# Preference-Based Learning for User-Guided HZD Gait Generation on Bipedal Walking Robots

Maegan Tucker[1], Noel Csomay-Shanklin[2], Wen-Loong Ma[1], and Aaron D. Ames[1,2]

***Abstract*— This paper presents a framework that unifies control theory and machine learning in the setting of bipedal locomotion. Traditionally, gaits are generated through trajectory optimization methods and then realized experimentally — a process that often requires extensive tuning due to differences between the models and hardware. In this work, the process of gait realization via hybrid zero dynamics (HZD) based optimization problems is formally combined with preference-based learning to systematically realize dynamically stable walking. Importantly, this learning approach does not require a carefully constructed reward function, but instead utilizes human pairwise preferences. The power of the proposed approach is demonstrated through two experiments on a planar biped AMBER-3M: the first with rigid point feet, and the second with induced model uncertainty through the addition of springs where the added compliance was not accounted for in the gait generation or in the controller. In both experiments, the framework achieves stable, robust, efficient, and natural walking in fewer than 50 iterations with no reliance on a simulation environment. These results demonstrate a promising step in the unification of control theory and learning.**

## I. Introduction

Despite advancements within robotics, realizing dynamic bipedal locomotion on hardware [1] remains a benchmark problem across the fields of control, engineering, high-performance computing and machine learning. The dynamics and control community has historically approached the challenge of walking from theory applied to real-world platforms, for example Raibert's seminal work on hopping robots [2]. Such theory includes locomotion stability, which has been well studied and realized experimentally from various control perspectives including *zero moment point* (ZMP) [3] and simplified models, such as LIP [4], SLIP [5], and centroidal dynamics [6]. These methods, although powerful, do not account for the full-order dynamics of the system.

Alternatively, the *hybrid zero dynamics* (HZD) framework reduces the full-order dynamics to a lower-dimensional zero dynamics manifold, through which stability of the overall system can be certified. This is accomplished by characterizing walking with hybrid systems that encode state jumps and Lyapunov methods robust to these jumps [7]–[9]. This approach has been demonstrated for walking [10], running [11], and quadrupedal locomotion [12]. To achieve experimental success, however, one needs more than the theoretic stability guarantees enjoyed by the theory — one must achieve robustness against unmodeled dynamics, which is difficult to formulate formally in the context of attributes of nonlinear controllers. This "last-mile mission" was historically solved by intensive parameter tuning, an arduous and nonintuitive process which inevitably affects the scalability of translating theory to hardware in a practical setting.

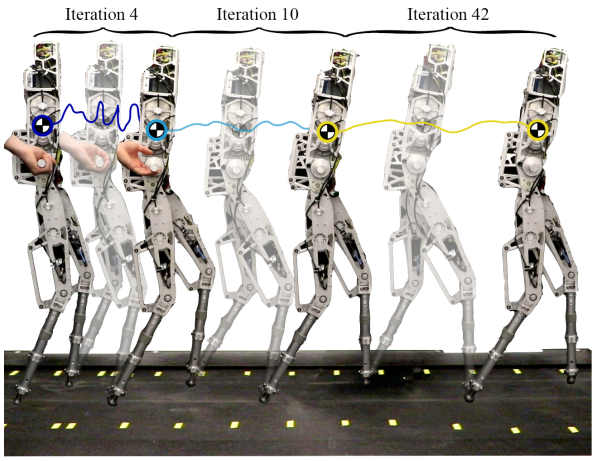


Fig. 1. Through 50 iterations of experiments, the proposed combination of preference-based learning and HZD optimization transforms failed gaits into robust walking on the AMBER-3M robot with a pair of compliant legs.

To circumvent this engineering empiricism, the field of machine learning has approached bipedal locomotion from many perspectives, including reinforcement leaning and imitation learning. Reinforcement learning simplifies the process of "learning to walk" [13] without prior knowledge [14]–[17], but because this method relies on a carefully crafted reward function, the behavior is exclusively determined by its construction. This motivates the second method, imitation learning, which infers the underlying reward function from expert demonstrations [18]–[20]. While both methods have demonstrated promising results, they heavily rely on physical engines such as Bullet [21], MuJoCo [22], and RaiSim [23]. As realistic as these rigid-body-dynamics based simulation environments have become, they still struggle with rough-terrain dynamics such as elastic impacts, slipping contacts, and granular media. These differences become more apparent when transferred to real-world systems.

As opposed to relying on just one field, this work explores combining the successes of both: the formality of stability from control theory and the ability to learn the relationship between complex parameter combinations and their resulting locomotive behavior from machine learning. This is accomplished by building upon our previous results [24], [25] and

This research was supported by NSF NRI award 1924526 and CMMI award 1923239, NSF Graduate Research Fellowship No. DGE-1745301, and the Caltech Big Ideas and ZEITLIN Funds.

[1]Authors are with the Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA 91125.

[2]Authors are with the Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125.

systematically integrating preference-based learning with gait generation via HZD optimization. The result is optimal walking on hardware based only on relative pairwise preferences from the human operator (i.e. the user prefers gait A over gait B). We demonstrate the power of this framework through two experiments on a planar biped AMBER-3M portrayed in Fig. 1. In both experiments stable, robust, efficient, and natural walking is achieved in fewer than 50 iterations with no reliance on a simulation environment. Notably, this is the first experimental demonstration of compliant walking through learning.

## II. HZD Gait Generation

The underlying control scheme of the proposed learning framework is based around two concepts: (1) hybrid zero dynamics (HZD) [7], [8], which theoretically addresses locomotion stability, and (2) trajectory optimization, namely direct collocation [26], which produces a walking trajectory (gait) that encodes the stability of the closed-loop system. We will briefly review this methodology in this section.

### A. Hybrid Zero Dynamics Method

Inherently, locomotion consists of alternating sequences of continuous-time dynamics and discrete-time impacts, which can be encoded as a hybrid control system [27]. Specifically, consider a robotic system of dimension $n$ with $q \in \mathcal{Q} \subset \mathbb{R}^n$ the configuration coordinates and $x = (q, \dot{q}) \in \mathcal{X} \subset T\mathcal{Q}$ the full system state. The continuous-time control system is given by

$$D(q)\ddot{q} + H(q, \dot{q}) = Bu \tag{1}$$

where $D(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $H(q, \dot{q}) \in \mathbb{R}^n$ is the drift vector, $B \in \mathbb{R}^{n\times m}$ is the actuation matrix, and $u \in \mathcal{U} \subset \mathbb{R}^m$ is the input. Here we present the "pinned" model for notional simplicity, but the "unpinned model" could similarly be considered [28]. Note that $m < n$ for underactuated robotic systems, such as the one under consideration in this paper.

As the robot's foot strikes the ground, an instantaneous change in velocity occurs causing the system state to suddenly jump. Taking $z : \mathcal{Q} \to \mathbb{R}$ to represent the height of the swing foot, the admissible states are given by the *domain*: $\mathcal{D} := \{(q, \dot{q}) \in \mathcal{X} : z(q) \geq 0\} \subset \mathcal{X}$. The region where this instantaneous change in velocity occurs is given by the *switching surface* $\mathcal{S} \subset \mathcal{D}$ defined by:

$$\mathcal{S} := \{(q, \dot{q}) \in \mathcal{X} \mid z(q) = 0, \dot{z}(q, \dot{q}) < 0\}. \tag{2}$$

Taking $x := (q, \dot{q})$, the discrete dynamics during this impact event are encoded by the *reset map* $\Delta : \mathcal{S} \to \mathcal{X}$, defined as:

$$x^+ = \Delta(x^-), \qquad x^- \in \mathcal{S} \tag{3}$$

where the $x^+$ and $x^-$ denote the pre- and post-impact state respectively. Finally, one can convert (1) to a *control system*: $\dot{x} = f(x) + g(x)u$ where when combined with (2) and (3) yields the single-domain hybrid control system:

$$\mathcal{HC} = \begin{cases} \dot{x} = f(x) + g(x)u & x \in \mathcal{D} \subset \mathcal{X} \\ x^+ = \Delta(x^-), & x^- \in \mathcal{S} \subset \mathcal{D} \end{cases} \tag{4}$$

which can be extended to the multi-domain case; for more details, refer to [7].

The HZD framework reduces the system $\mathcal{HC}$ to the lower-dimensional system. Consider the *zero dynamics surface*:

$$\mathcal{Z}_\alpha := \{x \in \mathcal{D} \mid y(q, \alpha) = 0,\ \dot{y}(q, \alpha) = 0\},$$

where $y : \mathcal{Q} \to \mathbb{R}^m$ is defined through the following *outputs* or *virtual constraints* (encoding desired behavior):

$$y(q, \alpha) = y^a(q) - y^d(\tau(q), \alpha). \tag{5}$$

Here, $y^a(q)$ is the actual measured output of the system, and $y^d(\tau(q), \alpha)$ is the desired output. For the following discussion, we take the desired output to be parameterized by the state-based timing variable $\tau(q)$ and a collection of Bézier coefficients $\alpha$. Through the use of a stabilizing controller $u^*(x)$, e.g., given by feedback linearizaiton or control Lyapunov functions [8], [9], [27] one can drive $y \to 0$ exponentially. The end result is the *closed-loop dynamics*: $\dot{x} = f_{\text{cl}}(x) = f(x) + g(x)u^*(x)$. In order to guarantee stability of a hybrid system, a hybrid invariance condition must be satisfied, encoded through the *HZD condition* [8]:

$$\Delta(\mathcal{S} \cap \mathcal{Z}_\alpha) \subset \mathcal{Z}_\alpha. \tag{6}$$

The remaining step to achieving hybrid invariant walking is to generate an $\alpha$ that satisfies the HZD condition.

### B. Trajectory Optimization

To obtain $\alpha$, we use a direct collocation-based optimization algorithm, FROST [26], which has been previously utilized for efficient gait generation of walking [29], running [11], and quadrupedal locomotion [30]. Direct collocation is an implicit Runge–Kutta method to approximate the numerical solution of certain dynamical systems, namely differential-algebraic equations and partial differential equations. The trajectory optimization problem is stated as

**HZD Optimization:**

$$\begin{aligned} \{\alpha^*, X^*\} = \underset{\alpha, X}{\text{argmin}}\ & \Phi(X) \\ \text{s.t.}\quad & \dot{x} = f_{cl}(x) && \text{(Closed-loop Dynamics)} \\ & \Delta(\mathcal{S} \cap \mathcal{Z}_\alpha) \subset \mathcal{Z}_\alpha && \text{(HZD Condition)} \\ & X_{\min} \preceq X \preceq X_{\max} && \text{(Decision Variables)} \\ & c_{\min} \preceq c(X) \preceq c_{\max} && \text{(Physical Constraints)} \\ & a_{\min} \preceq p(X) \preceq a_{\max} && \text{(Essential Constraints)} \end{aligned}$$

where $X = (x_0, ..., x_N, T)$ is the collection of all decision variables with $x_i$ the state at the $i^{th}$ discretization and $T$ the duration, $\Phi(X)$ is the cost function, and $c(X)$ is the set of physical constraints on the optimization problem. These physical constraints are included in every gait generation framework to encode the physical laws of real-word, such as the friction cone condition, workspace limit, and motor capacity [29]. In this work, we specify a subset of these constraints, $p(X)$, termed in this work as *essential constraints*, which are discussed further in Sec. II-C. With this optimization formulation, we can use nonlinear programming (NLP) solvers such as IPOPT [31], and SNOPT [32] to

efficiently synthesize an optimal walking gait. The end result is a stable periodic solution to the walking dynamics that is parameterized by some static parameter $\alpha^*$.

### C. Essential Constraints

Traditionally, operators manually select the bounds on key constraints such as walking frequency, walking velocity, step length, foot clearance, and impact velocity. These constraints are essential to achieving experimental robustness, and hence are termed here *essential constraints*. Often, practitioners derive intuition from years of experience on how to shape these constraints. One example of how this intuition relates to stability is Raibert-type controllers [2], which tune the relationship between step length and walking velocity based on a simplified model. These essential constraints are normally posed as inequality constraints (path constraints), where $a_{\min}$ and $a_{\max}$ are tuned manually. In this paper, we present a systematic approach to optimize essential constraints using preference-based learning. To do so, the inequality constraints are reformulated as:

$$a - \delta \preceq p(X) \preceq a + \delta$$

with $\delta < m$ (defined in Alg. 1). Here, $a$ is the vector of essential constraints defined as an action in Sec. III.

## III. Learning Framework

Let $a \in \mathbb{R}^v$ be a vector representing essential constraint values where $v$ is the number of constraints selected by the user to optimize over. This work is interested in learning the action $a^*$ that maximizes how stable, robust, and natural a walking gait is when executed on hardware. To set up the learning problem, upper and lower bounds on $a$ ($a_{max}$ and $a_{min}$, respectively) along with the granularity of discretization for each dimension $d_\kappa, \kappa = 1, ..., v$ are chosen by the operator. The entire search space of the algorithm is given by $\mathbf{A} \subset \mathbb{R}^d$ defined as the finite set of all constraint combinations where $d = \prod_{\kappa=1,...,v} d_\kappa$.

By naïvely searching over $\mathbf{A}$, the algorithm quickly faces the curse of dimensionality. Thus, to facilitate computational tractability, each iteration, $i$, only explores a subset of actions $\mathbf{S}_i \subset \mathbf{A}$. To learn $a^*$, we introduce a framework built around a high-dimensional preference-based learning algorithm LineCoSpar [25] that learns a Bayesian model over a user's preferences by sequentially constructing $\mathbf{S}_i$ . The underlying assumption of the algorithm is that the user's preferences are dictated by some unknown utility function $f : \mathbb{R}^v \to \mathbb{R}$ where $f(a)$ is termed the *latent utility* and $\boldsymbol{f} : \mathbf{A} \to \mathbb{R}$ represents the collection of all latent utilities. Finally, let $\boldsymbol{f}_B$ be the restriction of $f$ on some set $B \subset \mathbf{A}$.

In general, Bayesian optimization is a powerful tool for optimizing black-box functions by maintaining a model posterior over the unknown function. Feedback typically used in Bayesian optimization is numerical, however previous work has extended Bayesian optimization to pairwise preferences and other qualitative feedback mechanisms [24], [33]–[36].

LineCoSpar has previously been demonstrated to locate an optimal exoskeleton gait over a pre-computed gait library

**Algorithm 1** LineCoSparNLP

**procedure** LineCoSparNLP(select $v$ essential constraints, $a_{\min}$, $a_{\max}$, $d_\kappa$ for $\kappa = 1, ..., v$, and $n$)
  $\mathbf{D}_0 = \emptyset$ ▷ $\mathbf{D}_i$ : Preference feedback including iteration $i$
  $\mathbf{E}_0 = \emptyset$ ▷ $\mathbf{E}_i$ : Executed actions including iteration $i$
  Obtain $n$ uniformly-random actions: $\mathbf{A}_1 = \{a_{1j}, ...., a_{1n}\}$
  Execute outputs of NLP for $\mathbf{A}_1$ on the system
  Query operator for preference feedback: $y_1$
  Append executed actions: $\mathbf{E}_1 = \mathbf{E}_0 \cup \mathbf{A}_1$
  Append preference feedback: $\mathbf{D}_1 = \mathbf{D}_0 \cup y_1$
  **for** $i = 2, 3, \ldots, I$ **do**
    Obtain covariance $[\Sigma]_{j,k} = \mathcal{K}(a_j, a_k)$ for $a \in \mathbf{E}_{i-1}$
    Update posterior over $\mathbf{E}_{i-1}$ to obtain $(\mu_{i-1}, \Sigma_{i-1})$ given $\mathbf{D}_{i-1}$
    Update $a^*_{i-1} = \underset{\boldsymbol{a} \in \mathbf{E}_{i-1}}{\text{argmax}} \mu_{i-1}(\boldsymbol{a})$
    $\mathbf{L}i$ = random line through $a^*_{i-1}$, discretized via $m_v$
    Update subset $\mathbf{S}_i = \mathbf{L}_i \cup \mathbf{E}_{i-1}$
    Obtain covariance $[\Sigma]_{j,k} = \mathcal{K}(a_j, a_k)$ for $a \in \mathbf{E}_i$
    Update posterior over $\mathbf{S}_i$ to obtain $(\mu_i, \Sigma_i)$ given $\mathbf{D}_{i-1}$
    Draw $j = 1, ...n$ samples: $f_j \sim \mathcal{N}(\mu_i, \Sigma_i)$
    Update actions $\mathbf{A}_i = \{a_j = \underset{a \in \mathbf{E}_i}{\text{argmax}} f_j(a) | j = 1, ...n\}$
    Run NLP for actions $\mathbf{A}_i$ to obtain
    Execute outputs of NLP for $\mathbf{A}_i$ on the system
    Query operator for preference feedback $y_i$
    Append executed actions: $\mathbf{E}_i = \mathbf{E}_{i-1} \cup \mathbf{A}_i$
    Append preference feedback: $\mathbf{D}_i = \mathbf{D}_{i-1} \cup y_i$
  **end for**
  Obtain covariance $[\Sigma]_{j,k} = \mathcal{K}(a_j, a_k)$ for $a \in \mathbf{E}_I$
  Update posterior over $\mathbf{D}_I$ to obtain $(\mu_I, \Sigma_I)$
  Obtain $a^*_I = \underset{\boldsymbol{a} \in \mathbf{E}_I}{\text{argmax}} \mu_N(\boldsymbol{a})$
**end procedure**

[25]. In this work, we extend the learning framework to include the nonlinear optimization problem directly which eliminates the need for pre-computed gait libraries and allows for a more diverse set of behaviors. We term this slightly modified algorithm LineCoSparNLP where the goal is to identify the optimal action $a^* \in \mathbf{A}$ that maximizes $f(a)$ of the human operator in as few iterations as possible. For example, $f(a)$ may account for stability, and robustness to perturbations and model uncertainty. This multivariate utility function often admits no mathematical description; rather, the operator has intuition about what looks "right". Information about $f(a)$ is obtained through pairwise preferences. To learn from these preferences, we adopt the dueling bandit setting [37] in which the algorithm selects actions to execute on the system and receives a user's relative preferences between the executed actions.

### A. The LineCoSparNLP Algorithm

The procedure of the LineCoSparNLP algorithm (Alg. 1) is as follows. First, the algorithm begins by randomly selecting and executing $\mathbf{A}_1$ where $\mathbf{A}_i \in \mathbb{R}^{v \times n}$ is the $n$ actions selected to execute on the system for iteration $i$. The parameter $n$ can be changed depending on how many actions the operator would like to sample in each iteration. Since the actions are compared in pairs, $n$ actions equates to $m = \binom{n}{2}$ pairwise preferences. These actions are given to the NLP and $n$ corresponding gaits are generated, which are then executed on hardware. We define the set of actions

executed on hardware up to and including those sampled in iteration $i$ as $\mathbf{E}_i := \bigcup_{j=1,\dots,i} \mathbf{A}_j \subset \mathbf{A}$.

After demonstrating the gaits on hardware, the human operator is queried for $m$ pairwise preferences, denoted as $y_i$ for iteration $i$. $\mathbf{D}_i = \bigcup_{j=1,\dots,i} y_j \in \mathbb{R}^{N_p^{(i)}}$ is defined as the dataset of all preference feedback up to and including iteration $i$ with corresponding number of preferences $N_p^{(i)}$. Preference feedback is omitted when all sampled actions do not converge, or when the user chooses to give feedback of "no preference". Thus, $N_p^{(i)} = im - N_{\text{omit}}$ where $N_{\text{omit}}$ is number of omitted preferences.

In each iteration, the posterior is updated over the previously executed actions $\mathbf{E}_{i-1}$ given the dataset $\mathbf{D}_{i-1}$ by modeling the underlying utilities of $\mathbf{E}_{i-1}$, denoted as $\boldsymbol{f}_{\mathbf{E}_{i-1}}$, through the Gaussian posterior:

$$\mathbb{P}(\boldsymbol{f}_{\mathbf{E}_{i-1}}|\mathbf{D}_{i-1}) \propto \mathbb{P}(\mathbf{D}_{i-1}|\boldsymbol{f}_{\mathbf{E}_{i-1}})\mathbb{P}(\boldsymbol{f}_{\mathbf{E}_{i-1}}). \tag{7}$$

Thus, to calculate this posterior, we first calculate the Gaussian process prior defined over the utilities $\boldsymbol{f}_{\mathbf{E}_{i-1}}$ as:

$$\mathbb{P}(\boldsymbol{f}_{\mathbf{E}_{i-1}}) = \frac{1}{(2\pi)^{\frac{|\mathbf{E}_{i-1}|}{2}}|\Sigma_i^{\text{pr}}|^{1/2}} \exp\left(-\frac{1}{2}\boldsymbol{f}_{\mathbf{E}_{i-1}}(\Sigma_i^{\text{pr}})^{-1}\boldsymbol{f}_{\mathbf{E}_{i-1}}\right),$$

where $\Sigma_i^{\text{pr}} \in \mathbb{R}^{|\mathbf{E}_{i-1}|\times|\mathbf{E}_{i-1}|}$, $[\Sigma_i^{\text{pr}}]_{j,k} = \mathcal{K}(a_{ij}, a_{jk})$, and $\mathcal{K}$ is a kernel of choice. In this work we utilize a squared exponential kernel. Second, the likelihood $\mathcal{P}(\mathbf{D}_{i-1}|\boldsymbol{f}_{\mathbf{E}_{i-1}})$ is computed as:

$$\mathbb{P}(\mathbf{D}_{i-1}|\boldsymbol{f}_{\mathbf{E}_{i-1}}) = \prod_{k=1}^{N_p^{i-1}} g_{\text{sig}}\left(\frac{f(a_{k_1}) - f(a_{k_2})}{c}\right)$$

with $g_{\text{sig}}(x)$ as the chosen link function. This link function can be replaced with any monotonically-increasing function, but we found empirically that the heavy-tailed sigmoid distribution $g_{\text{sig}}(x) := \frac{1}{1+e^{-x}}$ improves performance. Finally, the posterior (7) is estimated via the Laplace approximation as in [38] which yields a multivariate Gaussian, $\mathcal{N}(\mu_{i-1}, \Sigma_{i-1})$. The posterior is then used to update the optimal action:

$$a_{i-1}^* = \underset{a\in\mathbf{E}_{i-1}}{\text{argmax}}\ \mu_{i-1}(a).$$

A random linear subspace $\mathbf{L}_i \subset \mathbf{A}$ is then generated to intersect $a_{i-1}^*$. The underlying utilities of the actions within the subset $\mathbf{S}_i := \mathbf{L}_i \cup \mathbf{E}_{i-1} \subset \mathbf{A}$, denoted as $\boldsymbol{f}_{\mathbf{S}_i}$, are then modeled through the Gaussian posterior:

$$\mathbb{P}(\boldsymbol{f}_{\mathbf{S}_i}|\mathbf{D}_{i-1}) \propto \mathbb{P}(\mathbf{D}_{i-1}|\boldsymbol{f}_{\mathbf{S}_i})\mathbb{P}(\boldsymbol{f}_{\mathbf{S}_i}),$$

which is calculated as with the posterior over $\mathbf{E}_{i-1}$.

Next, $n$ new actions are sampled using the Self Sparring approach [39] to Thompson sampling, a regret minimization sampling method. This method selects actions to execute by first sampling $n$ utility functions, denoted $f_j$ for $j = 1, ..., n$, from the posterior distribution $\mathcal{N}(\mu_i, \Sigma_i)$. The selected actions, $\mathbf{A}_i$, are those which maximize the sampled functions. These are then given to the NLP, whereby corresponding gaits are generated, the outputs are executed on the robot, and $\mathbf{A}_i$ is appended to $\mathbf{E}_i$. The operator is again queried for preference feedback which is added to $\mathbf{D}_i$.

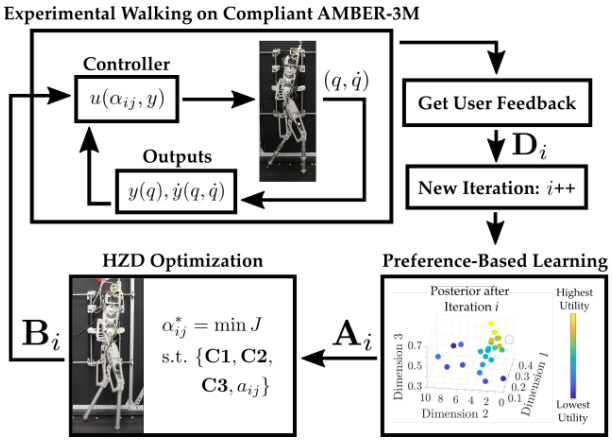


Fig. 2. The experimental procedure is illustrated in terms of each iteration $i$ with $n$ denoting the number of gaits compared in each iteration. The experiments presented in this work used $n = 2$. Using this notation, the set of $n$ actions given to the HZD optimization is denoted: $\mathbf{A}_i = \{a_{ij}|j = 1, \dots, n\}$. The resulting $n$ sets of Bézier coefficients given to the controller are denoted $\mathbf{B}_i = \{\alpha_{ij}|j = 1, ..., n\}$.

### B. Changes to LINECOSPAR for use with a NLP

Three notable changes were made to the algorithm LINECOSPARNLP in comparison to LINECOSPAR. First, the LINECOSPAR algorithm selects $\mathbf{L}_i$ to intersect $a_{i-2}^*$. To take advantage of more recent preference feedback, LINECOSPARNLP selects $\mathbf{L}_i$ to intersect $a_{i-1}^*$. This change requires two posterior updates in each iteration. Second, LINECOSPAR uses a buffer method to compare executed actions with previously executed actions which results in higher sample-efficiency. However, when considering preference-based learning towards gait generation, it is important to account for the computation time required to obtain gaits. For this reason, we modify the LINECOSPARNLP algorithm to sample and query $n$ actions in each iteration. This results in worse sample-efficiency, but allows for batched gait generation that enables the generated gaits to be executed on hardware back to back. Lastly, in LINECOSPAR, coactive feedback is also added to the dataset $\mathbf{D}_i$ to improve sample-efficiency. However, coactive feedback, otherwise known as user suggestions, relies on a known mapping between the actions $a$ and the underlying utility function $f(a)$. Since this mapping is rarely well-understood for the parameters of a nonlinear optimization problem, LINECOSPARNLP does not utilize coactive feedback.

## IV. LEARNING TO WALK IN EXPERIMENTS

We experimentally deploy LINECOSPARNLP on the planar bipedal robot, AMBER-3M [40]. This custom research platform has three interchangeable lower-limb configurations (flat-foot, point-foot, and spring-foot) and was originally built to study how leg configurations effect energy efficiency. We specifically selected this platform because of its engineering reliability [12], enabling consistent data collection to isolate the effects of various gaits in the learning process. The controller for AMBER-3M is implemented on an off-board i7-6700HQ CPU @ 2.6GHz with 16 GB RAM, which computes desired torques and communicates them with the

ELMO motor drivers. The motor driver communication and the control logic run at ∼1kHz, each on a separate core.

### A. Experimental Procedure

In the experiments (video: [41]), walking gaits are generated by the HZD trajectory optimization method presented in Sec. II. We take $y^a(q) := q_a \in \mathbb{R}^4$ as the position of the four motorized joints of AMBER-3M, $\tau(q)$ to be the linearized forward hip position, and use a $5^{th}$-order Bézeir polynomial ($\alpha \in \mathbb{R}^{4\times 6}$) to describe the desired output trajectories. Additionally, the cost function is selected to be the mechanical cost of transport (MCOT), a common metric for locomotion efficiency defined by:

$$MCOT = \int_{t_0}^{t_f} \frac{P(t)}{mgv} dt, \tag{8}$$

where $P(t) = \|u(t)^T \dot{q}_a(t)\|_2$, the 2-norm sum of positive power. The essential constraints selected to tune through learning are:

1) Average forward velocity of the torso $(m/s)$
2) Phase variable value at which to enforce minimum foot clearance, $\tau_c$
3) Minimum nonstance foot clearance enforced at $\tau_c$ $(m)$
4) Downward velocity enforced at impact $(m/s)$
5) Step length, i.e. the forward distance between swing foot and stance foot at impact $(m)$

The average optimization run time is 0.1 second per iteration, with each gait averaging 160 iterations. The experimental procedure is illustrated in Fig. 2. In our experiments, the learning was conducted for $n = 2$, corresponding to two gaits being compared in each iteration. This was chosen because fewer pairwise comparisons are easier to give feedback between, and thereby result in less noise in the preferences. Note that other applications may benefit in a higher $n$, which would increase the rate of learning.

Each trial began by initializing AMBER-3M in a stance configuration, starting the treadmill, and attempting to push the robot into its periodic orbit. If the robot seemed like it would fall, extra precaution was taken to give the gait the best chance at succeeding. Once the gait reached its orbit, the robot was released and the robustness of the gait to various disturbances was investigated. After both gaits were executed on the physical robot, preferences were collected from the human operator observing the physical realization of the walking. In some iterations, video footage was also reviewed before giving a preference. The criteria used to determine preferences between gaits were the following (in order of prioritization):

- Capable of walking
- Robust to perturbations in treadmill speeds
- Robust to external forces
- Does not exhibit harsh noise (e.g. during impact)
- Is visually appealing (intuitive judgment from operator)

### B. Procedure specific to AMBER-P and AMBER-S

In this work, we leverage two configurations of the robot: 1) the point-foot configuration (1.373 m tall, 21.3 kg), AMBER-P; and 2) the spring-foot configuration (1.430 m tall, 23.5 kg), AMBER-S. We first demonstrate the learning framework on AMBER-P, with the corresponding robot model used in the gait generation. To emphasize the scalability of LINECOSPARNLP, we also repeat the exact same procedure applied to AMBER-S while still using the AMBER-P model in the trajectory optimization process. We intentionally do not account for these changes and instead gaits were was still generated assuming the rigid body model and executed on hardware using the same controller with unmodified PD gains. Even though we don't account for these changes, the addition of compliance in the legs increases the degrees of freedom of the system, adds a double support domain to the hybrid dynamics, and increases the stiffness of the dynamics. Historically, robots with compliance are difficult to generate gaits for because of these added complexities, with past success relying on sophisticated models [42]. Therefore, the fact that the algorithm converges despite these unmodeled complexities highlights the effectiveness of the LINECOSPARNLP learning method towards achieving experimental robustness.

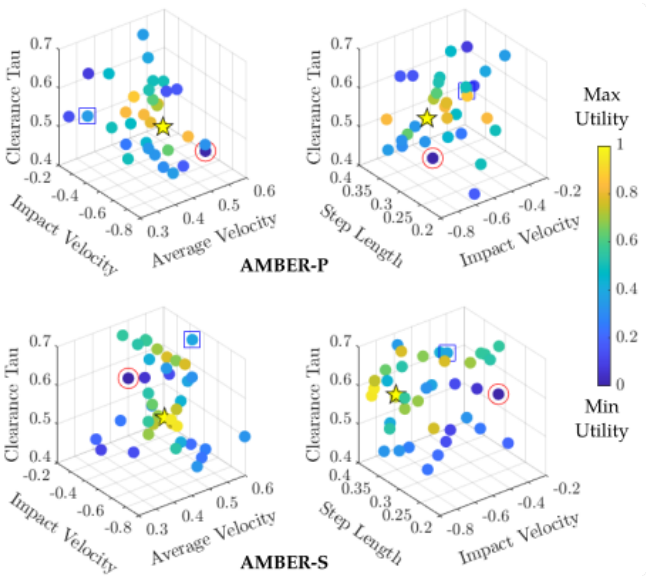


Fig. 3. An illustration of the final utility values obtained for the 5-dimensional visited actions, averaged over the two dimensions not shown on each subplot. The optimal action is illustrated by the yellow star( $[0.4399, 0.5425, 0.0759, -0.6040, 0.3190]$ for AMBER-P and $[0.4105, 0.5930, 0.0833, -0.7020, 0.3504]$ for AMBER-S). The other two actions depicted in Fig. 4 are denoted with a red circle (worst gait) and a blue square (middle gait).

### C. Results

During the experiment on AMBER-P, the gaits quickly met the first criterion of being able to walk; therefore, as the trials progressed, the efficiency, robustness, and naturalness became the key criteria in determining preferences between gaits. The experiment using the rigid model was run for a total of 30 iterations and sampled 27 unique gaits. The final posterior over the 27 sampled actions is illustrated in the top row of Fig. 3. To elucidate the success of the LINECOSPARNLP, three gaits are selected from the experiment for careful investigation corresponding to the

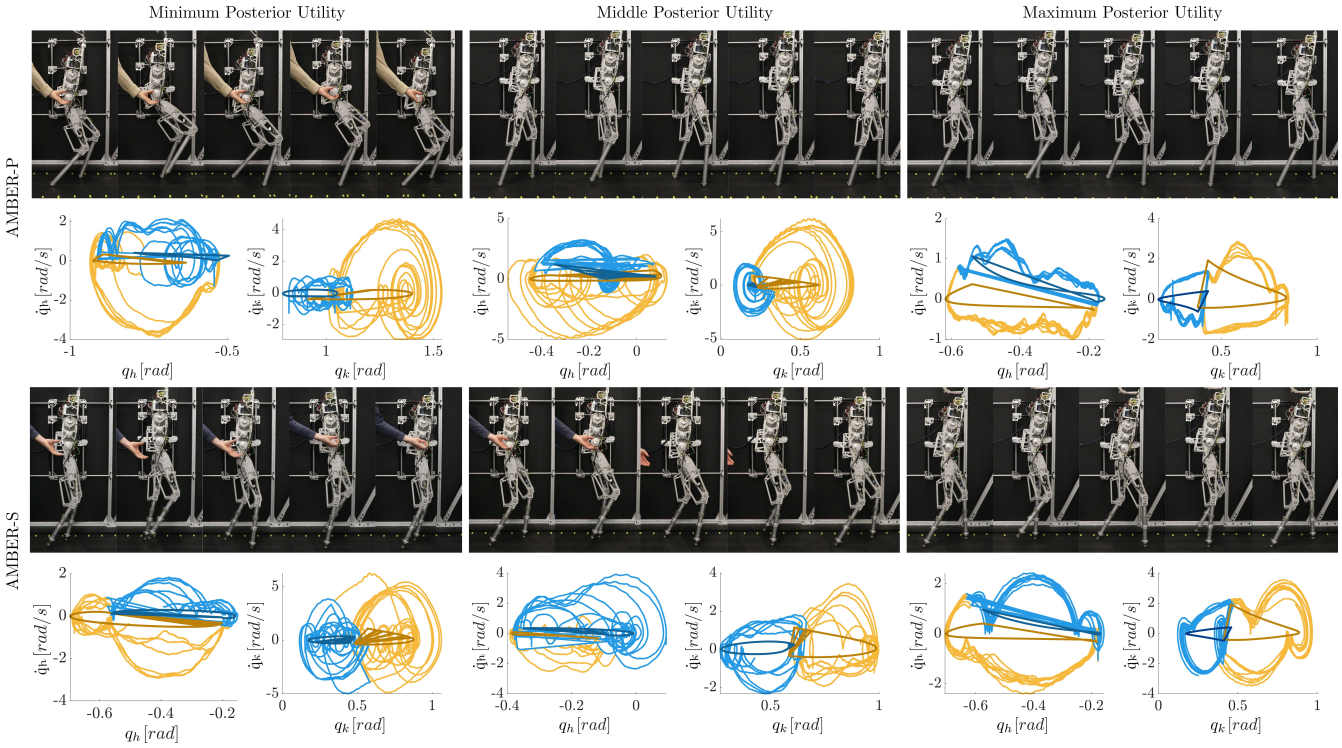


Fig. 4. Gait tiles with increasing posterior utility values from left to right are shown for the the rigid model (top) and spring model (bottom). The phase portraits of the hip ($q_h$) and knee ($q_k$) of the stance leg (blue) and swing leg (yellow) are shown below each corresponding gait, plotted over 10 seconds of data. The phase portraits clearly indicate that for both AMBER-P and AMBER-S the gaits evolved to be more experimentally robust.

minimum, a middle, and the maximum posterior utility. The iteration numbers corresponding to when these gaits were first sampled is 1, 21, and 26, respectively.

The power of this method is truly demonstrated by its ability to facilitate the experimental realization of robust gaits. The initial gaits tried on hardware, although optimal subject to the imposed constraints, resulted in inferior trajectory tracking and power consumption. As the algorithm progressed, the gaits became significantly smoother, more robust to disturbance, and energy efficient. This is exemplified in Figure 4 by noting the significantly lower velocity overshoot for all of the limbs and tighter tracking shown in the phase portraits for the gaits with higher posterior utility. The improvement in energy efficiency is illustrated by the decreasing MCOT values, which were 0.74, 0.95, and 0.26 corresponding to the three gaits, respectively.

When the procedure was repeated on the AMBER-S platform, many of the initial tested gaits were unable to walk due to the unmodeled compliance. Thus, gaits exhibiting periodic walking behavior were strongly preferred. This second experiment was conducted for 50 iterations and sampled 37 unique gaits with the obtained posterior illustrated in the bottom row of Fig. 3. As with the previous experiment, three gaits are selected for further discussion corresponding to the minimum, a middle, and the maximum posterior utility values. Gait tiles and phase portraits for these are again shown in Figure 4. The iterations when these gaits were first sampled are 4, 10, and 42. Once again, the algorithm converges to gaits with superior trajectory tracking and lower MCOT (1.16, 0.38, and 0.33, respectively).

## V. Conclusion

In this work, we present and experimentally demonstrate a preference-based learning framework, LineCoSparNLP, specifically designed for use towards gait generation via HZD optimization. The success of the proposed learning method is demonstrated through its ability to experimentally realize gaits that are stable, robust to model uncertainty, robust to external perturbations, efficient, and natural looking with no requirement for simulation within 50 experimental iterations.

LineCoSparNLP incorporates preference-based learning with HZD optimization to leverage the theoretical benefits of HZD without the challenge of parameter tuning. Furthermore, preference-based learning is a sample-efficient learning method that does not require the user to mathematically define a metric for "good" walking. Instead, the framework relies on easy to provide pairwise preferences. The success of LineCoSparNLP is demonstrated by achieving robust walking with unmodeled compliant legs, a historically challenging control task.

Future work includes extending this framework to more robotic platforms, such as quadrupeds and 3D bipedal robots, as well as improving the sample-efficiency of the framework through additional qualitative feedback mechanisms such as ordinal labels [43]. The experimental results presented in this paper demonstrate the rich potential lying in the boundary between machine learning and control theory. It is well-known that control theory provides necessary structure to bipedal platforms, but machine learning can play a critical role in shaping the final behavior of the system.

## REFERENCES


[1] E. Krotkov, D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, and C. Orlowski, "The darpa robotics challenge finals: Results and perspectives," *Journal of Field Robotics*, vol. 34, no. 2, pp. 229–240, 2017.

[2] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[3] T. Sugihara, Y. Nakamura, and H. Inoue, "Real-time humanoid motion generation through zmp manipulation based on inverted pendulum control," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2. IEEE, 2002, pp. 1404–1409.

[4] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1. IEEE, 2001, pp. 239–246.

[5] I. Poulakakis and J. W. Grizzle, "The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper," *IEEE Transactions on Automatic Control*, vol. 54, no. 8, pp. 1779–1793, 2009.

[6] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous robots*, vol. 35, no. 2-3, pp. 161–176, 2013.

[7] J. W. Grizzle, C. Chevallereau, A. D. Ames, and R. W. Sinnet, "3d bipedal robotic walking: models, feedback control, and open problems," *IFAC Proceedings Volumes*, vol. 43, no. 14, pp. 505–532, 2010.

[8] A. D. Ames, "Human-inspired control of bipedal walking robots," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1115–1130, 2014.

[9] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[10] K. Sreenath, H.-W. Park, I. Poulakakis, and J. W. Grizzle, "A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1170–1193, 2011.

[11] W.-L. Ma, S. Kolathaya, E. R. Ambrose, C. M. Hubicki, and A. D. Ames, "Bipedal robotic running with durus-2d: Bridging the gap between theory and experiment," in *Proceedings of the 20th international conference on hybrid systems: computation and control*, 2017, pp. 265–274.

[12] W.-L. Ma, Y. Or, and A. D. Ames, "Dynamic walking on slippery surfaces: Demonstrating stable bipedal gaits with planned ground slippage," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3705–3711.

[13] "NeurIPS 2019: Learn to move - walk around," https://www.aicrowd.com/challenges/neurips-2019-learning-to-move-walk-around.

[14] G. A. Castillo, B. Weng, A. Hereid, Z. Wang, and W. Zhang, "Reinforcement learning meets hybrid zero dynamics: A case study for rabbit," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 284–290.

[15] K. Hitomi, T. Shibata, Y. Nakamura, and S. Ishii, "Reinforcement learning for quasi-passive dynamic walking of an unstable biped robot," *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 982–988, 2006.

[16] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," *arXiv preprint arXiv:2002.08550*, 2020.

[17] J. Morimoto, G. Cheng, C. G. Atkeson, and G. Zeglin, "A simple reinforcement learning algorithm for biped walking," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 3. IEEE, 2004, pp. 3030–3035.

[18] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[19] S. Tirumala, S. Gubbi, K. Paigwar, A. Sagi, A. Joglekar, S. Bhatnagar, A. Ghosal, B. Amrutur, and S. Kolathaya, "Learning stable manoeuvres in quadruped robots from expert demonstrations," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 1107–1112.

[20] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative reinforcement learning based design of dynamic locomotion skills for cassie," *arXiv preprint arXiv:1903.09537*, 2019.

[21] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[22] 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*. IEEE, 2012, pp. 5026–5033.

[23] "Raisim," https://github.com/raisimTech/raisimlib, 2020.

[24] M. Tucker, E. Novoseller, C. Kann, Y. Sui, Y. Yue, J. W. Burdick, and A. D. Ames, "Preference-based learning for exoskeleton gait optimization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2351–2357.

[25] M. Tucker, M. Cheng, E. Novoseller, R. Cheng, Y. Yue, J. W. Burdick, and A. D. Ames, "Human preference-based learning for high-dimensional optimization of exoskeleton walking gaits," *arXiv preprint arXiv:2003.06495*, 2020.

[26] A. Hereid and A. D. Ames, "Frost: Fast robot optimization and simulation toolkit," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 719–726.

[27] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2018.

[28] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic humanoid locomotion: A scalable formulation for hzd gait optimization," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 370–387, 2018.

[29] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3d dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1447–1454.

[30] W.-L. Ma, N. Csomay-Shanklin, and A. D. Ames, "Coupled control systems: Periodic orbit generation with application to quadrupedal locomotion," *IEEE Control Systems Letters*, 2020.

[31] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[32] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[33] E. Bıyık, N. Huynh, M. J. Kochenderfer, and D. Sadigh, "Active preference-based gaussian process regression for reward learning," in *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[34] N. Thatte, H. Duan, and H. Geyer, "A method for online optimization of lower limb assistive devices with high dimensional parameter spaces," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–6.

[35] N. Wilde, D. Kulic, and S. L. Smith, "Active preference learning using maximum regret," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2020.

[36] L. Qian, J. Gao, and H. Jagadish, "Learning user preferences by adaptive pairwise comparison," *Proceedings of the VLDB Endowment*, vol. 8, no. 11, pp. 1322–1333, 2015.

[37] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims, "The k-armed dueling bandits problem," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1538–1556, 2012.

[38] W. Chu and Z. Ghahramani, "Preference learning with gaussian processes," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 137–144.

[39] Y. Sui, V. Zhuang, J. W. Burdick, and Y. Yue, "Multi-dueling bandits with dependent arms," *arXiv preprint arXiv:1705.00253*, 2017.

[40] E. Ambrose, W. Ma, C. Hubicki, and A. D. Ames, "Toward benchmarking locomotion economy across design configurations on the modular robot: Amber-3m," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, 2017, pp. 1270–1276.

[41] "Video of the experimental results." https://vimeo.com/473917519.

[42] A. Hereid, S. Kolathaya, M. S. Jones, J. Van Why, J. W. Hurst, and A. D. Ames, "Dynamic multi-domain bipedal walking with atrias through slip based human-inspired control," in *Proceedings of the 17th international conference on Hybrid systems: computation and control*, 2014, pp. 263–272.

[43] W. Chu and Z. Ghahramani, "Gaussian processes for ordinal regression," *Journal of machine learning research*, vol. 6, no. Jul, pp. 1019–1041, 2005.