TEST-TIME COMPUTE FOR HUMANOID BEHAVIOR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Humanoid Behavior Foundation Models (BFMs) can execute impressive motions, but downstream applications such as loco-manipulation demand high whole-body positional accuracy. Existing BFMs suffer from a systematic command–execution gap, where executed motions deviate from desired ones. We show that additional test-time compute can improve execution without retraining the BFM. First, we introduce Test-Time Command Optimization (TTCO), which adapts the input command to the specific BFM to produce an executed trajectory closer to the desired motion. TTCO finds such commands by sampling-based optimization through closed-loop simulations of the BFM and robot dynamics. Across three BFMs on the AMASS dataset, TTCO reduces mean global body-position error by 79–84% in simulation. Second, we analyze test-time compute scaling and derive an approximate compute-optimal strategy that can sustain performance gains as the compute budget increases. Third, TTCO can generate BFM-specific action labels from recorded robot trajectories without requiring the original teleoperation body-motion commands. Simulation and real-robot experiments show that these labels are effective for training high-level loco-manipulation policies. This method also supports reusing existing demonstrations across BFMs. The project page is available at https://bfm-ttc.github.io/

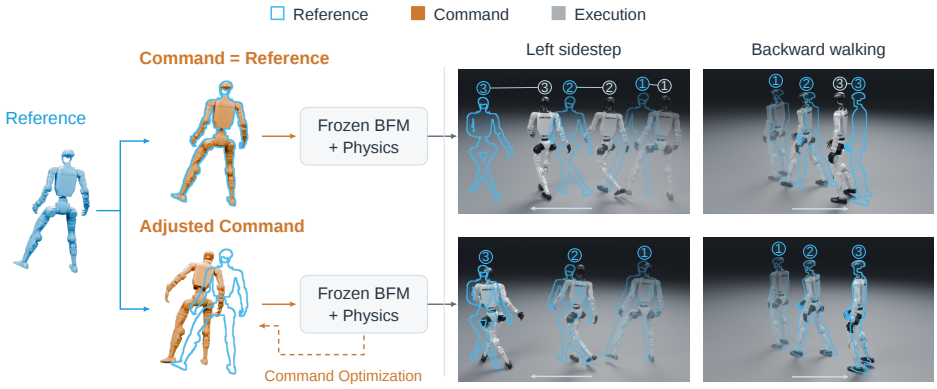

Figure 1: **Changing the command can bring execution closer to the same reference.** Directly using the reference as the command can leave a command–execution gap. With the BFM fixed, our command optimization adjusts commands to bring execution closer to the same reference. The adjustment of commands is exaggerated for demonstration.

1 INTRODUCTION

Recent advances in humanoid Behavior Foundation Models (BFMs) enable the successful execution of diverse motions, including complex martial arts and dance sequences (Liao et al., 2026; Luo et al., 2026; Xie et al., 2025; Ma et al., 2026; Zeng et al., 2026a; Wang et al., 2026c; Li et al., 2026b; Sleiman et al., 2026; Yuan et al., 2026; Yin et al., 2026). These models support downstream applications such as loco-manipulation (Luo et al., 2026; Hu et al., 2026; Wang et al., 2026b; Ze et al., 2026; Xie et al., 2026; Wu et al., 2026; Chen et al., 2026), which demand substantially higher whole-body positional accuracy (Jiang et al., 2026), ideally matching the accuracy of robotic arm position controllers. Due to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

training data quality, regularization design and other constraints (Luo et al., 2026), current humanoid BFMs exhibit systematic deviations between desired and executed motions, a discrepancy we call the *command–execution gap*. A robot can complete a movement while its body drifts from the intended global position or its hands and feet miss the intended contact locations. Prior work addresses these errors by scaling training datasets, model size, and computational resources (Luo et al., 2026; Qi et al., 2026; Zeng et al., 2026b), as well as improving training procedures and network architectures (Liu et al., 2026a; Wang et al., 2026c; Ma et al., 2026; Han et al., 2026).

We show that additional test-time compute can improve motion execution without changing the BFM. At inference time, the desired motion need not be supplied directly as the command to the BFM. Adapting the input command to the specific BFM can yield an executed trajectory that more closely matches the original desired motion (Figure 1). We introduce *Test-Time Command Optimization* (TTCO), which finds such commands through sampling-based optimization with closed-loop simulations of the BFM and robot dynamics. We evaluate TTCO on 12,273 AMASS (Mahmood et al., 2019) reference motions using HEFT (Liu et al., 2026a), SONIC (Luo et al., 2026), and TWIST2 (Ze et al., 2026). Compared with directly using the reference motion as the command, TTCO reduces mean global body-position error by 79–84% across these three BFMs in simulation.

This framework provides a direct way to scale up test-time compute. The gains from command optimization depend on both the amount of compute and its allocation across candidates, optimization iterations, rollout horizon, and replanning. This raises two questions: *To what extent can additional compute improve performance, and how should a fixed compute budget be allocated to achieve the best performance?* To answer these questions, we analyze the compute-allocation trade-offs and construct a surrogate cost model fitted to limited measurements on a small motion set. This model yields an approximate compute-optimal strategy that sustains performance gains as compute increases (Figure 2) and generalizes beyond the motions and BFM used for fitting.

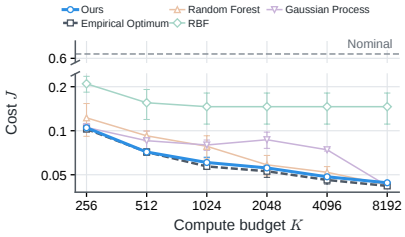

Figure 2: **Scaling test-time compute with an approximate compute-optimal strategy.** Our proposed strategy achieves the best overall performance among four strategies.

The benefits of command optimization extend to downstream humanoid loco-manipulation. Because of the command–execution gap, a recorded robot trajectory generally cannot serve as an action label for high-level policy learning. Existing pipelines therefore train high-level policies with teleoperation commands tied to the BFM used for demonstration collection (Hu et al., 2026; Ze et al., 2026; Luo et al., 2026). In contrast, we apply TTCO offline to optimize body-motion commands against the recorded trajectories while keeping the target BFM fixed. Experiments in HumanoidArena (Wang et al., 2026a) and on real robots show that optimized commands provide effective supervision for high-level policy learning, reducing dependence on the original body-motion labels and supporting demonstration reuse across BFMs.

In summary, our contributions to test-time compute for humanoid BFMs are:

- **We introduce TTCO to improve motion-tracking fidelity in existing humanoid BFMs through test-time compute, without further training.** Across three pretrained BFMs and 12,273 AMASS reference motions, TTCO reduces mean global body-position error by 79–84% in simulation.
- **We analyze test-time compute scaling and derive an approximate compute-optimal allocation strategy.** Our surrogate cost model captures allocation trade-offs and yields a strategy that generalizes across motions and BFMs.
- **We extend test-time compute to high-level policy learning and demonstration reuse across BFMs.** We validate TTCO-generated body-action labels for both uses in simulation and on real robots.

2 Related Work

Humanoid Behavior Foundation Models. Humanoid BFMs build on advances in whole-body motion tracking, enabling the execution of diverse and complex skills (Liao et al., 2026; Han et al., 2026; Ma et al., 2026; Zhu et al., 2026; Li et al., 2026c; Wang et al., 2026c). These models provide a

reusable motion-command interface (Zeng et al., 2026a;b; Li et al., 2026b; Luo et al., 2026), through which high-level policies (Ze et al., 2026; Hu et al., 2026) can specify target motions to perform downstream loco-manipulation. Recent work scales these BFMs at training time through more compute, data and capacity (Luo et al., 2026; Qi et al., 2026; Chen et al., 2026; Zeng et al., 2026b). We study inference compute for humanoid BFMs in general and how tracking quality benefits from scaling that compute.

Sampling-Based Methods for Robot Control. Sampling-based methods optimize control sequences through physics rollouts without requiring derivatives of the dynamics or cost (Williams et al., 2018; Howell et al., 2022; Xue et al., 2025; Liu et al., 2026b; Pan et al., 2025; Dhédin et al., 2026; Jordana et al., 2025). Sumo (Zhang et al., 2026) applies this approach in the command space of a pretrained whole-body policy, using policy-in-the-loop simulations to plan object interactions. BFM-Zero (Li et al., 2026b) also optimizes latent prompt sequences in simulation to improve tracking, with discussion limited to its own model. We apply sampling-based optimization to pretrained humanoid BFMs in general, to optimize motion commands and reduce execution error relative to a fixed reference motion.

Test-Time Compute Scaling and Allocation. Language-model studies show that test-time performance depends on both the compute budget and its allocation to sampling, refinement, and verification (Brown et al., 2024; Zhang et al., 2025; Snell et al., 2025; Wu et al., 2025; Setlur et al., 2025). Embodied systems use test-time compute to sample and select actions (Kwok et al., 2025) or executable motions (Cao et al., 2026), and some adaptively allocate that compute between action refinement and parallel sampling in a frozen generative policy (Li et al., 2026a). We measure test-time compute by the number of simulated BFM transitions, then study the scaling and allocation of that budget for pretrained humanoid BFMs.

3 METHOD

3.1 PROBLEM SETTING

Behavior Foundation Model. We consider a humanoid Behavior Foundation Model (BFM) π controlling a robot to track a reference motion. Let $\mathcal{M}_{1:T}$ denote the motion-sequence space over time steps $1:T$. Let $\tau^{\mathrm{ref}}_{1:T} \in \mathcal{M}_{1:T}$ be the desired reference motion, where $\tau^{\mathrm{ref}}_t \in \mathbb{R}^{2n_q+13}$ represents the robot's generalized coordinates and velocities at time t, and n_q is the number of joints. Let s_t, o_t, and a_t be the humanoid state, policy observation, and action. The BFM π receives a *motion command* sequence $c_{1:T} \in \mathcal{M}_{1:T}$ as its task input and produces action a_t based on observation history $o_{0:t-1}$ at each time step. We assume that both the policy π and the environment dynamics f are deterministic and the initial state is fixed to s_0. Then we have $a_t = \pi(o_{0:t-1}, c_t)$, $(s_t, o_t) = f(s_{t-1}, a_t)$.[1] With $z_t = (s_t, o_{0:t})$, the BFM policy and environment dynamics jointly define the closed-loop transition $z_t = F_\pi(z_{t-1}, c_t)$. For the fixed initial state, we use $\mathcal{T}_\pi(c)$ to denote the executed trajectory induced by the motion command c. Appendix A.1 describes the motion-tracking formulation, training, and control interface underlying this abstraction.

Command-Execution Gap. Standard BFM inference directly uses the desired reference motion as the motion command input (Luo et al., 2026; Ze et al., 2026). We call it the *nominal motion command*, $c^{\mathrm{nom}} = \tau^{\mathrm{ref}}_{1:T}$, which produces the executed trajectory $\tau^{\mathrm{nom}} = \mathcal{T}_\pi(c^{\mathrm{nom}})$. In general, a BFM is trained to minimize the tracking loss between the executed and reference trajectories. However, a BFM cannot execute τ^{ref} exactly because reference motions may be physically infeasible, the training motion data may have limited coverage, policy optimization may leave residual errors, and the regularization and domain randomization used for sim-to-real transfer may trade tracking precision for robustness. Consequently, the executed trajectory τ^{nom} systematically deviates from the desired reference motion τ^{ref}, creating a *command–execution gap*.

Command Optimization. To address this gap, our key insight is that *the motion command that produces the most accurate execution need not equal the desired reference motion*. Compensating for

[1] While some BFMs also use past and future command frames, we consider only the current frame for notational simplicity.

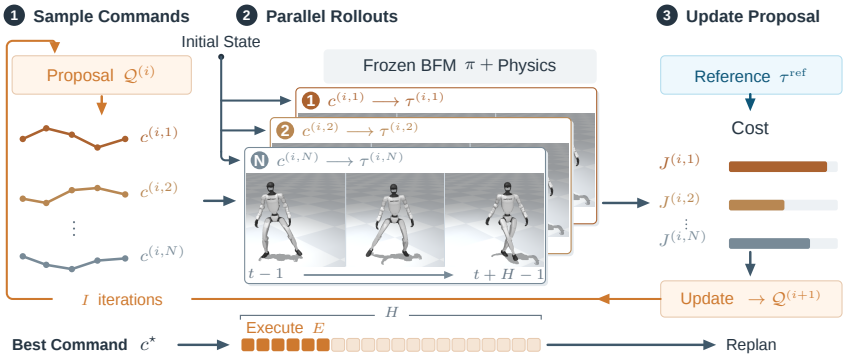

Figure 3: **Overview of test-time command optimization.** We evaluate N candidates through parallel H-step rollouts from the same state and update the proposal using their local costs. After I iterations, we execute the first E steps of the lowest-cost candidate in simulation, then replan.

the systematic deviation of a BFM, we can find an alternative command $c^\star$ that deviates from c^{nom} and produces an executed trajectory $\mathcal{T}_\pi(c^\star)$ that matches τ^{ref} more closely than $\tau^{\text{nom}} = \mathcal{T}_\pi(c^{\text{nom}})$.

Formally, we measure the tracking error of a candidate motion command c as $\mathcal{L}^{\text{track}}(c, \tau^{\text{ref}}) = \sum_{t=1}^{T} \ell_t^{\text{track}}(\mathcal{T}_\pi(c)_t, \tau_t^{\text{ref}})$. To penalize excessive command changes and jitter, we add a regularization term $\mathcal{L}^{\text{reg}}(c, \tau^{\text{ref}}) = \sum_{t=1}^{T} \ell_t^{\text{reg}}(c_t, \mathcal{T}_\pi(c)_t, \tau_t^{\text{ref}})$ and define the cost function as $J(c, \tau^{\text{ref}}) = \mathcal{L}^{\text{track}}(c, \tau^{\text{ref}}) + \mathcal{L}^{\text{reg}}(c, \tau^{\text{ref}})$. The optimal motion command is then given by

$$c^\star = \operatorname*{argmin}_{c \in \mathcal{M}_{1:T}} J(c, \tau^{\text{ref}}). \tag{1}$$

We refer to this optimization as the *full-sequence command-optimization problem*. Exact definitions of all loss terms are provided in Appendix B.1.

3.2 Test-Time Command Optimization

Receding-Horizon Command Optimization. Direct optimization of the non-differentiable, long-horizon problem is intractable: the number of decision variables grows with T, and each candidate evaluation requires T simulated BFM transitions. We therefore approximate $c^\star$ using receding-horizon optimal control (Rawlings et al., 2020). For a length-T reference motion, we solve a sequence of finite-horizon local problems: at each planning step, we optimize an H-step motion-command segment, execute its first E steps in simulation, and formulate the next local problem from the resulting state (Figure 3).

At time t, the local problem is formulated as follows:

$$c^\star_{t:t+H-1} = \operatorname*{argmin}_{c_{t:t+H-1} \in \mathcal{M}_{t:t+H-1}} J_{\text{local}}(c_{t:t+H-1}, z_{t-1}, \tau^{\text{ref}}_{t:t+H-1}), \tag{2}$$

$$J_{\text{local}} = \sum_{h=0}^{H-1} \left(\ell_{t+h}^{\text{track}} + \ell_{t+h}^{\text{reg}}\right) + \ell_{t+H-1}^{\text{term}}, \tag{3}$$

$$\text{s.t.} \qquad z_k = F_\pi(z_{k-1}, c_k), \qquad k = t, \ldots, t+H-1. \tag{4}$$

The terminal cost $\ell^{\text{term}}_{t+H-1}$ includes the joint- and body-position tracking errors at the end of the planning horizon. It serves as a surrogate that accounts for the effect of costs beyond the finite planning horizon. For notational simplicity, we omit the arguments of the cost terms when they are clear from context.

Sampling-based Optimization. We solve each local problem with a sampling-based optimizer (Jordana et al., 2025). We evaluate each candidate command by running the BFM in a physics simulator. To simplify the optimization, we define the decision vector x as the command values at control knots spaced H_k steps apart and reconstruct the complete motion-command segment through interpolation, $c = \Phi(x)$ (Howell et al., 2022), with details in Appendix B.2.

The optimizer performs I iterations of the following three steps. (1) Sampling: At iteration i, it samples N candidates $x^{(i,n)} \sim \mathcal{Q}^{(i)}$, $n = 1, \ldots, N$, from the current proposal distribution $\mathcal{Q}^{(i)}$.

(2) Rollout: Each candidate is interpolated into $c^{(i,n)} = \Phi(x^{(i,n)})$ and rolled out from z_{t-1} through the BFM and physics simulator to evaluate its local cost $J^{(i,n)} = J_{\text{local}}(c^{(i,n)}, z_{t-1}, \tau^{\text{ref}}_{t:t+H-1})$. (3) Update: The resulting candidate–cost pairs are used to update the proposal distribution for the next iteration, $\mathcal{Q}^{(i+1)} = \text{Update}(\mathcal{Q}^{(i)}, \{x^{(i,n)}, J^{(i,n)}\}_{n=1}^{N})$. After I iterations, we execute the first E steps of the lowest-cost candidate found across these iterations.

We consider five sampling-based optimizers: Cross-Entropy Method (CEM) (de Boer et al., 2005), Predictive Sampling (PS) (Howell et al., 2022), Model Predictive Path Integral with covariance matrix adaptation (MPPI-CMA) (Jordana et al., 2025), CMA-ES (Hansen & Ostermeier, 2001), and Consensus-based Optimization (CBO) (Sun et al., 2026). Based on the analysis in Secs. 3.3 and 4.2, CEM achieves competitive performance overall and is selected for the subsequent analysis and experiments.

3.3 Scaling Test-Time Compute Optimally

Test-Time Compute-Optimal Strategy. Finite horizons and finite sampling make the problem tractable, but can increase execution cost relative to the full-sequence optimum. Longer horizons(H) account for more future cost, more frequent replanning(smaller E) revises the plan earlier, and more iterations(I) or candidates(N) can improve local search. These choices compete for a given compute budget. We therefore seek the algorithm and compute configuration that minimize full-sequence execution cost at each budget. For compute configuration $\xi = (I, N, H, E) \in \mathcal{X}$, where $\mathcal{X} = \{(I, N, H, E) \in \mathbb{N}_+^4 : 1 \le E \le H\}$, a length-$T$ motion requires $\lceil T/E \rceil$ planning calls, each using INH simulated BFM transitions to evaluate candidates. Thus $C(\xi, T) = \lceil T/E \rceil INH$ measures the total compute. To compare compute across motion lengths, we define the *compute budget ratio* $K(\xi) = INH/E$, so $C(\xi, T) \approx TK(\xi)$.

Let $J_{\text{TTC}}(\mathcal{A}, \xi)$ denote the expected full-sequence cost J per time step, averaged over the motion dataset, when using algorithm $\mathcal{A}$ and configuration ξ. The compute-optimal strategy maps each budget K to an algorithm and a configuration that solve

$$(\mathcal{A}^\star(K), \xi^\star(K)) = \underset{\mathcal{A},\, \xi \in \mathcal{X}:\, K(\xi) \le K}{\text{argmin}} J_{\text{TTC}}(\mathcal{A}, \xi). \tag{5}$$

We call $\xi^\star(K)$ the *compute-optimal configuration*.

Modeling Test-Time Performance. An exhaustive search over $(\mathcal{A}, \xi)$ is computationally prohibitive, since evaluating J_{TTC} for each configuration requires command optimization over the motion dataset. We therefore construct a heuristic *surrogate cost model* to predict costs for untested configurations from limited measurements. Its structure follows the finite-horizon and sampling approximations in Sec. 3.2. We represent the irreducible cost of the full-sequence problem by J_∞. The finite-horizon error $\epsilon_{\text{horizon}}$ can remain with exact local solutions; we model it as independent of $(\mathcal{A}, I, N)$. Finite local search adds the sampling error ϵ_{sample}:

$$\widehat{J}_{\text{TTC}}(\mathcal{A}, \xi) = J_\infty + \epsilon_{\text{horizon}}(H, E) + \epsilon_{\text{sample}}(\mathcal{A}, I, N \mid H, E). \tag{6}$$

The terminal cost in J_{local} approximates costs beyond H. Kunisch & Pfeiffer (2020) prove that, in their stabilization setting, the cost gap caused by this approximation has an upper bound that decays exponentially with $2H - E$ (in our notation). This motivates an exponential term that decreases with longer planning horizons and shorter execution intervals. Executing more steps before replanning can also reduce robustness to disturbances (Grüne & Palma, 2015). We account for this dependence with an empirical linear correction, giving the finite-horizon error model

$$\epsilon_{\text{horizon}}(H, E) = C_H \exp[-\alpha_H(2H - E)] + C_E(E - 1). \tag{7}$$

To model the sampling error, we first isolate the effect of solver effort (I, N) at fixed (H, E). Iterations refine the proposal distribution; candidates explore the command space within each update. At a fixed IN, more iterations leave fewer candidates per update. We represent this dependence by the solver-dependent error ϵ_{solver}, using additive inverse-power terms to capture diminishing returns (Hoffmann et al., 2022):

$$\epsilon_{\text{solver}}(\mathcal{A}, I, N) = C_I(\mathcal{A}) I^{-\alpha_I(\mathcal{A})} + C_N(\mathcal{A}) N^{-\alpha_N(\mathcal{A})}. \tag{8}$$

We next account for the effect of H on sampling error. With control-knot spacing H_k fixed, the dimension d of the decision vector x grows approximately in proportion to H. For Lipschitz

Table 1: **TTCO reduces mean cost and global position error at the price of higher joint acceleration, across three BFMs on AMASS.** Each entry reports nominal → optimized performance, averaged over all motions per BFM.

BFM	Mean cost ↓	Cost-win rate (%) ↑	Body-position error (m) ↓	Root-position error (m) ↓	Joint acceleration (rad/s^2) ↓
HEFT	0.419 → 0.046	62.27	0.271 → 0.057	0.264 → 0.048	10.104 → 17.246
SONIC	6.047 → 0.323	86.00	0.615 → 0.127	0.609 → 0.117	8.832 → 11.999
TWIST2	5.321 → 0.573	99.67	1.095 → 0.173	1.079 → 0.164	43.280 → 51.574

optimization with a fixed Lipschitz constant, the minimax rate $n^{-1/d}$ for n evaluations motivates raising ϵ_{solver} to η/H (Malherbe & Vayatis, 2017). This makes improvement with solver effort slower at longer horizons. A separate factor γ^H allows the error scale to vary with H, giving

$$\epsilon_{\text{sample}}(\mathcal{A}, I, N \mid H, E) = \epsilon_{\text{solver}}(\mathcal{A}, I, N)^{\eta/H}\gamma^H. \tag{9}$$

With positive parameters and fixed (H, E), the sampling error vanishes as $I, N \to \infty$, leaving $J_\infty + \epsilon_{\text{horizon}}$.

With measurements from different configurations, we can fit the surrogate cost model by minimizing the mean squared difference between predicted and measured log costs. Minimizing the fitted model under the same compute budget approximately solves Eq. 5:

$$(\widehat{\mathcal{A}}^\star(K), \widehat{\xi}^\star(K)) = \operatorname*{argmin}_{\mathcal{A},\, \xi \in \mathcal{X}:\, K(\xi) \le K} \widehat{J}_{\text{TTC}}(\mathcal{A}, \xi). \tag{10}$$

In our experiments, we restrict configurations to the practical search range specified in Appendix C.4.1.

4 Experiments

We address four questions:

- **Q1:** Does command optimization improve tracking across pretrained BFMs at scale?
- **Q2:** How does the allocation of test-time compute affect tracking performance?
- **Q3:** Can the cost model find compute-optimal configurations across budgets, BFMs, and motions?
- **Q4:** Does command optimization improve downstream loco-manipulation policies?

Experimental setup. We evaluate SONIC (Luo et al., 2026), HEFT (Liu et al., 2026a), and TWIST2 (Ze et al., 2026) on G1. MuJoCo Warp (Todorov et al., 2012; Newton Developers, n.d.) supports tracking and command optimization; simulation for loco-manipulation uses Isaac Sim.

4.1 Test-Time Command Optimization Improves Tracking Precision (Q1)

For each pretrained BFM, we compare tracking performance with nominal and optimized commands on the retargeted AMASS motion dataset (Mahmood et al., 2019). Appendix C.2 provides the evaluation protocol and metric definitions.

Command optimization reduces mean cost by 89.0–94.7% and mean global body-position error by 79.1–84.2% across all three BFMs (Table 1), with cost-win rates of 99.67% for TWIST2 and 86.00% for SONIC. Joint acceleration increases for all three BFMs, indicating a trade-off between tracking accuracy and motion smoothness. Note that we use a moderate compute budget ratio of K=640 to limit the cost of evaluating the full dataset; the results in Sections 4.2 and 4.3 suggest that larger budgets may further improve performance. As shown in the two selected real-robot sequences in Figure 4, optimized commands bring the resulting robot positions closer to the reference.

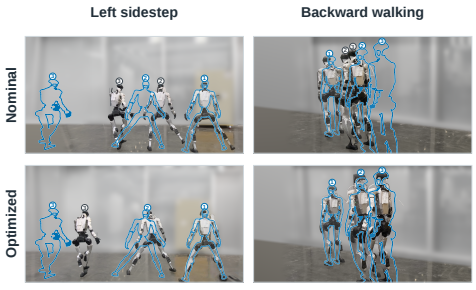

Figure 4: **Real-robot demonstration of motion tracking with nominal and optimized commands.** We compare nominal and optimized commands for left sidestep and backward walking executed open-loop on the real robot. In both examples, optimized commands bring the final displayed positions closer to the reference. Blue outlines show the reference motion.

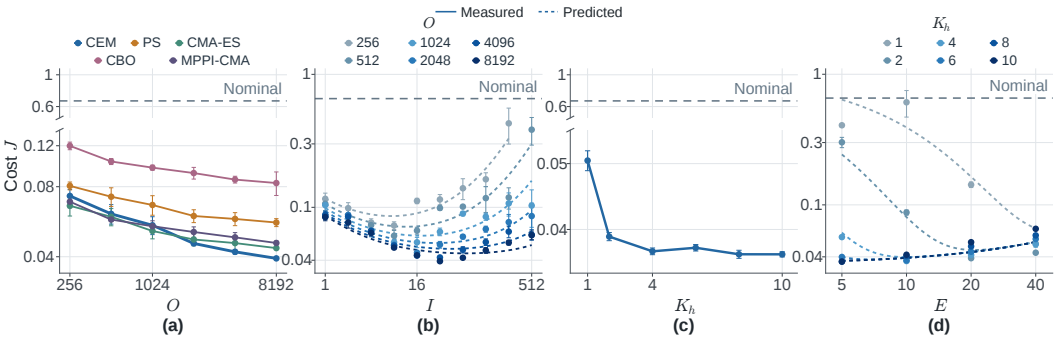

Figure 5: **Compute allocation governs performance through solver and time horizon parameters.** We examine these two parameter groups separately. **(a,b) Solver parameters** $(\mathcal{A}, I, N)$**.** (a) Each point reports the lowest measured cost over all tested (I, N) configurations for that algorithm at that budget. CEM achieves competitive performance overall, so we restrict the remaining analysis to it. (b) At fixed $O = IN$, its cost first falls and then rises as iterations increase and candidates per iteration decrease; the best measured configuration changes with the budget. **(c,d) Time horizon parameters** (H, E)**.** (c) The best measured cost over E levels off beyond a horizon ratio $K_h = H/E$ of 4. (d) Increasing E can either lower or raise cost depending on K_h, motivating joint selection of H and E. In (b,d), model curves follow the measurements, supporting the proposed forms of the solver and horizon error terms. Error bars show the standard deviation of the three seed means; horizontal dashed lines mark nominal-command cost.

4.2 Compute Allocation Governs the Gains (Q2)

We examine the trade-offs that govern compute allocation by separately varying solver parameters $(\mathcal{A}, I, N)$ and time horizon parameters (H, E), following the error decomposition in Sec. 3.3. Both sweeps use SONIC on the same 10 randomly sampled AMASS motions. The solver sweep fixes $(H, E) = (40, 20)$ and compares algorithms and iteration–candidate splits at each budget $O = IN$. To study the horizon error in isolation, we vary (H, E) and run CEM to convergence with $N = 512$, approximately treating the remaining solver error as negligible ($\epsilon_{\text{solver}} \approx 0$).

Solver parameters $(\mathcal{A}, I, N)$ **(Fig. 5(a,b)).** CEM achieves the lowest cost at the three largest budgets and remains competitive at the three smaller ones, so we restrict the remaining analysis to CEM. At a fixed solver budget $O = IN$, cost first decreases and then increases as more iterations leave fewer candidates per iteration. For example, at $O = 8192$, changing (I, N) from $(1, 8192)$ to $(32, 256)$ reduces cost by 53.42% without increasing compute. The best measured configuration also varies with the budget. The surrogate model captures this trade-off and fits costs with a mean absolute relative error of 11.38% (Fig. 5(b)), supporting our model of solver error.

Time horizon parameters (H, E) **(Fig. 5(c,d)).** The lowest measured cost across the tested E values levels off beyond $K_h = 4$ (Fig. 5(c)). The best measured configuration tends toward more frequent replanning as K_h increases, but the effect of E on cost varies across horizon ratios. For example, increasing E from 5 to 10 reduces cost by 34.43% at $K_h = 4$ and increases it by 12.87% at $K_h = 10$. These contrasting effects motivate joint selection of H and E. The horizon term of the surrogate model fits the measured costs with a mean absolute relative error of 12.77% (Fig. 5(d)), supporting our model of horizon error.

4.3 The Cost Model Estimates a Test-Time Compute Strategy (Q3)

We now use the surrogate cost model to derive an approximate compute-optimal strategy and ask whether it remains effective across BFMs and motions. To fit the full model, we combine configuration–cost pairs from the sweeps in Sec. 4.2 with measurements from additional jointly sampled configurations to capture interactions between solver and time horizon parameters, yielding 183 configuration–cost pairs in total. For comparison, we also fit Random Forest, Radial Basis Function (RBF), and Gaussian Process baselines on these data. At each budget K, we evaluate the configuration proposed by each of the four models alongside multiple random configurations satisfying $0.9K \leq K(\xi) \leq K$. We compare strategies using Regret: the percentage by which a

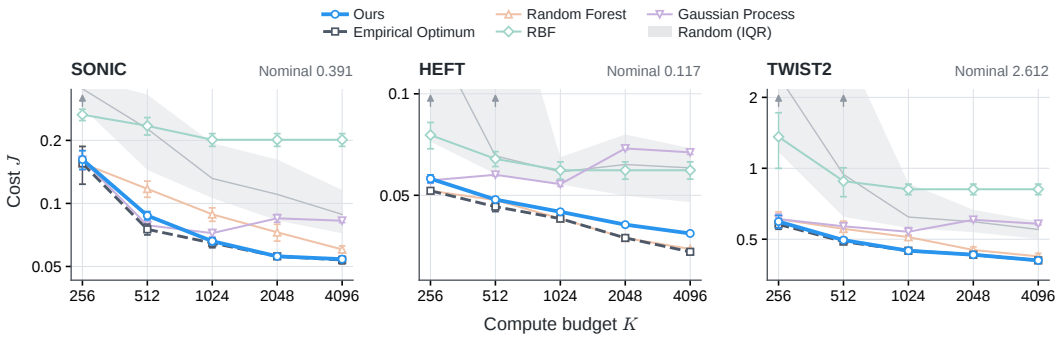

Figure 6: **The proposed strategy generalizes to unseen motions and BFMs.** Our strategy stays close to the empirical optimum and achieves the lowest average regret among the four strategies across three BFMs and five compute budgets. Faint lines and bands show median costs and 25th–75th percentiles across random configurations; nominal-command costs are labeled above the panels. Costs are averaged over five motions and three seeds; error bars show standard deviations of seed means. Each panel uses logarithmic axes.

strategy's cost exceeds that of the best measured configuration (Empirical Optimum) at the same budget. Appendix C.4 describes the fitting and evaluation procedures.

The surrogate cost model identifies a near-optimal strategy in the fitting regime. We compare the proposed configuration with 200 random configurations at the same compute budget, on SONIC and on the motions used to fit the cost model. Our strategy achieves the lowest mean regret of 4.02% among the four strategies across six budgets, compared with 18.44% for random forest, the next-best strategy. Scaling test-time compute from $K = 256$ to $K = 8192$ consistently lowers the measured cost (Fig. 2) under the proposed strategy, a total reduction of 58.0%.

The proposed strategy generalizes across BFMs and motions. We further apply the proposed strategy unchanged to SONIC, HEFT, and TWIST2 on five new motions, comparing its proposed configurations with 50 random configurations (Fig. 6). Across the three BFMs and five budgets, our strategy achieves the lowest mean regret: 4.60%, compared with 12.27% for random forest. Its mean regrets are 4.72%, 8.24%, and 0.86% on SONIC, HEFT, and TWIST2, respectively, making it the only strategy below 10% on all three BFMs. These results indicate that our surrogate cost model, fitted to limited measurements from a single BFM, can select near-optimal configurations outside its fitting regime.

4.4 The Benefits of Test-Time Compute Extend to Loco-Manipulation (Q4)

A high-level humanoid policy maps visual observations and optional language instructions to motion commands executed by a BFM. Learning this mapping requires demonstration labels compatible with that interface (Appendix A.2). We test whether optimized commands can serve as effective labels in both simulation and real-robot scenarios. We keep demonstration images, proprioception, and hand labels fixed and compare three body-action label sources: *original commands* issued during collection (Original), recorded *robot states* expressed as commands (State), and *optimized commands* (Ours). For Ours, we optimize commands offline with the BFM that will execute the learned policy's commands (target BFM), using the recorded robot motion as the reference. We randomize simulation dynamics during optimization to improve robustness (Appendix C.5.2). We also test cross-BFM settings, where the BFM used to collect demonstrations (source BFM) differs from the target BFM.

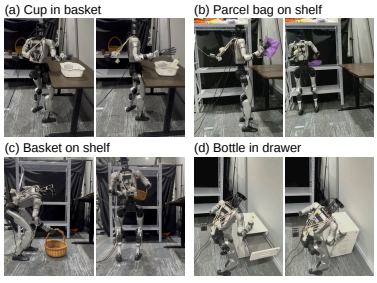

Figure 7: **Illustration of real-robot loco-manipulation tasks.**

For each label source, we train Diffusion Policy (Chi et al., 2025) on five HumanoidArena tasks (Wang et al., 2026a) following the benchmark protocol (Appendix C.5.3), and fine-tune GR00T N1.7 (NVIDIA, 2026) on SONIC demonstrations of four real-robot loco-manipulation tasks (Fig. 7; Appendix C.5.4).

Table 2: **Simulation task success rates (%) on HumanoidArena.** Each arrow points from the source BFM to the target BFM. Results are reported as mean ± sample standard deviation across three training seeds, with 60 evaluation rollouts per task and seed.

Labels	Boxing	Open Door	Pick & Place Box	Sit Sofa	Vision Navigation	Mean
SONIC → SONIC						
Original	$83.9_{\pm 1.0}$	$84.4_{\pm 1.9}$	$61.1_{\pm 1.0}$	$88.9_{\pm 1.0}$	$60.6_{\pm 1.0}$	$75.8_{\pm 0.5}$
Ours	$63.9_{\pm 1.0}$	$51.1_{\pm 1.0}$	$54.4_{\pm 1.0}$	$66.7_{\pm 1.7}$	$47.2_{\pm 1.0}$	$56.7_{\pm 0.3}$
State	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
SONIC → TWIST2						
Original	$48.9_{\pm 1.9}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$17.8_{\pm 1.9}$	$11.7_{\pm 1.7}$	$15.7_{\pm 0.6}$
Ours	$50.0_{\pm 1.7}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$21.7_{\pm 1.7}$	$7.2_{\pm 1.9}$	$15.8_{\pm 0.4}$
State	$33.9_{\pm 1.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$14.4_{\pm 1.0}$	$1.7_{\pm 1.7}$	$10.0_{\pm 0.3}$
SONIC → HEFT						
Original	$46.7_{\pm 1.7}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$22.8_{\pm 1.9}$	$16.1_{\pm 1.0}$	$17.1_{\pm 0.4}$
Ours	$57.2_{\pm 1.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$67.8_{\pm 1.0}$	$18.9_{\pm 1.0}$	$28.8_{\pm 0.2}$
State	$38.9_{\pm 1.9}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$36.1_{\pm 1.0}$	$27.8_{\pm 1.9}$	$20.6_{\pm 0.7}$

Table 3: **Real-robot task success rates (%).** Each arrow points from the source BFM to the target BFM. Each task entry reports the mean success rate over 60 evaluation rollouts.

	SONIC → SONIC			**SONIC → HEFT**		
Task	Original	**Ours**	State	Original	**Ours**	State
Cup in basket	55.0	55.0	0.0	0.0	40.0	13.3
Parcel bag on shelf	83.3	91.7	10.0	6.7	55.0	20.0
Basket on shelf	53.3	50.0	0.0	0.0	50.0	8.3
Bottle in drawer	50.0	60.0	0.0	0.0	50.0	6.7
Mean	60.4	64.2	2.5	1.7	48.8	12.1

Optimized commands serve as effective action labels. When the source and target BFMs are the same, policies trained on original commands provide a performance baseline because those commands generated the recorded motion. Command optimization uses the frozen BFM to infer alternative command labels from that motion. Policies trained on these labels achieve higher mean success rates than those trained on robot-state labels and approach the performance of policies trained on original commands in simulation and on hardware (Tables 2 and 3). This performance recovery supports optimized commands as a partial substitute for the original action labels.

Command optimization enables demonstration reuse across BFMs. When the target BFM changes, re-optimizing the command labels preserves the recorded motion as the reference while adapting supervision to the new controller. The resulting policies outperform or approach both original-command and robot-state baselines in mean success rate for most simulation pairings and on all four real-robot tasks (Tables 3 and 25). They also retain part of the success rate of policies trained on optimized labels for the source BFM and evaluated with that same BFM. This indicates that label adaptation reduces dependence on the source BFM.

5 Conclusion

Our study shows that an existing humanoid BFM can track a motion more accurately when its command differs from the desired motion. Test-Time Command Optimization finds such commands, with gains governed by both compute budget and allocation. The same optimization recovers useful action labels from recorded motions, extending the benefits to downstream learning and demonstration reuse across BFMs. By bringing the distinction between desired motion and input command into focus, our study reveals substantial room to improve existing BFMs through test-time compute.

Limitations. Our current implementation is not yet fast enough to deliver satisfactory real-time performance, limiting our study of closed-loop command optimization on real robots. Optimizing motion commands improves tracking accuracy but can introduce temporal irregularities in the commands and jitter in the executed motion. Residual execution errors can still limit downstream tasks that require precise contact and coordinated whole-body movement.

AI USE STATEMENT

In this work, we used generative AI tools for designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis. We have not used generative AI tools for generating synthetic datasets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, or interpreting results. Additionally, we used generative AI tools for creating or modifying scientific figures or images, creating or editing software code, drafting parts of the paper, and editing the paper to improve readability. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we provide the code and example data as part of the supplementary material. We further expand upon the experimental details explained in the main text in Appendix B and C.

REFERENCES

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Jianuo Cao, Yuxin Chen, Yuzhen Song, Masayoshi Tomizuka, Chenran Li, and Thomas Tian. TEXEDO: Test time scaling for controller-aware language-conditioned humanoid motion generation. *arXiv preprint arXiv:2606.22998*, 2026.

Maiyue Chen, Kaihui Wang, Bo Zhang, Xihan Ma, Zhiyuan Yang, Yi Ren, Qijun Huang, Zihao Zhu, Yucheng Wang, and Zhizhong Su. HoloMotion-1 technical report, 2026. URL https://arxiv.org/abs/2605.15336.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668. URL https://doi.org/10.1177/02783649241273668.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005. doi: 10.1007/s10479-005-5724-z.

Victor Dhédin, Ilyass Taouil, Shafeef Omar, Dian Yu, Kun Tao, Angela Dai, and Majid Khadiv. DynaRetarget: Dynamically-feasible retargeting using sampling-based trajectory optimization. *IEEE Robotics and Automation Letters*, pp. 1–8, 2026. doi: 10.1109/LRA.2026.3730386.

William Edwards, Gao Tang, Giorgos Mamakoukas, Todd Murphey, and Kris Hauser. Automatic tuning for data-driven model predictive control. In *IEEE International Conference on Robotics and Automation*, 2021. URL https://motion.cs.illinois.edu/papers/ICRA2021_Edwards_AutoMPC.pdf.

Marco Forgione, Dario Piga, and Alberto Bemporad. Efficient calibration of embedded MPC. *IFAC-PapersOnLine*, 53(2), 2020. doi: 10.1016/j.ifacol.2020.12.1188.

Lars Grüne and Vryan Gil Palma. Robustness of performance and stability for multistep and updated multistep MPC schemes. *Discrete and Continuous Dynamical Systems*, 35(9):4385–4414, 2015. doi: 10.3934/dcds.2015.35.4385.

Jinrui Han, Weiji Xie, Jiakun Zheng, Jiyuan Shi, Weinan Zhang, Ting Xiao, and Chenjia Bai. KungfuBot2: Learning versatile motion skills for humanoid whole-body control. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026. URL https://arxiv.org/abs/2509.16638.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022. doi: 10.52202/068431-2176. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c1e2faff6f588870935f114ebe04a3e5-Abstract-Conference.html.

Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive sampling: Real-time behaviour synthesis with MuJoCo. *arXiv preprint arXiv:2212.00541*, 2022.

Yingdong Hu, Haodong Zhu, Boyuan Zheng, Yihang Hu, Tong Zhang, Zunhao Chen, Junming Zhao, Ruiqian Nai, and Yang Gao. OpenHLM: An empirical recipe for whole-body humanoid loco-manipulation, 2026. URL https://arxiv.org/abs/2606.22174.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002. ISSN 1046-8188. doi: 10.1145/582415.582418. URL https://doi.org/10.1145/582415.582418.

Haoran Jiang, Jin Chen, Qingwen Bu, Li Chen, Modi Shi, Yanjie Zhang, Delong Li, Chuanzhe Suo, Chuang Wang, Zhihui Peng, and Hongyang Li. WholeBodyVLA: Towards unified latent VLA for whole-body loco-manipulation control. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ff7373914a96956f2a7cacbdf3b0b8d8-Abstract-Conference.html.

Armand Jordana, Jianghan Zhang, Joseph Amigo, and Ludovic Righetti. An introduction to zero-order optimization techniques for robotics. *arXiv preprint arXiv:2506.22087*, 2025.

M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 06 1938. ISSN 0006-3444. doi: 10.1093/biomet/30.1-2.81. URL https://doi.org/10.1093/biomet/30.1-2.81.

Karl Kunisch and Laurent Pfeiffer. The effect of the terminal penalty in Receding Horizon Control for a class of stabilization problems. *ESAIM: Control, Optimisation and Calculus of Variations*, 26:58, 2020. doi: 10.1051/cocv/2019037. URL https://numdam.org/articles/10.1051/cocv/2019037/.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. RoboMonkey: Scaling test-time sampling and verification for vision-language-action models. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3200–3217. PMLR, 2025.

Andrew Zou Li, Gokul Swamy, Yonatan Bisk, and Andrea Bajcsy. ELASTIC: Efficiently learning to adaptively scale test-time compute for generative control policies. *arXiv preprint arXiv:2606.31132*, 2026a.

Yitang Li, Zhengyi Luo, Tonghe Zhang, Cunxi Dai, Anssi Kanervisto, Andrea Tirinzoni, Haoyang Weng, Kris Kitani, Mateusz Guzek, Ahmed Touati, Alessandro Lazaric, Matteo Pirotta, and Guanya Shi. BFM-Zero: A promptable behavioral foundation model for humanoid control using unsupervised reinforcement learning. In *International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=jkhl2oI0g5.

Yixuan Li, Le Ma, Yutang Lin, Yushi Du, Mengya Liu, Kaizhe Hu, Jieming Cui, Yixin Zhu, Wei Liang, Baoxiong Jia, and Siyuan Huang. OmniClone: Engineering a robust, all-rounder whole-body humanoid teleoperation system. *arXiv preprint arXiv:2603.14327*, 2026c.

Qiayuan Liao, Takara E. Truong, Xiaoyu Huang, Yuman Gao, Guy Tevet, Koushil Sreenath, and C. Karen Liu. BeyondMimic: From motion tracking to versatile humanoid control via guided diffusion. *Science Robotics*, 11(117):eadx8924, 2026. doi: 10.1126/scirobotics.adx8924.

Chenxin Liu, Qingzhou Lu, Guangxiao Yang, Xuanyang Shi, Chenghan Yang, Yanjiang Guo, and Jianyu Chen. HEFT: Heavy-payload full-size humanoid teleoperation with privileged motion guidance and windowed payload curriculum. *arXiv preprint arXiv:2607.02332*, 2026a.

Xin Liu, Yulin Li, Ziming Li, Pengyu Jing, Zhenhao Huang, Bingyang Zhou, Ziqiu Zeng, Siyuan Luo, Chenkun Qi, and Fan Shi. Enabling robust cloth manipulation via inference-time simulator-in-the-loop refinement. *arXiv preprint arXiv:2606.24552*, 2026b.

Zhengyi Luo, Ye Yuan, Tingwu Wang, Chenran Li, Fernando Castañeda, Sirui Chen, Zi-Ang Cao, Jiefeng Li, David Minor, Qingwei Ben, Jinhyung Park, David Sami, Zi Wang, Xingye Da, Runyu Ding, Cyrus Hogg, Lina Song, Edy Lim, Eugene Jeong, Tairan He, Haoru Xue, Wenli Xiao, Simon Yuen, Jan Kautz, Yan Chang, Umar Iqbal, Linxi "Jim" Fan, and Yuke Zhu. SONIC: Supersizing motion tracking for natural humanoid whole-body control. *Science Robotics*, 11(117):eaed4592, 2026. doi: 10.1126/scirobotics.aed4592.

Yubiao Ma, Han Yu, Jiayin Xie, Changtai Lv, Qiang Luo, Chi Zhang, Yunpeng Yin, Boyang Xing, Xuemei Ren, and Dongdong Zheng. Robust and generalized humanoid motion tracking. *arXiv preprint arXiv:2601.23080*, 2026.

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5442–5451, 2019.

Cédric Malherbe and Nicolas Vayatis. Global optimization of Lipschitz functions. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2314–2323. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/malherbe17a.html.

Newton Developers. MuJoCo Warp (MJWarp). https://github.com/google-deepmind/mujoco_warp, n.d. Computer software. Accessed: 2026-09-22.

NVIDIA. NVIDIA Isaac GR00T N1.7. https://huggingface.co/collections/nvidia/gr00t-n17, 2026. Official model collection, accessed September 14, 2026.

Chaoyi Pan, Changhao Wang, Haozhi Qi, Zixi Liu, Homanga Bharadhwaj, Akash Sharma, Tingfan Wu, Guanya Shi, Jitendra Malik, and Francois Hogan. SPIDER: Scalable physics-informed dexterous retargeting. *arXiv preprint arXiv:2511.09484*, 2025.

Zekun Qi, Xuchuan Chen, Jilong Wang, Chenghuai Lin, Yunrui Lian, Wenyao Zhang, Xinqiang Yu, He Wang, and Li Yi. Humanoid generative pre-training for zero-shot motion tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20834–20844, 2026.

James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2 edition, 2020. URL https://sites.engineering.ucsb.edu/~jbraw/mpc/.

Raymond Ros and Nikolaus Hansen. A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity. In *10th International Conference on Parallel Problem Solving From Nature*, Dortmund, Germany, September 2008. URL https://inria.hal.science/inria-00287367.

Amrith Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. Scaling test-time compute without verification or RL is suboptimal. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54058–54094. PMLR, 2025.

Jean Pierre Sleiman, He Li, Alphonsus Adu-Bredu, Robin Deits, Arun Kumar, Kevin Bergamin, Mohak Bhardwaj, Scott Biddlestone, Nicola Burger, Matthew A. Estrada, Francesco Iacobelli, Twan Koolen, Alexander Lambert, Erica Lin, M. Eva Mungai, Zach Nobles, Shane Rozen-Levy, Yuyao Shi, Jiashun Wang, Jakob Welner, Fangzhou Yu, Mike Zhang, Alfred Rizzi, Jessica Hodgins, Sylvain Bertrand, Yeuhi Abe, Scott Kuindersma, and Farbod Farshidian. ZEST: Zero-shot embodied skill transfer for athletic robot control. *Science Robotics*, 11(117):eaec7695, 2026. doi: 10.1126/scirobotics.aec7695. URL https://www.science.org/doi/abs/10.1126/scirobotics.aec7695.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025.

C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. ISSN 00029556. URL http://www.jstor.org/stable/1412159.

Xudong Sun, Armand Jordana, Massimo Fornasier, Jalal Etesami, and Majid Khadiv. Consensus-based optimization (CBO): Towards global optimality in robotics. *arXiv preprint arXiv:2602.06868*, 2026.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Taowen Wang, Zikang Xie, Bin Yang, Yunheng Wang, Zizhao Yuan, Yuetong Fang, Yixiao Feng, Yichi Wang, Xingyu Chen, Haodong Chen, Qiwei Wu, Weisheng Xu, Lihan Chen, Lusong Li, Zecui Zeng, and Renjing Xu. HumanoidArena: Benchmarking egocentric hierarchical whole-body learning. *arXiv preprint arXiv:2606.17833*, 2026a. URL https://arxiv.org/abs/2606.17833.

Yen-Jen Wang, Jiaman Li, Sirui Chen, Takara E. Truong, Pei Xu, Pieter Abbeel, Rocky Duan, Koushil Sreenath, Angjoo Kanazawa, Carmelo Sferrazza, Guanya Shi, and Karen Liu. VLK: Learning humanoid loco-manipulation from synthetic interactions in reconstructed scenes, 2026b. URL https://arxiv.org/abs/2606.30645.

Yunshen Wang, Shaohang Zhu, Peiyuan Zhi, Yuhan Li, Jiaxin Li, Yong-Lu Li, Yuchen Xiao, Xingxing Wang, Baoxiong Jia, and Siyuan Huang. OmniXtreme: Breaking the generality barrier in high-dynamic humanoid control, 2026c. URL https://arxiv.org/abs/2602.23843.

Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018. doi: 10.1109/TRO.2018.2865891.

Bingqian Wu, Zicheng Xu, Xianghui Fan, Dayu Li, and Xiangru Huang. Teleopit: A full-embodiment humanoid teleoperation system, 2026. URL https://arxiv.org/abs/2608.01834.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *International Conference on Learning Representations*, 2025.

Tianyi Xie, Haotian Zhang, Jinhyung Park, Zi Wang, Bowen Wen, Jiefeng Li, Xueting Li, Qingwei Ben, Haoyang Weng, Yufei Ye, David Minor, Tingwu Wang, Chenfanfu Jiang, Sanja Fidler, Jan Kautz, Linxi Fan, Yuke Zhu, Zhengyi Luo, Umar Iqbal, and Ye Yuan. GRAIL: Generating humanoid loco-manipulation from 3D assets and video priors, 2026. URL https://arxiv.org/abs/2606.05160.

Weiji Xie, Jinrui Han, Jiakun Zheng, Huanyu Li, Xinzhe Liu, Jiyuan Shi, Weinan Zhang, Chenjia Bai, and Xuelong Li. KungfuBot: Physics-based humanoid whole-body control for learning highly-dynamic skills. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2089.

Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based MPC for torque-level locomotion control via diffusion-style annealing. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4974–4981, 2025. doi: 10.1109/ICRA55743.2025.11127320.

Kangning Yin, Weishuai Zeng, Ke Fan, Minyue Dai, Zirui Wang, Qiang Zhang, Zheng Tian, Jingbo Wang, Jiangmiao Pang, and Weinan Zhang. UniTracker: Learning universal whole-body motion tracker for humanoid robots. *IEEE Robotics and Automation Letters*, 11(7):8124–8131, 2026. doi: 10.1109/LRA.2026.3692091. URL https://doi.org/10.1109/LRA.2026.3692091.

Mingqi Yuan, Tao Yu, Wenqi Ge, Xiuyong Yao, Dapeng Li, Huijiang Wang, Jiayu Chen, Bo Li, Wei Zhang, Wenjun Zeng, Hua Chen, and Xin Jin. A survey of behavior foundation model: Next-generation whole-body control system of humanoid robots. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(4):4909–4927, 2026. doi: 10.1109/TPAMI.2025.3649177. URL https://doi.org/10.1109/TPAMI.2025.3649177.

Yanjie Ze, Siheng Zhao, Weizhuo Wang, Angjoo Kanazawa, Rocky Duan, Pieter Abbeel, Guanya Shi, Jiajun Wu, and C. Karen Liu. TWIST2: Scalable, portable, and holistic humanoid data collection system. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026. URL https://arxiv.org/abs/2511.02832.

Weishuai Zeng, Shunlin Lu, Kangning Yin, Xiaojie Niu, Minyue Dai, Jingbo Wang, and Jiangmiao Pang. Behavior foundation model for humanoid robots. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026a. URL https://arxiv.org/abs/2509.13780.

Weishuai Zeng, Kangning Yin, Xiaojie Niu, Shunlin Lu, Weixiang Zhong, Jiahe Chen, Feiyu Jia, Xiao Chen, Zirui Wang, Furui Xu, Ming Zhou, Kailin Li, Weinan Zhang, He Wang, Li Yi, Dahua Lin, Jiangmiao Pang, and Jingbo Wang. Scaling behavior foundation model for humanoid robots. *arXiv preprint arXiv:2607.15163*, 2026b.

John Z. Zhang, Maks Sorokin, Jan Brüdigam, Brandon Hung, Stephen Phillips, Dmitry Yershov, Farzad Niroui, Tong Zhao, Leonor Fermoselle, Xinghao Zhu, Chao Cao, Duy Ta, Tao Pang, Jiuguang Wang, Preston Culbertson, Zachary Manchester, and Simon Le Cléac'h. Sumo: Dynamic and generalizable whole-body loco-manipulation. *arXiv preprint arXiv:2604.08508*, 2026.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

Tengjie Zhu, Guanyu Cai, Zhaohui Yang, Guanzhu Ren, Haohui Xie, ZiRui Wang, Junsong Wu, Jingbo Wang, Xiaokang Yang, Yao Mu, and Yichao Yan. CLOT: Closed-loop global motion tracking for whole-body humanoid teleoperation. *arXiv preprint arXiv:2602.15060*, 2026.

APPENDIX CONTENTS

A Preliminaries **16**
A.1 Humanoid motion tracking . . . 16
A.2 Learning Humanoid Visuomotor Policies . . . 17

B Method Details **18**
B.1 Rollout Cost . . . 18
B.2 Command Parameterization and Controller Inputs . . . 18
B.3 Sampling Optimizers . . . 19
B.4 Allocation Under the Fitted Model . . . 20

C Additional Experimental Setup and Results **21**
C.1 Experimental Setup . . . 21
C.2 Test-Time Command Optimization Improves Tracking Precision (Q1) . . . 21
C.2.1 AMASS Evaluation and Tracking Metrics . . . 21
C.2.2 Per-Motion Tracking Precision and Cost . . . 22
C.3 Compute Allocation Governs the Gains (Q2) . . . 22
C.3.1 Solver Parameters . . . 23
C.3.2 Time Horizon Parameters . . . 23
C.3.3 Fitting the Solver and Horizon Error Models . . . 23
C.4 The Cost Model Estimates a Test-Time Compute Strategy (Q3) . . . 24
C.4.1 Fitting the Surrogate Cost Model and Selecting Configurations . . . 24
C.4.2 Baselines . . . 26
C.4.3 Evaluation Metrics . . . 27
C.4.4 Compute Scaling on the Fitting Motions . . . 28
C.4.5 Generalization to New Motions and BFMs . . . 29
C.5 The Benefits of Test-Time Compute Extend to Loco-Manipulation (Q4) . . . 32
C.5.1 Label Comparison and Controller Interfaces . . . 32
C.5.2 Dynamics Randomization During Optimization . . . 32
C.5.3 HumanoidArena Data, Training, and Evaluation . . . 32
C.5.4 Real-Robot Data, Training, and Evaluation . . . 33
C.5.5 Complete Simulation Results . . . 34

D Compute Resources **36**

A PRELIMINARIES

Hierarchical humanoid control connects two learned policies: a motion controller converts motion commands and robot feedback into control actions, while a visuomotor policy predicts those commands from sensory observations and task instructions (Ze et al., 2026; Luo et al., 2026; Hu et al., 2026). The following formulation describes how each policy is learned and how their interfaces connect.

A.1 HUMANOID MOTION TRACKING

Humanoid motion tracking is the task of reproducing a reference motion through feedback control under the robot's dynamics. A reference $\tau^{\mathrm{ref}}_{1:T}$ specifies the desired evolution of the robot's configuration and velocity. In the representation of Section 3.1, each frame has $2n_q + 13$ entries: three root-position coordinates, a unit quaternion for root orientation, n_q joint positions, and the corresponding three-dimensional root linear and angular velocities and n_q joint velocities. Human motion data can supply these references after retargeting to the robot's kinematics; body positions and orientations follow from forward kinematics (Liao et al., 2026; Luo et al., 2026). The actual system state s_t evolves during execution. The policy receives feedback observations o_t, such as joint positions, joint velocities, and inertial measurements, rather than direct access to every state variable (Liao et al., 2026; Ze et al., 2026).

A motion command c supplies the movement to execute. Combined with feedback, it determines the low-level action through

$$a_t = \pi(o_{0:t-1}, c_t), \qquad (s_t, o_t) = f(s_{t-1}, a_t). \tag{11}$$

Here t indexes the target frame reached from state s_{t-1}. The policy interface converts physical motion coordinates into its native inputs through coordinate transforms, feature construction, or learned encoding (Liao et al., 2026; Luo et al., 2026). The compact notation suppresses controller-specific extraction of temporal context from c: an input may encode a motion window rather than only the current frame (Luo et al., 2026). For controllers with joint-position actions, a_t specifies setpoints that proportional–derivative joint controllers convert into actuator torques (Liao et al., 2026). This actuation step is included in f. Thus, c_t specifies the movement to reproduce, whereas a_t is the control output selected from the current feedback to realize it. Repeated feedback updates produce the executed trajectory $\mathcal{T}_\pi(c)$.

Motion-tracking controllers are commonly learned through reinforcement learning in physics simulation (Liao et al., 2026; Luo et al., 2026; Ze et al., 2026). Training samples reference motions and starting phases, initializes the robot, and updates the policy from its simulated execution (Liao et al., 2026). The sampled reference supplies both the motion input and the tracking target. With p_{train} denoting the training-motion distribution, a standard objective is

$$\max_{\pi} \; \mathbb{E}_{\tau^{\mathrm{ref}} \sim p_{\mathrm{train}},\, \pi,\, f} \left[\sum_{t=1}^{T} \gamma^{t-1} \big(r_t^{\mathrm{track}} + r_t^{\mathrm{reg}} \big) \right], \tag{12}$$

where $\gamma \in (0, 1]$ is a discount factor and the expectation also includes sampled initial conditions and training perturbations. Tracking rewards compare simulated and reference positions, orientations, or velocities; regularization terms can penalize abrupt action changes, joint-limit violations, and undesired contacts (Liao et al., 2026; Luo et al., 2026). Local pose matching and global position tracking constrain different aspects of the motion, so the tracked quantities and coordinate frames are part of the training objective (Zeng et al., 2026b; Zhu et al., 2026). Dynamics randomization and observation perturbations expose the controller to variability encountered during deployment (Liao et al., 2026; Ze et al., 2026). This objective trains the controller's parameters; J in Section 3.1 evaluates commands for an already frozen controller.

Training a shared controller across diverse motions makes whole-body tracking a reusable capability. Humanoid Behavior Foundation Models (BFMs) develop this capability through broad motion training and expose control interfaces that support different motion specifications (Zeng et al., 2026a;b; Luo et al., 2026). The pretrained policy supplies the low-level control needed to execute new motion inputs, allowing higher-level policies to learn task-dependent commands while keeping the motion controller fixed (Ze et al., 2026; Luo et al., 2026).

A.2 Learning Humanoid Visuomotor Policies

A humanoid visuomotor policy learns to select motion commands from visual input and robot-related observations (Ze et al., 2026). A vision-language-action (VLA) model additionally conditions on a language instruction specifying the task (Bjorck et al., 2025). In a hierarchical system, these predictions drive the motion controller through its command interface (Ze et al., 2026; Luo et al., 2026). Let y_t denote high-level observations, ℓ an optional instruction, and u_t the high-level action. The body component of u_t represents a motion command or its encoding; any hand actions are sent through the corresponding hand-control interface. The BFM's output a_t remains the low-level control action.

Demonstrations provide paired observations and action labels for learning this mapping. During teleoperation, the sensory observations and commands issued to the robot can be recorded together (Ze et al., 2026). The issued command and the motion actually achieved are distinct records: one specifies the controller's input, while the other describes its physical response. Consequently, an action label must be interpreted in the input representation of the controller that will execute it (Ze et al., 2026; Hu et al., 2026). This representation may contain physical motion targets or learned motion tokens (Ze et al., 2026; Luo et al., 2026).

Supervised policy learning fits the relationship between demonstration observations and action sequences (Chi et al., 2025). For a prediction length L on the high-level time grid, a conditional policy generates an action chunk as

$$u_{t:t+L-1} \sim p_\theta(\cdot \mid y_{1:t}, \ell). \tag{13}$$

Here θ denotes the high-level policy parameters learned from the demonstrations. Diffusion policies learn conditional action sequences through denoising (Chi et al., 2025); VLA models can combine visual and language representations with generative action prediction (Bjorck et al., 2025).

At inference, a prefix of the predicted chunk is executed before the policy is queried with updated observations. Body actions are converted to the BFM's input representation and control-time grid, while the BFM continually updates its output from robot feedback (Chi et al., 2025; Ze et al., 2026). The high-level prediction length L and execution interval describe this policy's inference schedule; they are separate from the command optimizer's horizon H and replanning interval E. Concrete action representations and deployment protocols are given in Appendix C.5.

B METHOD DETAILS

This appendix specifies the rollout cost, command parameterization, controller inputs, and sampling optimizers, then describes compute allocation under the fitted surrogate cost model.

B.1 ROLLOUT COST

For a robot with n_q actuated joints, let q^A and $\dot{q}^A$ denote the joint positions and velocities in motion frame A, and use the same notation for frame B. For each of the B_0 evaluated bodies, let p_b be its world position and u_b its unit quaternion. The joint and body discrepancies are

$$e_q(A,B) = \frac{1}{n_q}\sum_{j=1}^{n_q}(q_j^A - q_j^B)^2, \qquad e_{\dot{q}}(A,B) = \frac{1}{n_q}\sum_{j=1}^{n_q}(\dot{q}_j^A - \dot{q}_j^B)^2, \tag{14}$$

$$e_x(A,B) = \frac{1}{B_0}\sum_{b=1}^{B_0}\|p_b^A - p_b^B\|_2^2, \quad e_R(A,B) = \frac{1}{B_0}\sum_{b=1}^{B_0}\left[2\arccos\big(\min(1, |(u_b^A)^\top u_b^B|)\big)\right]^2. \tag{15}$$

Body-position error sums the three squared coordinate errors before averaging over bodies. The absolute quaternion inner product makes orientation error invariant to quaternion sign.

Tracking errors compare the executed robot frame with the reference frame; deformation errors compare the decoded command frame with the reference. Body positions and orientations come from forward kinematics and are expressed in world coordinates. Using superscripts track and deform for these comparisons, the running and terminal terms in Eq. 3 are

$$\ell_t^{\mathrm{track}} = w_q e_{q,t}^{\mathrm{track}} + w_{\dot{q}} e_{\dot{q},t}^{\mathrm{track}} + w_x e_{x,t}^{\mathrm{track}} + w_R e_{R,t}^{\mathrm{track}}, \tag{16}$$

$$\ell_t^{\mathrm{reg}} = \bar{w}_q e_{q,t}^{\mathrm{deform}} + \bar{w}_x e_{x,t}^{\mathrm{deform}} + \bar{w}_R e_{R,t}^{\mathrm{deform}} + \frac{w_s}{n_q}\|\dot{q}_t\|_2^2, \tag{17}$$

$$\ell_{t+H-1}^{\mathrm{term}} = w_q^{\mathrm{term}} e_{q,t+H-1}^{\mathrm{track}} + w_x^{\mathrm{term}} e_{x,t+H-1}^{\mathrm{track}}. \tag{18}$$

Here $\dot{q}_t$ denotes the executed robot's joint velocities. Each running loss ℓ_t scores the state reached after applying c_t for one control period against reference frame τ_t^{ref}. The terminal loss uses the same state and reference frame as the last running loss, at $t+H-1$. The full-sequence cost J sums the running losses over executed steps.

We use the following cost weights:

$$\begin{aligned}(w_q, w_{\dot{q}}, w_x, w_R) &= (1, 0, 1, 0.5), \qquad (\bar{w}_q, \bar{w}_x, \bar{w}_R) = (0.1, 0.1, 0.05),\\ w_s &= 0.01, \qquad\qquad (w_q^{\mathrm{term}}, w_x^{\mathrm{term}}) = (5, 5).\end{aligned} \tag{19}$$

B.2 COMMAND PARAMETERIZATION AND CONTROLLER INPUTS

We optimize offsets from the nominal command using n_{knot} control knots. For an H-step plan, we set

$$n_{\mathrm{knot}}(H) = \max\left(2, \left\lfloor \frac{5H}{40} + \frac{1}{2} \right\rfloor\right). \tag{20}$$

Each row of the decision array x represents one knot, with root-position offsets, a three-dimensional rotation vector, and joint-position offsets. The knots are uniformly spaced at $\kappa_j = j(H-1)/(n_{\mathrm{knot}} - 1)$, $j = 0, \ldots, n_{\mathrm{knot}} - 1$.

We clip the knot coordinates to $[-1, 1]$, multiply them by the diagonal matrix of physical scales D, and linearly interpolate them to the H control steps with L_H. The resulting command offsets are

$$(\Delta p_h, \Delta r_h, \Delta q_h) = \left[L_H(\mathrm{clip}(x, -1, 1)D)\right]_h, \qquad h = 0, \ldots, H-1. \tag{21}$$

Clipping applies when constructing commands; optimizer updates use the original knot coordinates and the costs of their decoded commands. At each step, we add Δp_h and Δq_h to that step's nominal root and joint positions. The rotation offset is converted to a quaternion and left-multiplied with the nominal root orientation, $u_h^c = \mathrm{quat}(\Delta r_h) \otimes u_h^{\mathrm{ref}}$. Here quat maps a rotation vector to a unit quaternion, and $\otimes$ denotes quaternion multiplication. Forward kinematics then gives the body poses.

With control period Δt, we derive command velocities from the decoded poses. For a joint coordinate or root-position coordinate v_h,

$$\dot{v}_h = \frac{v_{r_h} - v_{l_h}}{(r_h - l_h)\Delta t}, \qquad l_h = \max(h-1, 0), \quad r_h = \min(h+1, H-1). \tag{22}$$

This uses centered differences in the interior and one-sided differences at the endpoints. Root angular velocity comes from the quaternion difference over the same interval; forward kinematics provides body linear and angular velocities.

B.3 Sampling Optimizers

The first planning call starts with mean $\mu = 0$. Each later call initializes the mean by shifting the previous best candidate's clipped knot offsets forward by E steps, using linear interpolation and a zero-filled tail. The remaining optimizer state is reset at each planning call.

CEM (de Boer et al., 2005). At each iteration, we draw candidate knot arrays $x_n = \mu + \sigma \odot \epsilon_n$ from a Gaussian with coordinate-wise mean μ and standard deviation σ. The noise arrays $\epsilon_n \sim \mathcal{N}(0, \mathbf{I})$ are independent, and $\odot$ denotes element-wise multiplication. The M lowest-cost candidates form the elite set $\mathcal{E}$, which updates the distribution:

$$\mu^+ = \frac{1}{M}\sum_{n\in\mathcal{E}} x_n, \qquad \sigma^+ = \max\left(\sigma_{\min}, \sqrt{\frac{1}{M}\sum_{n\in\mathcal{E}}(x_n - \mu^+)^2}\right). \tag{23}$$

Here $\sigma_{\min}$ is the minimum standard deviation. All operations act coordinate-wise, and the variance uses divisor M.

Predictive sampling (Howell et al., 2022). We draw candidate knot arrays $x_n = \mu + \sigma_{\mathrm{PS}}\epsilon_n$, with fixed noise scale σ_{PS} and independent standard Gaussian arrays ϵ_n. The current mean is included as $x_1 = \mu$. If J_n is a candidate's sum of running and terminal costs, the lowest-cost candidate becomes the next mean: $\mu^+ = x_{n^*}$, where $n^* = \mathrm{argmin}_n J_n$. The noise scale stays fixed within and across planning calls.

MPPI-CMA (Jordana et al., 2025). Each knot j has a mean μ_j and a full covariance Σ_j. Knots are sampled independently, $x_{n,j} \sim \mathcal{N}(\mu_j, \Sigma_j)$, so the complete covariance is block diagonal across knots. Each candidate's summed running and terminal cost J_n determines its weight,

$$w_n = \frac{\exp[-(J_n - J_{\min})/\vartheta]}{\sum_{m=1}^{N}\exp[-(J_m - J_{\min})/\vartheta]}, \qquad J_{\min} = \min_n J_n. \tag{24}$$

where temperature ϑ controls how strongly the weights favor lower costs. With adaptation rate ρ, the mean and covariance become

$$\begin{aligned} \mu_j^+ &= \sum_{n=1}^{N} w_n x_{n,j}, \\ \widetilde{\Sigma}_j &= (1-\rho)\Sigma_j + \rho\sum_{n=1}^{N} w_n (x_{n,j} - \mu_j)(x_{n,j} - \mu_j)^\top. \end{aligned} \tag{25}$$

The covariance update uses deviations from the previous mean μ_j. The next covariance Σ_j^+ floors each eigenvalue of $\widetilde{\Sigma}_j$ at $\sigma_{\min}^2$, where $\sigma_{\min}$ is the minimum noise scale. Each planning call starts from an isotropic covariance.

CMA-ES (Hansen & Ostermeier, 2001). The separable variant (Ros & Hansen, 2008) optimizes all control-knot coordinates together. Its population contains $N \geq 2$ candidates. Sampling and updates use the unclipped population and its costs, retaining the evolution state within each planning call.

CBO (Sun et al., 2026). The initial particles are $x_n = \mu + \sigma_{\mathrm{init}}\epsilon_n$, with noise scale σ_{init} and independent standard Gaussian arrays ϵ_n. At each iteration, a particle's summed running and terminal cost J_n gives weight $w_n \propto \exp[-(J_n - J_{\min})/\vartheta]$, normalized so that $\sum_n w_n = 1$. Here $J_{\min} = \min_n J_n$ and ϑ is the temperature. The weighted mean $\bar{x} = \sum_n w_n x_n$ is the consensus. With independent standard Gaussian arrays ζ_n, the next particles are

$$x_n^+ = x_n - \lambda\Delta s(x_n - \bar{x}) + \sigma_{\mathrm{CBO}}\sqrt{\Delta s}\,|x_n - \bar{x}| \odot \zeta_n, \tag{26}$$

Here λ is the consensus weight, σ_{CBO} is the noise weight, and Δs is the step size. The absolute value acts coordinate-wise. The updated particles are evaluated in the next iteration; the consensus is stored as the optimizer mean. Execution uses the best evaluated particle across iterations.

B.4 Allocation Under the Fitted Model

Let $J_{\mathrm{TTC}}(\mathcal{A}, \xi, \tau)$ denote the expected full-sequence cost on a motion τ of length T_τ. Its time-normalized mean over the reference-motion dataset $\mathcal{D}$ is

$$J_{\mathrm{TTC}}(\mathcal{A}, \xi) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} J_{\mathrm{TTC}}(\mathcal{A}, \xi, \tau)/T_\tau. \tag{27}$$

The expectation is over the sampling optimizer. Equation 5 defines the algorithm and allocation that minimize this dataset cost under each compute budget.

For a fixed sampling optimizer and its fitted surrogate parameters, consider a solver budget $O = IN$ at fixed H, E. The solver term becomes

$$P_O(I) = C_I I^{-\alpha_I} + C_N O^{-\alpha_N} I^{\alpha_N}. \tag{28}$$

The horizon cost is constant, and the sampling term is a strictly increasing function of P_O. In a continuous relaxation with $I > 0$, setting the derivative to zero gives

$$I^*_{\mathrm{cont}} = \left(\frac{\alpha_I C_I}{\alpha_N C_N} \right)^{1/(\alpha_I + \alpha_N)} O^{\alpha_N/(\alpha_I + \alpha_N)}, \qquad N^*_{\mathrm{cont}} = \frac{O}{I^*_{\mathrm{cont}}}. \tag{29}$$

As a function of $\log I$, P_O is the sum of two strictly convex exponentials, so this stationary point is its unique minimum.

For integer configurations, define $P = C_I I^{-\alpha_I} + C_N N^{-\alpha_N} > 0$. At fixed (I, H, E),

$$\frac{\partial \widehat{J}}{\partial N} = -\frac{\eta \alpha_N C_N}{H} \gamma^H P^{\eta/H - 1} N^{-\alpha_N - 1} < 0. \tag{30}$$

All fitted coefficients and exponents are positive. We therefore enumerate feasible integer triples (I, H, E) and evaluate the largest candidate count allowed by the budget and resource constraints. The configuration with the lowest predicted cost among these triples is the allocation selected by the fitted model.

C Additional Experimental Setup and Results

We begin with the shared simulation setup (Appendix C.1), followed by motion-tracking results (Appendix C.2) and the effects of solver and time horizon parameters (Appendix C.3). The cost-model experiments in Appendix C.4 cover fitting, configuration selection, evaluation metrics, and tests on fitting motions and new motions across BFMs. Appendix C.5 gives the loco-manipulation data, training, and evaluation protocols for simulation and hardware, followed by the complete simulation results.

C.1 Experimental Setup

Robot and simulation. The G1 has 29 actuated joints. The physics time step is 0.005 s, and each BFM runs at 50 Hz with four physics steps per action. Thus the control period is $\Delta t = 0.02$ s.

We use the official released SONIC G1 (Luo et al., 2026), HEFT PMG (Liu et al., 2026a), and TWIST2 1017–20k (Ze et al., 2026) checkpoints.

Command settings. Each control knot has 35 coordinates. The physical scales in Appendix B.2 are 0.055 m for each horizontal root coordinate, 0.0275 m for root height, 0.0825 rad for each rotation-vector coordinate, and 0.1375 rad for each joint.

C.2 Test-Time Command Optimization Improves Tracking Precision (Q1)

C.2.1 AMASS Evaluation and Tracking Metrics

Evaluation protocol. Each BFM executes all 12,273 retargeted AMASS motions once with nominal commands and once with optimized commands. Each pair starts from the first reference frame with identical robot states, controller states, and observation histories. We use CEM with $(I, N, H, E) = (4, 80, 40, 20)$, nine elites, and the noise scales in Table 7. For the mean metrics, we first average within each motion, then across motions with equal weight.

Cost-win rate. This is the percentage of motions for which optimized commands yield a lower cost than nominal commands.

Position errors. Global body-position error is the mean Euclidean distance between executed and reference positions in world coordinates, averaged over control steps and 14 bodies. These are the pelvis, torso, and the left and right hip-roll, knee, ankle-roll, shoulder-roll, elbow, and wrist-yaw links. Root-position error uses the pelvis alone.

Joint acceleration. For motion m with F_m frames, let $T_m = F_m - 1$ be the number of executed steps. We compute joint acceleration from consecutive executed joint velocities with $\Delta t = 0.02$ s, then average its absolute value over time and $n_q = 29$ joints:

$$\ddot{q}_{m,t,j} = \frac{\dot{q}_{m,t+1,j} - \dot{q}_{m,t,j}}{\Delta t}, \qquad A_m = \frac{1}{n_q(T_m - 1)} \sum_{t=1}^{T_m - 1} \sum_{j=1}^{n_q} |\ddot{q}_{m,t,j}|. \tag{31}$$

We report acceleration in rad/s^2.

C.2.2 Per-Motion Tracking Precision and Cost

Figure 8 and Table 4 show the frequency and size of the improvements that contribute to the mean reductions in Table 1.

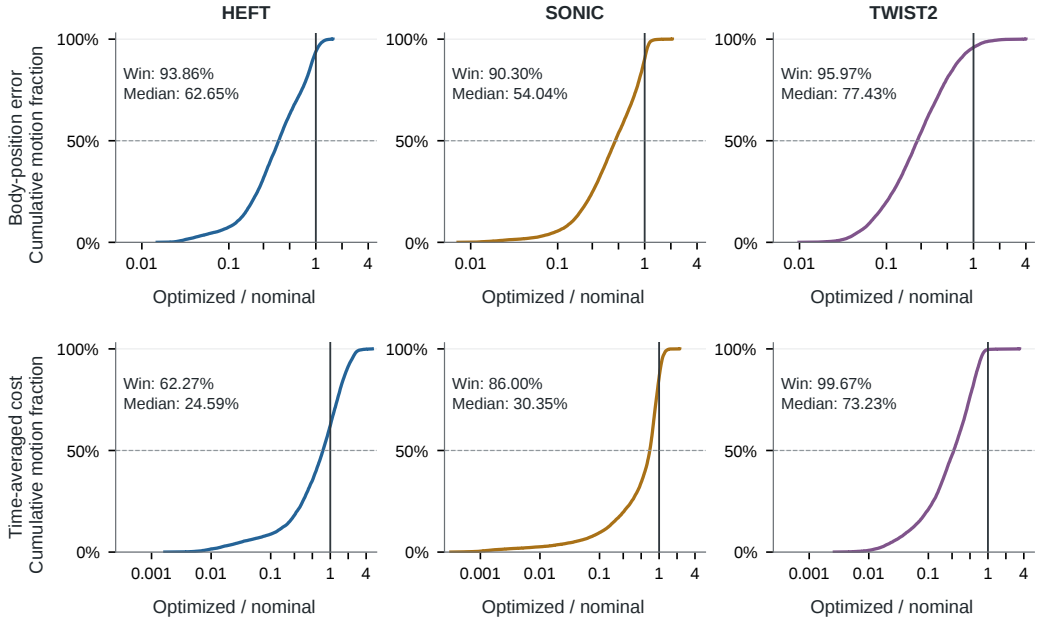

Figure 8: **Motion-level tracking performance on AMASS.** Empirical cumulative distributions of optimized-to-nominal global body-position error (top) and full-sequence cost per time step (bottom) across 12,273 motions per BFM. Ratios below one indicate improvement. Horizontal axes are logarithmic; reference lines mark a ratio of one and the 50th percentile. Panels report win rates and median percentage reductions.

Table 4: **Motion-level reductions and increases in tracking error and cost.** Global body-position error and full-sequence cost per time step for 12,273 motions per BFM. Median reduction is the median of the per-motion percentage reductions, $100(1 - X_m^{\mathrm{opt}}/X_m^{\mathrm{nom}})$, for metric X. Win rate is the percentage of motions with a lower error or cost after optimization. P_{95} is the 95th percentile of $X_m^{\mathrm{opt}} - X_m^{\mathrm{nom}}$ among motions with increased error or cost, in the metric's units.

BFM	Metric	Median nominal → optimized	Median reduction (%)	Win (%)	Increase P_{95}
HEFT	Body-position error (cm)	13.88 → 4.80	62.65	93.86	1.20
	Cost	0.0529 → 0.0367	24.59	62.27	0.0148
SONIC	Body-position error (cm)	11.51 → 4.86	54.04	90.30	1.09
	Cost	0.0677 → 0.0416	30.35	86.00	0.0054
TWIST2	Body-position error (cm)	62.68 → 13.15	77.43	95.97	23.76
	Cost	1.8983 → 0.4709	73.23	99.67	6.4616

To relate improvements to nominal tracking difficulty, we sort motions by nominal global body-position error within each BFM and divide them into four quartiles (Table 5). For each quartile, we report the median change and its contribution to the total reductions and increases.

C.3 Compute Allocation Governs the Gains (Q2)

Table 6 lists the ten motions used in Sec. 4.2. Every configuration uses three optimizer seeds. We estimate the dataset cost in Eq. 27 by averaging the three seed means.

Table 5: **Tracking performance grouped by nominal body-position error.** Quartiles run from lowest (Q1) to highest (Q4) nominal global body-position error within each BFM. Error reduction is nominal minus optimized error. For each contribution column, we sum only changes in the indicated direction: error decreases, error increases, or cost decreases. We divide each group sum by the corresponding total over all motions of the same BFM and express the result as a percentage.

				Contribution to total (%)		
BFM	Group	Nominal error range (cm)	Median error reduction (cm)	Error reduction	Error increase	Cost reduction
HEFT	Q1	1.89–6.66	0.57	0.86	78.53	0.01
	Q2	6.66–13.88	5.11	6.00	21.33	0.26
	Q3	13.88–27.78	13.78	16.60	0.14	2.92
	Q4	27.79–843.14	38.04	76.53	0.00	96.81
SONIC	Q1	1.89–5.63	0.22	0.23	43.60	0.02
	Q2	5.63–11.51	3.09	1.67	8.29	0.05
	Q3	11.51–28.56	11.64	6.33	3.73	0.23
	Q4	28.57–3065.46	60.81	91.77	44.38	99.70
TWIST2	Q1	3.72–31.43	7.87	2.27	39.10	2.29
	Q2	31.43–62.68	31.62	8.74	19.57	4.97
	Q3	62.71–130.24	70.86	19.74	26.58	11.28
	Q4	130.31–1412.99	206.07	69.25	14.75	81.45

Table 6: Reference motions used in the compute-allocation and strategy experiments.

Source sequence	Frames	Duration (s)
`CMU/63/63_01_poses`	200	3.98
`BMLmovi/Subject_22_F_MoSh/Subject_22_F_3_poses`	159	3.16
`KIT/883/wipe_leg_bigcircle03_poses`	267	5.32
`BMLmovi/Subject_45_F_MoSh/Subject_45_F_19_poses`	199	3.96
`BioMotionLab_NTroje/rub061/0014_knocking2_poses`	507	10.12
`CMU/102/102_20_poses`	265	5.28
`KIT/8/WalkInClockwiseCircle01_poses`	464	9.26
`BioMotionLab_NTroje/rub043/0010_normal_jog2_poses`	84	1.66
`KIT/424/bend_left09_poses`	335	6.68
`CMU/62/62_14_poses`	575	11.48

C.3.1 Solver Parameters

We fix $(H, E) = (40, 20)$ and evaluate solver budgets $O \in \{256, 512, 1024, 2048, 4096, 8192\}$. At each budget, we test

$$I \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}, \qquad I \le O, \qquad N = O/I. \tag{32}$$

CMA-ES uses only configurations with $N \ge 2$. Table 7 gives the optimizer settings.

C.3.2 Time Horizon Parameters

We evaluate all 24 pairs formed by $E \in \{5, 10, 20, 40\}$ and $K_h = H/E \in \{1, 2, 4, 6, 8, 10\}$. We use CEM with $N = 512$ and stop each local solve after three consecutive iterations without a strict decrease in its best cost.

C.3.3 Fitting the Solver and Horizon Error Models

We fit the curves in Fig. 5(b,d) separately. Each fit minimizes the mean squared log ratio between predicted and measured cost, with equal weight per configuration.

For the solver sweep, at fixed $(H, E) = (40, 20)$, the full surrogate reduces to the six-parameter form

$$\widehat{J}_S(I, N) = c + \left(aI^{-u} + bN^{-v}\right)^g. \tag{33}$$

Here c absorbs the fixed horizon contribution, $g = \eta/40$, and the positive coefficients a, b absorb the constant factor γ^{40}. We fit all six parameters jointly to the 59 CEM configuration means.

Table 7: Fixed optimizer settings in the solver sweep. Noise scales apply to the normalized knot coordinates before clipping and physical scaling. The CEM settings also apply to the horizon and strategy experiments.

Optimizer	Settings
CEM	Initial standard deviation 1.6; minimum standard deviation 0.075; $\max(1, \lfloor 0.1N \rfloor)$ elites; exploration fraction 0.
PS	Gaussian noise scale 1.2.
MPPI-CMA	Initial noise scale 1.6; minimum noise scale 0.075; temperature 0.1; covariance adaptation rate 0.1. Covariance eigenvalues are floored at 0.075^2.
CMA-ES	Separable CMA-ES; initial scale 1.6; minimum scale 0.05; maximum scale 2.0.
CBO	Initial noise scale 1.0; temperature 0.1; consensus weight 1.0; noise weight 0.3; step size 0.1.

Table 8: Parameters of the separate component fits in Fig. 5(b,d), rounded to five significant figures for display.

Solver parameter	Value	Horizon parameter	Value
c	0.036137	J_∞	0.035463
a	0.44596	C_H	1.0288
b	0.58164	α_H	0.10719
u	0.19475	C_E	0.00042258
v	0.35423		
g	3.9471		

For the horizon sweep, we fit $J_\infty + \epsilon_{\text{horizon}}(H, E)$ from Eq. 7 to the 24 configuration means, using the zero-solver-error approximation. Table 8 gives the fitted parameters.

For each fitting dataset $\mathcal{D}$, we report the mean absolute relative error as

$$\text{MARE} = \frac{100\%}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \left| \frac{\widehat{J}_d - J_d}{J_d} \right|. \tag{34}$$

C.4 The Cost Model Estimates a Test-Time Compute Strategy (Q3)

This section gives the fitting and evaluation details for Sec. 4.3. The two evaluations report the costs and allocation regret underlying Figs. 2 and 6, followed by additional analysis of configuration rankings. We write $J(\xi)$ for the measured estimate of $J_{\text{TTC}}(\text{CEM}, \xi)$ and $\widehat{J}(\xi)$ for the proposed surrogate's prediction, with $\xi = (I, N, H, E)$.

C.4.1 Fitting the Surrogate Cost Model and Selecting Configurations

Fitting data. We use the motion set and cost aggregation from Appendix C.3.

We fit the proposed cost model and the three regression baselines to the same 183 equally weighted training points: 24 horizon configurations and 59 fixed-horizon CEM configurations from Sec. 4.2, and 100 joint configurations that vary all four parameters.

The joint configurations and subsequent allocation search use the integer domain $\mathcal{X}_{\text{exp}}$:

$$\begin{gathered} 1 \le I \le 512, \quad N \ge 2, \quad 256 \le IN \le 8192, \\ 5 \le E \le 40, \quad E \le H \le 10E, \quad NH \le 327680, \qquad I, N, H, E \in \mathbb{N}_+. \end{gathered} \tag{35}$$

To generate joint configurations, we draw i, o, e, r independently from log-uniform distributions on $[1, 512]$, $[256, 8192]$, $[5, 40]$, and $[1, 10]$, respectively. Here o represents IN and r represents H/E.

Table 9: Initial values for the horizon fit and the solver coordinates of the joint fit. Each column lists five values in the stated parameter order.

$(J_\infty, C_H, \alpha_H, C_E)$	$(C_I, C_N, \alpha_I, \alpha_N)$
$(0.02, 0.5, 0.10, 0.0001)$	$(0.03, 0.10, 0.10, 0.30)$
$(0.02, 2, 0.20, 0.0005)$	$(0.03, 0.25, 0.30, 0.60)$
$(0.035, 0.5, 0.10, 0.0001)$	$(0.10, 0.10, 0.60, 0.30)$
$(0.035, 2, 0.20, 0.0005)$	$(0.10, 0.25, 0.30, 0.60)$
$(0.04, 1, 0.30, 0.001)$	$(0.25, 0.25, 0.60, 0.60)$

Table 10: Parameter bounds and fitted values of the complete CEM surrogate. Fitted values are rounded to five significant figures.

Parameter	Bounds	Fitted value
J_∞	$[10^{-8}, 1]$	0.034123
C_H	$[10^{-8}, 20]$	1.8011
α_H	$[10^{-4}, 2]$	0.094838
C_E	$[10^{-8}, 0.1]$	0.00047245
C_I	$[10^{-8}, 2]$	0.49438
C_N	$[10^{-8}, 2]$	0.83919
α_I	$[0.005, 8]$	0.40234
α_N	$[0.005, 8]$	0.34379
η	$[0.1, 160]$	160.00
γ	$[0.9, 1.1]$	0.98201

We reject a draw if $i \geq o$. With the rounding rule $R(x) = \lfloor x + 1/2 \rfloor$, set

$$\begin{aligned} I &= R(i), \qquad E = R(e), \qquad H = \mathrm{clip}(R(Er), E, 10E), \\ N_{\min} &= \max\bigl(2, \lceil 256/I \rceil\bigr), \\ N_{\max} &= \min\bigl(\lfloor 8192/I \rfloor, \lfloor 327680/H \rfloor\bigr), \\ N &= \mathrm{clip}\bigl(R(o/I), N_{\min}, N_{\max}\bigr). \end{aligned} \tag{36}$$

The clip operation limits its first argument to the specified interval. We reject empty intervals and tuples outside the domain. We reject duplicate tuples and previously tested configurations.

Proposed cost model. We fit the ten positive CEM parameters θ of the surrogate in Eq. 6. Write $\widehat{J}_\theta(\xi)$ for its full prediction and $B_\theta(H, E) = J_\infty + \epsilon_{\mathrm{horizon}}(H, E)$ for its horizon-only prediction. Denote the horizon, solver and joint training sets by $\mathcal{D}_H$, $\mathcal{D}_S$ and $\mathcal{D}_C$, respectively. The fitting objective is

$$\begin{aligned} \mathcal{L}(\theta) = \frac{1}{183}\Bigg[& \sum_{d \in \mathcal{D}_H} \left(\log \frac{B_\theta(H_d, E_d)}{J_d} \right)^2 \\ & + \sum_{d \in \mathcal{D}_S} \left(\log \frac{\widehat{J}_\theta(I_d, N_d, 40, 20)}{J_d} \right)^2 \\ & + \sum_{d \in \mathcal{D}_C} \left(\log \frac{\widehat{J}_\theta(\xi_d)}{J_d} \right)^2 \Bigg]. \end{aligned} \tag{37}$$

The horizon data use the zero-solver-error approximation.

We first fit the horizon parameters using the five initial values in Table 9. We combine the best horizon fit with each of the five solver initial values in that table and each of the twelve (η, γ) pairs in

$$\bigl(\{5, 20\} \times \{0.98, 1\}\bigr) \cup \bigl(\{40, 80\} \times \{0.98, 1, 1.02\}\bigr) \cup \bigl(\{140\} \times \{1, 1.02\}\bigr). \tag{38}$$

These combinations give 60 starts for joint optimization of all ten parameters. We optimize the parameters in natural-log coordinates by bounded nonlinear least squares, using the bounds in Table 10. We retain the converged fit with the lowest objective; Table 10 lists its parameters.

Configuration selection. We evaluate budgets $\mathcal{K} = \{256, 512, 1024, 2048, 4096, 8192\}$. Table 11 lists the six proposed allocations and their actual compute ratios INH/E.

Table 11: CEM allocations selected by the surrogate cost model for Sec. 4.3.

Budget K	I	N	H	E	Actual INH/E	$\widehat{J}$
256	8	32	40	40	256	0.11652350
512	8	32	36	18	512	0.07089407
1024	9	31	33	9	1023	0.05898193
2048	10	48	34	8	2040	0.05236377
4096	13	54	35	6	4095	0.04772997
8192	15	78	35	5	8190	0.04439349

Table 12: Configurations (I, N, H, E) selected by each regression baseline. The same configuration applies to both motion sets and all BFMs at a given budget. The new-motion evaluation uses the first five budgets.

K	Random forest	RBF	Gaussian process
256	$(8, 32, 35, 35)$	$(128, 2, 40, 40)$	$(8, 32, 40, 40)$
512	$(8, 62, 35, 34)$	$(256, 2, 40, 40)$	$(8, 32, 80, 40)$
1024	$(14, 71, 35, 34)$	$(512, 2, 40, 40)$	$(14, 37, 79, 40)$
2048	$(15, 78, 35, 20)$	$(512, 2, 40, 40)$	$(6, 46, 191, 26)$
4096	$(18, 130, 35, 20)$	$(512, 2, 40, 40)$	$(7, 73, 191, 24)$
8192	$(30, 140, 39, 20)$	$(512, 2, 40, 40)$	$(16, 105, 39, 8)$

C.4.2 BASELINES

We compare the proposed cost model with random forest, radial basis function, and Gaussian process regression. All three baselines use the fitting data in Appendix C.4.1. We select baseline hyperparameters by mean squared prediction error in natural-log cost on a separate validation set of 50 joint configurations.

Each baseline maps $x = (1/I, 1/N, H, E)$ to the natural logarithm of closed-loop cost. We linearly scale the input using fixed bounds $l = (0, 0, 5, 5)$ and $u = (1, 1/2, 400, 40)$, so $\widetilde{x}_j = 2(x_j - l_j)/(u_j - l_j) - 1$. For the 24 points from the horizon sweep, we set the first two unscaled coordinates to zero, representing $I, N \to \infty$ and zero solver error.

Random forest (RF). Following AutoMPC (Edwards et al., 2021), we use a regression forest with bootstrap sampling and squared-error splits. We test 27 combinations of tree count $\{10, 30, 100\}$, features per split $\{4, 3, 2\}$, and minimum leaf size $\{1, 3, 5\}$. The minimum split size is 2. Each leaf averages natural-log costs directly. Validation selects 10 trees, three features per split, and leaf size 1.

Radial basis functions (RBF). We follow the GLIS surrogate used for embedded MPC calibration (Forgione et al., 2020). It applies the inverse-quadratic kernel $\phi(r) = (1 + \epsilon^2 r^2)^{-1}$ to scaled inputs. We compute its weights by truncated singular value decomposition, with no polynomial tail. The 18 candidates combine $\epsilon \in \{0.125, 0.25, 0.5, 1, 2, 4\}$ with singular-value thresholds $\{10^{-6}, 10^{-8}, 10^{-10}\}$. Validation selects $\epsilon = 0.125$ and threshold 10^{-6}.

Gaussian process (GP). We fit exact GP regression with a learned constant mean and Gaussian observation noise. We test Matérn $5/2$ and squared-exponential kernels, each with either a shared length scale or a separate length scale per input coordinate. Three initial length scales $\{0.25, 1, 4\}$ and three initial noise standard deviations $\{0.001, 0.03, 0.3\}$ give 36 candidates. We optimize training marginal likelihood with 250 Adam updates at learning rate 0.05, retaining the iterate with the highest likelihood. The log length scale, log signal standard deviation, and log noise standard deviation are bounded by $[-6, 6]$, $[-8, 8]$, and $[-12, 2]$, respectively; diagonal jitter is 10^{-10}. Validation selects the Matérn $5/2$ kernel with separate length scales, initialized at length 0.25 and noise standard deviation 0.3.

Configuration selection. For each baseline, we minimize predicted log cost over the domain in Eq. 35 subject to $INH/E \le K$, using the same six budgets as the proposed model. We evaluate every feasible (I, N, H, E) configuration. Table 12 lists the selected configurations.

C.4.3 Evaluation Metrics

Allocation regret. The four strategies are $\mathcal{M} = \{\text{Ours}, \text{RF}, \text{RBF}, \text{GP}\}$. Let $J_a(K)$ denote strategy a's measured mean cost and $\mathcal{R}_K$ the random configurations at budget K on the fitting motions. Define the empirical optimum and proposed cost by

$$J_{\text{best}}(K) = \min\left\{\min_{\xi\in\mathcal{R}_K} J(\xi), \min_{a\in\mathcal{M}} J_a(K)\right\}, \qquad J_{\text{prop}}(K) = J(\widehat{\xi}^{\star}(K)). \tag{39}$$

The relative allocation regret is

$$R_K = \frac{J_{\text{prop}}(K) - J_{\text{best}}(K)}{J_{\text{best}}(K)}. \tag{40}$$

We report regret as a percentage. This is the comparison used in Sec. 4.3; the baseline regrets replace $J_{\text{prop}}(K)$ with $J_a(K)$ in Eq. 40.

On the five new motions, we compute the same regret separately for each BFM b, using 50 random configurations and the four strategies as the reference set. We denote this regret by $R_b(K)$. When strategies select the same tuple, each retains its measured mean; the Empirical Optimum error bar uses the entry with the lowest mean.

Mean regret weights budgets equally: six on the fitting motions and five on the new motions. The overall generalization mean also weights the three BFMs equally. Maximum regret is the largest regret across the corresponding budgets.

Additional analysis: configuration rankings. We also measure agreement between predicted and measured costs across the random configurations, using Kendall's τ_b, Spearman's ρ, and NDCG@5. For the fitting-motion evaluation, these metrics use the $n = 200$ configurations in $\mathcal{R}_K$ at each budget. Their measured and predicted costs are $y_i = J(\xi_i)$ and $\widehat{y}_i = \widehat{J}(\xi_i)$, with ascending ranks r_i and $\widehat{r}_i$, respectively. Rank one denotes the lowest cost. Tied costs receive their average rank. All costs in each measured or predicted vector are distinct in this experiment.

Kendall's τ_b. A pair is concordant if both orderings agree on which cost is lower and discordant if they disagree. Let n_c and n_d count these pairs, $n_0 = n(n-1)/2$, and let t_y and $t_{\widehat{y}}$ count pairs tied in measured and predicted cost. Then

$$\tau_b = \frac{n_c - n_d}{\sqrt{(n_0 - t_y)(n_0 - t_{\widehat{y}})}}. \tag{41}$$

Pairs tied in both costs contribute to both tie counts. For distinct costs, $\tau_b = (n_c - n_d)/n_0$ (Kendall, 1938). A value of one means complete agreement, zero means equal numbers of concordant and discordant pairs, and minus one means reverse order. For these distinct costs, the fraction of correctly ordered pairs is $n_c/n_0 = (1 + \tau_b)/2$.

Spearman's ρ. Spearman correlation (Spearman, 1904) is the Pearson correlation between the two rank vectors:

$$\rho = \frac{\sum_{i=1}^{n}(r_i - \bar{r})(\widehat{r}_i - \bar{\widehat{r}})}{\sqrt{\sum_{i=1}^{n}(r_i - \bar{r})^2}\sqrt{\sum_{i=1}^{n}(\widehat{r}_i - \bar{\widehat{r}})^2}}. \tag{42}$$

The bars denote the means of the rank vectors. A value of one means matching ranks, while minus one means reverse ranks.

NDCG@5 (Järvelin & Kekäläinen, 2002). To assess the top of the predicted ranking, assign linear relevance $g_i = n + 1 - r_i = 201 - r_i$. Let $\widehat{\pi}(j)$ denote the configuration at predicted position j, and $\pi(j)$ the configuration at measured position j. Then

$$\begin{aligned} \text{DCG@5} &= \sum_{j=1}^{5} \frac{g_{\widehat{\pi}(j)}}{\log_2(j+1)}, \quad \text{IDCG@5} = \sum_{j=1}^{5} \frac{g_{\pi(j)}}{\log_2(j+1)}, \\ \text{NDCG@5} &= \frac{\text{DCG@5}}{\text{IDCG@5}}. \end{aligned} \tag{43}$$

Higher relevance means lower measured cost. The discount gives more weight to earlier predicted positions, and the denominator is the ideal discounted gain. A value of one attains the ideal top-five ranking under this gain.

On the five new motions, ranking metrics use the 50 random configurations per BFM and budget, with predictions from the SONIC-fitted surrogate. With $n = 50$, NDCG@5 relevance becomes $g_i = n + 1 - r_i = 51 - r_i$. The tie rules are unchanged.

We compute each ranking metric within a budget, then average equally over the six budgets on the fitting motions or the five budgets for each BFM on the new motions.

C.4.4 Compute Scaling on the Fitting Motions

Evaluation protocol. We evaluate the four models' selected configurations on the ten SONIC fitting motions (Table 6) at each of the six budgets in $\mathcal{K}$. The comparison uses three seeds and 200 distinct random configurations per budget, each satisfying $0.9K \leq INH/E \leq K$.

For $K > 256$, we sample configurations using the procedure in Appendix C.4.1, subject to the budget band and without repeating configurations across budgets. Previously tested configurations remain eligible. At $K = 256$, we sample without replacement from feasible configurations with $IN = 256$ and $H = E$, using the probabilities induced by the same continuous sampling and rounding procedure.

Costs and regret in the main comparison. Tables 13 and 14 give the costs and regret for the four strategies compared in Fig. 2 and Sec. 4.3.

Table 13: Measured costs on the ten SONIC fitting motions at matched compute budgets; lower is better. Entries report mean cost $\pm$ one sample standard deviation across the three seed means. Empirical Optimum uses the configuration with the lowest mean cost among the four strategies and 200 random configurations.

K	Ours	Random forest	RBF	Gaussian process	Empirical Optimum
Ten fitting motions, SONIC					
256	0.1048 ± 0.0063	0.1226 ± 0.0310	0.2096 ± 0.0260	0.1053 ± 0.0068	0.1027 ± 0.0028
512	0.0715 ± 0.0034	0.0923 ± 0.0073	0.1554 ± 0.0360	0.0855 ± 0.0140	0.0715 ± 0.0034
1024	0.0607 ± 0.0052	0.0783 ± 0.0141	0.1461 ± 0.0350	0.0799 ± 0.0065	0.0570 ± 0.0017
2048	0.0557 ± 0.0021	0.0585 ± 0.0093	0.1461 ± 0.0350	0.0868 ± 0.0110	0.0528 ± 0.0049
4096	0.0484 ± 0.0010	0.0520 ± 0.0049	0.1461 ± 0.0350	0.0741 ± 0.0015	0.0461 ± 0.0028
8192	0.0441 ± 0.0007	0.0425 ± 0.0005	0.1461 ± 0.0350	0.0419 ± 0.0009	0.0419 ± 0.0009

Table 14: Allocation regret (%) on the ten SONIC fitting motions. The reference is Empirical Optimum in Table 13. The Mean row weights the six budgets equally. Bold marks the lowest regret in each row.

BFM	K	Ours	Random forest	RBF	Gaussian process
Figure 2: ten fitting motions					
SONIC	256	**2.10**	19.36	104.13	2.60
	512	**0.00**	29.08	117.26	19.52
	1024	**6.40**	37.27	156.16	40.12
	2048	**5.38**	10.65	176.43	64.25
	4096	**5.15**	12.90	217.10	60.90
	8192	5.10	1.35	248.25	**0.00**
	Mean	**4.02**	18.44	169.89	31.23

Additional analysis: configuration rankings. Table 15 reports the proposed model's ranking accuracy on the random configurations at each budget.

Table 15: **Additional ranking analysis in the SONIC scaling experiment.** Ranking metrics use the 200 random configurations at each budget; regret refers to the proposed configuration. The Mean row weights the six budgets equally.

K	$\tau \uparrow$	$\rho \uparrow$	NDCG@5↑	Regret (%)↓
256	0.722	0.901	0.954	2.10
512	0.731	0.902	0.987	0.00
1024	0.754	0.917	0.935	6.40
2048	0.562	0.730	0.798	5.38
4096	0.552	0.711	0.981	5.15
8192	0.572	0.736	0.944	5.10
Mean	0.649	0.816	0.933	4.02

C.4.5 Generalization to New Motions and BFMs

Evaluation protocol. Figure 6 compares the four strategies on SONIC, HEFT, and TWIST2 using five AMASS motions disjoint from the fitting set (Table 16). This experiment uses $\mathcal{K}_{\text{test}} = \{256, 512, 1024, 2048, 4096\}$. Each model uses its first five allocations from Tables 11 and 12 on all three BFMs.

At each budget, all three BFMs use the same 50 distinct random configurations from the fitting-motion experiment. We evaluate these and the selected configurations on the new motions with three optimizer seeds, using the 14 bodies listed in Appendix C.2.

We reuse measurements when a configuration is selected at several budgets for the same BFM and motion set. RBF selects $(512, 2, 40, 40)$ for every $K \geq 1024$, using actual compute 1024 and producing a flat cost curve.

Table 16: New motions used to evaluate strategy generalization. Durations count executed control steps at 0.02 s per step.

Source sequence	Frames	Duration (s)
`CMU/143/143_14_poses`	617	12.32
`BMLmovi/Subject_57_F_MoSh/Subject_57_F_2_poses`	277	5.52
`EKUT/300/PushBeamPedar_15_poses`	247	4.92
`BioMotionLab_NTroje/rub012/0009_normal_jog1_poses`	109	2.16
`CMU/74/74_01_poses`	400	7.98

The nominal-command costs labeled in Fig. 6 average the paired baseline executions equally over the five motions and three seeds.

Costs and regret in the main comparison. Tables 17 and 18 give the costs and regret for all four strategies in Fig. 6.

Table 17: Measured costs on five new motions at matched compute budgets; lower is better. Entries report mean cost ± one sample standard deviation across the three seed means. Empirical Optimum uses the configuration with the lowest mean cost among the four strategies and 50 random configurations on each BFM.

BFM	K	Ours	Random forest	RBF	Gaussian process	Empirical Optimum
Five new motions, SONIC						
SONIC	256	0.1621 ± 0.0166	0.1553 ± 0.0318	0.2653 ± 0.0167	0.1617 ± 0.0171	0.1553 ± 0.0318
SONIC	512	0.0875 ± 0.0040	0.1174 ± 0.0104	0.2346 ± 0.0225	0.0786 ± 0.0053	0.0752 ± 0.0045
SONIC	1024	0.0660 ± 0.0026	0.0887 ± 0.0065	0.2014 ± 0.0141	0.0722 ± 0.0028	0.0647 ± 0.0033
SONIC	2048	0.0559 ± 0.0004	0.0728 ± 0.0067	0.2014 ± 0.0141	0.0849 ± 0.0035	0.0559 ± 0.0004
SONIC	4096	0.0542 ± 0.0011	0.0603 ± 0.0023	0.2014 ± 0.0141	0.0827 ± 0.0016	0.0537 ± 0.0022
Five new motions, HEFT						
HEFT	256	0.0560 ± 0.0015	0.0516 ± 0.0002	0.0755 ± 0.0068	0.0554 ± 0.0011	0.0516 ± 0.0002
HEFT	512	0.0486 ± 0.0008	0.0484 ± 0.0009	0.0641 ± 0.0033	0.0575 ± 0.0005	0.0463 ± 0.0015
HEFT	1024	0.0447 ± 0.0004	0.0426 ± 0.0006	0.0593 ± 0.0035	0.0540 ± 0.0008	0.0426 ± 0.0006
HEFT	2048	0.0409 ± 0.0005	0.0374 ± 0.0002	0.0593 ± 0.0035	0.0689 ± 0.0006	0.0374 ± 0.0002
HEFT	4096	0.0385 ± 0.0003	0.0346 ± 0.0001	0.0593 ± 0.0035	0.0671 ± 0.0009	0.0340 ± 0.0004
Five new motions, TWIST2						
TWIST2	256	0.5942 ± 0.0371	0.6087 ± 0.0446	1.3601 ± 0.3610	0.6090 ± 0.0319	0.5780 ± 0.0269
TWIST2	512	0.4958 ± 0.0076	0.5534 ± 0.0432	0.8815 ± 0.1238	0.5672 ± 0.0129	0.4885 ± 0.0123
TWIST2	1024	0.4468 ± 0.0070	0.5107 ± 0.0110	0.8148 ± 0.0421	0.5377 ± 0.0224	0.4468 ± 0.0070
TWIST2	2048	0.4296 ± 0.0015	0.4501 ± 0.0137	0.8148 ± 0.0421	0.6045 ± 0.0062	0.4296 ± 0.0015
TWIST2	4096	0.4065 ± 0.0048	0.4228 ± 0.0124	0.8148 ± 0.0421	0.5814 ± 0.0065	0.4065 ± 0.0048

Table 18: Allocation regret (%) on five new motions. The reference is Empirical Optimum in Table 17. Each Mean row weights the five budgets equally; the Figure 6 mean also weights BFMs equally. Bold marks the lowest regret in each row.

BFM	K	Ours	Random forest	RBF	Gaussian process
Figure 6: five new motions					
SONIC	256	4.34	**0.00**	70.81	4.07
	512	16.27	56.07	211.91	**4.52**
	1024	**2.07**	37.10	211.23	11.52
	2048	**0.00**	30.22	260.27	51.91
	4096	**0.91**	12.36	275.05	53.99
	Mean	**4.72**	27.15	205.86	25.20
HEFT	256	8.59	**0.00**	46.34	7.36
	512	4.98	**4.59**	38.61	24.34
	1024	4.77	**0.00**	39.16	26.57
	2048	9.48	**0.00**	58.75	84.27
	4096	13.37	**1.94**	74.55	97.28
	Mean	8.24	**1.31**	51.48	47.96
TWIST2	256	**2.80**	5.31	135.31	5.36
	512	**1.49**	13.29	80.45	16.12
	1024	**0.00**	14.30	82.36	20.36
	2048	**0.00**	4.77	89.64	40.70
	4096	**0.00**	4.03	100.45	43.05
	Mean	**0.86**	8.34	97.64	25.12
Figure 6 mean		**4.60**	12.27	118.33	32.76

Additional analysis: configuration rankings. Table 19 reports the SONIC-fitted surrogate's ranking accuracy on the random configurations for each BFM and budget.

Table 19: **Additional ranking analysis on new motions and BFMs.** Costs average five motions and three seeds. Ranking metrics use the 50 random configurations per budget; regret refers to the proposed configuration. Each Mean row averages five budgets, and each Max row gives the largest regret among those budgets.

BFM	K	$J_{\text{prop}} \downarrow$	$J_{\text{best}} \downarrow$	Regret (%)↓	$\tau \uparrow$	$\rho \uparrow$	NDCG@5↑
SONIC	256	0.1621	0.1553	4.34	0.726	0.886	0.985
SONIC	512	0.0875	0.0752	16.27	0.758	0.916	0.956
SONIC	1024	0.0660	0.0647	2.07	0.678	0.842	0.902
SONIC	2048	0.0559	0.0559	0.00	0.479	0.631	0.691
SONIC	4096	0.0542	0.0537	0.91	0.527	0.703	0.945
SONIC	Mean	0.0851	0.0810	4.72	0.634	0.795	0.896
SONIC	Max	—	—	16.27	—	—	—
HEFT	256	0.0560	0.0516	8.59	0.806	0.944	0.945
HEFT	512	0.0486	0.0463	4.98	0.729	0.891	0.974
HEFT	1024	0.0447	0.0426	4.77	0.154	0.223	0.691
HEFT	2048	0.0409	0.0374	9.48	0.265	0.375	0.566
HEFT	4096	0.0385	0.0340	13.37	0.360	0.500	0.909
HEFT	Mean	0.0457	0.0424	8.24	0.463	0.586	0.817
HEFT	Max	—	—	13.37	—	—	—
TWIST2	256	0.5942	0.5780	2.80	0.615	0.804	0.946
TWIST2	512	0.4958	0.4885	1.49	0.758	0.919	0.983
TWIST2	1024	0.4468	0.4468	0.00	0.675	0.839	0.897
TWIST2	2048	0.4296	0.4296	0.00	0.448	0.595	0.601
TWIST2	4096	0.4065	0.4065	0.00	0.456	0.611	0.941
TWIST2	Mean	0.4746	0.4699	0.86	0.591	0.754	0.874
TWIST2	Max	—	—	2.80	—	—	—

C.5 The Benefits of Test-Time Compute Extend to Loco-Manipulation (Q4)

C.5.1 Label Comparison and Controller Interfaces

The source BFM is used during demonstration collection; the target BFM executes commands predicted by the learned high-level policy. In simulation, Original and State each produce one dataset per source and task, so their trained policies are reused across SONIC, TWIST2, and HEFT. Ours produces a separate set of labels and a trained policy for each target BFM.

C.5.2 Dynamics Randomization During Optimization

We optimize commands in MJWarp for execution in Isaac Sim or on the physical robot. To account for differences in dynamics, each candidate is evaluated in $R = 4$ randomized physics scenarios. The fixed CEM configuration is $(I, N, H, E) = (7, 39, 37, 5)$ with nominal budget $K = 2048$; including all scenarios gives a rollout budget of $RINH/E$.

At each planning update, we draw dynamics multipliers and initial-velocity perturbations from Table 20. All candidates share this scenario set across optimizer iterations, and the next planning update draws a new set. The demonstrated reference motion stays fixed. Each candidate's score is the arithmetic mean of its four scenario costs.

Table 20: Randomization distributions used in downstream label optimization. Multipliers scale nominal dynamics parameters; velocity perturbations are additive, zero-mean Gaussian offsets.

Parameter	Distribution
Contact-friction multiplier	$\mathcal{U}(0.8, 1.2)$
Joint-armature multiplier	$\mathcal{U}(0.8, 1.2)$
Joint-friction-loss multiplier	$\mathcal{U}(0.8, 1.2)$
Actuation-strength multiplier	$\mathcal{U}(0.9, 1.1)$
Root linear velocity offset	$\mathcal{N}(0, 0.02^2)$ per component
Root angular velocity offset	$\mathcal{N}(0, 0.05^2)$ per component
Joint velocity offset	$\mathcal{N}(0, 0.05^2)$ per component

C.5.3 HumanoidArena Data, Training, and Evaluation

We use five HumanoidArena (Wang et al., 2026a) tasks with the officially released LeRobot demonstrations. Closed-loop evaluation runs in Isaac Sim 5.0.0 with Isaac Lab 2.2.0 and Unitree Dex3 hands. Object randomization, lighting, and textures follow the collection conditions. All target BFMs operate at 50 Hz with frozen pretrained parameters.

Command labels. State encodes the recorded achieved motion into 40-dimensional actions comprising local displacement increments, root height, a six-dimensional orientation representation, 29 joint targets, and two source hand actions. Every frame encodes its achieved pose, including reset frames.

For Ours, the first 38 label dimensions encode the optimized motion commands applied to the target BFM during optimization using the same representation. Reset frames encode the achieved pose, and the two hand-action dimensions retain the source labels.

Training and evaluation. We train Diffusion Policy (Chi et al., 2025) with three seeds for each condition, using the frozen source–task training configuration. Training uses 100,000 updates with batch size 64, a ResNet18 visual encoder, and Adam with learning rate 10^{-4}. The action chunk horizon is 24 steps, of which 20 are executed before the next policy query. We evaluate the 100,000-step checkpoint for every seed using 60 rollouts, for a total of 180 rollouts per source–task–target and label condition.

Each rollout starts from the task's fixed default robot pose with randomized objects; initializations are matched across label conditions. For the Mean column, we first average the five task success rates within each seed, then compute the mean and sample standard deviation across seeds.

An episode ends when the task succeeds, the robot falls, or the task's step limit is reached. Table 21 gives the success conditions and limits.

Table 21: Success conditions and episode limits for the five HumanoidArena tasks.

Task	Success condition	Max steps
Boxing	The hand/fist proxy enters the target sphere around the punching bag.	900
Open Door	The latch is unlocked and the door opens at least 60°, with root height ≥ 0.45 m and up-axis $z \geq 0.60$.	1,800
Pick & Place Box	The box rests on the shelf support surface.	1,450
Sit Sofa	Sitting-related links enter the seat region with contact, or satisfy the geometric occupancy criterion.	2,000
Vision Navigation	Both feet reach the target region while the robot remains standing.	1,800

C.5.4 Real-Robot Data, Training, and Evaluation

We use 50 SONIC-collected demonstrations per task, approximately 20 minutes of data, to fine-tune GR00T N1.7-3B (NVIDIA, 2026). The physical robot is a Unitree G1 with a 29-DoF body, BrainCo Revo2 hands, and a ZED-M camera. SONIC execution uses the released low-latency whole-body controller (Luo et al., 2026). Table 22 summarizes the policy interface. Each policy query uses one egocentric image, a language task description, and proprioception to predict 40 action steps. SONIC decodes the body tokens into motor-angle targets at approximately 50 Hz. Hand commands follow a separate control path.

Table 22: Real-robot observations and actions at the policy interface. The body-and-hand state contains 41 values, with three additional projected-gravity components computed from the IMU orientation.

Signal	Size	Representation
Language	Text	Task description
Egocentric frame	$376 \times 672 \times 3$	ZED-M left eye (H×W×C)
Body state	29	Joint values
Hand state	$6 + 6$	BrainCo Revo2, normalized to $[0, 1]$
Projected gravity	3	Computed from IMU orientation
Body action	64	SONIC motion token
Hand action	$6 + 6$	BrainCo Revo2, normalized to $[0, 1]$

Labels and fine-tuning. During demonstration collection, we record the G1 motion references supplied to SONIC before token encoding, together with measured joint positions and root poses obtained from robot state estimation.

For SONIC execution, Original retains the stored collection-time SONIC motion tokens. State encodes the recorded achieved motion into 64-dimensional SONIC tokens. Ours first exports the optimized commands applied to the BFM during offline optimization, then encodes these commands into motion-token labels. Temporal windows, frame alignment, and reset handling follow the SONIC low-latency defaults.[2]

For HEFT execution, we use the released pretrained controller with frozen weights. Original expresses the recorded source G1 references in HEFT's motion-command representation. State uses the achieved motion assembled from the estimated root poses and measured joint positions. Ours uses commands optimized offline with the frozen HEFT controller to track the same achieved motion. All three conditions use the same action representation and retain the demonstration observations and hand-command labels.

The high-level policy predicts 29 joint targets and a nine-dimensional root-motion encoding per action step, together with the same 12 hand commands. The root encoding comprises sine and cosine-minus-one pairs for roll and pitch, the wrapped yaw increment, three position increments in the current yaw frame, and root height. At deployment, the adapter reconstructs root trajectories using the maintained reference XY position and the measured IMU heading, and supplies these trajectories and joint targets to HEFT. The root-pose estimates recorded during collection support offline label construction; deployment does not require external root-position odometry.

[2] https://nvlabs.github.io/GR00T-WholeBodyControl/tutorials/vla_workflow.html

We fine-tune the action head with the vision and language backbone frozen, using the Isaac-GR00T implementation.[3] Table 23 lists the shared fine-tuning parameters for SONIC execution.

Table 23: Shared GR00T fine-tuning configuration for the real-robot experiments with SONIC execution.

Parameter	Value
Optimizer	AdamW
Learning rate	10^{-4}
Global batch size	160
Learning-rate schedule	Cosine, 30,000 updates
Warmup	5% (1,500 updates)
Weight decay	10^{-5}
Maximum gradient norm	1.0
State dropout	0.2
Evaluated checkpoint	20,000 updates

Tasks and success. Table 24 describes the four tasks and their completion conditions. We evaluate each checkpoint for 60 rollouts. A trial starts near the demonstration's initial scene configuration and ends when the task succeeds or failure is judged.

Table 24: Real-robot task sequences and success conditions. Shelf placements use the designated target location.

Task	Execution sequence	Success condition
Cup in basket	Approach the table with one or two short steps, grasp the cup with the right hand, and place it in the basket to its right.	The cup is picked up and placed in the basket.
Parcel bag on shelf	Grasp the bag from the table, move and turn left toward the shelf, and place it at the target location.	The bag is picked up and placed at the target shelf location.
Basket on shelf	Squat to grasp the low basket with the right hand, rise while lifting it, then move and turn left to place it on the shelf.	The basket is picked up and placed at the target shelf location.
Bottle in drawer	Grasp the bottle from the cabinet top, lower it into the initially open drawer, and push the drawer closed.	The bottle is inside the drawer and the drawer is closed.

C.5.5 Complete Simulation Results

Table 25 reports the complete five-task comparison. The Mean column equally weights the five tasks. The Mean column is computed before rounding task success rates.

[3] https://github.com/NVIDIA/Isaac-GR00T

Table 25: HumanoidArena task success rates (%) for all evaluated source–target pairings in the five-task subset. Source is the demonstration-collection BFM and target is the execution BFM. Original, State, and Ours denote the three label conditions. Results are reported as mean ± sample standard deviation across three training seeds, with 60 evaluation rollouts per task and seed.

Source	Target	Labels	Boxing	Open Door	Pick & Place Box	Sit Sofa	Vision Navigation	Mean
SONIC	SONIC	Original	83.9 ± 1.0	84.4 ± 1.9	61.1 ± 1.0	88.9 ± 1.0	60.6 ± 1.0	75.8 ± 0.5
SONIC	SONIC	State	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
SONIC	SONIC	Ours	63.9 ± 1.0	51.1 ± 1.0	54.4 ± 1.0	66.7 ± 1.7	47.2 ± 1.0	56.7 ± 0.3
SONIC	TWIST2	Original	48.9 ± 1.9	0.0 ± 0.0	0.0 ± 0.0	17.8 ± 1.9	11.7 ± 1.7	15.7 ± 0.6
SONIC	TWIST2	State	33.9 ± 1.0	0.0 ± 0.0	0.0 ± 0.0	14.4 ± 1.0	1.7 ± 1.7	10.0 ± 0.3
SONIC	TWIST2	Ours	50.0 ± 1.7	0.0 ± 0.0	0.0 ± 0.0	21.7 ± 1.7	7.2 ± 1.9	15.8 ± 0.4
SONIC	HEFT	Original	46.7 ± 1.7	0.0 ± 0.0	0.0 ± 0.0	22.8 ± 1.9	16.1 ± 1.0	17.1 ± 0.4
SONIC	HEFT	State	38.9 ± 1.9	0.0 ± 0.0	0.0 ± 0.0	36.1 ± 1.0	27.8 ± 1.9	20.6 ± 0.7
SONIC	HEFT	Ours	57.2 ± 1.0	0.0 ± 0.0	0.0 ± 0.0	67.8 ± 1.0	18.9 ± 1.0	28.8 ± 0.2
TWIST2	TWIST2	Original	55.0 ± 1.7	71.1 ± 1.0	41.7 ± 2.9	59.4 ± 1.0	59.4 ± 1.9	57.3 ± 1.2
TWIST2	TWIST2	State	38.3 ± 2.9	0.0 ± 0.0	0.0 ± 0.0	72.8 ± 1.9	44.4 ± 2.5	31.1 ± 0.4
TWIST2	TWIST2	Ours	50.0 ± 1.7	43.3 ± 1.7	42.8 ± 1.0	48.3 ± 2.9	55.6 ± 2.5	48.0 ± 1.7
TWIST2	SONIC	Original	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	2.8 ± 1.9	36.7 ± 2.9	7.9 ± 0.8
TWIST2	SONIC	State	5.0 ± 2.9	0.0 ± 0.0	0.0 ± 0.0	22.8 ± 2.5	50.6 ± 2.5	15.7 ± 1.2
TWIST2	SONIC	Ours	37.8 ± 2.5	0.0 ± 0.0	0.0 ± 0.0	56.7 ± 2.9	34.4 ± 1.9	25.8 ± 1.4
TWIST2	HEFT	Original	36.7 ± 1.7	0.0 ± 0.0	0.0 ± 0.0	31.1 ± 1.0	4.4 ± 1.0	14.4 ± 0.2
TWIST2	HEFT	State	59.4 ± 1.0	0.0 ± 0.0	0.0 ± 0.0	46.7 ± 2.9	50.6 ± 2.5	31.3 ± 1.2
TWIST2	HEFT	Ours	47.8 ± 1.9	0.0 ± 0.0	0.0 ± 0.0	35.0 ± 1.7	24.4 ± 2.5	21.4 ± 0.8

D COMPUTE RESOURCES

We run experiments on NVIDIA GeForce RTX 3080, RTX 3080 Ti, RTX 4090, and H100 80GB GPUs. For the TTCO runs, wall-clock time on an RTX 3080 Ti can be approximated by

$$t(C, N) = \left(58.1 + 0.009961 \frac{C}{N}\right) \text{ s}, \qquad R^2 = 0.890, \tag{44}$$

The $1/N$ factor reflects GPU parallelism across candidates. For large datasets, motions share finite GPU capacity, so total computation C remains relevant to compute-optimal allocation.

The runs in Secs. 4.1–4.3 used approximately 1,800, 380, and 1,700 RTX 3080 Ti-equivalent GPU hours, respectively. The runs in Sec. 4.4 used approximately 700 H100 GPU hours.