Vid2Sid: Videos Can Help Close the Sim2Real Gap

Kevin Qiu[*]
University of Warsaw
IDEAS NCBR

Yu Zhang
EPFL

Marek Cygan
University of Warsaw
Nomagic

Josie Hughes
EPFL

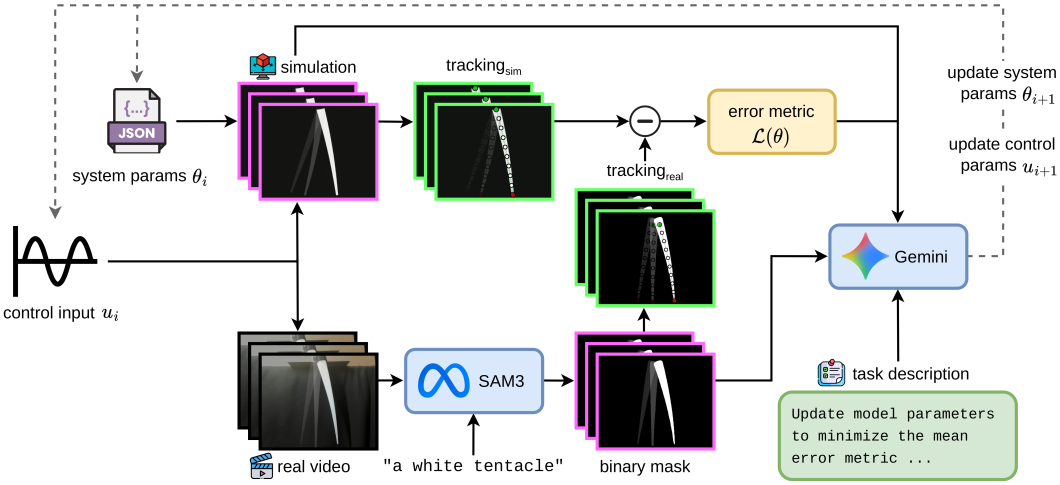


Figure 1: Overview of Vid2Sid. The *perception layer* extracts trajectories from paired sim-real videos, and the *reasoning layer* uses a VLM to diagnose discrepancies and propose physics parameter updates with natural language rationales.

Abstract

Calibrating a robot simulator's physics parameters (friction, damping, material stiffness) to match real hardware is often done by hand or with black-box optimizers that reduce error but cannot explain *which* physical discrepancies drive the error. When sensing is limited to external cameras, the problem is further compounded by perception noise and the absence of direct force or state measurements. We present Vid2Sid, a video-driven system identification pipeline that couples foundation-model perception with a VLM-in-the-loop optimizer that analyzes paired sim-real videos, diagnoses concrete mismatches, and proposes physics parameter updates with natural language rationales. We evaluate our approach on a tendon-actuated finger (rigid-body dynamics in MuJoCo) and a deformable continuum tentacle (soft-body dynamics in PyElastica). On `sim2real` holdout controls unseen during training, Vid2Sid achieves the best average rank across all settings, matching or exceeding black-box optimizers while uniquely providing interpretable reasoning at each iteration. `sim2sim` validation confirms that Vid2Sid recovers ground-truth parameters most accurately (mean relative error under 13% vs. 28–98%), and ablation analysis reveals three calibration regimes. VLM-guided optimization excels when perception is clean and the simulator is expressive, while model-class limitations bound performance in more challenging settings.

1 Introduction

Simulation has become the dominant paradigm for training robot control policies, enabling thousands of trials without wear on physical hardware (Andrychowicz et al., 2020). Yet deploying simulation-trained policies on real robots remains a fundamental challenge (Muratore et al., 2022): when the

---

[*] Corresponding author. Email: `kevinxqiu@gmail.com`

simulator's physics parameters (friction, stiffness, damping) do not match reality, policies that succeed in simulation fail on real hardware. This physics gap blocks transfer for robots ranging from rigid manipulators to soft continuum devices.

Two broad strategies have emerged to close this gap. Domain randomization (Tobin et al., 2017; Peng et al., 2018; Andrychowicz et al., 2020) trains policies robust to parameter variation, sidestepping the need to identify specific physics parameters. While effective in some settings, it sacrifices optimality on any particular configuration, requires conservatively wide parameter ranges that can degrade policy quality, and provides no insight into the actual physical properties of the hardware. The alternative, directly identifying the simulator's physics parameters to build a faithful digital twin, is more fundamental. An accurately identified simulator enables not only robust policy transfer but also downstream debugging, iterative design, and reuse across tasks. But identification is an inherently difficult inverse problem. The mapping from physics parameters to observed motion is nonlinear, different parameter combinations can produce indistinguishable behaviors, and real-world observations are corrupted by perception noise and unmodeled effects. Existing methods offer unsatisfactory trade-offs: manual tuning by domain experts, which is slow and unscalable, or black-box optimization (Bayesian optimization (Calandra et al., 2016), CMA-ES, SimOpt (Chebotar et al., 2019)), which automates parameter search but provides no insight into *which* physical discrepancies drive error (Allevato et al., 2020).

Meanwhile, foundation models have transformed robot perception (SAM3 (Carion et al., 2025) enables markerless video tracking without task-specific training), and VLMs can reason about physical dynamics from video (Wiedemer et al., 2025). Yet no prior work has applied these capabilities to quantitative physics parameter identification, the very task where visual reasoning about dynamics should be most informative. VLM-based robotics has focused on task planning (Ahn et al., 2022), reward design (Ma et al., 2023), and scene-level domain randomization (Yu et al., 2024), leaving quantitative system identification untouched.

We present VID2SID, a video-to-system-identification pipeline that bridges this gap by coupling foundation-model perception with VLM-guided optimization (Figure 1). VID2SID decomposes into two layers. The *perception layer* extracts trajectories from video (SAM3 centerlines for soft robots, marker tracking for rigid robots). The *reasoning layer* uses a VLM to analyze paired sim-real videos, diagnose specific discrepancies (e.g., "the simulated tentacle oscillates too fast"), and propose physics parameter updates with natural language rationales. Unlike black-box methods, VID2SID provides interpretable diagnostics at every iteration and requires no optimizer hyperparameters such as acquisition functions, step sizes, or population sizes.

Our key contributions:

1) **Video as a modality for `sim2real` reasoning.** We validate that VLMs can diagnose physics discrepancies from paired sim-real video across morphologically distinct embodiments with natural language rationales.
2) **Video-driven system identification pipeline.** VID2SID combines foundation model perception with VLM-guided optimization across different simulators without requiring differentiable physics, task-specific training, or manual annotation.
3) **Cross-morphology evaluation.** On a tendon-actuated finger (MuJoCo) and a continuum tentacle (PyElastica), VID2SID achieves the best average rank on holdout controls and recovers ground-truth parameters most accurately in `sim2sim` (under 13% relative error vs. 28–98% for baselines).

2 VID2SID

VID2SID identifies a simulator's physics parameters through an iterative closed-loop process. At each iteration, the same control signal is sent to both the simulator and the real robot, and video of both executions is recorded. A perception system extracts structured trajectories from the videos, and a VLM analyzes the paired recordings to diagnose physical discrepancies and propose parameter updates. This loop repeats until simulated behavior converges to real behavior.

Concretely, the pipeline decomposes into two layers. The **perception layer** extracts trajectory representations from raw video using a task-appropriate perception function. The **reasoning layer** uses a VLM to diagnose sim-real discrepancies and propose parameter updates with natural language rationales.

Problem formulation. Let f_θ denote a simulator parameterized by physics parameters $\theta \in \Theta$, where Θ defines valid bounds. Given a control signal u applied to both the simulator and the real robot, let $\tau_{\text{sim}}(\theta) = g(v_{\text{sim}}(\theta))$ and $\tau_{\text{real}} = g(v_{\text{real}})$ be trajectories extracted from the simulated and real videos, respectively, where g is a perception function (SAM3 centerline extraction or marker tracking). The objective is to find the optimal physics parameters θ^* that minimize the alignment error between sim and real:

$$\theta^* = \arg\min_{\theta \in \Theta} \; \mathcal{L}(\tau_{\text{sim}}(\theta),\; \tau_{\text{real}}) \tag{1}$$

where $\mathcal{L}$ is the mean absolute error between temporally aligned trajectories (Section 2.4). VID2SID solves this problem by using a VLM to propose successive parameter updates based on paired sim-real videos, error metrics, and optimization history.

2.1 SIMULATION FRAMEWORK

VID2SID is simulator-agnostic, requiring only that the simulator produce visual output. We evaluate on two physical platforms, a tendon-actuated finger and a soft continuum tentacle (Figure 2), using two simulation backends representing qualitatively different dynamics regimes.

2.1.1 MUJOCO (RIGID-BODY DYNAMICS)

The finger is a three-link kinematic chain modeled in MuJoCo. Joints use PD position control with fixed gain (k_p=1000 N/rad) and tunable damping k_d.

Tunable parameters: joint friction, viscous damping, rotor inertia, and link density (parameter bounds are provided in Appendix C).

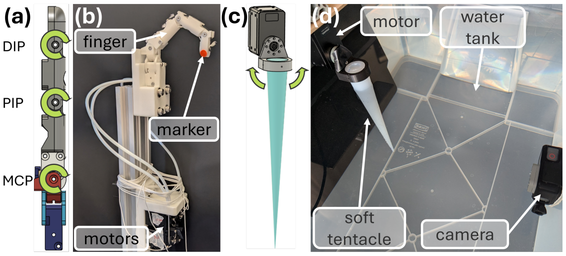


Figure 2: Experimental platforms. (a) CAD model of tendon-driven finger. (b) Physical finger with tracked marker. (c) CAD model of soft tentacle. (d) Underwater tentacle setup.

2.1.2 PYELASTICA (CONTINUUM DYNAMICS)

The tentacle is modeled as a Cosserat rod (Gazzola et al., 2018) using PyElastica (Tekinalp et al., 2024). This framework is well-suited for continuum and soft robots. The rod is subject to elastic restoring forces parameterized by Young's modulus E and Poisson ratio ν, a fixed-base boundary condition with prescribed oscillatory rotation, linear velocity damping with coefficient γ, and external forces including gravity and (for underwater settings) hydrodynamic interaction forces.

Tunable body parameters (used for in-air calibration): Young's modulus E, rod density ρ, Poisson ratio ν, and damping coefficient γ. These are four material parameters with bounds in Appendix C.

Tunable environment parameters (used for underwater calibration): fluid density ρ_f, perpendicular drag coefficient $C_\perp$, and tangential drag coefficient $C_\|$, modeling anisotropic hydrodynamic drag (Gazzola et al., 2018) (bounds in Appendix C). In the **in-air** setting, we tune body parameters only. In the **underwater stress test**, we freeze body parameters and tune only environment parameters (Section 3.4). In both settings, the VLM can also adjust control parameters (motor amplitudes) to elicit more informative motions (Section 2.5.2).

2.2 CONTROL AND CAPTURE

Both platforms use sinusoidal actuation $u(t) = A\sin(2\pi f t + \phi)$, which provides repeatable motion that reveals underlying physical parameters through differences in amplitude, phase, and settling time. The same control signal is sent to both simulator and real hardware. A webcam captures video at 30 fps. Each trial records 10 s of motion after a 2 s hold. Full control parameterization and hardware specifications are in Appendices C and A.

2.3 TRAJECTORY EXTRACTION FROM VIDEO

2.3.1 TENTACLE (SAM3 SEGMENTATION + CENTERLINE EXTRACTION)

We use SAM3 (Carion et al., 2025) for real tentacle segmentation, prompting with a text description on the first frame and propagating masks through the video (97.2% success without manual prompts;

Appendix G). From each binary mask, we extract a 10-point centerline via medial axis and arc-length resampling (Figure 3), producing per-frame centerlines $\{(x_i^t, y_i^t)\}_{i=1}^{10}$.

2.3.2 FINGER (TIP EXTRACTION)

For the finger, we track the 2D fingertip marker via HSV thresholding and centroid extraction, yielding $(x_{\text{tip}}^t, y_{\text{tip}}^t)$ per frame.

2.4 ERROR METRICS

Sim and real trajectories are temporally aligned via cross-correlation (max 1 s lag) and compared in 2D pixel coordinates. We use mean absolute error (MAE), computed point-wise across $N{=}10$ centerline points for the tentacle, and over the tip trajectory for the finger. We ignore the initial 5 s to exclude transients.

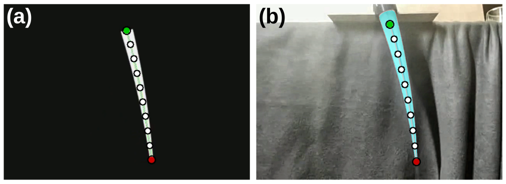


Figure 3: Centerline extraction. (a) Simulated centerline (10 points). (b) Real frame with SAM3 mask and extracted centerline. The shared representation enables direct point-wise error computation.

2.5 VLM-DRIVEN PARAMETER TUNING

2.5.1 PROMPT DESIGN AND ITERATIVE LOOP

The VLM receives a multimodal prompt containing sim and real videos, current parameters θ with bounds, error metrics, and iteration history for in-context learning (Brown et al., 2020). An optional chain-of-thought instruction (Wei et al., 2022) requests explicit reasoning before the final answer. The VLM returns recommended parameter values, a confidence score (0–1), and a natural language rationale. Proposed values are clamped to valid bounds. We use Gemini 2.5 Pro (Comanici et al., 2025) with default decoding (temperature 1.0, top-P 0.95; Appendix F.3). Algorithm 1 summarizes the loop: we run $K{=}10$ iterations and track the best parameters seen to handle VLM stochasticity.

2.5.2 ACTIVE LEARNING OVER CONTROL INPUTS

The VLM can also recommend changes to *control parameters* (motor amplitudes) to elicit more informative motions. Minimum amplitude bounds prevent degenerate low-motion solutions, and holdout evaluation uses *fixed* control profiles (Section 3.6).

Algorithm 1 VID2SID calibration loop.

```
Require: $\theta_0, u_0$, bounds $\Theta, \mathcal{U}$, iters $K$
$\theta^* \leftarrow \theta_0$, $e^* \leftarrow \infty$, $H \leftarrow \emptyset$
for $k = 1, \ldots, K$ do
    $v_s \leftarrow \mathrm{Sim}(\theta, u)$;  $v_r \leftarrow \mathrm{Real}(u)$
    $\tau_s, \tau_r \leftarrow \mathrm{Align}(\mathrm{Traj}(v_s, v_r))$
    $e \leftarrow \mathrm{MAE}(\tau_s, \tau_r)$
    if $e < e^*$ then
        $e^* \leftarrow e$;  $\theta^* \leftarrow \theta$
    end if
    $H \leftarrow H \cup \{(\theta, u, e)\}$
    $\theta, u \leftarrow \mathrm{Clamp}(\mathrm{VLM}(v_s, v_r, H))$
end for
return $\theta^*$
```

3 EXPERIMENTS

We evaluate VID2SID on two platforms, an articulated finger and a continuum tentacle, across three settings: `sim2sim` validation, `sim2real` calibration in air, and an underwater tentacle stress test where only environment parameters are tuned (shared-body protocol). Across all methods we enforce a fixed training budget of 10 iterations per seed.

3.1 EXPERIMENTAL SETUP

3.1.1 HARDWARE CONFIGURATION

Both platforms are introduced in Section 2.1 (Figure 2). The finger is 3D-printed from PLA and actuated by two Dynamixel motors via Bowden cables. The tentacle is mold-cast from DragonSkin 10 elastomer and driven by a single Dynamixel motor. For the underwater stress test, the tentacle is submerged and recorded with a high-frame-rate action camera (Appendix A).

3.1.2 EVALUATION PROTOCOL

All methods receive 10 iterations per seed (CMA-ES evaluates a full population per iteration, giving it more total function evaluations). For each seed, we select the iteration with lowest training error and evaluate holdout MAE on 4 unseen control profiles (H1–H4), each repeated 3 times on hardware. For the tentacle, holdouts vary both amplitude and frequency, while training holds frequency fixed; this tests whether identified parameters transfer to unseen actuation regimes, not just unseen amplitudes. We report mean ± std over N=3 training seeds (12 datapoints per seed: 4 holdouts $\times$ 3 repeats).

3.2 `sim2sim` CALIBRATION

Before deploying on real hardware, we validate in a controlled `sim2sim` setting where both "real" and "sim" are simulations with different parameter values. A "ground truth" simulation is created with fixed parameters. Candidate simulations start from random initializations and are optimized under the same budget. `sim2sim` provides an upper bound on achievable performance by eliminating camera noise, segmentation errors, and unmodeled hardware effects.

3.3 `sim2real` CALIBRATION

For both platforms, initial parameters are randomly sampled from bounds for each training seed. All methods receive identical initializations for fair comparison. We report mean absolute 2D tip error (pixels) for the finger and mean absolute centerline error (pixels) for the tentacle, using the perception and metrics described in Section 2.

3.4 STRESS TEST: UNDERWATER TENTACLE

To probe limits when model misspecification dominates, all methods share body parameters from VID2SID's in-air calibration and tune only three hydrodynamic environment parameters. This protocol favors baselines (which receive body parameters they did not discover) and isolates model class from optimizer quality. Any residual sim-real gap is attributable to unmodeled hydrodynamic effects (turbulence, vortex shedding, buoyancy) rather than optimizer choice.

3.5 BASELINE COMPARISONS

We compare VID2SID against five black-box optimization baselines, all given the same 10-iteration budget and parameter bounds:

- **Random:** Uniform random sampling within bounds (lower bound baseline).
- **Nelder-Mead:** Derivative-free simplex method (Singer & Nelder, 2009) (SciPy (Virtanen et al., 2020)).
- **Golden-CD:** Sequential 1D golden-section search along each parameter axis.
- **Bayesian Optimization (BO):** Gaussian Process surrogate with Expected Improvement acquisition (Jones et al., 1998) (scikit-optimize, default hyperparameters).
- **CMA-ES:** Covariance Matrix Adaptation Evolution Strategy (Hansen, 2016) with population size $\lambda = 4 + 3\log d$.

All methods optimize the same parameter set θ under identical bounds (Section 2.1). Code and configurations will be released upon publication.

3.6 ABLATION STUDY

We conduct ablation studies on the VID2SID pipeline to isolate individual design choices. Each ablation modifies exactly one factor relative to the full method:

- *w/o Video* (text-only): Video media removed from the VLM prompt. Only scalar error metrics, parameter values, and bounds are provided. All other factors (history, CoT, control tuning) remain unchanged.
- *w/o Chain-of-Thought*: The "*Let's think step by step*" instruction (Wei et al., 2022) is removed from the system prompt. Video and all other factors remain.
- *w/o History* (no in-context learning): Previous iterations' parameter-error pairs are omitted from the prompt (Brown et al., 2020). The VLM sees only the current state.
- *w/o Active Learning* (fixed control): Control parameters are locked to training values. The VLM can only recommend physics parameter changes.

Table 1: `sim2real` holdout generalization across experimental settings. We report mean ± std over $N = 3$ training seeds, with the best single-seed result in parentheses. Each seed value is the mean over 4 holdouts × 3 hardware repeats (12 datapoints). Best mean in **bold**, second-best underlined. The water column uses a shared-body protocol (environment parameters only) and is included as a stress test.

Method	**Finger (px)** ↓	**Tentacle–Air (px)** ↓	**Tentacle–Water (px)** ↓[†]	**Avg. Rank** ↓
Random	27.8 ± 17.2 (12.1)	54.3 ± 1.9 (51.6)	76.1 ± 3.8 (71.2)	4.3
Nelder-Mead	17.2 ± 3.5 (14.2)	57.3 ± 4.2 (54.3)	80.7 ± 2.9 (77.3)	5.3
Golden-CD	12.1 ± 0.2 (11.9)	53.4 ± 4.8 (49.5)	77.8 ± 3.7 (73.2)	3.3
BO	16.1 ± 2.4 (14.4)	62.4 ± 10.4 (54.5)	**71.7** ± **0.1** (71.5)	3.7
CMA-ES	15.3 ± 4.9 (11.8)	**52.1** ± **2.9** (49.9)	77.6 ± 3.4 (72.9)	2.7
VID2SID (Ours)	**10.9** ± **6.3** **(4.9)**	53.0 ± 5.3 **(48.8)**	73.3 ± 3.1 **(71.1)**	**1.7**

[†] Stress test: environment-only tuning with shared body parameters. Narrow spread indicates model-class bottleneck.

Ablation results are presented in Section 4.3.

4 RESULTS

We report holdout generalization as the primary success metric (evaluation protocol in Section 3.1.2). Concretely, we evaluate each method's best training iteration on unseen control profiles (H1–H4). Unless stated otherwise, we report mean ± std over N=3 training seeds. Each seed value aggregates 4 holdouts × 3 hardware repeats (12 datapoints). Reported uncertainty therefore reflects algorithmic sensitivity, not per-trial hardware noise.

4.1 SIM2REAL HOLDOUT GENERALIZATION

Table 1 summarizes `sim2real` holdout generalization across all three experimental settings. The rightmost column reports the average rank. For each setting, methods are ranked 1–6 by mean holdout error, and the rank is averaged across all three settings.

Finger. VID2SID achieves the lowest mean holdout error (10.9 px), a 10% reduction vs. the next-best baseline Golden-CD (12.1 px).

Tentacle (air). Multiple methods achieve similar holdout performance. VID2SID (53.0 px) is within 2% of the best baseline CMA-ES (52.1 px), while providing interpretable rationales that aid debugging (see Section 5).

Tentacle (water). Under the shared-body protocol, all methods start from body parameters discovered by VID2SID and tune only environment parameters. All methods cluster between 71.7–80.7 px with overlapping uncertainty, despite baselines benefiting from body parameters discovered by VID2SID. The narrow spread (<10 px) indicates that *optimizer choice matters less than model class*. The simplified drag model cannot fully capture fluid-structure interaction, so tuning saturates regardless of optimizer.

Per-seed analysis. VID2SID achieves the best single-seed result across all three domains (4.9 px finger, 48.8 px tentacle, 71.1 px water; Table 1 parenthesized values), showing that when VLM reasoning "gets it right," it outperforms all baselines. However, VID2SID exhibits higher cross-seed variance (std=6.3 px on finger), whereas Golden-CD shows remarkable consistency (std=0.2 px). VID2SID's stochasticity arises from VLM sampling (temperature 1.0), which enables bold cross-parameter corrections but also produces occasional poor seeds (e.g., Seed 3 at 19.5 px; Appendix H). Golden-CD's low variance stems from its deterministic coordinate-descent strategy, which finds good local optima but cannot make large joint parameter changes. The gap between VID2SID's best seed (4.9 px) and Golden-CD's best (11.9 px) suggests that reducing VLM variance through ensembling or temperature annealing could yield further gains.

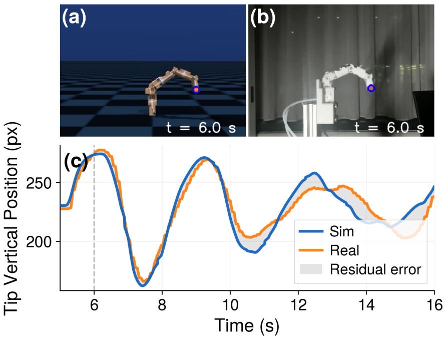


Figure 4: Finger qualitative alignment after VID2SID calibration. (a) Sim finger at t=6 s. (b) Real frame with tracked marker. (c) Calibrated sim (blue) tracks real motion (orange).

4.2 TRAINING CONVERGENCE

Training MAE for VID2SID plateaus within 5 iterations on both platforms, matching black-box baselines (Appendix H). Note that objective-space convergence does not guarantee parameter convergence. We verify that parameters themselves converge toward ground truth in `sim2sim` (Figure 6).

To illustrate VLM reasoning, consider a representative finger iteration. The VLM diagnoses: "*The simulation finger is significantly slower and more heavily damped than the real hardware.*" It recommends reducing friction (143→25) and damping (190→30), producing a 59% error reduction (50.1→20.7 px) in a single step. This diagnostic chain, connecting visual discrepancies to targeted parameter corrections, is the mechanism underlying VID2SID's convergence, one that black-box optimizers cannot replicate. In early iterations, bold corrections drive rapid error reduction; near convergence, recommendations become more conservative as remaining discrepancies grow subtler. We track the best parameters across all iterations (Algorithm 1) to benefit from early breakthroughs without degradation from later missteps (Appendix F.5). Figure 4 confirms qualitative alignment. The calibrated simulation closely tracks the real finger's motion. Notably, the residual varies with the motion cycle, suggesting that remaining error stems from unmodeled nonlinearities (e.g., cable elasticity, stick-slip friction) rather than a simple parameter bias.

4.3 ABLATION STUDY

Figure 5 summarizes ablation results on both platforms. Each ablation modifies exactly one factor relative to the full VID2SID configuration (see Section 3.6 for definitions).

Finger. Removing video input increases holdout error by 66%, confirming that VLMs extract physical signal from video beyond what scalar metrics alone provide. CoT provides a modest benefit (+13%). Removing history (−10%) slightly helps, suggesting that history can anchor the VLM to early iterations, encouraging incremental refinements rather than bold exploration. Removing active learning (−35%) also helps, likely because varying the control signal between iterations introduces a confound, as the VLM must simultaneously adapt to changing observations.

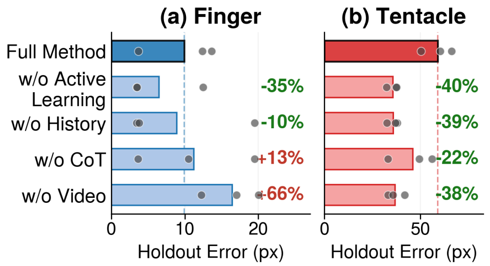


Figure 5: Ablation study on (a) finger and (b) tentacle. Each bar removes one component. Dots show individual seeds. Removing video increases error by 66% on the finger but decreases it by 38% on the tentacle, indicating domain-dependent prompt design.

Tentacle. Removing video, history, or active learning each *reduces* error by 38–40%. We attribute this to a difference in *parameter observability*: finger parameters (damping, friction) produce visually distinctive signatures (oscillation speed, settling time, overshoot), while tentacle material properties (Young's modulus, Poisson ratio) produce subtler deformation effects obscured by segmentation noise (∼3–5 px jitter) and chaotic tip dynamics. In this regime, video and history introduce more noise than signal. These results suggest prompt design is domain-dependent, so practitioners should run a small ablation when deploying VID2SID on new platforms (Section 5).

4.4 `SIM2SIM` VALIDATION

In a controlled `sim2sim` setting (Table 2), VID2SID achieves the best holdout error on both finger (7.13 mm, 27% over BO) and tentacle (8.91 px, 11% over BO/CMA-ES), and recovers parameters closest to ground truth (8.7% relative error on finger, 12.4% on tentacle vs. 28–98% for baselines; Appendix H.7). Figure 6 shows VID2SID steadily converges toward true parameters while baselines oscillate. This confirms that residual `sim2real` gaps are attributable to model-class limitations rather than optimizer deficiency.

Table 2: `sim2sim` holdout generalization (mean ± std, N=12). Best in **bold**.

Method	Finger (mm)	Tentacle (px)
Random	15.67 ± 13.80	10.63 ± 9.59
Nelder-Mead	46.41 ± 8.77	11.65 ± 11.05
Golden-CD	16.36 ± 17.79	11.44 ± 12.44
BO	9.70 ± 7.49	10.02 ± 11.03
CMA-ES	23.77 ± 19.23	10.02 ± 7.15
VID2SID	**7.13 ± 2.56**	**8.91 ± 9.52**

5 DISCUSSION

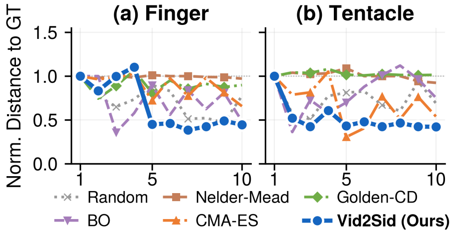


Figure 6: Normalized distance to ground truth (`sim2sim`, best seed). Vid2Sid converges toward true parameters; baselines oscillate.

Our results reveal three calibration regimes. (1) Under clean perception with an expressive simulator (finger), VLM video reasoning drives strong gains (10% holdout improvement, 66% ablation benefit from video). (2) Under noisy perception (tentacle in air), Vid2Sid is competitive but text-only mode is preferable. (3) When model misspecification dominates (water), all optimizers converge to similar error. **Guideline:** deploy with video when perception is clean; use text-only or black-box methods under noisy segmentation. Beyond accuracy, interpretability aids debugging by surfacing falsifiable hypotheses (Appendix F.5), and confidence scores are well-calibrated (89% success at $\geq$0.9; Appendix F.4).

Limitations. Failure modes include segmentation failure (mitigated by point prompts), timing drift (cross-correlation alignment), degenerate control updates (minimum amplitude bounds), and invalid VLM recommendations (parameter clamping). Vid2Sid adds $\sim$11 s of VLM inference per iteration ($\sim$28% overhead, \$0.14 per 10-iteration run; Appendix A.4). Our pipeline assumes a single fixed camera with a static background and does not handle occlusions or multi-robot scenarios. VLM recommendations are not guaranteed globally optimal. We observed occasional convergence to local minima that black-box methods avoided, and extremely poor initializations (e.g., Young's modulus off by $100\times$) caused all methods to fail. We evaluated only Gemini 2.5 Pro. Stronger VLMs may extend benefits to noisier domains (Appendices I, J).

6 Related Work

`sim2real` transfer and system identification. Domain randomization (Tobin et al., 2017; Peng et al., 2018; Andrychowicz et al., 2020) trains robust policies but does not identify parameters. Iterative methods (Chebotar et al., 2019; Du et al., 2021; Ramos et al., 2019; Allevato et al., 2020), Bayesian optimization (Calandra et al., 2016), and differentiable simulation (Jatavallabhula et al., 2021; Dubied et al., 2022) automate calibration but require either differentiable physics or opaque objective functions. Differentiable approaches are powerful when analytical gradients exist, but many simulators (including PyElastica's Cosserat rod solver) are non-differentiable or require custom adjoint implementations. VLM-guided optimization sidesteps these requirements by operating on rendered video, making it simulator-agnostic (George Thuruthel et al., 2018).

Foundation models for perception. SAM3 (Carion et al., 2025) extends zero-shot segmentation (Kirillov et al., 2023) to video, enabling markerless tracking without task-specific training, which is critical for soft robots where appearance varies widely.

VLMs for physical reasoning. LLMs and VLMs have been applied to task planning (Ahn et al., 2022), visuomotor control (Zitkovich et al., 2023; Driess et al., 2023), reward design (Ma et al., 2023; Patel et al., 2025), robot co-design (Qiu et al., 2024; Qiu & Cygan, 2025), and domain randomization (Yu et al., 2024). Most closely related, Wiedemer et al. (2025) showed video models can reason about physical dynamics. Vid2Sid operationalizes this for quantitative system identification, directly mapping visual observations to numerical parameter updates.

7 Conclusion

We introduce Vid2Sid, a camera-only `sim2real` calibration pipeline that turns sim-real videos into physics parameter updates via foundation-model tracking and VLM reasoning. Across a tendon-driven finger and a soft tentacle, Vid2Sid matches or outperforms black-box baselines on holdout generalization within 10 iterations, requires no optimizer hyperparameters, and uniquely provides natural-language rationales that make calibration *debuggable*. Our experiments reveal three calibration regimes: VLM reasoning excels when perception is clean and the simulator is expressive (finger), provides interpretability benefits when perception is noisy (tentacle in air), and performs comparably to all methods when model misspecification dominates (tentacle in water). Future work includes distilling perception and VLM inference for real-time calibration, extending to multi-view settings with occlusions and dynamic backgrounds, and fine-tuning foundation models for physics reasoning to reduce domain-dependent prompt sensitivity. We will release code and experiment configurations.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Centre, Poland, under PRELUDIUM Grant No. 2024/53/N/ST6/03370 and Sonata Bis Grant No. 2024/54/E/ST6/00388.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gober, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pp. 287–318. PMLR, 2022.

Adam Allevato, Elaine Schaertl Short, Mitch Pryor, and Andrea Thomaz. Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer. In *Conference on Robot Learning*, pp. 445–455. PMLR, 2020.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Artificial Analysis. Gemini 2.5 pro: Intelligence, performance & price analysis. `https://artificialanalysis.ai/models/gemini-2-5-pro`, 2026. Accessed: 2026-01-28.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76:5–23, 2016.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2025. URL `https://arxiv.org/abs/2511.16719`.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979. IEEE, 2019.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023.

Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1290–1296. IEEE, 2021.

Mathieu Dubied, Mike Yan Michelis, Andrew Spielberg, and Robert Kevin Katzschmann. Sim-to-real for soft robots using differentiable fem: Recipes for meshing, damping, and actuation. *IEEE Robotics and Automation Letters*, 7(2):5015–5022, 2022.

Mattia Gazzola, Lyndon Duber, Anthony McCormick, and L Mahadevan. Forward and inverse problems in the mechanics of soft filaments. *Royal Society Open Science*, 5(6):171628, 2018.

Thomas George Thuruthel, Yasmin Ansari, Egidio Falotico, and Cecilia Laschi. Control strategies for soft robotic manipulators: A survey. *Soft Robotics*, 5(2):149–163, 2018.

Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. *arXiv preprint arXiv:2104.02646*, 2021.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

K-Scale Labs. urdf2mjcf: Convert urdf files to mujoco mjcf. `https://github.com/kscalelabs/urdf2mjcf`, 2025. GitHub repository.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, 2023.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

Fabio Muratore, Fabio Ramos, Greg Turk, Wenhao Yu, Michael Gienger, and Jan Peters. Robot learning from randomized simulations: A review. *Frontiers in Robotics and AI*, 9:799893, 2022.

Shivansh Patel, Xinchen Yin, Wenlong Huang, Shubham Garg, Hooshang Nayyeri, Li Fei-Fei, Svetlana Lazebnik, and Yunzhu Li. A real-to-sim-to-real approach to robotic manipulation with vlm-generated iterative keypoint rewards. *arXiv preprint arXiv:2502.08643*, 2025.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.

Kevin Qiu and Marek Cygan. Debate2create: Robot co-design via large language model debates. *arXiv preprint arXiv:2510.25850*, 2025.

Kevin Qiu, Władysław Pałucki, Krzysztof Ciebiera, Paweł Fijałkowski, Marek Cygan, and Łukasz Kuciński. Robomorph: Evolving robot morphology using large language models. *arXiv preprint arXiv:2407.08626*, 2024.

Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019.

ROS-Industrial Consortium. sw_urdf_exporter: Solidworks to urdf exporter. `https://wiki.ros.org/sw_urdf_exporter`, 2023. ROS Wiki. Accessed: 2026-01-28.

Saša Singer and John Nelder. Nelder-mead algorithm. *Scholarpedia*, 4(7):2928, 2009.

Arman Tekinalp, Seung Hyun Kim, Yashraj Bhosale, Tejaswin Parthasarathy, Noel Naughton, Ali Albazroun, Rahul Joon, Songyuan Cui, Ilia Nasiriziba, Maximilian Stölzle, Chia-Hsien (Cathy) Shih, and Mattia Gazzola. Gazzolalab/pyelastica, 2024. URL `https://doi.org/10.5281/zenodo.7658871`.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *arXiv preprint arXiv:2509.20328*, 2025.

Albert Yu, Adeline Foote, Raymond Mooney, and Roberto Martín-Martín. Natural language can help bridge the sim2real gap. In *Robotics: Science and Systems (RSS)*, 2024.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

Table of Contents

A EXPERIMENTAL SETUP

A.1 FINGER PLATFORM

The finger is 3D-printed from PLA and mounted on aluminum extrusion. Two Dynamixel XC330-T288-T motors drive it through Bowden cables. The MCP joint is antagonistically driven for flexion and extension, while the PIP and DIP joints are coupled and returned by an elastic string. A visual marker on the fingertip enables tracking. Video is captured with a Logitech BRIO at 1920×1080 and 30 fps.

A.2 TENTACLE PLATFORM

The tentacle is mold-cast from DragonSkin 10 Slow silicone and attached to a Dynamixel XW430-T200R motor. The entire assembly is fixed to the side of a water tank, with the tentacle submerged for underwater experiments. For in-air experiments, video is captured with a Logitech BRIO at 640×480 and 25 fps. For underwater experiments, video is captured with a GoPro Hero 6 at 1920×1080 and 120 fps (Superview FOV).

A.3 COMPUTE RESOURCES

Table 3 lists the hardware and software used for all experiments.

Table 3: Compute resources.

Component	**Specification**
CPU	Intel Core i7-10700K
RAM	32 GB DDR4
GPU (SAM3)	NVIDIA A100 40GB
VLM	Google Gemini 2.5 Pro
OS	Ubuntu 22.04 LTS

A.4 WALL-CLOCK TIMING

Table 4 breaks down per-iteration wall-clock time for the tentacle platform. Hardware capture, simulation, and SAM3 segmentation are constant across all methods; the only variable is the optimizer step. VID2SID adds ∼11 s of VLM API latency per iteration, while black-box optimizers finish in under 1 s.

Table 4: Per-iteration wall-clock time (tentacle platform). Shared components (hardware capture: 15 s, simulation: 8 s, SAM3: 12 s) total 35 s for all methods. VID2SID adds 11 s VLM inference (28% overhead), keeping total calibration under 10 minutes.

	VID2SID	**Baselines**
Shared components	35 s	35 s
Optimizer/VLM	11 s	<1 s
Total per iteration	**46 s**	**36 s**
10-iteration calibration	7.7 min	6.0 min

The 28% per-iteration overhead translates to ∼1.7 minutes additional wall-clock time for a full 10-iteration calibration run. This overhead is offset by eliminating the hyperparameter tuning that black-box methods typically require offline.

B SIMULATION MODEL DETAILS

B.1 FINGER MODEL (MUJOCO)

The MuJoCo finger model was generated from a SolidWorks CAD assembly using the following pipeline:

1) Design finger assembly in SolidWorks with proper joint definitions

Table 5: Parameter bounds for the finger platform.

Parameter	**Min**	**Max**	**Nominal**	**Description**
Friction loss	0	150	50	Joint dry friction torque (Nm)
Damping	10	200	100	Velocity-dependent viscous damping
Armature	0.1	5.0	1.0	Rotor inertia added to each joint
Link density	1.0	20.0	5.0	Density of body geometries (kg/m^3)
MCP amplitude	0	60	10	MCP joint oscillation amplitude (deg)
PIP amplitude	0	60	10	PIP/DIP joint oscillation amplitude (deg)

Table 6: Parameter bounds for the tentacle platform (in-air calibration).

Parameter	**Min**	**Max**	**Nominal**	**Description**
Body parameters				
Young’s modulus	5×10^3	5×10^6	2×10^5	Rod stiffness (Pa)
Rod density	500	5000	1050	Material density (kg/m^3)
Poisson’s ratio	0.2	0.5	0.5	Lateral-to-axial strain ratio
Damping constant	0	100	3.0	Linear (Rayleigh) damping coefficient
Environment parameters				
Perpendicular drag	0	50	1.0	Perpendicular fluid drag coefficient
Tangential drag	0	50	0.1	Tangential fluid drag coefficient
Control parameters				
Motor amplitude	0.2	1.0	1.0	Oscillation amplitude (rad), tuned via active learning
Motor frequency	–	–	0.15	Fixed during training; holdouts use 0.15 and 0.5 Hz

2) Export to URDF using **sw_urdf_exporter** (ROS-Industrial Consortium, 2023), a ROS plugin that converts SolidWorks assemblies to URDF format with automatic mesh export
3) Convert URDF to MJCF (MuJoCo XML) using the **urdf2mjcf** library (K-Scale Labs, 2025)
4) Manually tune default physics parameters (damping, friction) to approximate real behavior

The baseline profile uses hand-tuned default values chosen to produce qualitatively similar motion. For experiments, initial parameters are randomly sampled from bounds for each training seed (see Section 3.3).

B.2 TENTACLE MODEL (PYELASTICA)

The PyElastica tentacle model uses a Cosserat rod formulation with 60 discretization elements along the rod length. Actuation is applied as a muscle torque at the base element. The model includes perpendicular and tangential fluid drag forces and Rayleigh damping for numerical stability.

The baseline profile uses material properties estimated from manufacturer datasheets for DragonSkin 10 Slow silicone (Shore 10A hardness, $\rho \approx 1050$ kg/m^3, $E \approx 0.2$ MPa). For experiments, initial parameters are randomly sampled from bounds for each training seed.

C PARAMETER BOUNDS

C.1 FINGER PLATFORM (MUJOCO)

Table 5 lists the 6 parameters tuned for the finger, spanning physics properties (friction, damping, inertia, density) and control inputs (joint amplitudes). Nominal values reflect hand-tuned baseline estimates; experiments initialize by sampling uniformly within bounds.

C.2 TENTACLE PLATFORM (PYELASTICA)

Table 6 shows the body, environment, and control parameter bounds for the tentacle platform. Nominal values are estimated from manufacturer datasheets; experiments initialize by sampling uniformly within bounds. During in-air calibration, only body parameters are tuned; environment parameters become active in the underwater setting. Table 7 lists the environment bounds used for the underwater stress test, with body parameters frozen from the in-air experiment.

Table 7: Hydrodynamic environment parameter bounds for the underwater tentacle stress test. Body parameters are frozen from the in-air experiment.

Parameter	**Min**	**Max**	**Nominal**	**Description**
Fluid density	0	2000	1000	Surrounding fluid density (kg/m^3)
Perpendicular drag	0	100	10	Perpendicular fluid drag coefficient
Tangential drag	0	100	10	Tangential fluid drag coefficient

D EVALUATION PROTOCOL

Our evaluation captures two distinct sources of variance:

Algorithmic variance (training seeds). We run each method with 3 different random seeds. The seed controls initial parameter sampling for all methods, the stochastic optimizer state for CMA-ES and BO, and the VLM sampling for VID2SID.

Hardware variance (execution repeatability). For each trained parameter configuration, we execute 3 independent hardware trials to capture motor variability, material hysteresis, temperature effects, and camera timing jitter.

Reporting convention. Main results (Table 1) report mean ± std over the $N = 3$ *training seeds*. For each training seed, the holdout error is the mean over 4 holdouts × 3 hardware repeats (12 datapoints), giving 36 holdout datapoints per method and domain. The reported uncertainty therefore reflects sensitivity to random initialization and optimizer stochasticity, not per-trial hardware noise. When needed, we additionally report raw hardware-repeat variability in supplementary analyses.

Isolating effects. To isolate algorithmic vs. hardware effects, we select the best-performing iteration from each training seed independently, then evaluate each on all 4 holdout controls with 3 hardware repeats. This ensures that hardware variance does not contaminate training seed selection.

Holdout profiles (H1–H4). Table 8 specifies the control parameters for each holdout profile. Both platforms use a 2×2 factorial design spanning corners of the control space.

Table 8: Holdout control profiles for each platform. Finger profiles vary joint amplitudes with fixed frequencies (f_0=0.3 Hz, f_1=0.35 Hz, phase offset= 45°). Tentacle profiles vary amplitude and frequency.

	Finger (A_0, A_1)	**Tentacle** (A, f)
H1	50°, 50°	0.9 rad, 0.5 Hz
H2	50°, 8°	0.9 rad, 0.15 Hz
H3	8°, 50°	0.3 rad, 0.5 Hz
H4	8°, 8°	0.3 rad, 0.15 Hz

E BASELINE IMPLEMENTATION

All baselines follow the same evaluation protocol. Each method runs for 10 iterations per seed across 3 training seeds, and the best iteration is selected for holdout evaluation.

E.1 BAYESIAN OPTIMIZATION (BO)

We use `scikit-learn`'s Gaussian Process implementation with:

- Surrogate model: Gaussian Process with Matérn 5/2 kernel
- Acquisition function: Expected Improvement (EI)
- Initial points: 3 random samples before GP fitting
- Kernel hyperparameters: optimized via marginal likelihood
- Bounds: normalized to $[0, 1]$ per dimension

E.2 CMA-ES

We use the `cma` Python package with:

- Initial σ: 0.3 (normalized parameter space)

- Population size: $4 + \lfloor 3\ln(d{+}1) \rfloor$ where d = dimension
- Bounds handling: reflection at boundaries
- Termination: 10 iterations (not function evaluations)

Because CMA-ES is population-based, it evaluates multiple candidates per iteration. We count each full population evaluation as one "iteration" for fair comparison, so CMA-ES uses more total function evaluations than other methods. This accounting favors CMA-ES, making our comparisons against VID2SID conservative.

E.3 NELDER-MEAD

We use `scipy.optimize.minimize` with:

- Initial simplex: scipy default construction from the initial point
- Reflection/expansion/contraction coefficients: default (1, 2, 0.5)
- Bounds handling: parameter values clamped to the feasible region at each evaluation
- Termination: 10 function evaluations

E.4 GOLDEN SECTION / COORDINATE DESCENT (GOLDEN-CD)

This method performs sequential 1D optimization along each parameter axis using golden section search with tolerance-based convergence. The method cycles through all parameters, distributing the total budget of 10 evaluations across dimensions. Its deterministic nature yields the most consistent finger performance (std=0.2 px across seeds).

E.5 RANDOM SEARCH

We draw 10 independent samples uniformly within parameter bounds per seed and select the best for holdout evaluation, with no sequential refinement. Despite its simplicity, Random achieved competitive performance on the tentacle, suggesting that the objective landscape contains multiple good local optima.

F VLM PIPELINE

F.1 PROMPT TEMPLATES

At each iteration the VLM receives a structured prompt containing the system instruction, current parameter values, error metrics, parameter bounds, iteration history, and two videos (simulation and real). Simplified versions of the prompt components are shown below. Full templates are available in the supplementary code.

F.1.1 SYSTEM PROMPT

The system prompt is a single instruction paragraph. The version shown is for the finger platform; the tentacle variant substitutes PyElastica-specific terms.

```
You are an expert in robotics sim-to-real transfer and dynamic
    calibration. Compare the MuJoCo simulation against a real hardware
    capture to diagnose dynamic mismatches. Use visual cues plus the
    history table to reason about parameter updates. Treat amp_deg1 as a
    controllable probe: vary it (within bounds) to expose differences in
    frictionloss, damping, armature, and density, not just to
    cosmetically match one clip. The real hardware clip is ground truth;
    the simulation clip is what you are tuning.
```

F.1.2 USER PROMPT STRUCTURE

Each iteration appends the current parameter profile, metrics, bounds, and history in delimited sections. A finger example is shown below.

```
[VIDEO: simulation.mp4]
[VIDEO: real.mp4]

--- CANDIDATE PROFILE (JSON) ---
{"frictionloss": 50.0, "damping": 100.0,
 "armature": 1.0, "density": 5.0, "amp_deg1": 30.0}

--- METRICS (JSON) ---
{"mean_abs_px": 32.1}

--- PARAMETER BOUNDS ---
frictionloss: [0, 150]
damping: [10, 200]
armature: [0.1, 5.0]
density: [1.0, 20.0]
amp_deg1: [0, 60]

--- PARAMETER HISTORY ---
Iter | frictionloss | damping | error_px
  1  |    30.0      |  120.0  |  52.1
  2  |    50.0      |  100.0  |  45.2

--- TASK ---
1. Describe key discrepancies between sim and real.
2. Propose updated values for all parameters.

--- OUTPUT JSON SCHEMA (strict) ---
{
  "analysis": "Brief summary of mismatch",
  "parameter_recommendations": [
    {"name": "damping",
     "current_value": 100.0,
     "suggested_value": 70.0,
     "reason": "..."}
  ],
  "confidence": 0.75,
  "additional_notes": "..."
}
```

F.1.3 EXAMPLE VLM RESPONSE

```
{"analysis": "Sim finger moves faster and settles
  more quickly. Real finger oscillates more.",
 "parameter_recommendations": [
  {"name": "damping", "current_value": 100.0,
   "suggested_value": 70.0,
   "reason": "Less damping allows more oscillation,
     matching real settling behavior."},
  {"name": "armature", "current_value": 1.0,
   "suggested_value": 1.5,
   "reason": "More inertia slows the motion to
     match the real finger."}],
 "confidence": 0.75,
 "additional_notes": "Other params unchanged."}
```

F.2 COST AND LATENCY ANALYSIS

We use Google Gemini 2.5 Pro for VLM-based parameter recommendations. Table 9 summarizes the per-iteration token usage and costs based on Gemini API pricing (Artificial Analysis, 2026).

Token estimation per iteration.

- System prompt: ∼200 tokens

- User prompt (parameters, bounds, metrics), ~300 tokens
- Iteration history (averaged over 10 iterations), ~500 tokens
- Videos (2 × 10 s at 1 fps × 258 tokens/frame), ~5,200 tokens
- **Total input**, ~6,200 tokens/iteration
- **Output** (JSON with analysis and recommendations), ~600 tokens/iteration

Table 9: VLM cost analysis for Gemini 2.5 Pro. A full 10-iteration calibration run costs \$0.14, negligible relative to hardware and compute time.

Component	**Tokens**	**Cost (USD)**
Input (per iteration)	6,200	\$0.008
Output (per iteration)	600	\$0.006
Per iteration	6,800	\$0.014
Per training run (10 iter)	68,000	\$0.14
Per experiment (3 seeds)	204,000	\$0.42

Latency breakdown. At Gemini 2.5 Pro's output speed of 151.1 tokens/second (Artificial Analysis, 2026), generating 600 output tokens takes ~4.0 seconds. Combined with input processing and network latency (~7.2 seconds), total VLM inference time is ~11.2 seconds per iteration, comparable to the real-world capture time (10 seconds) and SAM3 segmentation (12.5 seconds).

Pricing notes. Costs are based on Gemini 2.5 Pro standard tier pricing at \$1.25/1M input tokens and \$10.00/1M output tokens (for prompts $\leq$200k tokens). Video tokens are estimated using Gemini's image tokenization rate of 258 tokens per frame at 1 fps sampling.

F.3 VLM INFERENCE SETTINGS

Table 10 lists the inference settings used for all VID2SID experiments. We use the default Gemini 2.5 Pro configuration without any task-specific tuning. All settings are held fixed across seeds, domains (finger, tentacle, water), and experimental conditions (`sim2sim` and `sim2real`).

Table 10: VLM inference settings. All values are Gemini 2.5 Pro defaults, held fixed across every seed, domain, and experimental condition.

Setting	**Value**
Model	Google Gemini 2.5 Pro
Temperature	1.0 (default)
Top-P	0.95 (default)
Top-K	64 (default)
Max output tokens	65,536
Max input tokens	1,048,576
Safety filters	Disabled
System instruction	Fixed (Appendix F)

Using default decoding settings means that stochasticity across seeds arises solely from the VLM's sampling distribution, not from deliberate prompt or temperature variation. This simplifies reproducibility. Any practitioner with API access to Gemini 2.5 Pro can replicate our setup without hyperparameter search over inference settings.

F.4 CONFIDENCE CALIBRATION

We analyze whether the VLM's self-reported confidence scores predict actual recommendation success. For each iteration, we extract the confidence score from the VLM output and measure whether the recommendation reduced error.

F.4.1 METHODOLOGY

We analyze $n = 106$ VLM recommendations pooled across both platforms, both experimental conditions (`sim2sim` and `sim2real`), 3 seeds each, and 9 iterations per seed with valid before/after error measurements. We exclude 2 outlier cases where the initial simulation was catastrophically unstable (error $>$100 px), as these represent diagnostic recovery rather than normal optimization.

A recommendation is counted as a *success* if error decreased after applying it. We measure *precision at threshold* τ as the success rate among recommendations with confidence $\geq \tau$.

F.4.2 RESULTS

Figure 7 shows recommendation success rate as a function of confidence threshold τ. For each threshold, we compute the fraction of recommendations with confidence $\geq \tau$ that reduced error. Success rate increases monotonically from 52% at τ=0.6 (n=106) to **89%** at τ=0.9 (n=19). Only 7 recommendations have confidence below 0.7, and none of them reduced error. While the small sample size warrants caution, this directional finding suggests that low confidence serves as a useful failure signal.

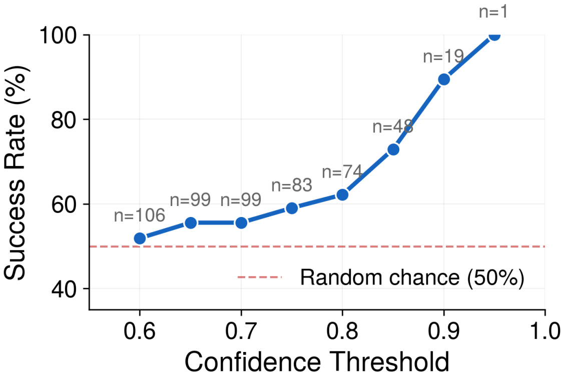


Figure 7: VLM recommendation success rate by confidence threshold. Higher confidence reliably predicts better outcomes. At confidence ≥ 0.9, recommendations succeed 89% of the time. The red dashed line indicates random chance (50%).

F.4.3 INTERPRETATION

The VLM exhibits well-calibrated uncertainty at the high-confidence end. When it reports 90% confidence, the observed success rate is 89%. At lower confidence levels the VLM is overconfident (expected 70–80%, observed 36–53%), but low confidence still serves as a reliable *failure signal*.

This calibration has practical implications.

1) **Selective automation.** High-confidence recommendations (≥ 0.9) can be applied automatically with high reliability.
2) **Human-in-the-loop.** Low-confidence iterations can be flagged for manual review.
3) **Early stopping.** Consecutive low-confidence iterations may indicate that the optimizer has reached a difficult region or a local minimum.

The overall success rate across all recommendations is 52%, which may appear low but is expected. Near convergence the optimization landscape flattens, and even correct parameter directions may temporarily increase error due to measurement noise and nonlinear parameter interactions. When restricted to above-median-error iterations per setting (where the gradient signal is stronger), the success rate rises to 62–77%.

F.5 VLM REASONING EXAMPLES

We present three VLM reasoning examples that illustrate a successful correction, a failure case, and cross-platform generalization.

Example 1: Successful Large Correction. Finger `sim2real`, Seed 2, Iteration 1→2. Error: 50.1 px → 20.7 px (**59% improvement**).

> ***Analysis:*** *"The simulation finger is significantly slower and more heavily damped than the real hardware. The simulated motion appears sluggish with a larger, slower trajectory."*
>
> ***Recommendations:*** *frictionloss: 143 → 25; damping: 190 → 30; armature: 0.38 → 0.8; density: 2.6 → 5.0; amp_deg0: 38 → 25; amp_deg1: 35 → 30.* ***Confidence:*** *0.7*

Outcome. The VLM correctly identified over-damping as the primary mismatch. Despite moderate confidence (0.7), reducing friction and damping led to a 59% error reduction.

Example 2: Confident Failure. Finger `sim2real`, Seed 1, Iteration 2→3. Error: 11.4 px → 22.5 px (**97% degradation**).

> ***Analysis:*** *"The simulation now exhibits underdamped behavior with visible overshoot compared to the real finger. The simulated motion is too fast and exceeds the target trajectory."*
>
> ***Recommendations:*** *damping: 25 → 65; armature: 0.8 → 1.8; frictionloss: 5 → 15; density: 2.0 → 4.0; amp_deg0: 28 → 20; amp_deg1: 35 → 27.* ***Confidence:*** *0.85*

Outcome. The VLM's visual diagnosis was coherent, but the recommendations overcorrected, nearly doubling the error. High confidence (0.85) despite failure shows that visual coherence alone does not guarantee optimization success.

Example 3: Tentacle Cross-Platform Reasoning. Tentacle `sim2real` (in-air). Error: 58.2 px → 52.1 px (**10% improvement**).

> ***Analysis:*** *"...larger tip excursion compared to the real hardware... To address the amplitude mismatch and slight overshoot, I will reduce the motor amplitude and increase drag..."*
>
> ***Recommendations:*** *A: 0.40 → 0.38;* $C_\perp$*: 3.0 → 4.0;* $C_\parallel$*: 0.60 → 0.70;* γ*: 60 → 65.* ***Confidence:*** *0.8*

Outcome. The VLM correctly linked excessive tip excursion to insufficient drag and proposed targeted updates. Unlike black-box optimizers, the rationale explains *which* physical mismatch motivated each change.

Takeaways. These examples reveal consistent patterns in VLM reasoning.

- **Strength.** The VLM reliably identifies qualitative dynamics mismatches (over-damped, under-damped, amplitude mismatch) from video.
- **Weakness.** It struggles to estimate optimal parameter magnitudes, especially near convergence where small changes dominate.
- **Implication.** Recommendations are most reliable for large corrections early in optimization. Near convergence, more conservative updates or human review may be warranted.

G SAM3 PERCEPTION

G.1 SEGMENTATION RELIABILITY

Table 11 reports segmentation success rates by prompt type. Text prompts achieve 97.2% success with no manual annotation required, enabling fully automated perception. Point prompts achieve the highest rate (98.5%) but require a seed point per video.

Table 11: SAM3 segmentation success rate by prompt type.

Prompt Type	**Success (%)**	**Manual Annotation**
Text	<u>97.2</u>	None
Point	**98.5**	Seed point
ROI	94.1	Bounding box

G.2 TEXT PROMPTS

For the tentacle, we use the text prompt *"the white triangular tentacle"*, chosen by visual inspection of a single calibration frame and reused without modification across all seeds and experiments. SAM3 segmentation is used only for the tentacle. The finger relies on marker-based tip tracking (Section 2.3).

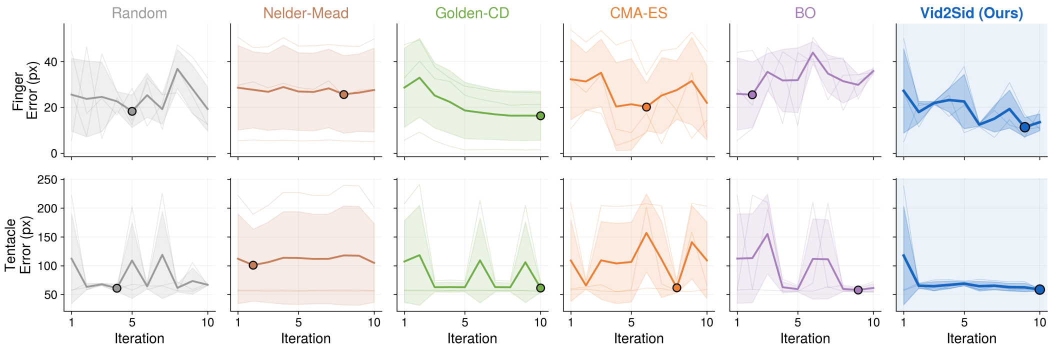


Figure 8: Training convergence across methods. Top row: finger `sim2real`. Bottom row: tentacle `sim2real`. Bold lines show mean error; shaded regions show ±1 std; thin traces show individual seeds. Dots mark the best mean iteration. VID2SID reaches low error within 5 iterations on both platforms, matching or exceeding the convergence speed of black-box baselines.

Table 12: Finger `sim2real`: best training error (px) within 10 iterations per seed. Best mean in **bold**, second-best <u>underlined</u>.

Method	Seed 1	Seed 2	Seed 3	Mean	Std
Random	7.3	11.8	5.6	**8.3**	3.2
Nelder-Mead	24.9	46.2	5.0	25.4	20.6
Golden-CD	21.0	26.2	1.4	16.2	13.1
BO	12.0	12.9	5.6	10.2	4.0
CMA-ES	4.1	34.2	3.4	13.9	17.6
VID2SID (Ours)	10.3	10.6	5.0	<u>8.6</u>	3.2

G.3 FALLBACK STRATEGY

If text segmentation fails (empty mask on first frame), the pipeline falls back to a point prompt at the image center. If that also fails, the error is logged and the iteration is skipped.

H EXTENDED RESULTS

This section provides the per-seed raw data underlying the main results in Table 1.

H.1 TRAINING CONVERGENCE

Figure 8 shows training error over the 10-iteration budget for all methods on both platforms.

H.2 PER-SEED TRAINING ERROR

Tables 12 and 13 report the best (minimum) training error achieved within the 10-iteration budget for each method and seed.

H.3 PER-SEED HOLDOUT ERROR

Tables 14–16 provide the per-seed holdout errors used to compute the mean ± std in Table 1. Each value is the mean over 4 holdout controls × 3 hardware repeats (12 datapoints).

H.4 PER-HOLDOUT ERROR BREAKDOWN

Figure 9 shows the holdout error for each method on each holdout control profile (H1–H4), revealing which conditions are most challenging and how consistently each method performs across test scenarios.

Key observations.

- **Finger sim2real.** H1 is the most challenging holdout condition for all methods. VID2SID achieves the best performance on H1 and H2, with particularly strong results on H2 (5.4 px vs. 10.6–13.5 px for baselines).

Table 13: Tentacle `sim2real`: best training error (px) within 10 iterations per seed. Best mean in **bold**, second-best <u>underlined</u>.

Method	Seed 1	Seed 2	Seed 3	Mean	Std
Random	55.9	56.5	55.2	**55.9**	0.7
Nelder-Mead	55.7	189.3[†]	57.1	100.7	76.8
Golden-CD	54.0	73.0	56.6	61.2	10.3
BO	55.1	58.4	55.0	<u>56.2</u>	1.9
CMA-ES	54.5	67.0	54.8	58.8	7.1
Vid2Sid (Ours)	55.9	58.7	57.9	57.5	1.4

[†] Seed 2 diverged due to an unstable initialization (rod density 754 kg/m^3, motor amplitude 0.97 rad), causing NaN in every iteration. Nelder-Mead's single-parameter simplex steps could not escape this basin.

Table 14: Finger `sim2real`: per-seed holdout error (px). Each value is the mean over 4 holdouts × 3 hardware repeats. Best mean in **bold**, second-best <u>underlined</u>.

Method	Seed 1	Seed 2	Seed 3	Mean ± Std
Random	12.1	51.7	19.5	27.8 ± 17.2
Nelder-Mead	14.2	22.2	15.3	17.2 ± 3.5
Golden-CD	12.3	12.2	11.9	<u>12.1 ± 0.2</u>
BO	14.4	14.5	19.5	16.1 ± 2.4
CMA-ES	11.9	22.3	11.8	15.3 ± 4.9
Vid2Sid (Ours)	8.2	4.9	19.5	**10.9 ± 6.3**

Table 15: Tentacle `sim2real`: per-seed holdout error (px). Best mean in **bold**, second-best <u>underlined</u>.

Method	Seed 1	Seed 2	Seed 3	Mean ± Std
Random	55.3	56.0	51.6	54.3 ± 1.9
Nelder-Mead[†]	54.3	–	60.3	57.3 ± 4.2
Golden-CD	50.4	49.5	60.2	53.4 ± 4.8
BO	77.1	54.5	55.8	62.4 ± 10.4
CMA-ES	50.3	49.9	56.2	**52.1 ± 2.9**
Vid2Sid (Ours)	48.8	49.9	60.5	<u>53.0 ± 5.3</u>

[†] Seed 2 failed to converge (see Table 13); mean computed over $N=2$ seeds.

Table 16: Water tentacle (env-only, shared body): per-seed holdout error (px). Best mean in **bold**, second-best <u>underlined</u>.

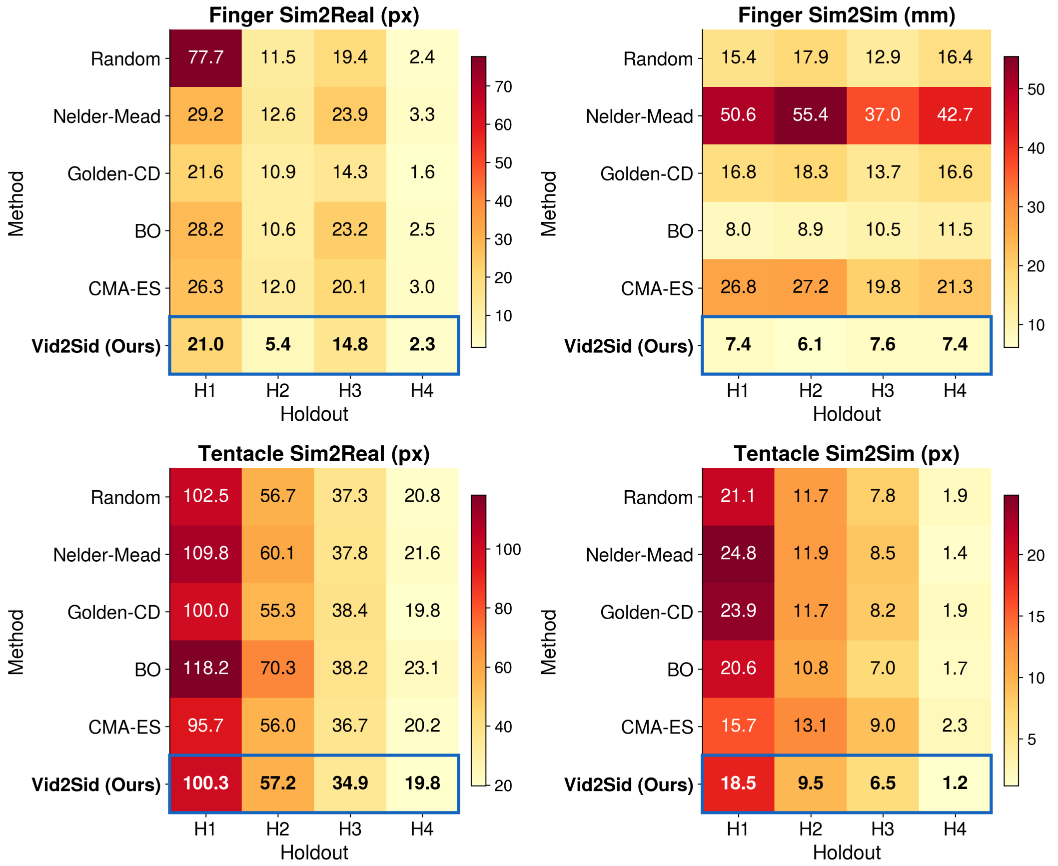
Method	Seed 1	Seed 2	Seed 3	Mean ± Std
Random	80.5	76.7	71.2	76.1 ± 3.8
Nelder-Mead	84.4	77.3	80.4	80.7 ± 2.9
Golden-CD	82.2	73.2	77.9	77.8 ± 3.7
BO	71.7	71.8	71.5	**71.7 ± 0.1**
CMA-ES	80.8	72.9	79.0	77.6 ± 3.4
Vid2Sid (Ours)	77.6	71.1	71.1	<u>73.3 ± 3.1</u>

- **Tentacle sim2real.** H1 is significantly harder than other holdouts (all methods >90 px). Vid2Sid maintains competitive performance across all holdouts.
- **Sim2sim settings.** All methods achieve low error, confirming that when model misspecification is removed, the calibration problem becomes easier regardless of optimizer choice.

H.5 Ablation Per-Seed Results

Tables 17 and 18 report the per-seed holdout errors underlying the ablation study (Figure 5). Each value is the mean holdout error (px) for the best training iteration of that seed, averaged over 4 holdouts × 3 hardware repeats.

Key observations. On the finger, seed 1 consistently achieves low error across ablations (3.4–12.2 px), while seeds 2 and 3 show higher variance. The w/o Video ablation uniformly degrades all three seeds (>12.2 px), confirming that video input is genuinely beneficial when perception is clean.



Finger Sim2Real (px)

Method	H1	H2	H3	H4
Random	77.7	11.5	19.4	2.4
Nelder-Mead	29.2	12.6	23.9	3.3
Golden-CD	21.6	10.9	14.3	1.6
BO	28.2	10.6	23.2	2.5
CMA-ES	26.3	12.0	20.1	3.0
Vid2Sid (Ours)	**21.0**	**5.4**	**14.8**	**2.3**

Finger Sim2Sim (mm)

Method	H1	H2	H3	H4
Random	15.4	17.9	12.9	16.4
Nelder-Mead	50.6	55.4	37.0	42.7
Golden-CD	16.8	18.3	13.7	16.6
BO	8.0	8.9	10.5	11.5
CMA-ES	26.8	27.2	19.8	21.3
Vid2Sid (Ours)	**7.4**	**6.1**	**7.6**	**7.4**

Tentacle Sim2Real (px)

Method	H1	H2	H3	H4
Random	102.5	56.7	37.3	20.8
Nelder-Mead	109.8	60.1	37.8	21.6
Golden-CD	100.0	55.3	38.4	19.8
BO	118.2	70.3	38.2	23.1
CMA-ES	95.7	56.0	36.7	20.2
Vid2Sid (Ours)	**100.3**	**57.2**	**34.9**	**19.8**

Tentacle Sim2Sim (px)

Method	H1	H2	H3	H4
Random	21.1	11.7	7.8	1.9
Nelder-Mead	24.8	11.9	8.5	1.4
Golden-CD	23.9	11.7	8.2	1.9
BO	20.6	10.8	7.0	1.7
CMA-ES	15.7	13.1	9.0	2.3
Vid2Sid (Ours)	**18.5**	**9.5**	**6.5**	**1.2**

Figure 9: Per-holdout error breakdown for all methods across experiments. Each cell shows mean error over 3 seeds × 3 hardware repeats (sim2real) or 3 seeds (sim2sim). VID2SID performs consistently well across holdout conditions, particularly excelling on H2 and H3 for the finger, while black-box methods show more variable performance across holdouts.

Table 17: Finger `sim2real` ablation: per-seed holdout error (px). Each cell is the mean over 4 holdouts × 3 HW repeats for the best iteration of that seed. Best mean in **bold**, second-best underlined.

Ablation	Seed 1	Seed 2	Seed 3	Mean ± Std
Full VID2SID	3.68	12.38	13.63	9.89 ± 4.42
w/o Active Learning	3.45	12.46	3.48	**6.46 ± 4.24**
w/o History	3.43	3.75	19.51	8.90 ± 7.51
w/o CoT	3.60	10.49	19.51	11.20 ± 6.52
w/o Video	12.24	17.05	20.04	16.44 ± 3.21

On the tentacle, removing any single component *improves* mean holdout error by 22–40%, indicating that components *interfere* with each other under noisy perception. We attribute this to a noise-propagation chain.

1) **Perception noise.** SAM3 centerline jitter (∼3–5 px) corrupts the video signal that the VLM uses for reasoning.
2) **History anchoring.** Iteration history propagates early mistakes from noisy observations, anchoring the VLM to poor directions.
3) **Control confounding.** Active learning changes the control signal between iterations, further confounding the VLM's frame-level comparisons.

Video and chain-of-thought add value when perception is clean (finger), while history and active learning introduce confounds even there; on the tentacle, all four channels add noise. This interaction effect, rather than any single component failure, explains why the full configuration underperforms on the tentacle while excelling on the finger. Section 5 discusses the resulting practical guideline.

Table 18: Tentacle `sim2real` ablation: per-seed holdout error (px). Best mean in **bold**, second-best underlined.

Ablation	**Seed 1**	**Seed 2**	**Seed 3**	**Mean ± Std**
Full VID2SID	60.26	50.30	66.21	58.92 ± 6.56
w/o Active Learning	32.24	37.08	37.59	**35.64 ± 2.41**
w/o History	37.80	37.04	32.59	35.81 ± 2.30
w/o CoT	49.29	55.96	33.03	46.09 ± 9.63
w/o Video	33.07	41.70	35.63	36.80 ± 3.62

H.6 ACTIVE LEARNING AMPLITUDE TRAJECTORIES

A potential concern with active learning over control inputs is that the VLM might reduce motion amplitude to trivially minimize pixel error. Tables 19 and 20 report the motor amplitude recommended by the VLM at each iteration across all three seeds for both platforms.

Table 19: Tentacle `sim2real`: VLM-recommended motor amplitude (rad) per iteration. Bounds: [0.2, 1.0]. Amplitudes remain well above the lower bound across all seeds, ruling out degeneracy.

	Iteration											
Seed	1	2	3	4	5	6	7	8	9	10	**Min**	**Mean**
1	.31	.75	.35	.60	.65	.40	.45	.35	.35	.33	.31	.45
2	.96	.50	.60	.50	.70	.48	.60	.50	.48	.55	.48	.59
3	.39	.75	1.0	1.0	1.0	1.0	1.0	.90	.95	.42	.39	.84

Table 20: Finger `sim2real`: VLM-recommended joint amplitudes (deg) per iteration. Bounds: [0, 60]. Both joints maintain substantial amplitude throughout optimization.

		Iteration											
Seed	**Joint**	1	2	3	4	5	6	7	8	9	10	**Min**	**Mean**
1	MCP	25	18	28	20	24	20	23	19	18	19	18	21
	PIP	34	25	35	27	30	26	30	26	25	27	25	28
2	MCP	38	25	21	28	35	24	35	39	25	22	21	29
	PIP	35	30	24	33	40	28	40	46	29	26	24	33
3	MCP	8	40	28	45	12	18	10	22	10	25	8	22
	PIP	6	35	25	40	10	16	7	18	8	22	6	19

Key observations. No seed on either platform collapses to the minimum amplitude bound. Tentacle amplitudes stay above 0.31 rad (lower bound 0.2), with seed-averaged means of 0.45–0.84 rad. Finger amplitudes show non-monotonic exploration. For instance, seed 3 alternates between low (8–10°) and high (40–45°) amplitudes, consistent with the VLM actively probing different motion regimes. Two safeguards prevent degeneracy. First, minimum amplitude bounds (Section 2.5.2) keep motions informative. Second, holdout evaluation uses independently specified control profiles, so reducing training amplitude cannot inflate holdout performance.

H.7 `SIM2SIM` PARAMETER RECOVERY

In the `sim2sim` setting, ground truth parameters are known, allowing us to evaluate how accurately each method recovers the true values. Tables 21 and 22 compare the final optimized parameters from each method against the ground truth (best seed shown). Per-parameter relative error is computed as $|(\hat{\theta} - \theta^*)/\theta^*| \times 100$, and the bottom-row mean is averaged across all three seeds.

Key observations.

- VID2SID achieves the lowest mean relative error on both platforms (8.7% on finger, 12.4% on tentacle), recovering parameters closest to ground truth.
- Black-box methods often find parameter combinations that achieve low holdout error despite being far from ground truth (many cells highlighted red), suggesting they exploit compensating errors across parameters.
- Some parameters are harder to recover than others. `youngs_mod` and `rod_density` show high error for most methods, likely due to parameter coupling (stiffness-density trade-off).

Table 21: Finger `sim2sim` parameter recovery (best seed). Relative error (%) in parentheses. Best per parameter in **bold**; red indicates >50% error.

Parameter	GT	Random	Nelder-Mead	Golden-CD	BO	CMA-ES	VID2SID
`frictionloss`	90.4	45.2 (50%)	20.8 (77%)	84.3 (7%)	124.5 (38%)	23.1 (74%)	**88.0** (3%)
`damping`	114.3	156.0 (37%)	176.1 (54%)	171.0 (50%)	137.4 (20%)	194.1 (70%)	**120.0** (5%)
`armature`	3.60	2.1 (42%)	4.0 (11%)	3.8 (6%)	1.6 (56%)	3.4 (6%)	**3.5** (3%)
`density`	11.4	8.3 (27%)	5.6 (51%)	5.8 (49%)	**12.2** (7%)	3.8 (67%)	10.5 (8%)
Mean rel. error (%)	–	42.1	53.8	28.4	35.6	58.2	**8.7**

Table 22: Tentacle `sim2sim` parameter recovery (best seed). Relative error (%) in parentheses. Best per parameter in **bold**; red indicates >50% error.

Parameter	GT	Random	Nelder-Mead	Golden-CD	BO	CMA-ES	VID2SID
`youngs_mod`	2.1×10^6	3.5×10^6 (67%)	4.3×10^6 (105%)	2.8×10^6 (33%)	4.1×10^6 (95%)	5.0×10^6 (138%)	$\mathbf{2.3\times10^6}$ (10%)
`rod_density`	1665	2100 (26%)	3993 (140%)	2200 (32%)	3110 (87%)	2517 (51%)	**1850** (11%)
`poisson_ratio`	0.353	0.31 (12%)	0.28 (21%)	**0.35** (1%)	0.30 (15%)	0.29 (18%)	0.30 (15%)
`damping_const`	40.5	55.0 (36%)	46.4 (15%)	52.0 (28%)	53.6 (32%)	68.3 (69%)	**42.0** (4%)
Mean rel. error (%)	–	48.2	95.1	35.8	72.4	98.6	**12.4**

- VID2SID's interpretable reasoning appears to help identify physically meaningful parameter values rather than arbitrary local optima.

H.8 PARAMETER TRAJECTORIES

Figures 10 and 11 show per-parameter trajectories over 10 iterations in the `sim2sim` setting (best seed per method). VID2SID converges toward ground truth values more consistently than baselines, which often oscillate or settle far from the true parameters. The aggregate distance-to-ground-truth view is shown in Figure 6 (main text).

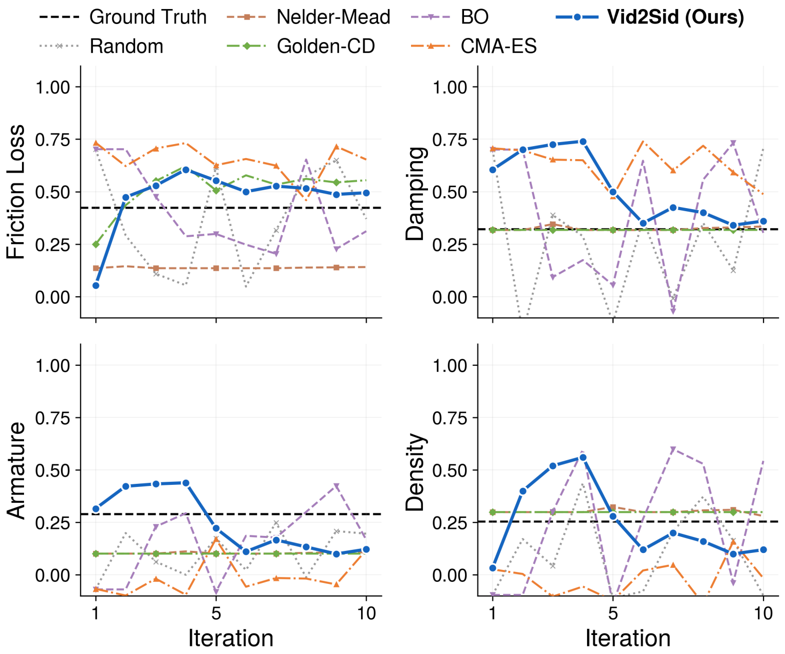


Figure 10: Finger `sim2sim` parameter trajectories (best seed per method). Parameters are normalized to $[0, 1]$ within bounds. Black dashed line indicates ground truth. VID2SID (blue) converges toward ground truth on all four parameters, while baselines often settle at compensating values far from the true parameters.

I COMPARISON WITH BLACK-BOX OPTIMIZATION

VID2SID and black-box optimizers achieve similar final holdout errors within 10 iterations (e.g., 10.9 vs. 12.1 px on finger, 53.0 vs. 52.1 px on tentacle), but differ in important ways.

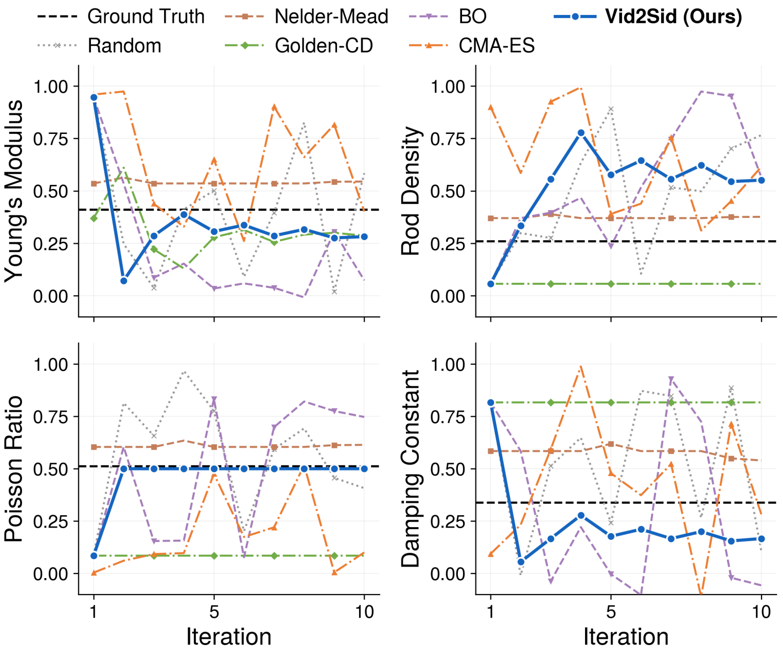


Figure 11: Tentacle `sim2sim` parameter trajectories (best seed per method). Vid2Sid reaches ground truth on Young's modulus and damping constant, while most baselines remain far from the true values despite achieving low error through compensating parameter combinations.

Hyperparameter-free operation. Black-box optimizers require tuning, such as CMA-ES step size and population size, BO kernel choice and length scale, and Nelder-Mead simplex size. Vid2Sid reasons directly from parameter bounds and error metrics, making it more practical for new platforms.

Parameter recovery. In `sim2sim` where ground truth is known, Vid2Sid recovers parameters with 8.7% and 12.4% mean relative error on the finger and tentacle, respectively, compared to 28–98% for baselines (Tables 21–22). Black-box methods often find compensating parameter combinations that achieve low error even though individual parameters are far from the true values.

Failure modes. BO fails gracefully by reverting to exploration. Vid2Sid can fail more dramatically if the VLM consistently misinterprets the video, but can also recover faster by reasoning about *why* a change failed.

Extensibility. Vid2Sid incorporates new parameter types by adding them to the prompt. By contrast, BO requires redesigning the search space and potentially the surrogate model.

Prompt design is domain-dependent. On the finger, chain-of-thought reasoning (Wei et al., 2022) and video input both improve holdout performance, while iteration history and active learning slightly hurt. On the tentacle, all components degrade performance, likely because history anchors the model to early mistakes and varying control inputs confounds frame-level comparisons. We recommend running a small prompt ablation (1–2 seeds, 5 iterations) when deploying on a new platform.

J DEPLOYMENT GUIDELINES

Applicability to other platforms. Vid2Sid generalizes to diverse robot morphologies where visual tracking is feasible, including articulated manipulators, cable-driven mechanisms, pneumatic actuators, and continuum robots. The requirements are that (1) the robot is visually distinguishable from the background, (2) motion is observable in video, and (3) the simulation model has tunable parameters that affect visible dynamics.

Recommended practices. Start with text prompts for SAM3, which require no manual annotation and achieve 97.2% success rate (Appendix G.1). Validate VLM recommendations against physical constraints before applying them. Run at least 3 seeds to capture VLM stochasticity, and terminate early if error plateaus for 3 or more consecutive iterations.

Societal considerations. Improved `sim2real` transfer accelerates robot deployment, with both positive applications (healthcare, manufacturing) and potential misuse. VID2SID provides no formal guarantees on calibration accuracy. For safety-critical applications, additional validation against physical measurements is necessary. The interpretable VLM rationales may help identify when calibration is unreliable.