

Residual Learning on a Physics Pivoting Controller: Faster Search, and Where the Quasi-Static Model Breaks

Clement Aidoo
CANLABS, Ghana
aidooclement613@gmail.com

October 2026 (simulation study, preprint draft)

Abstract

A parallel-jaw gripper can reorient a grasped rod by loosening its grip and letting gravity pivot it. In earlier work we built a model-based controller for this task that measures the rod angle from wrist motor current and steps the grip force down using a quasi-static friction model. That controller is accurate but slow: it searches for the slip threshold in small fixed steps, and each measurement costs a 10s wrist sweep. Here we ask whether a learned residual on top of the controller makes it faster, at what cost in accuracy, and how it compares with a policy learned without the model. In MuJoCo simulation we trained linear policies with evolution strategies (1560 episodes per run, three seeds per method) that adjust each grip step and the release rate. On 40 held-out objects, residual policies cut the measurements per pivot from 18.0 to 9.3 and pivot time by 24%, with almost all of the saved steps in the search phase, but overshoot the target more often (88% vs 95% of objects within 0.05 rad). Policies learned without the model were slower than the residual policies, and less accurate with noise-model sensing. Most overshoots happened on a single step: when the rod first slips it rotates in one jump (median 0.46 rad), which tended to be larger for larger force steps and faster releases, and which overshoots shallow targets. A guard that bounds the search step by the target angle and, once the rod moves, lets the residual be only more cautious than the model brought overshoots back to the model controller’s level (93% within 0.05 rad) while keeping a 19% time saving; the guard was designed after we saw test results, so we confirmed it once on 40 fresh objects, where the guarded policies reached 98, 95 and 90% within 0.05 rad against 95% for the model controller and were 18% faster. With the full motor-current sensing pipeline the guarded residual was 32% faster than the model controller (3.73 vs 5.49 simulated minutes) at 97% vs 100% within 0.05 rad. One of three seeds stayed clearly worse with noise-model sensing, a release-rate cap added after training made every policy worse, and on heavier out-of-range objects the residual and model controllers degraded alike while pure RL degraded more. All results are from simulation.

1 Introduction

Gravitational pivoting reorients a long object held in a parallel-jaw gripper by reducing grip force until gravity overcomes the torsional friction at the contacts [1, 2, 3]. In a previous study [4] we showed that the angle of the pivoting object can be measured from the motor current of the wrist joint alone, and we built a closed-loop controller on that measurement. The controller alternates three actions: release the grip slowly to a chosen force, re-clamp, and measure the angle with a slow wrist sweep. It chooses each new force from a quasi-static soft-finger model in which the equilibrium angle θ satisfies $\cos \theta \approx F/F_h$, where F_h is the grip force at which the rod starts to slip from horizontal.

The controller reaches its targets, but slowly. Before the rod first moves, F_h is unknown, so the controller lowers the force in fixed 6% steps until it sees motion; objects with high torsional friction need many such steps, and with motor-current sensing each step costs a 10 s measurement sweep. This is a natural place for learning. Residual reinforcement learning [7, 8] keeps a hand-designed controller and learns a correction to its output, which typically needs less data than learning from scratch and starts from the controller’s performance rather than from zero.

This paper tests that idea on the pivoting controller and reports what happened, including where it failed. The contributions are:

1. A controlled comparison of the model controller, a residual policy on top of it, and a policy learned without the model, trained with the same algorithm, observations, objects and budget, and evaluated on held-out objects with both a sensor noise model and the full motor-current pipeline of [4] (Sections 3–4).
2. A diagnosis of the learned policies’ main failure, target overshoot: it happens mostly on the first-slip step, where the rod rotates in one jump that grows with the force step and the release rate, partly because the quasi-static equilibrium is very steep near the slip threshold, and partly because the rod rotates further than the quasi-static model predicts, mostly through a static model mismatch with a smaller rate-dependent swing on top (Section 4.4).
3. A model-based guard that bounds the search step by the target and makes the residual only more cautious than the model once the rod is moving. It kept most of the time saving with accuracy not significantly different from the model controller; because it was designed after we had seen test results, we confirmed it on a fresh set of objects. We also report that a release-rate cap added after training made every policy worse (Section 4.5).

2 Related Work

Pivoting. Holladay et al. planned open-loop pivoting with a friction model [1]; Viña et al. added adaptive control with visual and tactile feedback [2]; Toskov et al. learned gravitational pivoting with tactile sensing [3]. Waltersson and Karayiannidis built a sensorized parallel gripper that measures contact forces and sliding velocities and used it for gravity-assisted rotational and linear slippage control [5]. Our previous work replaced dedicated tactile sensing with wrist motor current [4]; this paper keeps that sensing and studies the controller.

Residual learning. Residual reinforcement learning adds a learned policy to a conventional controller and trains only the correction [7]; residual policy learning applies the same idea to imperfect initial policies in manipulation [8]. Both report better sample efficiency than learning from scratch. Closer to manipulation by slip and rolling, Chen et al. add physics-based priors, a grasp-quality reward and a fingertip curvature design, to reinforcement learning for in-hand rolling without external sensing [6]. Our setting differs in that the base controller is already accurate and the residual is rewarded for speed at a fixed exchange rate against accuracy, so the question is how much accuracy it gives up.

Constraining learned policies. Shielding enforces a safety specification by overriding a learner’s actions at run time [9]. Our guard is related in spirit but much simpler: a hand-derived clip of the residual against the base controller’s own proposal, with no formal guarantee.

Optimizer. We use antithetic evolution strategies [10] on linear policies, which are competitive with deep RL on continuous-control benchmarks [11] and suit our small, episodic problem with a sparse return.

3 Task and Methods

3.1 Simulation testbed

The testbed is the MuJoCo [12] model of [4], unchanged: a parallel-jaw gripper on a position-controlled wrist hinge holds a rod off-centre with two spherical pads that use a soft-finger contact (Coulomb and torsional friction). Objects are randomized in mass $m \in [0.05, 0.2]$ kg, grasp offset $d \in [0.02, 0.06]$ m, sliding friction $\mu \in [0.4, 0.8]$ and torsional friction $\mu_t \in [0.002, 0.006]$ m; target angles of the centre-of-mass (COM) direction are drawn from $[-1.3, -0.3]$ rad below horizontal. The task starts from a clamped, horizontal grasp.

3.2 Base controller

Each step of the controller of [4]: (i) ramp quickly to $1.05F_{k-1}$, (ii) release slowly to the new force F_k at rate r (0.3 N/s), (iii) hold for 1 s, (iv) re-clamp with a smooth 0.5 s ramp and a 0.5 s settle, (v) measure the angle $\hat{\theta}_k$. Before motion is seen, $F_k = 0.94F_{k-1}$ (search phase). Once the rod has moved, the controller fits $c = \cos \hat{\theta}_k / F_k$ and aims 70% of the remaining angle: $F_{k+1} = \cos(\hat{\theta}_k + 0.7(\theta^* - \hat{\theta}_k)) / c$. It stops when $\hat{\theta}_k \leq \theta^* + 0.03$ rad, or after 30 steps. The initial force is $F_0 = \min(30, \hat{A} / (2\mu_{t,\min}))$ N, where $\hat{A}$ is the gravity-torque amplitude from the first probe.

3.3 Learned policies

All policies are linear, $a = \tanh(Wo) \in [-1, 1]^2$, on an observation o with remaining angle $\hat{\theta} - \theta^*$, $\hat{\theta}$, θ^* , $\log(F_{k-1}/F_0)$, step index, a moved flag and a bias. (The code also passes a feature intended as the model’s fit residual, but it is computed from the same measurement as the fit and is identically zero, so the policies effectively see these seven inputs.) The two actions set the grip step and the release rate.

- *Residual*: $F_k = F_k^{\text{model}} e^{0.25a_0}$ and $r_k = 0.3 e^{1.4a_1}$ N/s. With $W = 0$ this is exactly the base controller.
- *Pure*: $F_k = F_{k-1} e^{-0.25(1+a_0)}$ with the same rate action. It keeps the step loop, measurement, stopping rule and model-based initial force F_0 , but uses no friction model to choose later forces.
- *Guarded residual*: as Residual, but once the rod has moved, $F_k \geq F_k^{\text{model}}$ and $r_k \leq 0.3$ N/s, so the residual can only hold more grip or release more slowly than the model. In the search phase we also apply the bound $F_k \geq F_{k-1} \cos \theta^*$: if the rod did not move at F_{k-1} then $F_h \leq F_{k-1}$, and a quasi-static release to F_k cannot rotate it past $\arccos(F_k/F_{k-1})$. *Model + guard* applies the same search bound to the base controller, as an ablation.

In every mode the force is capped at $0.995F_{k-1}$ so each step releases.

3.4 Reward and training

An episode’s return is $-|\theta_{\text{final}} - \theta^*|/0.05 - 0.5t/\text{min}$, where t is simulated time, or -10 if the rod is dropped. Both learned modes were trained from $W = 0$ with antithetic evolution strategies [10]: 6 perturbation pairs, $\sigma = 0.1$, rank-normalized fitness, Adam step 0.06, 6 random objects per iteration shared by all perturbations, 20 iterations, i.e. 1560 episodes per run, three seeds per mode. Both modes saw the same objects for the same seed.

During training the angle measurement was the true COM direction plus Gaussian noise with $\sigma = 0.008$ rad, the self-estimate error of the motor-current pipeline measured in [4], and $\hat{A}$ was the true amplitude with a uniform $\pm 10\%$ error. Running the full pipeline inside training would have made each episode several times slower; the final evaluation also uses the full pipeline.

3.5 Evaluation protocol

Test objects were drawn with a seed never used in training: 40 in-distribution objects with the noise model, 20 of the same objects with the full motor-current pipeline, and 30 out-of-range objects (heavier, $m \in [0.2, 0.3]$ kg; longer offset, $d \in [0.06, 0.08]$ m; $\mu \in [0.3, 0.9]$; $\mu_t \in [0.004, 0.008]$ m; targets in $[-1.4, -0.2]$ rad), rejection-sampled so that the required grip $mgd/(2\mu_t)$ is at most 25 N, below the 30 N clamp. A separate validation set of 24 objects was used for designing the guard, for the release-rate-cap experiment of Section 4.5, and to pick the object for Fig. 3. We report root-mean-square (RMS) error over held objects, the fraction of objects within 0.05 rad of the target, overshoots ($\theta_{\text{final}} < \theta^* - 0.05$), drops, number of angle measurements and simulated time. Paired comparisons use a Wilcoxon signed-rank test on per-object differences averaged over the three seeds. These tests treat the three trained policies as fixed and ignore seed-to-seed variance, so they describe these policies on these objects rather than the training method; we report them without correction for multiple comparisons. Standard deviations are sample standard deviations over seeds. RMS errors exclude dropped objects; the paired tests on absolute error include them (only the out-of-range set has drops).

Chronology and possible leakage. The guard was not planned in advance. We designed it after the first test-set evaluation of the unguarded policies showed systematic overshoot. Its post-motion part was checked on the validation set before use. Its search-phase bound was not: on the validation set it never changed an outcome, and the one test object it affects most (a model-controller overshoot on a shallow target) was already visible in the first test evaluation. The guarded test results are therefore not a clean held-out test. To address this, after the guard and all code were frozen we drew a further 40 objects with a new seed (the *confirmation set*), ran the model controller and the residual policies with and without the guard once, and report the outcome as it came (Section 4.5).

4 Results

4.1 Residual learning makes the controller faster

Table 1 and Fig. 1 summarize the held-out results. Residual policies roughly halved the number of angle measurements, from 18.0 to 9.3 per pivot, and cut pivot time by 24% ($\Delta t = -0.57$ min, $p = 2e - 12$). With motor-current sensing, where each measurement carries a 10 s sweep, the cut was 37%. Almost all of the saved steps are in the search phase: the model controller took 12.5 steps before the step on which the rod first slipped and 2.5 after it, the residual policies 4.8 and 1.5. (We did not split time by phase; the step counts are the evidence.) Before motion the residual policies lowered the force by 13, 14 and 17% per step (seeds 0–2) instead of the model’s 6%; their release rates in that phase were 0.24, 0.38 and 0.35 N/s against the model’s 0.3 N/s (step sizes and rates measured on the first 10 test objects).

Method	RMS error (rad)	Within 0.05 rad	Overshoots	Measurements	Time (min)
<i>Noise-model sensing, 40 held-out objects</i>					
Model controller [4]	0.038	95%	2.0	18.0	2.42
Model + guard	0.025	95%	2.0	18.0	2.42
Pure RL	0.066 ± 0.002	$76 \pm 5\%$	9.3	12.7	2.01 ± 0.19
Residual RL	0.057 ± 0.037	$88 \pm 7\%$	5.0	9.3	1.85 ± 0.16
Residual RL + guard	0.038 ± 0.029	$93 \pm 5\%$	2.3	10.8	1.97 ± 0.14
<i>Full motor-current sensing pipeline, 20 held-out objects</i>					
Model controller [4]	0.015	100%	0.0	17.8	5.49
Model + guard	0.012	100%	0.0	18.0	5.53
Pure RL	0.063 ± 0.015	$80 \pm 5\%$	3.7	13.0	4.28 ± 0.37
Residual RL	0.049 ± 0.036	$87 \pm 3\%$	2.7	9.3	3.48 ± 0.21
Residual RL + guard	0.019 ± 0.004	$97 \pm 3\%$	0.7	10.4	3.73 ± 0.19

Table 1: Held-out results. Learned methods show the mean $\pm$ standard deviation over three training seeds; overshoots ($\theta_{\text{final}} < \theta^* - 0.05$ rad) and measurements are seed means. Measurements count every angle measurement, including two before the first step. No method dropped an object in these sets. Time is simulated time per pivot; with motor-current sensing each measurement includes a 10 s wrist sweep.

4.2 ... but less accurately

The unguarded residual policies overshoot more often than the model controller (5.0 vs 2.0 of 40 objects, seed mean) and their absolute error rose by $+0.014$ rad ($p = 2e - 04$). The effect was largest for one seed: seeds 0 and 1 reached 92% and 90% of objects within 0.05 rad, seed 2 only 80%. On this test set, all but 2 of the errors larger than 0.05 rad, across all methods and seeds, were overshoots.

Training did not reliably improve the objective it optimized. On the test set the mean return of the residual policies (-1.61) was no better than the model controller’s (-1.62): the time saving was bought with accuracy at roughly the exchange rate set by the reward. Seed 2’s training return fell from -1.45 over its first five iterations to -2.09 over its last five, and it ended below its own starting point, the model controller (test return -1.98). We did not use early stopping or select checkpoints; with six objects per iteration the training signal is noisy (Fig. 2).

4.3 Learning without the model

Pure RL, trained with the same budget, was faster than the model controller but less accurate (76% within 0.05 rad; absolute error $+0.022$ rad higher, $p = 1e - 05$). Against the residual policies it was both slower ($\Delta t = +0.16$ min, $p = 0.003$) and less accurate ($\Delta|e| = +0.008$ rad, $p = 0.010$) with noise-model sensing; with motor-current sensing it was slower ($p = 0.008$) but the accuracy difference was not significant ($p = 0.076$). After its first-slip step the pure policy needed 4.6 further steps against 1.5 for the residual, and about half of its overshoots (9.3 per seed, 4.7 at first slip) happened after first slip, where it has no model to aim with. Fig. 2 shows that the residual policy starts at the model controller’s level while the pure policy starts lower; at this budget the pure policies had not clearly converged, and a larger budget might narrow the gap.

4.4 Where the overshoot comes from: the first slip

We re-ran every test episode with trajectory logging (the re-run reproduced all results exactly). Most overshoots happened on one step, the step on which the rod first slipped: all 2.0 of the model

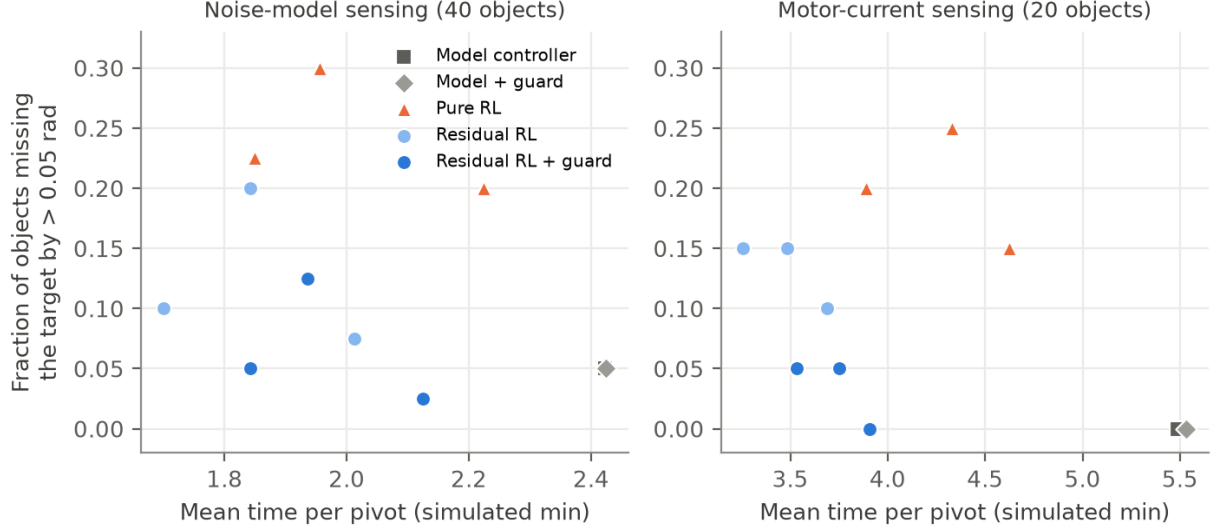


Figure 1: Accuracy against speed. Each learned-policy marker is one training seed. Lower left is better. Residual policies are faster than the model controller; the guard moves them down toward its accuracy.

controller’s, 4.0 of the residual policies’ 5.0, and all 2.3 of the guarded residual’s (seed means). On that step the rod rotated in a single jump with a median of 0.46 rad (model controller 0.39, residual 0.47 rad). Across the 440 first slips (40 objects, each under 11 policies, so not independent) the jump was weakly associated with the size of the force step (Spearman $\rho = 0.35$; partial $\rho = 0.29$ given the rate) and with the release rate ($\rho = 0.31$; partial $\rho = 0.24$ given the step); step and rate were themselves correlated ($\rho = 0.28$). Whether the jump overshoots depends mainly on the target. Pooled over all policies, it overshoot in 51% of episodes with targets shallower than 0.5 rad (only 5 test objects), 5% between 0.5 and 0.8 rad, and 0% deeper.

Part of the jump is expected from the quasi-static model itself. Near the slip threshold the equilibrium $\theta = -\arccos(F/F_h)$ is very steep: a release to 6% below F_h already gives 0.35 rad, 16% below gives 0.57 rad. Because the search does not know F_h , its last step before slip can end anywhere up to one full step below F_h , and the residual policies’ larger steps raise that ceiling. The rest is not explained by the quasi-static model: in 39% of first slips the rod rotated further than $\arccos(1 - \text{step})$, the most the model allows for that step. This excess is most frequent for the slow, small-step model controller (74% of its first slips, against 26% for the residual policies), so it is mostly a static mismatch between the simple model and the simulated soft contact, together with the 0.03 rad threshold used to detect motion, rather than a rate effect.

A rate effect does exist on top of that. Fig. 3 shows one validation object held at $1.08F_h$ and released to $0.80F_h$ at four rates. The quasi-static model predicts -0.644 rad. The rod ended at -0.706 , -0.750 , -0.810 and -0.994 rad for 0.05, 0.15, 0.3 and 0.5 N/s, i.e. 0.062 to 0.350 rad beyond the prediction: a static offset (already 0.062 rad at the slowest rate) plus a swing that grows with rate. This is one object at one release depth; it illustrates the mechanism rather than measuring it in general.

Across policies, release rate alone does not explain which seeds failed. Seed 1 released fastest in the search phase yet was among the most accurate; seed 2, the least accurate, took the largest force steps. Shallow targets are hard for every method because the first slip moves the rod a long way at once.



Figure 2: Training curves for three seeds per method. The return of the current policy on each iteration’s six random objects is noisy because the objects change every iteration. Residual RL starts at the model controller’s level because a zero residual is the model controller.

4.5 A model-based guard

The guard acts on both phases of a pivot. The search-phase bound $F_k \geq F_{k-1} \cos \theta^*$ limits the step on shallow targets, where the first-slip jump is dangerous; for the model controller’s 6% steps it is almost never active. The post-motion clip keeps the residual from aiming more aggressively than the model once the model is informative.

With the guard, overshoots fell to 2.3 of 40 (model controller: 2.0) and 93% of objects were within 0.05 rad, while pivot time stayed 19% lower than the model controller’s ($p = 4e - 10$). Against the model controller with the same search bound, the guarded residual’s absolute error was +0.005 rad higher, not a significant difference ($p = 0.31$), and its RMS error was higher (0.038 vs 0.025 rad). A non-significant difference on 40 objects is not evidence that the two are equally accurate. With motor-current sensing the guarded policies reached 100, 95 and 95% within 0.05 rad by seed (model controller: 100%), 32% faster; against the model controller with the bound the error difference was +0.004 rad ($p = 0.22$). The guard did not rescue seed 2 with noise-model sensing (88% within 0.05 rad): its remaining overshoots are first-slip jumps beyond the quasi-static bound, which the bound by construction cannot prevent. Guarded seeds 0 and 1 also still overshoot one shallow target, at release rates below the model’s, consistent with the static part of the mismatch. The search bound alone improved the model controller’s RMS error from 0.038 to 0.025 rad, mostly by halving one overshoot on a shallow target (from -0.208 to -0.107 rad at a target of -0.335 rad).

Confirmation on fresh objects. On the confirmation set, which played no part in designing anything, the pattern repeated. The model controller reached 95% within 0.05 rad (RMS 0.037 rad, 2.31 min per pivot). The unguarded residual policies were 22% faster but less accurate (88, 90 and 90% within 0.05 rad; absolute error +0.010 rad higher, $p = 0.011$). With the guard they reached 98, 95 and 90% (RMS 0.025, 0.025 and 0.050 rad) while staying 18% faster than the model controller ($p = 7e - 10$); the change in absolute error was +0.002 rad, not significant ($p = 0.075$). Seed 2 remained the weakest.

A guard added after training can hurt. On the validation set we also tried capping only the release rate of the trained residual policies at the model’s 0.3 N/s. Every seed got worse: 50% of

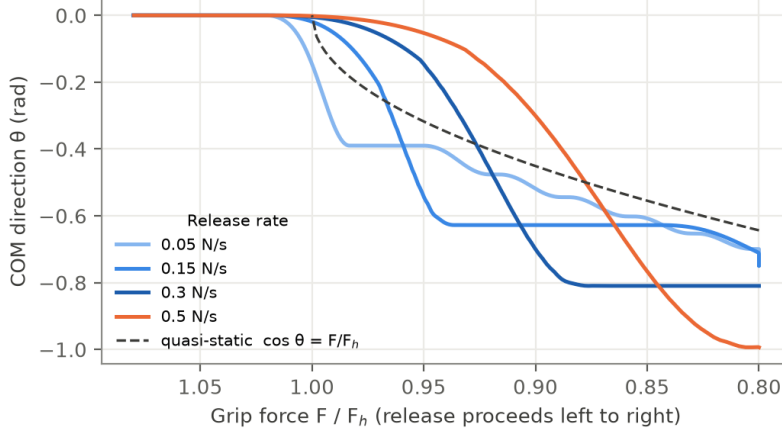


Figure 3: Breakaway dynamics on one object. Grip force is released from $1.08F_h$ to $0.80F_h$ at four rates; the dashed line is the quasi-static equilibrium. The rod overshoots the equilibrium, by more at higher rates.

objects within 0.05 rad against 90% unguarded and 96% for the model controller, with overshoots rising from 2.3 to 12.0 of 24. A likely explanation is that the learned force steps were tuned together with the rates they were trained with; we did not test this directly. Either way, a constraint that a policy will run under should be present during training.

4.6 Out-of-range objects

On heavier, longer objects every method degraded, including the model controller (53% within 0.05 rad, 4.0 drops of 30). The guarded residual was close to it (53%, 4.3 drops; error difference -0.001 rad, $p = 0.81$) and the unguarded residual was at 51%. Pure RL degraded more: 40% within 0.05 rad, RMS error 0.258 against 0.161 rad (error difference +0.055 rad, $p = 0.045$).

5 Discussion and Limitations

The study supports a narrow claim. A learned residual removed most of the wasted search steps of a model-based pivoting controller, and its main failure was concentrated on one identifiable step, the first slip, where the quasi-static model is least reliable. A simple guard tied to the model kept most of the time saving with accuracy close to the model controller’s for two of three seeds with noise-model sensing and all three with motor-current sensing; the seed-pooled error differences were not significant.

The limitations are substantial. Everything is in simulation, and the size of the first-slip jump depends on MuJoCo’s soft-contact parameters; on hardware it could be larger or smaller, and stick-slip with distinct static and kinetic friction, which our contact model does not include, could make it worse. We trained three seeds per method with a small budget (1560 episodes per run), linear policies and no checkpoint selection; one seed got worse than its starting point, and more seeds or a different optimizer could change the ranking between pure and residual policies. The training sensor was a noise model, not the motor-current pipeline. The reward weights were chosen once and not tuned, and on average the residual policies traded accuracy for speed rather than improving the reward. The guard was designed after we saw the unguarded test results (Section 3); the fresh confirmation set supports it, but it is a single set of 40 objects with noise-model sensing. The out-of-range objects are one fixed shift and do not cover changes in contact geometry or friction

model. Finally, one intended observation feature was identically zero, so the policies used seven inputs.

The first-slip finding suggests the next step: estimate F_h more cleverly before the first slip (for example from the probe amplitude and a prior on μ_t), slow the release only near the predicted threshold, and train the residual with the guard in place from the start.

6 Conclusion

We added a learned residual to a physics-based pivoting controller that senses through motor current. The residual roughly halved the number of measurements and cut pivot time by 24–37%, but overshoot targets more often. The overshoots came mostly from the first slip, a single jump that tended to be larger for larger force steps and faster releases, is larger than the quasi-static model predicts, and is harmful on shallow targets. A guard that bounds the search step by the target and keeps the residual more cautious than the model after first slip retained a 19–32% time saving with accuracy not significantly different from the model controller, a result that held on a fresh confirmation set drawn after the guard was fixed. Learning without the model was slower than residual learning at the same budget, and less accurate with noise-model sensing.

Reproducibility. The code for this study (environment, training, evaluation, analysis and figures) will be archived on Zenodo together with this preprint. It builds on the archived code of [4] (<https://doi.org/10.5281/zenodo.23264189>). Every number in this paper is generated from the result files by scripts, except the illustrative quasi-static values $\arccos(0.94)$ and $\arccos(0.84)$ in Section 4.4.

Use of AI tools. An AI assistant (Claude, Anthropic) was used to write and debug code and to draft this paper. The author directed the study, reviewed the code, results and text, and is responsible for the content.

References

- [1] A. Holladay, R. Paolini, and M. T. Mason, “A general framework for open-loop pivoting,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 3675–3681.
- [2] F. E. Viña B., Y. Karayiannidis, C. Smith, and D. Kragic, “Adaptive control for pivoting with visual and tactile feedback,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 399–406.
- [3] J. Toskov, R. Newbury, M. Mukadam, D. Kulić, and A. Cosgun, “In-hand gravitational pivoting using tactile sensing,” in *Proc. 6th Conf. Robot Learning (CoRL)*, PMLR vol. 205, 2023, pp. 2284–2293.
- [4] C. Aidoo, “Gravity pivoting from motor current: Calibration-free angle estimation and the sensing budget it requires,” preprint, engrXiv, 2026. <https://doi.org/10.31224/8492>. Code and data: <https://doi.org/10.5281/zenodo.23264189>
- [5] G. A. Waltersson and Y. Karayiannidis, “Perception, control, and hardware for in-hand slip-aware object manipulation with parallel grippers,” *Int. J. Robotics Research*, vol. 45, no. 10, pp. 1497–1519, 2026.

- [6] Y. Chen, S. Lu, E. Colgate, and K. Lynch, “Robust in-hand manipulation via priors in reinforcement learning and mechanical design,” arXiv:2607.12105, 2026.
- [7] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, “Residual reinforcement learning for robot control,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 6023–6029.
- [8] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, “Residual policy learning,” arXiv:1812.06298, 2018.
- [9] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, “Safe reinforcement learning via shielding,” in *Proc. AAAI Conf. Artificial Intelligence*, 2018, pp. 2669–2678.
- [10] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, “Evolution strategies as a scalable alternative to reinforcement learning,” arXiv:1703.03864, 2017.
- [11] H. Mania, A. Guy, and B. Recht, “Simple random search of static linear policies is competitive for reinforcement learning,” in *Advances in Neural Information Processing Systems 31 (NeurIPS)*, 2018.
- [12] 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.