Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

Consensus-based Optimization (CBO): Towards Global Optimality in Robotics

Xudong Sun*†, Armand Jordana‡, Massimo Fornasier*, Jalal Etesami*†, Majid Khadiv*†
*School of Computation, Information and Technology, Technical University of Munich, Munich, Germany
Email: {xudong.sun, massimo.fornasier, j.etesami, majid.khadiv}@tum.de
†Munich Institute of Robotics and Machine Intelligence (MIRMI), Munich, Germany
‡LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France Email: armand.jordana@laas.fr

***Abstract*—Zero-order optimization has recently received significant attention for designing optimal trajectories and policies for robotic systems. However, most existing methods (e.g., MPPI, CEM, and CMA-ES) are local in nature, as they rely on gradient estimation. In this paper, we introduce consensus-based optimization (CBO) to robotics, which is guaranteed to converge to a global optimum under mild assumptions. We provide theoretical analysis and illustrative examples that give intuition into the fundamental differences between CBO and existing methods. To demonstrate the scalability of CBO for robotics problems, we consider three challenging trajectory optimization scenarios: (1) a long-horizon problem for a simple system, (2) a dynamic balance problem for a highly underactuated system, and (3) a high-dimensional problem with only a terminal cost. Our results show that CBO is able to achieve lower costs with respect to existing methods on all three challenging settings. This opens a new framework to study global trajectory optimization in robotics.**

I. Introduction

State-of-the-art trajectory and policy optimization relies heavily on engineered heuristics, subtle design of cost functions, and careful weighting of different objectives [1, 2]. One alternative to the cumbersome tuning of cost functions is to train policies directly retargeting human demonstrations to robots [3, 4] or using large quantities of data collected via teleoperation [5, 6]. While these approaches have demonstrated impressive results in recent years, they can hardly guarantee any kind of optimality and might fail to obtain the full set of valid solutions to a given problem. Hence, it is desirable to obtain favorable behaviors in robotics without reliance on human motion and other forms of high quality demonstration. Achieving this goal requires a capable optimization algorithm for trajectory optimization under high state-control dimensions and long horizons.

Traditionally, trajectory optimization in robotics has relied on gradient-based methods [1]. However, these techniques are well known to struggle with problems involving complex contact interactions, as (rigid) contact is inherently a hybrid phenomenon. Thanks to recent advances in parallelized simulation [7, 8], it is now possible to run massive parallel rollouts. This enables the simultaneous evaluation of many different control trajectories. As a result, zero-order (gradient-free) optimization has become increasingly popular for trajectory and policy optimization in robotics [9, 10]. By relying solely on function evaluations, these techniques avoid the challenging development of a differentiable simulator and are naturally contact-implicit.

Popular zero-order optimization techniques used for trajectory optimization in robotics are model predictive path integral (MPPI) [11], cross entropy method (CEM) [12], and covariance matrix adaptation evolution strategy (CMA-ES) [13]. These methods have shown promise in solving complex robotic problems in real-time [14, 15, 16, 17]. The common ground among these techniques is that they iteratively perform a random search by generating samples and updating an estimate of the optimal solution based on the performance of each sample. However, the sampling distribution is local, typically a multivariate Gaussian centered at the current estimate. The update step then approximates a gradient descent step [18]; this leads to local optimality.

In this paper, we explore the use of consensus-based optimization (CBO) [19] via particle dynamics to achieve global optimality in trajectory optimization for robotics. Instead of sampling locally, CBO uses a population of particles to explore globally. Each particle follows a diffusion dynamic that drives it towards a consensus point defined by the weighted average of the population, where the weights are determined by the trajectory costs. This approach allows for a more flexible exploration of the solution space and has been shown to converge to global optima under mild assumptions [20, 21].

Our major contributions in this work are:

- We reinterpret zero-order optimization methods, widely used in robotics, under a general global optimization framework via surrogate objective minimization, which naturally highlights their weaknesses in achieving global optimality. We offer an intuitive explanation of how CBO works, how it addresses the weaknesses of other zero-order optimization methods, draw connections to current methods, and reinterpret other methods through the lens of the diffusion process dynamics.
- We demonstrate that the CBO algorithm can identify meaningful global solutions in robotic trajectory optimization, even for high-dimensional humanoid models and problems with long time horizons. In particular, we illustrate how techniques based on local sampling can get stuck in local minima, regardless of the number of particles used, while CBO can find a global solution as the number of particles increases.

It is important to emphasize that CBO has been successfully tested on many global optimization benchmarks [22] and applications [23]. This paper is the first to demonstrate its applicability to robotics problems and its superior performance compared to other zero-order optimization methods used in robotics. It is also important to note that particle-based evolutionary strategies have a long history in robotics for performing global optimization [24]. Rather than proposing a new optimization paradigm, here we introduce a unifying framework under which these meta-heuristic methods can be formally analyzed and studied within a rigorous mathematical setting.

The remainder of the paper is organized as follows. Section II presents a unified framework for widely used zero-order optimization methods in robotics and highlights their inherently local nature. Section III introduces the CBO formulation and provides intuition for its global behavior. The experimental results are presented in Section IV. Finally, Section V concludes the paper and summarizes our findings.

II. Preliminaries

A. Notations

In this section, we summarize the notions used throughout the paper to improve the readability:

- r: optimization iteration index
- $x_t \in \mathcal{S} \subset \mathbb{R}^{n_s}$: state of system at time t with state space $\mathcal{S}$ of dimension n_s.
- $u_t^{(r)} = (u_{t,1}^{(r)}, \dots, u_{t,n_a}^{(r)}) \in \mathcal{U} \subset \mathbb{R}^{n_a}$: control vector at time t with n_a number of actuations, at iteration r, where $\mathcal{U} \subset \mathbb{R}^{n_a}$ is the control signal space.
- $u_{0:T-1} \in \prod_1^T \mathcal{U} \subset \mathbb{R}^{T \times n_a}$: control signal (decision vector) corresponding to planning horizon T.
- $x_{0:T}$: state trajectory driven via control trajectory $u_{0:T-1}$ through system dynamic equation starting from state x_0.
- Particle i: We use $u_{0:T-1}^{(r,i)}$ to denote the ith control signal at iteration r, which we call the ith particle.
- $\mathbb{U}^{(r)}(u_{0:T-1})$: measure (a.k.a. distribution) of control signal $u_{0:T-1}$ at iteration r.
- $\bar{u}_{0:T-1}$: the location parameter of a distribution, i.e. mean for parametric distribution or center (e.g. target consensus point) of non-parametric distribution.
- $\rho > 0$: temperature parameter controlling the selectiveness of the softmax weighting scheme.

B. The big picture of global optimization

Trajectory optimization finds a control sequence $u_{0:T-1}$ that minimizes a cost function J given an initial state x_0:

$$\min_{u_{0:T-1}} \quad J(u_{0:T-1} \mid x_0), \tag{1}$$

where the cost is defined to be stagewise while satisfying the system's dynamics $x_{t+1} = dyn(x_t, u_t)$:

$$J(u_{0:T-1} \mid x_0) = \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \tag{2}$$

$$\text{s.t.} \quad x_0 = x_{init}, \;\; x_{t+1} = dyn(x_t, u_t), \tag{3}$$

where $\{\ell_t\}$ and $\ell_T(x_T)$ denote the running and terminal costs, respectively.

Achieving global optimality in trajectory optimization requires overcoming the limitations of traditional gradient-based optimization techniques that often get stuck in local minima of the objective function $J(u_{0:T-1}|x_0)$ [1]. One approach to mitigating this issue is to smooth the objective landscape by integrating the cost over a neighborhood of the current guess of the solution [25].

Formally, this smoothing can be defined given a probability distribution, $\mathbb{U}$, over the controls. More precisely, the smoothing surrogate of the objective function can be written as:

$$\xi(J \mid \mathbb{U}) = \int J(u_{0:T-1}|x_0) d\mathbb{U}(u_{0:T-1}). \tag{4}$$

Remark 1. *This smoothing can be interpreted as applying a low-pass filter to the cost landscape with a given bandwidth. A heavily smoothing filter may remove many local optima around the current solution and thus enable better solutions. However, it may also shift the global optimum of the surrogate landscape away from that of the original objective. Consequently, it is natural to begin the optimization with a strongly smoothing filter and gradually reduce its magnitude.*

Now, one may search for the probability distribution $\mathbb{U}$ that minimizes $\xi(J \mid \mathbb{U})$. This naturally leads to an iterative optimization procedure in the spirit of gradient descent.

$$\mathbb{U}^{(r+1)} = \mathbb{U}^{(r)} - \alpha \Delta \xi(J \mid \mathbb{U}^{(r)}), \tag{5}$$

where we use Δ to denote the variation operation to the distribution.

Remark 2. *Typically, the distribution of controls $\mathbb{U}$ is chosen to be a multivariate Gaussian distribution. This is often referred to as randomized smoothing [26]. In this setting, we can write $\mathbb{U}(\cdot \mid \bar{u}_{0:T-1}) = \mathcal{N}(\bar{u}_{0:T-1}, \Sigma)$, where $\bar{u}_{0:T-1}$ is the mean of the gaussian distribution and Σ is a fixed covariance matrix. In this case, the iterative procedure can be expressed as an iteration over the mean of the trajectory:*

$$\bar{u}_{0:T-1}^{(r+1)} = \bar{u}_{0:T-1}^{(r)} - \alpha \nabla \xi(J \mid \mathbb{U}(\cdot \mid \bar{u}_{0:T-1}^{(r)})) \tag{6}$$

Note that the gradient is taken with respect to $\bar{u}_{0:T-1}$. The control mean can then be viewed as an estimate for the solution of the original minimization problem defined in (1)*.*

Remark 3. *To enable global optimization, the following requirements need to be satisfied for representing $\mathbb{U}$:*

- *For high-dimensional decision variables, the probability distribution $\mathbb{U}$ should be able to concentrate on the "important" regions of the solution space with favorable objective values and facilitate finite sample size approximation.*
- *The probability distribution $\mathbb{U}$ should be able to adapt and shrink its support to enable a refined search for the global optima.*

C. Zero-order methods used in robotics

In this section, we cast the popular zero-order optimization methods used for trajectory optimization in robotics within the framework of (5). This perspective provides insight into their limitations and helps explain why our proposed method addresses them. We begin with the path integral framework.

1) Path integral methods: Path integral (PI) methods were originally introduced in [27] within the context of reinforcement learning. The method was derived from an information-theoretic perspective based on the Feynman-Kac lemma. Subsequently, a receding-horizon control variant, known as MPPI, was proposed in [28].

The original derivation of PI updates happen in the state space; here, we are mostly interested in the control space solution directly. One can show that the control update can be computed as an expectation over sampled control trajectories weighted by their costs. Given a current guess of the solution $\bar{u}_{0:T-1}^{(r)}$, the algorithm samples N random controls

$$u_{0:T-1}^{(r,i)} \sim \mathcal{N}(\bar{u}_{0:T-1}^{(r)}, \Sigma), \tag{7}$$

where i indexes the N independent samples, and Σ is a fixed covariance matrix. These random control inputs are then used to compute the next guess according to the following rule:

$$\bar{u}_{0:T-1}^{(r+1)} = \sum_{i=1}^{N} w^{(r,i)} u_{0:T-1}^{(r,i)}, \tag{8}$$

$$\text{where} \quad w^{(r,k)} = \frac{\exp(-\rho J(u_{0:T-1}^{(r,k)} \mid x_0))}{\sum_{j=1}^{N} \exp(-\rho J(u_{0:T-1}^{(r,j)} \mid x_0))}, \tag{9}$$

where $w^{(r,k)}$ is referred to as the normalized weight computed from the cost of the i^{th} trajectory.

The path integral update can be interpreted as a form of Gaussian smoothing over the objective function J [9]. To be more precise, one can define a surrogate of the form:

$$\xi(J \mid \mathbb{U}) = -\frac{1}{\rho} \log \left(\int e^{-\rho J(u_{0:T-1} \mid x_0)} d\mathbb{U}(u_{0:T-1}) \right). \tag{10}$$

Then, if we consider $\mathbb{U}(\cdot \mid \bar{u}_{0:T-1}) \sim \mathcal{N}(\bar{u}_{0:T-1}, \Sigma)$, the path integral updates in (8) can be recovered by applying (5), with the gradient taken with respect to the mean $\bar{u}_{0:T-1}$. The following proposition formalizes this result.

Proposition 1. *The update for the mean of distribution $\bar{u}_{0:T-1}$ under the PI framework in* (8) *can be written as*

$$\bar{u}_{0:T-1}^{(r+1)} = \bar{u}_{0:T-1}^{(r)} - \gamma \Delta(\bar{u}_{0:T-1}), \tag{11}$$

where $\Delta(\bar{u}_{0:T-1})$ is the solution to the following constrained optimization problem with γ being the Lagrange multiplier.

$$\arg \min_{\Delta(\bar{u}_{0:T-1})} \xi(J \mid \mathbb{U}(\cdot \mid \bar{u}_{0:T-1} + \Delta(\bar{u}_{0:T-1}))) \tag{12}$$

$$s.t.\ KL\Big(\mathbb{U}(u \mid \bar{u}_{0:T-1} + \Delta(\bar{u}_{0:T-1})) | \mathbb{U}(u \mid \bar{u}_{0:T-1})\Big) = \beta, \tag{13}$$

where β defines the Kullback–Leibler (KL) divergence metric between the current distribution and new distribution.

We invite readers who are interested in the details to find proof in Appendix A in the supplementary material.

Remark 4. *The Lagrange multiplier $\gamma = \frac{d\xi(J|\mathbb{U};\bar{u}_{0:T-1}+\Delta;\beta)}{d\beta}$ decides how much the optimal surrogate function will change per β change. In practice, this constrained optimization problem is not solved explicitly; instead, the update in* (8) *is applied directly, in which case the parameter β could vary per iterations.*

Remark 5 (Particle dynamic under parameterized distribution updates)**.** *Consider two arbitrarily chosen particles (random control sequences) from iterations r and $r+1$, the difference between them can be written as*

$$\begin{aligned} u_{0:T-1}^{(r,i)} - u_{0:T-1}^{(r+1,j)} &= \bar{u}_{0:T-1}^{(r)} + \Sigma^{\frac{1}{2}} \epsilon^{(r,i)} - \bar{u}_{0:T-1}^{(r+1)} - \Sigma^{\frac{1}{2}} \epsilon^{(r+1,j)} \\ &= \Sigma^{\frac{1}{2}} (\epsilon^{(r,i)} - \epsilon^{(r+1,j)}) + \frac{1}{\gamma} \Delta(\bar{u}_{0:T-1}), \end{aligned} \tag{14}$$

where $\{\epsilon^{(r,i)}\}$ are i.i.d. standard normal variables. Thus, the potential improvements of $u^{(r+1,i)}$ over $u^{(r,j)}$ come from the major effect of $\Delta(\bar{u}_{0:T-1})$ plus a random exploration term $\Sigma^{\frac{1}{2}}(\epsilon^{(r,i)} - \epsilon^{(r+1,j)})$. Since the particles are reset after each iteration, there is no distinction between them (particles are "forgotten" after each iteration), resulting in homogeneous exploration. Any potential improvements beyond $\Delta(\bar{u}_{0:T-1})$ comes from repeatedly exploring the same mechanism of random directions.

Remark 6. *Note that when ρ is big enough in* (10)*:*

$$\xi(J \mid \mathbb{U}(\cdot \mid \bar{u}_{0:T-1})) \approx \inf J(u_{0:T-1} \mid x_0). \tag{15}$$

Thus, optimizing the surrogate is approximately equivalent to optimizing the lower bound, while the constraint penalizes large changes by restricting how much the distribution can shift at each iteration. With a finite sample size, this involves choosing the best performing $u_{0:T-1}$ among the sampled population.

Remark 7 (Curse of dimensionality in finite sample approximation of Gaussian expectation)**.** *When the decision variable lies in an extremely high-dimensional space, a finite sample size cannot realistically approximate the distribution; thus, the lower bound of a finite sample does not approximate the mathematical formulation of* (10) *faithfully.*

2) Covariance matrix adaptation: One way to mitigate the issue highlighted in Remark 7 is to adapt the covariance matrix of the sampling distribution. This allows the sampled population to concentrate along promising directions around the current solution while reducing exploration in less useful directions. This idea is known as Covariance Matrix Adaptation (CMA), and CMA-ES builds on it with a widely used evolution-path mechanism [13].

For clarity, we use a simplified presentation of CMA-ES following the viewpoint of [9]; the complete algorithm, including the full evolution-path machinery, is treated in [13]. Specifically, CMA-ES maintains a Gaussian search distribution over the control sequences and adapts its mean, covariance, and step

size based on ranked samples [13]. At iteration r, offspring are sampled according to (16), then evaluated through the cost function J:

$$u_{0:T-1}^{(r,i)} = \bar{u}_{0:T-1}^{(r)} + \alpha^{(r)} A^{(r)} \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, I), \tag{16}$$

where A is the decomposition of the covariance matrix Σ as stated below:

$$\Sigma^{(r)} = (\alpha^{(r)})^2 A^{(r)} (A^{(r)})^\top. \tag{17}$$

The mean is then updated only using an elite proportion of the current population and a moving average trick to stabilize the updates, which we formulate in (18). The covariance is adapted to capture successful search directions as (19), using $y^{(r,i)}$ in (20), which is the deviation of particle i from the current population mean.

$$\bar{u}_{0:T-1}^{(r+1)} = (1-\alpha^{(r)})\bar{u}_{0:T-1}^{(r)} + \alpha^{(r)} \sum_{i=1}^{N_e} w^{(r,i)}\, u_{0:T-1}^{(r,i)} \tag{18}$$

$$\Sigma^{(r+1)} = (1-\alpha^{(r)})\Sigma^{(r)} + \alpha^{(r)} \sum_{i=1}^{N_e} w^{(r,i)}\, y^{(r,i)} y^{(r,i)\top} \tag{19}$$

$$y^{(r,i)} = u_{0:T-1}^{(r,i)} - \bar{u}_{0:T-1}^{(r)}, \tag{20}$$

where N_e is the number of elite particles. One choice of pertaining $w^{(r,i)}$, the weight for particle i, can be done via calculating (9). The step size $\alpha^{(r)}$ is adjusted by a separate evolution path to control the overall search scale. This mechanism allows CMA-ES to adaptively stretch or shrink the sampling distribution along promising directions.

As pointed out by [29, 18], the update rule of CMA-ES (without the evolution path [13]) can be recovered by applying (5) with the gradient defined with respect to both the mean and covariance parameters.

Remark 8 (Connection between CMA and CEM)**.** *Setting $w^{(r,i)} = \frac{1}{N_e}$ with elite particles $N_e < N$ in* (19) *gives the Cross Entropy Method (CEM) [12].*

Remark 9 (curse of distribution parametrization)**.** *When a parameterized distribution is employed, its shape can be adjusted only through a limited number of parameters, which constrains the expressive flexibility of $\mathbb{U}(u_{0:T-1})$. The Gaussian distribution has additional innate geometric structures, such as symmetry, which typically do not fit control signal distributions. For instance, flipping the control signal with respect to an arbitrary center might result in a control signal that leads to undesirable behavior. In Figure 1, we illustrate the difference between a Gaussian distribution and an irregular, non-parametric distribution. The irregular distribution can develop longer, potentially asymmetric tails along important directions, whereas a shrinking Gaussian is much less flexible in accommodating such behavior. In Section III, we introduce CBO and explain how an irregular, non-parameteric distribution can be achieved, which also has the ability to concentrate on important regions.*

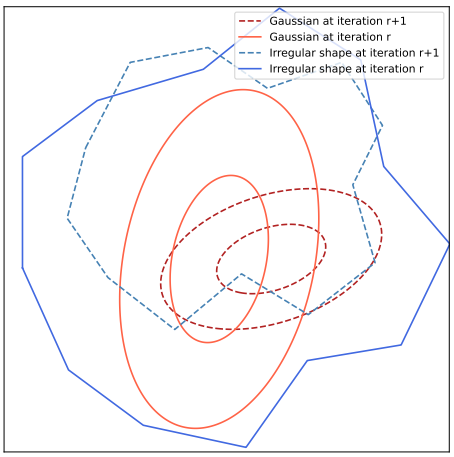


Fig. 1: An irregular, non-parametric distribution can focus probability on important directions by having longer tails, whereas a shrinking Gaussian is constrained by its fixed symmetric shape and has less flexibility.

III. Consensus based optimization (CBO)

To address the challenges of parameterized sampling-based optimization discussed in Section II, we introduce consensus-based optimization (CBO) to the robotics community. Instead of explicitly parameterizing the distribution $\mathbb{U}^{(r)}$ at iteration r, CBO uses a population of particles $\{u_{0:T}^{(r,i)}\}$ to approximate the distribution $\mathbb{U}^{(r)}$, where i indexes the particles of the population. Under mild assumptions, CBO is proven to converge to the global optimum [20, 21].

A. Basics of CBO

In CBO, each particle follows the following stochastic differential equation (SDE):

$$du_{0:T}^{(r,i)} = -\lambda \left(u_{0:T}^{(r,i)} - \bar{u}_{0:T}^{(r)}\right) dr + \sigma \left|u_{0:T}^{(r,i)} - \bar{u}_{0:T}^{(r)}\right| dW^{(r,i)}, \tag{21}$$

where r denotes the iteration index, λ is the decay rate governing how fast particle i drifts toward the consensus point $\bar{u}_{0:T}^{(r)}$ with drift rate $-\lambda(u_{0:T}^{(r,i)} - \bar{u}_{0:T}^{(r)})$. The consensus point is defined by

$$\bar{u}_{0:T}^{(r)} = \sum_i w^{(r,i)} u_{0:T}^{(r,i)}, \tag{22}$$

where $w^{(r,i)}$ denotes the weight of particle i at iteration r defined as in (9). $\sigma \in \mathbb{R}$ in (21) controls the intensity of exploration via Brownian motion $W^{(r,i)}$. In other words, larger σ leads to more exploration. In discrete time iteration, (21) becomes

$$\begin{aligned}\Delta u^{(r,i)} = u_{0:T-1}^{(r+1,i)} - u_{0:T-1}^{(r,i)} = -\lambda \left(u_{0:T-1}^{(r,i)} - \bar{u}_{0:T}^{(r)}\right) \Delta r \\ + \sigma\sqrt{\Delta r} \left|u_{0:T-1}^{(r,i)} - \bar{u}_{0:T}^{(r)}\right| \Delta W^{(r,i)}\end{aligned} \tag{23}$$

where $\Delta W^{(r,i)}$ is sampled from standard normal distribution $\mathcal{N}(0, I)$. Algorithm 1 summarizes the steps of CBO, where the particle-wise for loops can be parallelized on a GPU. This

parallel structure keeps CBO computationally fast in practice: in the experiment section, we demonstrate that empirically its per-iteration runtime is comparable to CMA-ES and MPPI, as shown in Table I in Section IV-C.

Algorithm 1: CBO for trajectory optimization

Input: Initial state $x_0 \in \mathcal{S}$, SDE integration time Δr, initial population $\{u^{(0,i)}_{0:T-1}\}$ of size N. σ, λ

while *stopping criterion is not met,* **do**
  for $i = 1, \ldots, N$ **do**
    Evaluate J value for particle i
  Calculate target consensus point $\bar{u}_{0:T}$ with (22)
  for $i = 1, \ldots, N$ **do**
    Calculate drift and exploration term in (23) to update particle

Output: final population $\{u^{(r^*,i)}_{0:T-1}\}$ at iteration r^*

Remark 10. *In Remark 5, we derived the particle dynamic for the PI update that reads as* $u^{(r,i)} - u^{(r+1,j)} = \Sigma^{\frac{1}{2}}(\epsilon^{(r,i)} - \epsilon^{(r+1,j)}) + \frac{1}{\gamma}\Delta(\bar{u}_{0:T-1})$ *and discussed the homogeneous drift behavior and exploration. In contrast, CBO dynamic in* (21) *indicates that different particles exhibit different behaviors depending on how far away the particle is from the target consensus point. The exploration term is also particle dependent, again depending on how far away the particle is from the target consensus point; particles farther away explore more.*

Remark 11 (λ and Δr)**.** *In* (23)*,* Δr *corresponds to Euler-Maruyama integration time for the stochastic differential equations. Numerically,* Δr *can be absorbed into* λ *as a lumped up parameter. When* $\lambda \Delta r = 1$*,* $\sigma = \sqrt{\Delta r}$*, the CBO dynamic in* (23) *reduces to*

$$u^{(r+1,i)}_{0:T-1} = \bar{u}^{(r)}_{0:T} + \sigma^2 \left| u^{(r,i)}_{0:T-1} - \bar{u}^{(r)}_{0:T} \right| \Delta W^{(r,i)}, \tag{24}$$

which indicates that at each iteration, each particle is resampled around the target consensus point with a variance proportional to its distance to the consensus point. This corresponds to a special case of CBO called consensus hopping [30].

Note that the specific type of consensus hopping we defined in (24) *is different from the sample generation in the PI update in* (7)*, where each particle at the current generation (iteration) is resampled around the target consensus point with a fixed covariance* Σ*, independent of the particle's distance to the consensus point.*

B. Advantages of CBO in global optimization

We use the example of Figure 2 as an illustration of the advantages of CBO where we constructed a cost function landscape in a contour plot with a unique global optimizer (marked as blue star) and several local optimizers (marked as orange disks). As a **hypothetical setting** for the purpose of illustration only, we assume at the current iteration, the target consensus is marked with a red cross in the figure. The following key observations explain the advantage of CBO over other sampling-based methods.

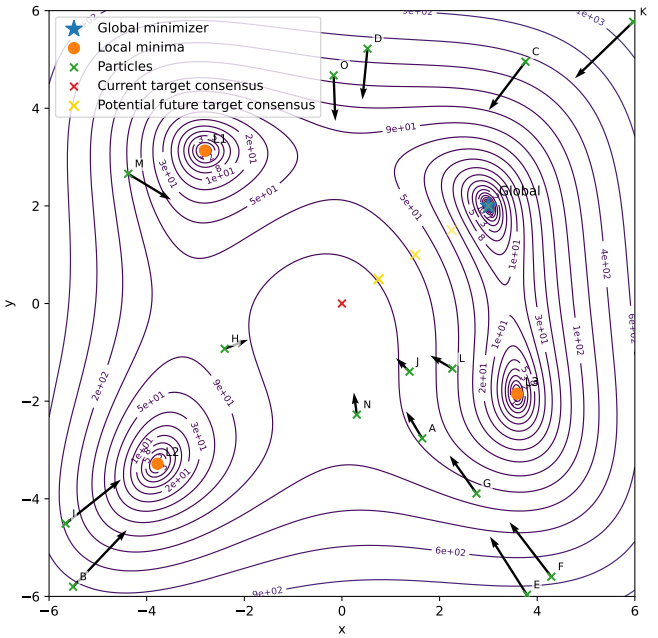


Fig. 2: A cost function landscape in a contour plot constructed with a unique global optimizer (marked star) and several local optimizers for illustration of how CBO works and its advantages. See text for more explanation.

- The dynamic of each particle is not directly driven by the gradient of the objective function, but by a drift term that "drags" or "attracts" the particle toward the target consensus point at the current iteration in (21). As a result, the particle dynamics are agnostic to local optima of the objective function. For instance, in the illustrative example of Figure 2, particle B at the lower-left corner can travel through the local minimum near $(-4, -3)$ since its dynamic aims to drive it toward the goal position of the target consensus point, instead of the local minimum.
- The inertia of each particle enables local exploration around the particle position from the previous iteration. In contrast, sampling-based methods discard particle histories (see (14)). For instance, in the illustrative Figure 2, the drift vectors (arrows in the plot) for particle K, C, O, D determines a dominating direction towards the current consensus point, however, the exploration term in Equation (21) adds noise on top, **proportional to the distance to the current consensus point** (Particles sitting on the consensus point has no exploration, a necessary condition for convergence. Particles far way get explored more to search for better solution), avoiding a straight-line behavior and allowing the particles to explore their neighborhood. This gives them a good chance to approach the vicinity of the global optimizer.
- As particles move toward the target consensus point, each particle explores along different directions. Along

these drifting paths, once a better solution is found, the target consensus point shifts toward that solution, and the drift terms of all particles change accordingly, inducing a new set of exploration directions. This results in a more diverse exploration mechanism than the homogeneous exploration used in the PI updates (see (14)). For instance, as particle K and C in Figure 2 being pulled toward the current target consensus point $(0, 0)$, they have a good chance to find better position with lower cost than the current consensus point since they are near the global optimizer.

- The non-parametric nature of CBO allows it to adapt to complex cost landscapes more effectively than parameterized distributions, which may struggle to capture intricate solution spaces. The particles gradually gather in the vicinity of the target consensus point, analogous to the covariance matrix shrinkage in CMA. Unlike CMA, however, the empirical distribution of CBO is not Gaussian-shaped, but instead exhibits an irregular shape as illustrated in Figure 1.

Remark 12 (How CBO alleviates the curse of dimensionality and the curse of distribution parameterization)**.** *In Remark 7, we pointed out that when the decision variables have high dimensions, faithfully approximating the cost surrogate in* (10) *becomes unrealistic. CMA tries to resolve this issue by adapting its covariance matrix to let finite samples better concentrate on favorable directions, with potential shrunken eigenvalues, but is still restricted by the parameterized form of the as we noted in Remark 9. Instead, the population for CBO is evolving via the dynamic* (21)*, which offers a more flexible adaptive capacity to capture interesting regions of the decision variable space. In Section III-C, we demonstrate such an irregular, non-parametric distribution of solutions from CBO is effective in achieving global optimality.*

C. Properties of CBO

First we show CBO falls into the global optimization framework of (5) with the surrogate function in (4) under the assumption that $J(u \mid x_0)$ satisfies the so-called inverse continuity property and relaxed Lipschitz continuous property, which we state below:

$$\frac{1}{L(u)}\|J(u \mid x_0) - J(u^* \mid x_0)\| \leq \|u - u^*\| \tag{25}$$

$$\leq \frac{1}{\eta}\|J(u \mid x_0) - J(u^* \mid x_0)\| \tag{26}$$

$$\forall u \in B_\kappa(u^*) \tag{27}$$

where $B_\kappa(u^*)$ is a neighborhood of the optima u^* with radius κ, and $\eta > 0$ is a constant.

Formally, to demonstrate the irregular, non-parametric distribution of particles of CBO is effective in achieving global optimality, we have the following results:

Proposition 2. *Without loss of generality, assume* $J(u^* \mid x_0) = 0$*. When Equations* (25) *to* (27) *hold, there exist* $\rho, r^* > 0$ *large enough, where*

$$r^* \propto \frac{1}{2\lambda - n_a \times T\sigma^2}, \tag{28}$$

such that, with high probability, CBO improves the surrogate function following the formulation in (5) *after at most* r^* *iterations, i.e., we have*

$$\int J(u_{0:T-1}|x_0) d\mathbb{U}^{(r^*)}(u_{0:T-1} \mid \bar{u}^{(r^*)}_{0:T-1}) < \int J(u_{0:T-1}|x_0) d\mathbb{U}^{(0)}(u_{0:T-1} \mid \bar{u}^{(0)}_{0:T-1}). \tag{29}$$

In addition, due to the structure of $\mathbb{U}$ *evolution driven by the CBO dynamic in* (21)*, the target consensus point converges to the global optimizer with high probability.*

Proof is in Appendix B of the supplementary material.

Remark 13 (choice of decay rate λ)**.** *Equation* (28) *gives us a convenient reference of selection of decay rate* λ *with respect to the noise level* σ^2 *to ensure convergence of the dynamics of* (23)*. Concretely, the convergence can be ensured by* $2\lambda > n_a \times T \times \sigma^2$*. For hyperparameter tuning, we recommend simply calculate the lowest* λ *that satisfies this formula and then try a grid of values, while ensuring* $\lambda\Delta r < 1$*. Increasing* λ *improves convergence rate but too large* λ *breaks the performance, see an empirical comparision of different* λ *in Figure 9 in the appendix to understand this tradeoff.*

IV. Experiment

In this section, we show how the CBO algorithm can enable finding better solutions on three challenging robotics problems, highlighting different challenges in trajectory optimization for robotics. In what follows, first, we explain the general setting of our experiments and then present the results.

A. Experiment setting

To ensure fair comparisons, we use the same environment and the same initial particle population for all methods compared within each experiment whenever possible. In addition, we align shared hyperparameters across algorithms, for example, by using the same temperature ρ and matching the σ parameter of CBO with the exploration terms of the other methods. We choose λ in (21) according to Remark 13.

For Section IV-C and Section IV-D, our implementation is based on *Hydrax* [31] and we use the implementation of CMA-ES from *Evosax* [32]. Our implementation of the algorithms and experiments can be found in https://github.com/Atarilab/cbo_to.

B. Long horizon planning

In this example, we study a trajectory optimization problem involving a simplified dynamical system with a low-dimensional state space but an extremely long planning horizon. As illustrated in Figure 4, the agent is modeled as a point mass but represented by a chassis for better visualization (shown as a small blue rectangle) that travels from an initial position (black square located at the center left) towards a

goal position (marked by a small red disk) through a confined space (depicted as a large green square). The tunnel has an upward-facing opening and is bounded by a wall on the left side (shown as black slats) that extends upward, as well as a shorter wall on the right side. We deliberately design the left wall to extend upward so that reaching it corresponds to a local minimum, since the agent cannot penetrate the wall. In many robotics control problems, feasible control signals lie in very "narrow" regions. To emulate this phenomenon, we randomly place obstacles (shown as dark green disks) such that only carefully chosen control signals allow the agent to successfully navigate through the obstacles and reach the tunnel.

We use the following simple first order dynamic model without inertia (i.e., velocity can be changed instantaneously):

$$q_x(t+1) = q_x(t) + v\cos(\theta)\delta t, \tag{30}$$
$$q_y(t+1) = q_y(t) + v\sin(\theta)\delta t. \tag{31}$$

To avoid optimization in the radian space, instead of optimizing the sequence of $\theta \in [0, 2\pi], v \in \mathbb{R}$, we use $v_x = v\cos(\theta), v_y = v\sin(\theta)$ as the decision variable. The decision making horizon is set to $T = 100$ steps, and the control dimension is $n_a = 2$. Despite the simplicity of the dynamic, the number of decision variables is $n_a \times T = 200$, making it a challenging long horizon planning problem.

The cost function is composed of the following terms: the primary term is the running cost, which is the distance to the goal (red disk) at each step. On top of that, we add the control loss term, plus a penalty term for not reaching the tunnel and further penalties when confronting the obstacles along the planned trajectories. In summary, the cost function is defined as:

$$J = \sum_{t=0}^{T-1} \left(\|q(t) - q_{goal}\|^2 + \gamma_v \|v(t)\|^2\right) + I_{\text{not in tunnel}}\left(q(T-1)\right) + \sum_{i=1}^{N_{obs}} I_{\text{obstacle}}(q_{0:T-1}), \tag{32}$$

where $I_{\text{not in tunnel}}$ is an indicator function that adds a large penalty if the final position is not inside the tunnel, and I_{obstacle} is an indicator function that adds a penalty if the trajectory confronts any obstacle. We set $\gamma_v = 10.0$, $I_{\text{not in tunnel}}\left(q(T-1)\right) = 1000$ if the agent is not inside the tunnel at the last step, $I_{obstacle} = 1000 \times n_c$, where n_c is the number of obstacles confronted along the trajectory. The parameters for penalty terms are chosen to ensure that circumventing the left wall and the obstacles is more cost-effective than confronting them directly, such that the optimal solution involves navigating through the tunnel and reaching the goal. Note that the velocity is not bounded but only regularized in the cost function with γ_v.

In terms of collision handling, the left wall acts as a barrier for the agent to reach the tunnel and goal (red disk). To simplify, we restrain the agent to bounce back to its original place if the current velocity would bring the agent into contact with the wall. On the other hand, when confronting the disk barrier, we simply add a penalty to the cost function but do not restrain the dynamic accordingly (i.e., the agent can penetrate the disk obstacle but incurs a large cost). The agent needs to make long term planning to avoid getting stuck in local minima, especially when the obstacles are placed closely.

For each repetition of the experiment, we create the same environment for competing algorithms by randomly sampling obstacles. We compare CBO with MPPI and CMA, where we implemented CMA according to (18) and (19). We set population size $N = 1000$. The covariance matrix for baselines is set to have initial diagonal value matching $\sigma = 10$ for CBO, while we use an exponential decay of the σ for CBO ensures the convergence behavior due to Remark 13.

The benchmark results are summarized in Figure 3. CBO outperforms CMA and MPPI by a large margin. In Figure 4, we visualize the trajectories generated by CBO, MPPI, and CMA, respectively. It can be observed that CBO is able to successfully navigate through the tunnel, avoiding all obstacles (in terms of obstacle avoidance, for simplicity, we consider the agent as a point mass), and reaches the tunnel, whereas MPPI and CMA get stuck in their local minima and fail to reach the tunnel.

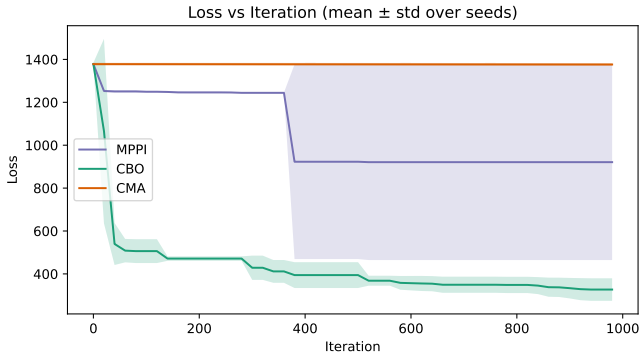
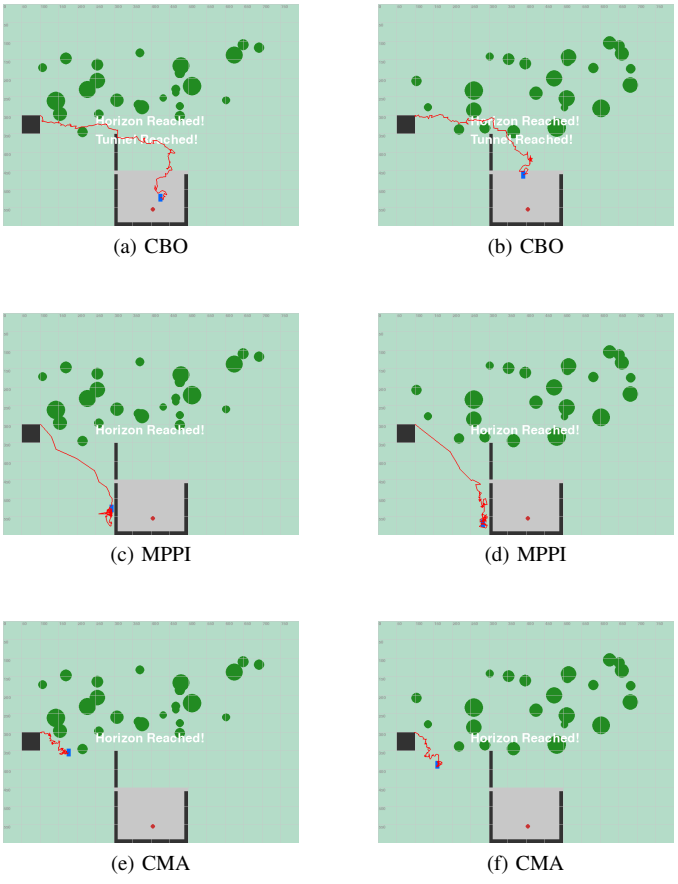


Fig. 3: Long horizon planning: comparison of the best loss across the population at each iteration for the long horizon planning problem. CBO wins by a large margin. CMA implemented according to (18) and (19).

Remark 14 (Why none of the algorithms reach the goal)**.** *This scenario is extremely difficult because of the problem's long-horizon nature and the extreme clutter in the scene. In this task, although CBO performs significantly better than other methods, it still fails to reach the exact goal indicated by the red point inside the tunnel. The performance of CBO can be further improved by increasing the number of particles. We refer the reader to Theorem 3.8 and Remark 3.9 in [21], where, to each optimization problem, there is an associated rate of convergence $C_{MFA}N^{-1}$. C_{MFA} characterizes the complexitiy of a problem. The corresponding C_{MFA} for this task is very large, implying the need of a larger number N of particles.*

C. Double Cartpole

The double cartpole (see an illustration in Figure 5) has the cart that is allowed to move within the range $[-3.8m, +3.8m]$



(a) CBO (b) CBO

(c) MPPI (d) MPPI

(e) CMA (f) CMA

Fig. 4: MPPI and CMA get stuck at a local minimum and fail to reach the tunnel. CBO succeeds in circumventing the left wall barriers and all obstacles (point mass agent) and reaches the tunnel. Each image shows the trajectory of a single experiment for different methods. Each column corresponds to one environment setting (spatial distribution of obstacles).

and the pole with the cart forming a revolute joint and another pole connected to the base pole via a revolute joint. The generalized coordinates are $\begin{bmatrix} q & \theta_1 & \theta_2 \end{bmatrix}^T$, where q is the cart position along the rail, θ_i with $i = 1, 2$ are the pole angles.

The initial state is

$$x_0 = \begin{bmatrix} q_0 \\ \dot{q}_0 \end{bmatrix} = \begin{bmatrix} q & \theta_1 & \theta_2 & \dot{q} & \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix}^T_{t=0} = 0, \qquad (33)$$

where $\theta_i = 0$ corresponds to the pole hanging downward, and the upright configuration is $\theta_i = \pi$. Slider joint (cart) damping is set to $d_x = 10^{-4}$. To make the problem more challenging, we set friction loss to 0 for both revolute joints. The running cost is set to be the bound violation cost of control according to [33] and the terminal cost is calculated as the summation of all the following terms with equal weighting:

- distance of the tip site from its upright target: $\|\texttt{tip}_z - 4\|^2$, where $4\,\mathrm{m}$ is the upright tip height: the cart joint is located at height $2\,\mathrm{m}$ in the MuJoCo scene and the two $1\,\mathrm{m}$ pole segments add another $2\,\mathrm{m}$ when fully upright.

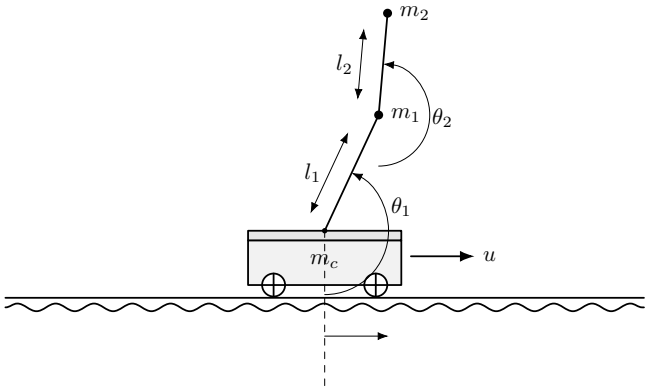


Fig. 5: Double cartpole with cart mass $m_c = 1.0$ kg. Pole masses: $m_1 = m_2 = 0.1$ kg. Each pole segment has length $l_1 = l_2 = 1.0$m, giving a total pole-chain length of 2.0m. The cart sits at 2m high in the world frame.

- $\|\texttt{tip}_x - q\|^2$: pole tip aligned directly above the cart, not leaning left or right.
- the distance of the cart to the center of the rail.
- the velocity of all degrees of freedom.

The control input is the actuator command u applied to the cart, with actuator force $F = 20u$, where $u \in [-u_{\max}, u_{\max}]$. With $u_{\max} = 0.5$, the nominal force range is $[-10, 10]$ N. To make the problem challenging, we deliberately set $u_{\max}$ as small as 0.5 to increase the control difficulty, while we use a long planning horizon of $16\,\mathrm{s}$ discretized by 100 zero-order-hold control knots. Population size $N = 5000$.

Note that we do not aim for a perfect swing-up solution here; rather, the goal is to demonstrate that under such a challenging setting, CBO exhibits clear advantages over competing methods in terms of convergence and consistency. In Figure 6, we present benchmark results of loss minimization among CBO, MPPI and CMA-ES (code from [32]) where CBO performs the best. CMA-ES experiences large fluctuations of cost during the iteration, while CBO consistently generates good performance.

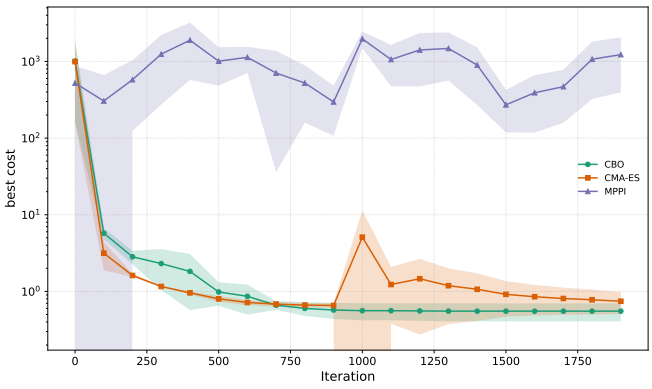


Fig. 6: Benchmark double-cartpole: On average, CBO performs the best with relatively small variance. CMA-ES is from implementation [32].

Remark 15 (CBO can be as fast as other zero-order methods)**.** *The bottleneck of the computation time for zero-order methods is the simulation rollout time. Thus the double cartpole experiment, due to its relative simplicity to be simulated, is a good example to compare the running time of the different algorithms. The statistics in Table I are generated on a NVIDIA GeForce RTX 5090 Laptop GPU with a 32 × AMD Ryzen 9-9955HX3D 16-Core processor running a Linux OS (6.17.0-109014-Tuxedo). Here we use only $N = 1000$ particles. We report JAX compilation time and per-iteration computation time, excluding auxiliary code overhead. The results show no significant difference between the tested zero-order methods in per-iteration computation time.*

TABLE I: Runtime comparison across algorithms.

Algorithm	Compile (ms)	Mean (ms)	Std (ms)
CBO	6638.30	336.43	1.02
CMA-ES	11369.38	351.28	12.53
MPPI	6548.47	372.48	35.96

D. Humanoid

Our last experiment is the 23 DoF Unitree G1 Humanoid, given $T = 6$ zero-order hold knots as PD target input q_{des} in (34) with the following parameters:

$$\tau^* = k_p\,(q_{\text{des}} - q) - k_d\,\dot{q}, \tag{34}$$

$$k_p = 500, \qquad k_d = 2\,\zeta\,\sqrt{k_p\,I_{\text{eff}}}, \qquad \zeta = 1, \tag{35}$$

where I_{eff} is the effective inertia of the actuator at neutral position. The decision variable has dimension $T \times n_a = 6 \times 23$. We aim to achieve **demonstration free** locomotion within 2 seconds: Starting from a standing position, we only provide the terminal base configuration as the goal, without any running cost. We use the $se(3)$ difference between the starting base configuration and the goal base configuration at the last step as the cost function.

We present the benchmark results in Figure 7 by comparing the best loss among the population of each iteration. This shows that CBO significantly outperforms MPPI and CMA-ES (code from [32]). Interestingly, CBO has a large variance. Some runs are able to generate much lower cost compared to others, though the upper boundary of CBO loss variance still overrates the lower cost variance boundary of the baselines.

In Figure 8, we show snapshots of the starting position, middle position, and end position of the humanoid from a trajectory optimized by CBO.

V. Conclusion and Future Work

We introduced CBO to the robotics community as a step toward global trajectory optimization. By presenting a general mathematical framework for global optimization, we analyzed the limitations of widely used zero-order optimization methods in robotics and explained how CBO addresses these shortcomings. Through three challenging trajectory optimization problems, we demonstrated that CBO can find solutions with significantly lower cost for the tasks considered. Future work includes exploring the use of CBO for discovering multi-modal behaviors, incorporating constraints, and conducting real-world robotic experiments.

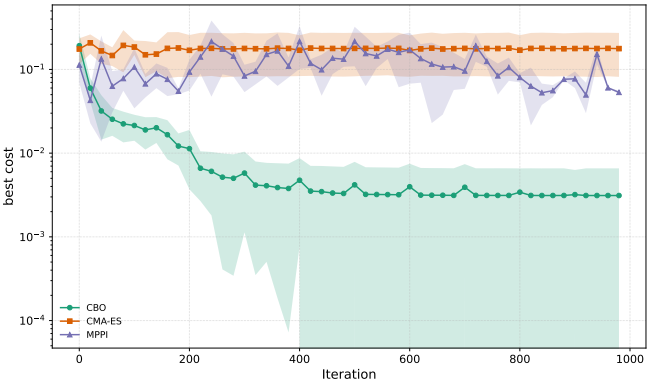


Fig. 7: Comparison of the best loss among the population of each iteration for the humanoid demonstration free locomotion experiment. Population size $N = 10,000$, CBO wins by large margin. CMA-ES is from implementation [32].

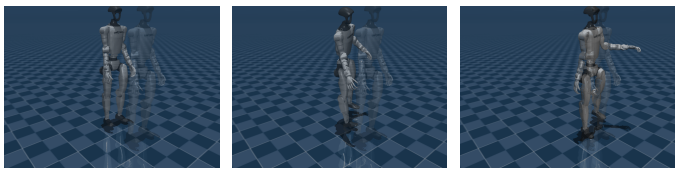


Fig. 8: CBO optimized trajectory: Three snapshots (start, middle, end) of the G1 humanoid trajectory in the free locomotion task for 2 seconds. The shadowed silhouette indicates the target position for the humanoid.

VI. Acknowledgments

Xudong Sun and Majid Khadiv acknowledge the support of the Huawei-TUM joint laboratory. Massimo Fornasier acknowledges the support of the Munich Center for Machine Learning and the ERC Advanced Grant NEITALG, grant agreement No. 101198055.

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. This project has received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement No. 101198055, project acronym NEITALG).

References

[1] Patrick M Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard, and Andrea Del Prete. Optimization-based control for dynamic legged robots. *IEEE Transactions on Robotics*, 40:43–63, 2023.

[2] Sehoon Ha, Joonho Lee, Michiel van de Panne, Zhaoming Xie, Wenhao Yu, and Majid Khadiv. Learning-based legged locomotion: State of the art and future perspectives. *The International Journal of Robotics Research*, 44(8):1396–1427, 2025.

[3] Arthur Allshire, Hongsuk Choi, Junyi Zhang, David McAllister, Anthony Zhang, Chung Min Kim, Trevor Darrell, Pieter Abbeel, Jitendra Malik, and Angjoo Kanazawa. Visual imitation enables contextual humanoid control. *arXiv preprint arXiv:2505.03729*, 2025.

[4] Lujie Yang, Xiaoyu Huang, Zhen Wu, Angjoo Kanazawa, Pieter Abbeel, Carmelo Sferrazza, C Karen Liu, Rocky Duan, and Guanya Shi. Omniretarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction. *arXiv preprint arXiv:2509.26633*, 2025.

[5] 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.

[6] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[7] MJX. Mujoco3, 2023.

[8] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

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

[10] Chaoyi Pan, Zeji Yi, Guanya Shi, and Guannan Qu. Sampling-based methods for optimal control: Theory, algorithms, and applications. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, pages 3775–3793. IEEE, 2025.

[11] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[12] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2004.

[13] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[14] 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)*, pages 4974–4981. IEEE, 2025.

[15] Hossein Keshavarz, Alejandro Ramirez-Serrano, and Majid Khadiv. Control of legged robots using model predictive optimized path integral. In *2025 IEEE-RAS 24th International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2025.

[16] Corrado Pezzato, Chadi Salmi, Elia Trevisan, Max Spahn, Javier Alonso-Mora, and Carlos Hernández Corbato. Sampling-based model predictive control leveraging parallelizable physics simulations. *IEEE Robotics and Automation Letters*, 2025.

[17] Pietro Noah Crestaz, Ludovic De Matteis, Elliot Chane-Sane, Nicolas Mansard, and Andrea Del Prete. Td-cd-mppi: Temporal-difference constraint-discounted model predictive path integral control. *IEEE Robotics and Automation Letters*, 11(1):498–505, 2025.

[18] Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *Journal of Machine Learning Research*, 18(18):1–65, 2017.

[19] René Pinnau, Claudia Totzeck, Oliver Tse, and Stephan Martin. A consensus-based model for global optimization and its mean-field limit. *Mathematical Models and Methods in Applied Sciences*, 27(01):183–204, 2017.

[20] José A Carrillo, Young-Pil Choi, Claudia Totzeck, and Oliver Tse. An analytical framework for consensus-based global optimization method. *Mathematical Models and Methods in Applied Sciences*, 28(06):1037–1066, 2018.

[21] Massimo Fornasier, Timo Klock, and Konstantin Riedl. Consensus-based optimization methods converge globally. *SIAM Journal on Optimization*, 34(3):2973–3004, 2024.

[22] Xin-She Yang. Momin Jamil. A literature survey of benchmark functions for global optimization problems. *Int. Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

[23] Massimo Fornasier, Hui Huang, Jona Klemenc, and Greta Malaspina. From consensus-based optimization to evolution strategies: Proof of global convergence. *arXiv:2602.11677*, 2026.

[24] Yuval Davidor. *Genetic Algorithms and Robotics: A heuristic strategy for optimization*, volume 1. World Scientific Publishing Company, 1991.

[25] Sebastien Labbe and Andrea Del Prete. Analytical integral global optimization. In *Proceedings of the 7th Annual Learning for Dynamics & Control Conference*, volume 283 of *Proceedings of Machine Learning*

Research, pages 711–722. PMLR, 04–06 Jun 2025.

[26] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, April 2017.

[27] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.

[28] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[29] Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. Bidirectional relation between cma evolution strategies and natural evolution strategies. In *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I 11*, pages 154–163. Springer, 2010.

[30] Konstantin Riedl, Timo Klock, Carina Geldhauser, and Massimo Fornasier. Gradient is all you need? *arXiv preprint arXiv:2306.09778*, 2023.

[31] Vince Kurtz. Hydrax: Sampling-based model predictive control on gpu with jax and mujoco mjx, 2024. https://github.com/vincekurtz/hydrax.

[32] Robert Tjarko Lange. evosax: Jax-based evolution strategies. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 659–662, 2023.

[33] Takato Miura, Naoki Akai, Kohei Honda, and Susumu Hara. Spline-interpolated model predictive path integral control with stein variational inference for reactive navigation. (arXiv:2404.10395), April 2024. arXiv:2404.10395 [cs].