 unloaded development runs and accumulated by CUSUM; the alarm threshold is the split-

TABLE II
PART I. MEDIAN FINAL TIP ERROR IN CM (RUNS NOT SETTLED, OF 24). C3-E AVERAGES FIVE NETWORKS.

Tasks	World	Data	C1	C2	C3	C3-E	Ref.
inside	W2	30 s	1.99 (24)	1.03 (11)	0.31 (4)	0.21 (2)	0.04 (0)
inside	W2	90 s	1.95 (22)	0.36 (4)	0.07 (1)	0.07 (0)	0.04 (0)
inside	W3	30 s	1.94 (24)	0.81 (9)	0.84 (8)	0.51 (6)	0.06 (0)
inside	W3	90 s	1.82 (22)	0.59 (6)	0.23 (4)	0.23 (3)	0.06 (0)
outside	W2	30 s	1.79 (20)	3.63 (22)	2.22 (21)	1.63 (17)	0.04 (0)
outside	W2	90 s	1.69 (19)	1.78 (18)	0.56 (6)	0.42 (2)	0.04 (0)
outside	W3	30 s	1.70 (18)	5.91 (24)	1.84 (18)	1.73 (20)	0.06 (0)
outside	W3	90 s	1.61 (17)	1.85 (21)	1.23 (15)	0.90 (10)	0.06 (0)

conformal quantile of the per-run maximum on 40 (two-link, monitoring) or 100 (handover, Panda) unloaded calibration runs, for a 5% false-alarm rate.

Handover. The policy acts; at each new chunk the monitor checks the run so far. On the first alarm, the MPC takes control with the payload mass estimated from every step of the run.

D. Protocol

Every setting was chosen on development material (separate data seeds, validation episodes and tasks) and then frozen. Test tasks have their own seeds, inside and outside the training region. Data is split by whole episodes. A run lasts 3 s; it is *not settled* if the tip is more than 1 cm from the target at the end. For Parts II and III, the expected outcome of every test was written and committed before the run.

IV. PART I: CORRECTING THE MODEL

Data is recorded by running C0 with random torque perturbations in the training region, in budgets of 9, 30 and 90 s of motion. There are 8 test tasks inside and 8 outside the region and three data seeds: 24 runs per method, world and budget.

Prediction. C1 is exact in W1, but in W2 and W3 it leaves a 0.2 s tip prediction error of 2.5–4 cm that more data does not reduce: it explains the offset and friction by overestimating the mass. C2 is far worse than C1 with 9 s and far better with 90 s. C3 is the most accurate throughout. Outside the training region the learned models degrade by about an order of magnitude; C1 does not.

Control inside the region. C0 stops 8–10 cm from the target and C1 about 2 cm in W2 and W3 (Fig. 2, Table II). With 30 s or more, learning does better; with 90 s, C3 reaches 0.07 cm in W2 against 0.04 cm for the reference. With 9 s, C2 is worse than C1. C1 is consistently a little off; the learned methods are usually precise but occasionally far off.

Control outside the region. C1 performs as inside, but the learned methods lose most of their advantage. In W2 with 90 s, C2 goes from 0.36 to 1.78 cm, no better than C1; C3 stays ahead (0.56 against 1.69 cm) but falls behind with 30 s. The worst C2 runs swing the arm through large loops and end 49–66 cm from targets that C1 reaches directly.

Why: optimism. Along the path the MPC chose, the learned models' one-period prediction error grows 8–13 times outside the region; C1's does not. The error has a direction. We define optimism as how much closer to the target the model expected the tip to be than it was. For every wrong model, physical or learned, inside or outside, optimism is half to all of the prediction error (Fig. 3). This is a selection effect: by choosing the action its model rates best, the MPC chooses actions whose outcome the model overestimates. Learned models outside their data differ only in the size of the error.

Ensembles. Averaging five networks mainly cuts the tails: outside the region in W2 with 90 s, unsettled C3 runs drop from 6 to 2, and with 30 s C3-E is no longer worse than C1. With 9 s it does not help. A penalty on network disagreement changed nothing measurable, probably because it was measured one step ahead, where disagreement is small. The ensemble needs about 28 ms per 20 ms period on the laptop CPU used.

V. PART II: MONITORING THE POLICY

The policy under load. Trained on the empty arm, CFDP settles within 0.4 cm unloaded but stops 2.7–3.1 cm from the target with 1 kg (32 inside tasks per world; Table III): gravity compensation assumes no load, and the arm sags. Outside the training region it fails completely (26–28 cm). A standard Diffusion Policy [11] gives the same picture (2.8–3.0 cm with 1 kg, 19–25 cm outside), so these failures do not depend on how the policy is built.

Which model should a monitor use? Fig. 4 shows the alarm rate against payload. When the model of the empty arm is exact (W1), every residual monitor detects every run from 0.05 kg. When it lacks the friction (W3), the threshold must rise to hold the false-alarm rate, and the physics monitor detects only 13% of runs at 0.05 kg; the learned monitor, whose network learned the friction from the unloaded demonstrations, detects 63%. The consistency monitor hardly sees the payload (at most 44% at 1 kg), because the policy's chunks stay consistent while the arm sags. Outside the training region it is the reverse: the consistency monitor alarms in 94–97% of runs, the residual monitors in W1 and W2 in at most 9%. The two kinds of monitor see two kinds of failure: of the execution and of the policy. In W1, the false-alarm rate on unloaded test runs was 16% against the 5% target, with 40 calibration runs; the handover experiment used 100.

Handover. Table III gives the result. In W1 with 1 kg, a residual-triggered handover reduces the error from 3.1 cm to 0.04 cm, equal to the MPC alone. In W2 and W3 it only reaches 1.3–1.8 cm: the MPC that takes over estimates only a mass, and on this torque-driven arm the remaining model error matters, as in Part I. Outside the region, only the consistency trigger recovers the policy (W1: 0.07 cm, 4 of 32 unsettled). Combining the two triggers at half the false-alarm budget each did not work: the consistency signal outside the region exceeds its threshold only by a small margin, and at the stricter threshold most alarms disappeared. Finally, a detected failure does not always justify a handover: in W3 at 0.1 kg, the learned monitor alarms in 97% of runs, but the MPC that takes over, which does not know the friction, is slightly worse than the policy, whose demonstrations came from an expert that knew it.

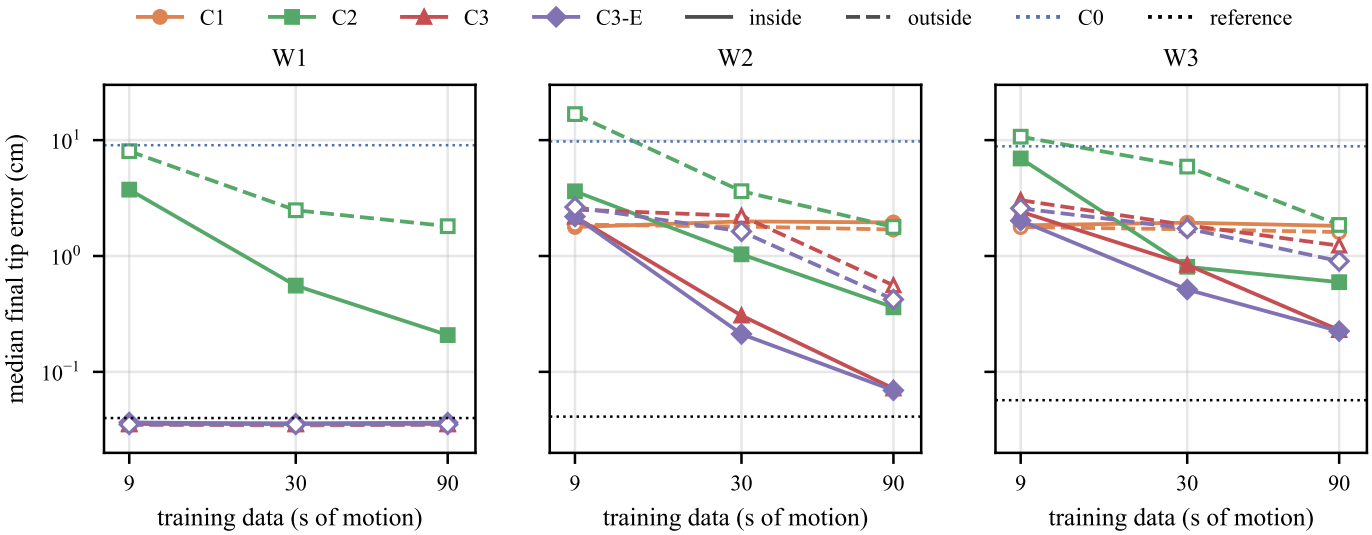


Fig. 2. Part I. Median final tip error over 24 runs against the amount of training data, for test tasks inside (solid) and outside (dashed) the training region. Dotted: C0 and the reference.

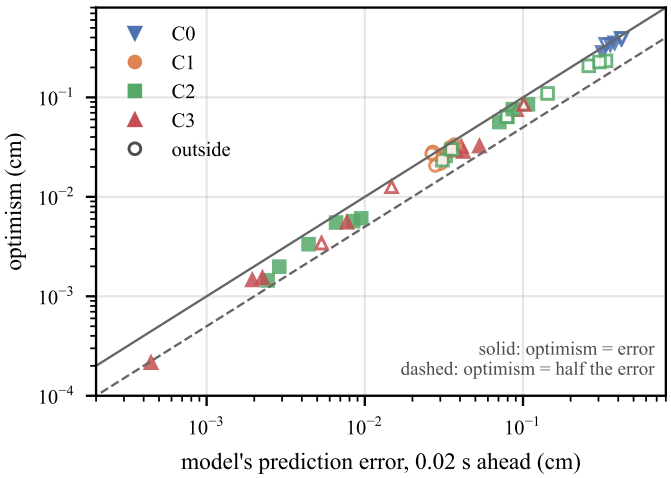


Fig. 3. Part I. Median optimism against median one-period prediction error along the executed path, one point per method, world, data budget and region (exact models omitted).

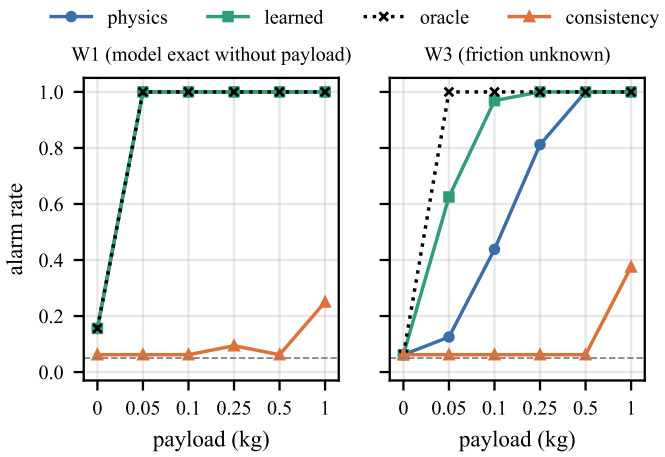


Fig. 4. Part II. Alarm rate of each monitor on the two-link arm, CFDP, 32 tasks inside the training region. At 0 kg the rate is the false-alarm rate (target 5%, dashed).

VI. Part III: The Franka Panda

The platform changes the role of the model. Table IV compares C0, C1 and the reference on the Panda. With stiff joints, C1 is within 0.04 cm of the reference in every world, although it underestimates the mass by 12–15%: the impedance loop absorbs most of the model error. With soft joints, C0's error roughly doubles and C1's excess over the reference rises to 0.05–0.35 cm, about four times its stiff value, in proportion to the stiffness. This excess is the most that any learned correction could gain for control here. In W3 without payload and with soft joints, C1 also takes friction for mass (up to 0.24 kg) and is slightly worse than C0.

Policy and monitors. CFDP settles within 0.16–0.51 cm unloaded and fails outside the region (11–25 cm). Under load it is about twice as bad as C0 (stiff, 2.5 kg: 3.1–3.7 against 1.7–1.9 cm), unlike on the two-link arm, where C0 drives torques without a joint loop. The policy copies setpoint patterns from

TABLE III
Part II, two-link arm. Median final error in cm (runs not settled, of 32): CFDP alone, the adaptive MPC alone, and handover triggered by the physics, learned or consistency monitor.

Tasks	W	kg	Policy	Adapt.	Phys.	Learn.	Cons.
ins.	W1	0	0.39 (1)	0.06 (0)	0.39 (1)	0.39 (1)	0.33 (0)
ins.	W1	1	3.09 (29)	0.04 (0)	0.04 (0)	0.04 (0)	2.69 (28)
ins.	W2	1	2.69 (29)	1.24 (20)	1.47 (23)	1.47 (23)	2.50 (30)
ins.	W3	0.1	0.72 (12)	0.83 (14)	0.89 (14)	0.86 (13)	0.72 (12)
ins.	W3	1	2.84 (27)	1.49 (23)	1.35 (24)	1.80 (25)	2.69 (27)
out.	W1	0	27.79 (31)	0.06 (0)	26.62 (30)	26.62 (30)	0.07 (4)
out.	W3	0	25.72 (32)	0.49 (6)	1.31 (18)	1.73 (22)	1.75 (18)

TABLE IV

PART III. MEDIAN FINAL ERROR IN CM ON THE PANDA, 16 TASKS INSIDE THE TRAINING REGION, STIFF AND SOFT JOINTS.

		stiff			soft		
W	kg	C0	C1	Ref.	C0	C1	Ref.
W1	1	0.80	0.10	0.10	1.79	0.08	0.08
W1	2.5	1.89	0.10	0.10	4.26	0.09	0.09
W2	1	0.71	0.11	0.10	1.58	0.13	0.08
W2	2.5	1.67	0.13	0.10	3.84	0.26	0.09
W3	1	0.69	0.08	0.08	1.64	0.23	0.07
W3	2.5	1.67	0.12	0.08	3.81	0.34	0.08

TABLE V

PART III. MEDIAN FINAL ERROR IN CM ON THE PANDA (RUNS NOT SETTLED), POOLED OVER THE THREE WORLDS (48 RUNS) OR W3 ONLY (16 RUNS).

Tasks	kg	W	Stiff.	Policy	Adapt.	Phys.	Learn.	Cons.
ins.	2.5	all	stiff	3.60 (48)	0.11 (0)	0.13 (0)	0.12 (0)	0.13 (0)
ins.	2.5	all	soft	6.29 (48)	0.22 (0)	0.16 (0)	0.17 (0)	0.20 (0)
ins.	1	all	stiff	1.83 (45)	0.10 (0)	0.11 (2)	0.10 (0)	1.43 (29)
ins.	1	all	soft	3.22 (47)	0.12 (0)	0.15 (3)	0.13 (0)	0.15 (2)
ins.	1	W3	stiff	1.48 (16)	0.08 (0)	0.16 (2)	0.11 (0)	1.43 (14)
ins.	1	W3	soft	2.86 (16)	0.23 (0)	0.50 (3)	0.19 (0)	0.28 (1)
out.	0	all	stiff	19.06 (42)	0.09 (0)	1.33 (25)	0.69 (23)	0.10 (3)
out.	0	all	soft	13.12 (46)	0.10 (0)	5.73 (32)	5.71 (31)	0.10 (3)

an unloaded arm and never raises its setpoints to compensate a sag; the MPC replans them. The monitoring results of Part II hold: in W1 and W2 the residual monitors detect 88–100% of runs from 0.1 kg at false-alarm rates of 0–6%; in W3 the physics monitor detects nothing below 0.25 kg, the learned monitor 69–100% at 0.25 kg. The learned monitor stays well below the oracle at 0.1 kg (19–25% against 88–100%). One difference: on the Panda the consistency monitor also detects large payloads inside the region (100% at 2.5 kg), because the loaded arm stops in states that the demonstrations rarely contain, where the policy is uncertain.

Handover. Fig. 5 and Table V show the result. With 2.5 kg, every handover recovers every run, within 0.04 cm of the adaptive MPC; the residual triggers mostly alarm at the first possible check (0.08 s). Outside the region without payload, the consistency trigger recovers the policy (3 of 48 unsettled at each stiffness); the residual triggers do so only in W3, where friction outside the calibrated motion makes them alarm. The learned model matters where the physics model is blind: in W3 with 1 kg, the physics trigger alarms after 1.6–1.8 s and leaves 2–3 of 16 runs unsettled; the learned trigger alarms after 0.16–0.24 s and recovers all. False-alarm handovers on unloaded runs were 6% overall but 4 of 16 in one cell (W2, soft); they did no harm, because the MPC that took over was more accurate than the policy.

VII. DISCUSSION AND LIMITATIONS

Expectations against results. Of the 26 written expectations, 16 held, 8 held in part and 2 failed. The failures were informative. Combined triggers at half the false-alarm budget lost most consistency alarms (Part II). And Diffusion Policy was *not* more

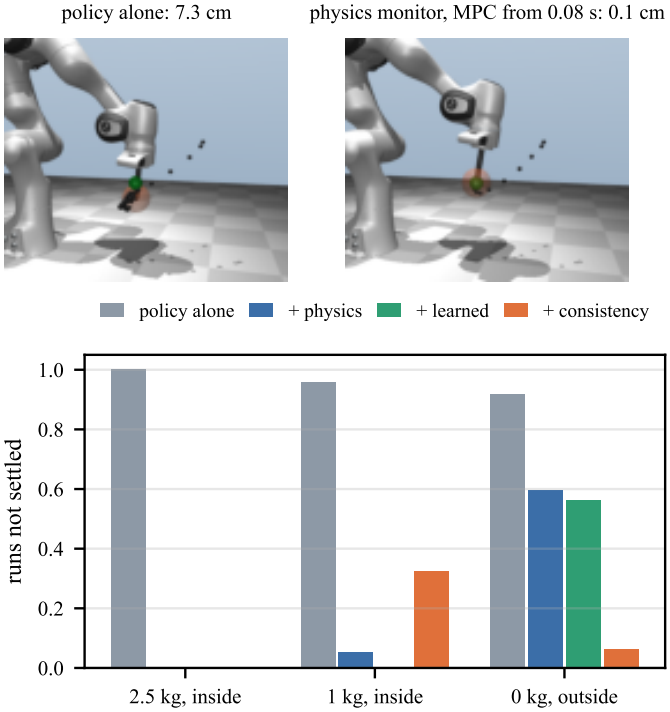


Fig. 5. Part III. Top: one typical test run with 2.5 kg (soft joints, W1), end of the run; dots show the tip path, the green sphere the target. Bottom: share of runs not settled within 1 cm, 96 runs per bar (16 tasks, 3 worlds, 2 stiffnesses).

consistent than CFDP outside the region: the consistency signal measures the policy's uncertainty about its current observation, not a property of CFDP. The largest partial miss was that, on the Panda, CFDP under load is worse than C0 rather than similar.

Practical reading. With little data, estimate the physical parameter; with more, add a learned residual and average several networks, and judge the result by its worst runs and where it will be used. To monitor a policy, use a model of the empty arm for execution faults, learned where the physics model has structural errors, and the consistency of the policy's own actions for unfamiliar situations; neither covers the other. A policy can also be confidently wrong: in one Panda run outside the region, the policy stopped 20 cm from the target with consistent chunks, and no monitor saw it.

Limitations. All results are from simulation. The tasks are reaches, for which a hand-written cost works, so the adaptive MPC alone is everywhere as good as the best handover: the policy is not needed here, and the study only asks whether monitoring and handover can recover one. The payload is present from the start rather than picked up. "Outside" is one region of the workspace, and W2 and W3 were chosen by hand. The ensemble MPC and CFDP do not run in real time as implemented (about 28 and 140 ms against 20 and 80 ms).

VIII. CONCLUSION

In controlled simulation where every model error is known, a learned correction made the MPC markedly more accurate when data was sufficient and the task stayed near the data, and less reliable otherwise, because the controller exploits whatever

errors its model has. When a learned policy acts instead, model-based monitors and the policy's own consistency detect different failures, and a handover to the MPC recovers both. On an impedance-controlled arm, the learned model's value moves from control to monitoring: it barely improves the final error but brings the alarm forward by more than a second where friction hides the payload.

References

[1] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Estimation of inertial parameters of manipulator loads and links," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 101–119, 1986.

[2] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 9784–9790.

[3] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019.

[4] R. Mishra and I. R. Manchester, "Training-free imitation learning with closed-form diffusion policies," arXiv:2606.01238, 2026.

[5] L. Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[6] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[7] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018, pp. 7559–7566.

[8] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[9] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[10] I. Abraham, A. Handa, N. Ratliff, K. Lowrey, T. D. Murphey, and D. Fox, "Model-based generalization under parameter uncertainty using path integral control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2864–2871, 2020.

[11] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proc. Robotics: Science and Systems (RSS)*, 2023.

[12] C. Agia, R. Sinha, J. Yang, Z. Cao, R. Antonova, M. Pavone, and J. Bohg, "Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress," in *Proc. Conf. Robot Learning (CoRL)*, ser. PMLR, vol. 270, 2024.

[13] R. Römer, A. Kobras, L. Worbis, and A. P. Schoellig, "Failure prediction at runtime for generative robot policies," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[14] C. Xu, T. K. Nguyen, E. Dixon, C. Rodriguez, P. Miller, R. Lee, P. Shah, R. Ambrus, H. Nishimura, and M. Itkina, "Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies," in *Proc. Robotics: Science and Systems (RSS)*, 2025.

[15] A. N. Angelopoulos and S. Bates, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," arXiv:2107.07511, 2021.

[16] H. Liu, S. Dass, R. Martín-Martín, and Y. Zhu, "Model-based runtime monitoring with interactive imitation learning," arXiv:2310.17552, 2023.

[17] Y. Zhou, R. Qiu, Y. Chen, J. Cui, and W. Zhi, "PATCH: Action-chunk-conditioned latent patch innovation monitoring for robot manipulation," arXiv:2606.16690, 2026.

[18] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[19] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

[20] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors, "MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo," https://github.com/google-deepmind/mujoco_menagerie, 2022.