SBTO: Sampling-Based Trajectory Optimization for Dynamically-Feasible Retargeting

Victor Dhédin*, Ilyass Taouil*, Majid Khadiv
Munich Institute of Robotics and Machine Intelligence (MIRMI)
Technical University of Munich (TUM), Germany
Email: {firstname.lastname}@tum.de

***Abstract*—In this paper, we introduce a novel Sampling-Based Trajectory Optimization (SBTO) framework that refines imperfect kinematic trajectories into dynamically feasible motions. SBTO incrementally advances the optimization horizon, enabling optimization over the entire trajectory for long-horizon tasks. We validate *SBTO* by successfully refining hundreds of humanoid–object demonstrations and achieving higher success rates than the state of the art. The framework also generalizes across varying object properties, such as mass, size, and geometry, using the same tracking objective. This ability to robustly retarget diverse demonstrations opens the door to generating large-scale synthetic datasets of humanoid loco-manipulation trajectories, addressing a major bottleneck in real-world data collection. A supplementary video demonstrating the results is available here. This paper is a short version of [1].**

I. Introduction

Generating feasible loco-manipulation behaviors is a highly complex problem, as it requires handling the underactuation of both the robot and the manipulated object, as well as complex contact interactions. Traditional frameworks mostly relied either on gradient-based optimization to generate optimal trajectories [2], or on deep reinforcement learning (RL) [3] to directly learn optimal policies. The main advantage of trajectory optimization (TO) lies in its efficiency at finding locally optimal trajectories; however, it often requires augmentation with search-based methods to enable sufficient exploration [4]–[6]. On the other hand, while RL is very effective at generating robust behaviors, exploration remains a major challenge, frequently leading to heavy reward shaping for each individual motion.

One plausible remedy to the exploration problem is the use of demonstrations. In particular, thanks to the similarity between human and humanoid morphology, this line of research has recently attracted significant attention and has led to many impressive results [7]–[9]. The main idea behind these works is to retarget human motions to humanoid robots and then use the resulting trajectories as inputs to an RL policy, using domain randomization for sim-to-real transfer. In this way, the exploration problem is largely alleviated, and the RL component requires only a small set of simple reward terms that are shared across motions.

While the RL structure is largely standard among these approaches, the retargeting module differs substantially. Most methods solve a kinematic optimization problem for retargeting [9]–[12], and are therefore susceptible to artifacts such as physical and geometric inconsistencies, particularly for loco-manipulation tasks. More recently, [8] proposed using sampling-based model predictive control (SBMPC) to improve retargeting quality. However, this approach solves only a short-horizon optimization problem at each iteration, making it difficult to handle long-horizon behaviors due to its inherently myopic nature.

In this paper, we introduce *SBTO*, which introduces a novel sampling-based trajectory optimization (SBTO) method that incrementally increases the optimization horizon to ultimately solve the original long-horizon problem. The generated trajectories are then fed to an RL module to learn robust tracking policies using domain randomization during training. Through extensive ablation studies and comparisons, we show that *SBTO* significantly improves success rates. For conciseness, we omit the RL results in this paper, those can be seen in the accompanying video.

II. Related Works

The work most closely related to ours is SPIDER [8], which employs an SBMPC framework [13]–[15] to improve the geometric and dynamic consistency of retargeted motions. Given a kinematically retargeted trajectory, SBMPC generates dynamically consistent motions by repeatedly optimizing control trajectories over a short horizon in a receding horizon fashion. However, because the optimization considers only a limited horizon at each step, the method is sensitive to imperfections in the demonstration. Failures can occur during the retargeting phase, as full-horizon consistency is not explicitly enforced.

III. Method

A. Optimal Control with Sampling-Based Optimization

Sampling-based (or *zero-order*) optimization algorithms address the generic optimization problem $\min_{x\in\mathbb{R}^n} f(x)$ by only point-wise evaluating a known function f to be minimized. The gradient of f is not required, which makes it popular for non-smooth and non-convex optimization problems, such as optimal control of robots for contact-rich tasks.

In optimal control, the objective is to find a control sequence $\{\mathbf{u}_0, \dots, \mathbf{u}_{T-1}\}$ minimizing a cost function J while satisfying

This work was partially supported by the Huawei-TUM joint laboratory.

the system's dynamics $\mathbf{x}_{t+1} = f_{dyn}(\mathbf{x}_t, \mathbf{u}_t)$, as formulated below:

$$\begin{aligned} &\min_{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}} J(\mathbf{x}_{0:T}, \mathbf{u}_{0:T-1}) \\ &\text{s.t.} \quad \mathbf{x}_0 = \mathbf{x}_{ini}, \;\; \mathbf{x}_{t+1} = f_{\text{dyn}}(\mathbf{x}_t, \mathbf{u}_t). \end{aligned} \tag{1}$$

$\mathbf{x}_t = (\mathbf{q}_t, \mathbf{v}_t) \in \mathbb{R}^{n_x}$ and $\mathbf{u}_t \in \mathbb{R}^{n_u}$ denote respectively the state (joint positions and velocities) and control of the system at time t. To satisfy the dynamics constraint, control sequences $\mathbf{u}_{0:T-1}$ are rolled-out in a single-shooting fashion from the initial state $\mathbf{x}_0$ using a simulator (treating it as a black box), which ultimately outputs state trajectories $\mathbf{x}_{0:T}$ needed to evaluate the cost $J(\mathbf{x}_{0:T}, \mathbf{u}_{0:T-1})$.

The most commonly used algorithms in robotics are Cross-Entropy Method (CEM) [16], a special case of Covariance Matrix Adaptation (CMA) [17], and Model Predictive Path Integral (MPPI) [14]. In general, these approaches are used in a receding horizon fashion [18], [19]. These algorithms rely on similar mechanisms, namely trying to approximate gradients at each iteration [20]; however, they differ in how the sampling distribution is updated, as outlined in algorithm 1.

To reduce the size of the sampling space, it is common to sample *interpolation knots* $\mathbf{k} \in \mathbb{R}^{K \cdot n_u}$ instead of the full control trajectory $\mathbf{u}_{0:T-1}$. Those knots are usually equally spread in time at steps $\boldsymbol{\tau} \in \mathbb{N}^K$, with $\tau_0 = 0$ and $\tau_{K-1} = T - 1$.

Algorithm 1 FHTO, Sampling-Based Fixed Horizon Trajectory Optimization

Inputs $\boldsymbol{\mu} \in \mathbb{R}^{K \cdot n_u}, \boldsymbol{\Sigma} \in \mathbb{R}^{K \cdot n_u \times K \cdot n_u}$, $\boldsymbol{\tau} \in \mathbb{N}^K$, N samples, I iterations
for iteration $i = 1, 2, \ldots, I$ **do**
  Sample N interpolation knots $\{\mathbf{k}^j\}_{j=1}^N \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
  for each sample j **do**
    Interpolate: $\mathbf{u}^j_{0:T-1} = \texttt{interp}(\mathbf{k}^j, \boldsymbol{\tau})$
    Roll-out dynamics: $\mathbf{x}^j_{t+1} = f_{\text{dyn}}(\mathbf{x}^j_t, \mathbf{u}^j_t)$
    Evaluate cost: $J^j = J(\mathbf{x}^j_{0:T}, \mathbf{u}^j_{0:T-1})$
  end for
  Update parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ using $\{J^j\}_{j=1}^N$
    ▷ CEM: from N_e elites ▷ MPPI: exponential average weights
end for
Return best control sequence $\mathbf{u}^*_{0:T-1}$

B. Issues with existing SBMPC-based retargeting methods

Successful works in the literature that use zero-order optimization for retargeting [8], [21], [22] solve a short-horizon problem in the form of (1) in an MPC fashion. However, such an approach suffers from three important issues:

1) As MPC repeatedly solves a short-horizon problem, it can exhibit myopic behavior in long-horizon tasks. This effect is further exacerbated when using imperfect references that are physically inconsistent. For instance, if the contact geometry in the reference trajectory is inaccurate, the resulting short-horizon optimal plans may also fail to establish the correct contacts, ultimately leading to task failure.
2) This is amplified as SBMPC-based retargeting behaves in a greedy fashion: once an action is executed, the system is simulated forward, and earlier actions cannot be re-optimized. For example, if SBMPC mistakenly drops the object during the early phases of a motion, recovery is highly unlikely, as doing so would require substantial deviation from the reference, which is penalized by the tracking cost.
3) The trajectories produced by SBMPC tend to be jerky, as they are generated through feedback control. This lack of smoothness can negatively affect both the training process and the performance of downstream RL policies.

Considering the full horizon of the problem would avoid these pitfalls. However, humanoid loco-manipulation is a high-dimensional problem; optimizing all control variables simultaneously with a single-shooting sampling-based optimizer is therefore likely to fail, as both the number of variables and the number of local minima increase with the horizon length. Our key observation is that the control variables toward the end of the trajectory strongly depend on those at the beginning. Updating the last control variables before the early ones are sufficiently optimized can lead to undesirable updates, making the optimization inefficient and potentially preventing convergence to desired behaviors. Based on this observation, in the next subsection, we introduce SBTO, a trajectory optimization framework that incrementally increases the optimization horizon.

C. Sampling-Based Trajectory Optimization

SBTO optimizes control variables incrementally. First, control knots $\mathbf{k}_0$ at time τ_0 are optimized, then control knots $\{\mathbf{k}_0, \mathbf{k}_1\}$ at time $\{\tau_0, \tau_1\}$ (warm-starting with the previous solution $\mathbf{k}^*_0$), and so on, until all knots are being optimized. The algorithm contains two nested loops: the outer loop incrementally increases the number of decision variables being optimized, while the inner loop repeatedly refines *all* the currently active variables. A skeleton of the algorithm can be seen in Algorithm 2.

To optimize knots $\{\mathbf{k}_0, \ldots, \mathbf{k}_k\}$, we do not perform the full horizon roll-out until T, but a partial rollout until τ_k, with τ_k corresponding to the time step of the last knot being optimized. Also, the optimization horizon τ_k grows incrementally. At each increment k, this procedure is equivalent to solving the Fixed-Horizon Trajectory Optimization (FHTO) from Algorithm 1 with truncated parameters $\boldsymbol{\mu}_{0:\kappa}, \boldsymbol{\Sigma}_{0:\kappa,0:\kappa}$ and knot time $\boldsymbol{\tau}_{0:k}$. $\kappa = (k+1)n_u - 1$ denotes the index of the last variable associated with knot $\mathbf{k}_k$. Since $\tau_0 = 0$, the process starts at $k = 1$ in practice.

Increments occur when the maximum diagonal value of the covariance matrix $\boldsymbol{\Sigma}$ is below a threshold σ_{min}, indicating that the optimization has sufficiently converged. This adaptive criterion allows SBTO to adjust to the growing number of variables, as the convergence rate can change when more variables are being optimized.

Intuitively, σ_{min} plays an important role in the convergence of the algorithm. Setting σ_{min} large enough ensures that

$\boldsymbol{\Sigma}_{0:\kappa,0:\kappa}$ has not fully converged before incrementing. This enables all variables in the current horizon window to be optimized, even after multiple increments, which prevents early convergence to local minima. However, a too small σ_{min} makes the sampling distribution shrink to a point distribution, which would prevent the first variables from escaping a potentially bad local minimum after incrementing. Conversely, a too large σ_{min} would make the increment happen too early with the variables too far from the optimum, which would almost be equivalent to solving the full-horizon TO from scratch (shown in the video.

In practice, earlier decision variables could converge before the full horizon is reached, causing the early part of the state trajectory to eventually become effectively fixed over iterations. We therefore propose an improved implementation that caches these initial trajectory segments and skips their recomputation in the following roll-outs (noted *SBTO_skip*). To do so, the algorithm identifies the latest knot κ_0 such that all earlier controls have already converged (i.e. their variances fall below a threshold σ_{skip}), and optimizes all knots between κ_0 and the last knot of the current increment κ. This can reduce computation quite significantly for longer-horizon problems, as it is not required to roll out from τ_0 at each step. This improvement can be seen in Algorithm 2.

Note that SBTO is tailored for problems having a dense cost J where even a short-horizon window provides a meaningful estimate of the optimal solution. Retargeting tasks satisfy this property, which motivates our evaluation of the proposed method in this context.

Algorithm 2 SBTO and **SBTO_skip**

Inputs $\boldsymbol{\mu} \in \mathbb{R}^{K \cdot n_u}, \boldsymbol{\Sigma} \in \mathbb{R}^{K \cdot n_u \times K \cdot n_u}$, $\boldsymbol{\tau} \in \mathbb{N}^K$, N samples, $I = 1$ iteration, $\sigma_{\min}$, σ_{skip}
for $k = 1, 2, \ldots, K-1$ **do**
  $\kappa \leftarrow (k+1)n_u - 1$
  while $\max(\text{diag}(\boldsymbol{\Sigma}_{0:\kappa,0:\kappa})) > \sigma_{\min}$ **do**
    if allow_skip **then**
      $k_0 \leftarrow \max\{j < k \mid \text{diag}(\boldsymbol{\Sigma}_{0:j \cdot n_u, 0:j \cdot n_u}) < \sigma_{\text{skip}}\}$
        ▷ All variables before k_0 have converged
    else
      $k_0 \leftarrow 0$
    end if
    $\kappa_0 \leftarrow k_0 \cdot n_u$
    $\mathbf{u}_{\tau_{k_0}:\tau_k} \leftarrow \text{FHTO}(\boldsymbol{\mu}_{\kappa_0:\kappa}, \boldsymbol{\Sigma}_{\kappa_0:\kappa,\kappa_0:\kappa}, \boldsymbol{\tau}_{k_0:k}, N, I)$
      ▷ Skip roll-out before τ_{k_0}
  end while
end for
return $\mathbf{u}_{0:T-1}$

IV. Evaluation

We evaluate SBTO on a trajectory refinement task using reference motions from the OmniRetarget dataset [9]. As input, we use kinematically retargeted trajectories from this dataset; in fact, SBTO performs *dynamic refinement*, correcting kinematically imperfect trajectories to produce dynamically feasible whole-body motions. The dataset contains hundreds of motions of a G1 humanoid robot interacting with a box, including pick-and-place, kicking, and pushing or dragging motions. Many of these trajectories exhibit missing contacts, penetrations, or discontinuities, making them challenging to refine.

In subsection IV-A, we provide implementation details of SBTO. Subsection IV-B compares SBTO's performance with a state-of-the-art SBMPC. Section IV-C demonstrates that SBTO can adapt to objects with properties (shape, etc.) different from the original demonstrations.

A. Implementation details

We implemented SBTO using the MuJoCo simulator [23] and leveraged its newly introduced `rollout` function to perform parallel rollouts on the CPU. We used a simulation timestep of $\Delta t = 0.01$ s and considered the full collision model of the robot. The time interval between knots is independent of the reference and is set to 0.25 s. The control sequence $\boldsymbol{u}_{0:T}$ corresponds to a PD target trajectory.

In all experiments, we use CEM [24] to update the sampling distribution. Following [18], we additionally retain a subset of elite samples across iterations ($N_{keep} = \lceil \rho_k \rho_e N \rceil$) and apply an exponentially weighted moving average (EWMA) with momentum parameters α_μ and α_Σ to prevent premature shrinking of the distribution [25]. The initial mean of the distribution is set to the joint positions at each knot time step from the reference, $\boldsymbol{\mu} = \mathbf{q}^{ref}_{0:T}[\boldsymbol{\tau}]$, while the initial covariance is set to $\boldsymbol{\Sigma} = \sigma_0^2 \mathbf{I}_{K \cdot n_u}$. We observed that considering the full covariance matrix improved convergence. Hyperparameter values are reported in Table II.

The cost function penalizes deviations in the state position $\mathbf{q}_{0:T}$ and velocity $\mathbf{v}_{0:T}$. Additional task-space terms enforce tracking of desired torso, foot, and hand poses. Contact-related terms discourage undesired collisions. The cost weights are summarized in Table III.

B. Performance evaluation

We evaluate SBTO on all motions from the OmniRetarget dataset [9] that are shorter than 9 s (285 motions in total). SPIDER [8], a recently released SBMPC that achieves state-of-the-art results on many dynamic refinement tasks, serves as our baseline. In SPIDER, the cost terms are based solely on the configurations $\boldsymbol{q}_{0:T}$ (and not their velocities). For a fair comparison, we also evaluate a variant of SBTO using a similar configuration - considering only the terms described in the first section of Table III and omitting the velocity terms — referred to as SBTO_pos. We compare the methods using three metrics: algorithm success rate, computational efficiency, and smoothness of the resulting trajectories (all described below). The results are reported in Table I.

We consider the refinement successful when the object trajectory has an average position error $E_{\text{pos}} < 10$cm and an

TABLE I

ALGORITHM PERFORMANCE COMPARISON. THE COMPUTATIONAL EFFICIENCY AND SMOOTHNESS ARE AVERAGED OVER THE SUCCESSFUL TRAJECTORIES ONLY. FOR THE COMPUTE, WE PROVIDE ABSOLUTE AND RELATIVE VALUES (SEPARATED BY —).

Algorithm	**Success** (%) ↑	**Smoothness** ↓	**Compute** η_{eff} ↓	E_{pos} (m) ↓	E_{rot} (rad) ↓
SBTO_skip	**76.8**	**1.41**	**1.18e7**—0.96	**0.11**±0.17	**0.32**±0.35
SBTO	74.6	1.7	4.06e7—3.3	0.15±0.28	0.36±0.44
SBTO_pos	62.1	2.7	4.46e7—3.6	0.24±0.37	0.42±0.45
SPIDER	37.9	3.4	1.23e7—1.	0.33±0.28	0.55±0.42

average rotation error $E_{\text{rot}} < 25°$. The error terms are defined as below:

$$E_{\text{pos}} = \frac{1}{T}\sum_{t=1}^{T} \left\| \mathbf{p}_{\text{obj},t} - \mathbf{p}^{\text{ref}}_{\text{obj},t} \right\|_2 \tag{2}$$

$$E_{\text{rot}} = \frac{180}{\pi}\frac{1}{T}\sum_{t=1}^{T} \arccos\!\left(2\langle \mathbf{q}_{\text{obj},t}, \mathbf{q}^{\text{ref}}_{\text{obj},t}\rangle^2 - 1\right) \tag{3}$$

We define computational efficiency η_{eff} as the total number of simulation steps required in the optimization, divided by the duration of the reference. This makes the metric independent of the machine or simulator, providing a more meaningful measure than one based on compute time. For SBTO, the computational cost depends on the number of knots being optimized at iteration i, denoted by $k(i)$, as can be seen below:

$$\eta_{\text{eff}} \;=\; \frac{N_{\text{sim}}}{T\Delta t} \overset{SBTO}{=} \frac{N}{T\Delta t}\sum_{i\in\mathcal{I}} \tau_{k(i)} \tag{4}$$

Finally, we define the trajectory smoothness S as the sum of accelerations of all *actuated* joints (obtained by finite differencing) over the full trajectory. For better interpretability, we normalize the result by the trajectory smoothness of its corresponding reference $\tilde{S} = \frac{S}{S_{\text{ref}}}$.

$$S \;=\; \sum_{t=2}^{T-1} \left\|\ddot{\mathbf{q}}_t\right\|_1, \quad \text{with} \quad \ddot{\mathbf{q}}_t = \frac{\mathbf{q}_{t+1} - 2\mathbf{q}_t + \mathbf{q}_{t-1}}{\Delta t^2} \tag{5}$$

As summarized in Table I, SBTO outperforms SPIDER, achieving nearly twice the success rate and producing smoother refined trajectories. Even SBTO_pos shows a clear improvement over SPIDER, highlighting the algorithmic advantages of SBTO. While SBTO is computationally expensive, the SBTO_skip variant reduces the computational cost by a factor of three, making it cheaper than SPIDER, while also achieving slightly better performance. In practice, the refinement process takes roughly 20 second per second of refined motion on a 112-core Intel(R) Xeon(R) Platinum 8480+ CPU. Failure cases typically occur when references are of poor quality, especially if they include sudden changes in hand–object contact or abrupt flips in object orientation.

C. Demonstration Augmentation

SBTO produces trajectories that deviate from the kinematic reference to ensure dynamic feasibility. One way to quantify how much it could deviate is to evaluate refinement performance under changes in object properties, such as mass, size, and geometry. This evaluation is also important in practice, as collecting new demonstrations is expensive.

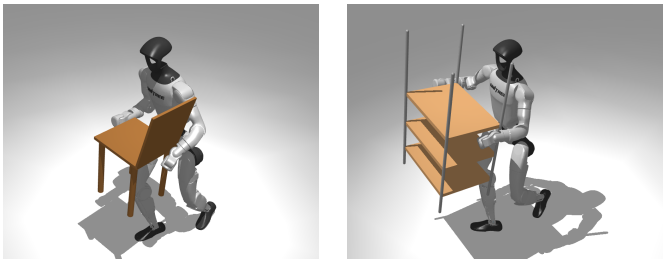

Fig. 1. Trajectory snapshots of `sub_10_largebox_084` with the original box geometry being replaced by a chair (left) and a shelf (right).

We evaluate the success rate of SBTO on a box with different masses and sizes, and geometries. All experiments are based on a single motion reference (`sub10_largebox_084_original`). The original object size is a cubic box of length 0.31cm and 0.6kg mass. We use the same cost terms and optimization settings as in previous experiments.

SBTO successfully handled boxes with masses ranging from 0.1 to 8kg and sizes ranging from 0.2m to 0.4m. Furthermore, the dynamic refinement process generalized beyond boxes and was also successful on a cylinder (diameter and height of 0.31m), a chair, and a shelf (as can be seen on Figure 1). This shows that a single demonstration can be refined into dynamically feasible motions across a diverse set of object geometries and physical properties.

V. CONCLUSION

In this work, we presented SBTO, a new sampling-based method that incrementally grows the optimization horizon, effectively warm-starting the full-horizon problem while still allowing early decision variables to be refined as the horizon grows. This strategy mitigates the convergence challenges of sampling-based methods on long-horizon high-dimensional problems, while simultaneously overcoming the short-sightedness of SBMPC.

We extensively evaluated SBTO on hundreds of motions and showed that, for the same computational cost, it achieves a substantially higher success rate than a state-of-the-art SBMPC baseline, while producing smoother trajectories.

REFERENCES

[1] V. Dhedin, I. Taouil, S. Omar, D. Yu, K. Tao, A. Dai, and M. Khadiv, "Dynaretarget: Dynamically-feasible retargeting using sampling-based trajectory optimization," 2026.

[2] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. D. Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, vol. 40, pp. 43–63, 2024.

[3] S. Ha, J. Lee, M. van de Panne, Z. Xie, W. Yu, and M. Khadiv, "Learning-based legged locomotion: State of the art and future perspectives," *The International Journal of Robotics Research*, vol. 44, no. 8, pp. 1396–1427, 2025.
[4] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," 2018.
[5] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multi-contact pattern generation with sequential convex approximations of the centroidal dynamics," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1661–1679, 2021.
[6] M. Ciebielski, V. Dhédin, and M. Khadiv, "Task and motion planning for humanoid loco-manipulation," in *2025 IEEE-RAS 24th International Conference on Humanoid Robots (Humanoids)*, pp. 1179–1186, IEEE, 2025.
[7] S. Zhao, Y. Ze, Y. Wang, C. K. Liu, P. Abbeel, G. Shi, and R. Duan, "Resmimic: From general motion tracking to humanoid whole-body loco-manipulation via residual learning," 2025.
[8] C. Pan, C. Wang, H. Qi, Z. Liu, H. Bharadhwaj, A. Sharma, T. Wu, G. Shi, J. Malik, and F. Hogan, "Spider: Scalable physics-informed dexterous retargeting," 2025.
[9] L. Yang, X. Huang, Z. Wu, A. Kanazawa, P. Abbeel, C. Sferrazza, C. K. Liu, R. Duan, and G. Shi, "Omniretarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction," 2025.
[10] H. Weng, Y. Li, N. Sobanbabu, Z. Wang, Z. Luo, T. He, D. Ramanan, and G. Shi, "Hdmi: Learning interactive humanoid whole-body control from human videos," 2025.
[11] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, Z. Yi, G. Qu, K. Kitani, J. Hodgins, L. J. Fan, Y. Zhu, C. Liu, and G. Shi, "Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills," 2025.
[12] Z. Luo, J. Cao, A. Winkler, K. Kitani, and W. Xu, "Perpetual humanoid control for real-time simulated avatars," 2023.
[13] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," *arXiv preprint arXiv:2212.00541*, 2022.
[14] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.
[15] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi, "Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing," 2024.
[16] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2004.
[17] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.
[18] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolinek, and G. Martius, "Sample-efficient cross-entropy method for real-time planning," 2020.
[19] V. Kurtz and J. W. Burdick, "Generative predictive control: Flow matching policies for dynamic and difficult-to-demonstrate tasks," *arXiv preprint arXiv:2502.13406*, 2025.
[20] A. Jordana, J. Zhang, J. Amigo, and L. Righetti, "An introduction to zero-order optimization techniques for robotics," 2025.
[21] A. T. Le, K. Nguyen, M. N. Vu, J. Carvalho, and J. Peters, "Model tensor planning," 2025.
[22] V. Kurtz, "Hydrax: Sampling-based model predictive control on gpu with jax and mujoco mjx," 2024. https://github.com/vincekurtz/hydrax.
[23] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033, IEEE, 2012.
[24] P.-T. Boer, D. Kroese, S. Mannor, and R. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, pp. 19–67, 02 2005.
[25] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "Chapter 3 - the cross-entropy method for optimization," in *Handbook of Statistics* (C. Rao and V. Govindaraju, eds.), vol. 31 of *Handbook of Statistics*, pp. 35–59, Elsevier, 2013.

VI. Appendix

A. Sampling hyperparameters

TABLE II
CEM AND SBTO HYPERPARAMETERS

Parameters	Value
Number of samples N	1024
Elite set proportion ρ_e	0.03
Keep elites proportion ρ_k	0.04
Mean momentum α_μ	0.95
Covariance momentum α_Σ	0.2
Initial std. σ_0	0.25
Increment threshold σ_{min}	0.055
Skip threshold σ_{skip}	10^{-4}

B. Cost function

TABLE III
COST TERMS AND CORRESPONDING WEIGHTS. $\ominus$ DENOTES THE SUBTRACTION BETWEEN QUATERNIONS IN THE TANGENT SPACE. COLLISION COSTS ARE COMPUTED AS THE NUMBER OF COLLISION EVENTS, WHICH CAN BE EASILY OBTAINED WITH MUJOCO.

Cost term	**Equation**	**Weight**
Motion Tracking		
Joint position	$\|\mathbf{q}_{act} - \mathbf{q}^{\text{ref}}_{act}\|^2$	0.25
Joint velocity	$\|\mathbf{v}_{act} - \mathbf{v}^{\text{ref}}_{act}\|^2$	0.01
Base position	$\|{}^W\mathbf{p}_{\text{base}} - {}^W\mathbf{p}^{\text{ref}}_{\text{base}}\|^2$	5.0
Base orientation	${}^W\mathbf{q}_{\text{base}} \ominus {}^W\mathbf{q}^{\text{ref}}_{\text{base}}$	1.0
Object position	$\|{}^W\mathbf{p}_{\text{object}} - {}^W\mathbf{p}^{\text{ref}}_{\text{object}}\|^2$	40.0
Object orientation	${}^W\mathbf{q}_{\text{object}} \ominus {}^W\mathbf{q}^{\text{ref}}_{\text{object}}$	4.0
Object linear velocity	$\|{}^W\mathbf{v}_{\text{object}} - {}^W\mathbf{v}^{\text{ref}}_{\text{object}}\|^2$	0.2
Motion Tracking (task space)		
Torso position	$\|{}^W\mathbf{p}_{\text{torso}} - {}^W\mathbf{p}^{\text{ref}}_{\text{torso}}\|^2$	30.0
Torso orientation	${}^W\mathbf{q}_{\text{torso}} \ominus {}^W\mathbf{q}^{\text{ref}}_{\text{torso}}$	3.0
Torso linear velocity	$\|{}^W\mathbf{v}_{\text{torso}} - {}^W\mathbf{v}^{\text{ref}}_{\text{torso}}\|^2$	0.3
Torso angular velocity	$\|{}^W\mathbf{w}_{\text{torso}} - {}^W\mathbf{w}^{\text{ref}}_{\text{torso}}\|^2$	0.1
Foot position	$\sum_{i\in\mathcal{F}} \|{}^W\mathbf{p}_{\text{foot}_i} - {}^W\mathbf{p}^{\text{ref}}_{\text{foot}_i}\|^2$	10.0
Hand position	$\sum_{j\in\mathcal{H}} \|{}^W\mathbf{p}_{\text{hand}_j} - {}^W\mathbf{p}^{\text{ref}}_{\text{hand}_j}\|^2$	5.0
Regularization		
Robot–object collision	$\sum_{c\in\mathcal{C}_{\text{ro}}} 1$	2.0
Self-collision	$\sum_{c\in\mathcal{C}_{\text{self}}} 1$	1.0