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 leverages both control theory and machine learning to obtain stable and robust bipedal locomotion without the need for manual parameter tuning. 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 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.**
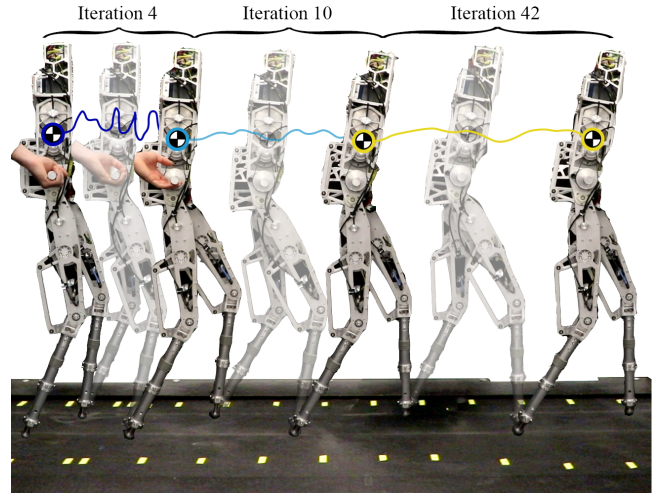
Fig. 1. Through 50 iterations of experiments, the proposed combination of preference-based learning and HZD optimization transforms failed gaits into robust walking on the AMBER-3M robot with a pair of compliant legs.

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 simple model-based methods, 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 first characterizing walking as a hybrid system with continuous dynamics and discrete state jumps. The HZD framework then uses Lyapunov methods to guarantee stability of the entire hybrid system [7]–[9]. This approach has been demonstrated

for walking [10]–[12], running [13], and quadrupedal locomotion [14]. To accomplish experimental success, however, one needs more than the theoretical stability guarantees — one must achieve robustness against unmodeled dynamics, which is especially difficult for model-based methods such as the HZD framework. 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.

To circumvent this engineering empiricism, the field of machine learning has approached bipedal locomotion from different perspectives, including reinforcement leaning and imitation learning. Reinforcement learning simplifies the process of "learning to walk" [15] without prior knowledge [16]–[19], but because this methodology 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 [20]–[22]. While both methods have demonstrated promising results, they heavily rely on physical engines such as Bullet [23], MuJoCo [24], and RaiSim [25]. 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 paper 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 [26], [27] and systematically integrating preference-based learning with gait generation via HZD optimization. The result is optimal walking on hardware based only on pairwise preferences from the operator (i.e. the user prefers gait A over gait B). We demonstrate the power of this framework through two experiments on a modular biped, AMBER-3M, shown in Fig. 1. In both experiments, stable, robust, efficient, and visually appealing walking is achieved on hardware in fewer than 50 iterations, with no reliance on a simulation environment.

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 [28], 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 [29]. Consider a robotic system with the configuration coordinates $q \in \mathcal{Q} \subset \mathbb{R}^n$ and the full system state $x = (q, \dot{q}) \in \mathcal{X} \subset T\mathcal{Q}$. 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 [30]. Note that $m < n$ for underactuated robotic systems, such AMBER-3M.

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*: $\mathcal{D} := \{(q, \dot{q}) \in \mathcal{X} \mid z(q) \geq 0\} \subset \mathcal{X}$. 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:

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

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:

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

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

The HZD framework reduces the system $\mathcal{HC}$ to a 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 α. Through the use of a stabilizing controller $u^*(x)$, e.g., given by feedback linearization or control Lyapunov functions [8], [9], [29], one can drive $y \to 0$ exponentially. The end result is the *closed-loop dynamics*: $\dot{x} = f_{cl}(x) = f(x) + g(x)u^*(x)$. In order to guarantee stability of a hybrid system, a hybrid invariance condition must be satisfied, encoded through the *HZD condition*:

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

The remaining step to achieving hybrid invariance is to generate α such that the HZD condition is satisfied.

B. Trajectory Optimization

To obtain α, we use a direct collocation based optimization algorithm, FROST [28], which has been previously utilized for efficient gait generation of walking [11], running [13], and quadrupedal locomotion [31]. 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:

$$\{\alpha^*, X^*\} = \underset{\alpha, X}{\operatorname{argmin}} \ \Phi(X)$$

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

---

where $X = (x_0, ..., x_N, T)$ is the collection of all decision variables with x_i the state at the ith 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 [12]. In this work, we specify a specific subset of physical constraints as $p(X)$, which we term *essential constraints* and discuss further in Sec. II-C. With this optimization formulation, we can use nonlinear programming (NLP) solvers, such as IPOPT [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 set of Bézier coefficients α^*.

C. Essential Constraints

Expert operators typically tune $a_{\min} \in \mathbb{R}^v$ and $a_{\max} \in \mathbb{R}^v$ of (Essential Constraints) in the hopes of guiding the HZD optimization towards a solution that maximizes the operators' subjective metric of "good" walking. Since the construction of these constraints is often essential towards achieving experimental robustness, we term them *essential constraints*. Traditionally, essential constraints consist of gait features such as average velocity, step length, foot clearance, and impact velocity. Often, practitioners derive intuition on how to shape essential constraints from years of experience. 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.

In this paper, we present a systematic approach towards tuning essential constraints using preference-based learning. To do so, we reformulate (Essential Constraints) as:

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

where $a \in \mathbb{R}^v$ consists of v constraint values, and $\delta \in \mathbb{R}^v$ defines the equality tolerance for each constraint. Thus, the goal of the learning is to identify $a^* := \operatorname{argmax}_{a \in \mathbb{R}^v} U(a)$, where $U : \mathbb{R}^v \to \mathbb{R}$ is the underlying utility function. In our work, we construct the components of a to be:

1) average forward velocity of the torso (m/s)
2) phase variable value at which to enforce minimum foot clearance, τ_c
3) minimum nonstance foot clearance enforced at τ_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),

which are defined over the search space of possible parameter combinations $\mathbf{A}$, a discretization of $\mathbb{R}^v$, as given in Table I.

D. Benefits of Preference-Based Learning

The traditional hand-tuning process requires a human operator to make assumptions about the underlying utility function U, which is difficult given the following: the non-intuitive relationship between parameter combinations and the resulting experimental behavior; and the need to account for numerous factors including stability, robustness to perturbations/model uncertainty, and visual appearance. Additionally, U admits no obvious mathematical description; eliminating the use of reward-based tuning methods.

Alternatively, we propose the use of preference-based learning to identify a^* using only pairwise preferences, which take advantage of a human's natural ability to combine many factors into a single judgment of "better" or "worse". Although this requires the human to provide feedback, there are two major benefits of our approach: 1) the duration of the tuning process is reduced significantly compared to hand-tuning; and 2) pairwise preferences are much easier for a naïve user to provide compared to manually navigating the complex search space of parameter combinations.

TABLE I
ESSENTIAL CONSTRAINT ACTION SPACE

Essential Constraint	Bounds $[a_{\min}, a_{\max}]$	Disc. d
Average Forward Velocity (m/s)	[0.3, 0.6]	0.05
Clearance Tau $(\cdot)$	[0.4, 0.7]	0.1
Minimum Foot Clearance (m)	[0.05, 0.19]	0.02
Impact Velocity (m/s)	[−0.8, −0.2]	0.1
Step Length (m)	[0.2, 0.4]	0.05

Algorithm 1 LINECOSPARNLP

1: Construct $\mathbf{A}$ using $a_{\min}$, $a_{\max}$, and d
2: Initialize datasets $\{\mathbf{D}_0, \mathbf{E}_0 = \emptyset\}$
3: **for all** $i = 1, \ldots, N$ **do**
4:     **if** $i == 1$ **then**
5:         Obtain $\boldsymbol{a}_1 = \{a_1^1, \ldots., a_1^n\}$ as uniform-random
6:     **else**
7:         Generate $\mathbf{L}_i :=$ random line through a_{i-1}^*
8:         Construct subset $\mathbf{S}_i = \mathbf{L}_i \cup \mathbf{E}_{i-1}$
9:         Approximate $\mathcal{P}(\boldsymbol{U}_{\mathbf{S}_i} | \mathbf{D}_{i-1})$ as $\mathcal{N}(\mu_{\mathbf{S}_i}, \Sigma_{\mathbf{S}_i})$
10:        Draw $k = 1, \ldots, n$ samples: $f^k \sim \mathcal{N}(\mu_{\mathbf{S}_i}, \Sigma_{\mathbf{S}_i})$
11:        Obtain $\boldsymbol{a}_i = \{a_i^k = \operatorname*{argmax}_{a \in \mathbf{S}_i} f^k(a) | k = 1, \ldots n\}$
12:     **end if**
13:     Execute outputs of NLP for $\boldsymbol{a}_i$ on the system
14:     Append executed actions: $\mathbf{E}_i = \mathbf{E}_{i-1} \cup \boldsymbol{a}_i$
15:     Query operator for preference feedback $\boldsymbol{p}_i$
16:     Append preference feedback: $\mathbf{D}_i = \mathbf{D}_{i-1} \cup \boldsymbol{p}_i$
17:     Approximate $\mathcal{P}(\boldsymbol{U}_{\mathbf{E}_i} | \mathbf{D}_i)$ as $\mathcal{N}(\mu_{\mathbf{E}_i}, \Sigma_{\mathbf{E}_i})$
18:     Update $a_i^* = \operatorname*{argmax}_{a \in \mathbf{E}_i} \mu_{\mathbf{E}_i}(a)$
19: **end for**

III. LEARNING FRAMEWORK

To learn the optimal action a^* in as few iterations as possible, we introduce a framework built around a high-dimensional preference-based learning algorithm LINECOSPAR [27] that learns a Bayesian posterior over the utility function U. The new framework, LINECOSPARNLP, still relies on pairwise preferences obtained from a human observing the experimental behavior of the robot, but embeds the learning directly into an HZD optimization problem, eliminating the need for a pre-computed gait library. We will first present LINECOSPARNLP, and then explicitly discuss the differences between the two frameworks.

A. The LINECOSPARNLP Algorithm

The procedure of the LINECOSPARNLP algorithm is shown in Alg. 1. First, to set up the learning problem, upper and lower bounds on $a \in \mathbb{R}^v$ along with the granularity of discretization $d \in \mathbb{R}_+^v$ are chosen by the operator. This leads to the discrete search space $\mathbf{A}$ with $|\mathbf{A}| = \prod d$. The corresponding set of utilities is defined as $U : \mathbf{A} \to \mathbb{R}$, with U_B used to denote the restriction of U on $B \subset \mathbf{A}$.

Each iteration i of the algorithm is as follows. First, n actions, denoted as the set $\boldsymbol{a}_i := \{a_i^1, \ldots, a_i^n\} \in \mathbb{R}^{v \times n}$, must be selected to give to the NLP. The parameter n can be changed depending on how many actions the operator would like to sample in each iteration. Because the actions are compared in pairs, n actions equates to $m = \binom{n}{2}$ pairwise preferences. In the first iteration, $\boldsymbol{a}_1$ is constructed using uniform-random actions. During every subsequent iteration, the algorithm utilizes a Self-Sparring approach [33]

to Thompson sampling which is a sample-efficient sampling method for regret-minimization. In general, to select n actions, Thompson sampling works by drawing n samples from a given distribution, such as the normal distribution $\mathcal{N}(\mu_B, \Sigma_B)$ over actions $a \in B \subset \mathbf{A}$:

$$f^k \sim \mathcal{N}(\mu_B, \Sigma_B) \quad \forall k = 1, \dots, n, \qquad (7)$$

and selecting the actions that maximize the samples:

$$a_i^k = \operatorname*{argmax}_{a \in B} f^k(a) \quad \forall k = 1, \dots, n. \qquad (8)$$

To be computationally tractable, LINECOSPARNLP performs Thompson sampling only considering the subset of actions $\mathbf{S}_i \subset \mathbf{A}$. This subset is defined as $\mathbf{S}_i := \mathbf{L}_i \cup \mathbf{E}_{i-1}$, with $\mathbf{E}_{i-1}$ being the dataset of previously executed actions and $\mathbf{L}_i \subset \mathbf{A}$ being a random linear subspace which intersects the best action from the previous iteration, a_{i-1}^*. Using this subset, Thompson sampling draws n samples from the posterior distribution $\mathcal{P}(\boldsymbol{U}_{\mathbf{S}_i} | \mathbf{D}_{i-1})$, where $\mathbf{D}_{i-1}$ is the preference dataset from the previous iteration. The posterior is modeled as proportional to the product of the preference likelihood and the Gaussian prior [34]:

$$\mathcal{P}(\boldsymbol{U}_{\mathbf{S}_i} | \mathbf{D}_{i-1}) \propto \mathcal{P}(\mathbf{D}_{i-1} | \boldsymbol{U}_{\mathbf{S}_i}) \mathcal{P}(\boldsymbol{U}_{\mathbf{S}_i}). \qquad (9)$$

The Gaussian process prior is computed as:

$$\mathcal{P}(\boldsymbol{U}_{\mathbf{S}_i}) = \frac{\exp\left(-\frac{1}{2} \boldsymbol{U}_{\mathbf{S}_i} (\Sigma_i^{\text{pr}})^{-1} \boldsymbol{U}_{\mathbf{S}_i}\right)}{(2\pi)^{\frac{|\mathbf{S}_i|}{2}} |\Sigma_i^{\text{pr}}|^{1/2}}, \qquad (10)$$

where $\Sigma_i^{\text{pr}} \in \mathbb{R}^{|\mathbf{S}_i| \times |\mathbf{S}_i|}$ with $[\Sigma_i^{\text{pr}}]_{j,k} = \mathcal{K}(a_{\mathbf{S}_i}^j, a_{\mathbf{S}_i}^k)$ for the set of actions $a_{\mathbf{S}_i}$ in $\mathbf{S}_i$, and $\mathcal{K}$ being a kernel of choice (taken as a squared exponential kernel in this work). The preference likelihood function is computed as:

$$\mathcal{P}(\mathbf{D}_{i-1} | \boldsymbol{U}_{\mathbf{S}_i}) = \prod_{j=1}^{i-1} \prod_{k=1}^{n} g\left(\frac{U(a_j^k) - U(a_j^k)}{c_p}\right), \qquad (11)$$

where $g : \mathbb{R} \to (0, 1)$ is a monotonically-increasing activation function, and $c_p > 0$ models the expected noisiness of the preference feedback. In this work, we select $g(x) := \frac{1}{1+e^{-x}}$ to be the heavy-tailed sigmoid function because it was empirically found to improve performance [27].

Equipped with (10) and (11), the posterior (9) can then be estimated via the Laplace approximation as in [34] which yields a multivariate Gaussian, $\mathcal{N}(\mu_{\mathbf{S}_i}, \Sigma_{\mathbf{S}_i})$. Finally, applying this distribution to (7) and (8) yields $\boldsymbol{a}_i$. These sampled actions are then given to the NLP, whereby corresponding gaits are generated, the outputs are executed on the robot, and $\boldsymbol{a}_i$ is appended to $\mathbf{E}_i$. We define the set of actions executed on hardware up to and including those sampled in iteration i as $\mathbf{E}_i := \{\boldsymbol{a}_1, \dots, \boldsymbol{a}_i\} \in \mathbb{R}^{v \times n \times i} \subset \mathbf{A}$.

After demonstrating the gaits on hardware, the human operator is queried for m pairwise preferences, denoted as $\boldsymbol{p}_i = \{p_i^1, \dots, p_i^m\} \in \mathbb{R}^m$. The collection of all preference feedback up to and including iteration i is denoted $\mathbf{D}_i := \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_i\} \in \mathbb{R}^{m \times i}$. Note that it is possible for $\boldsymbol{p}_i = \emptyset$ when all sampled actions do not converge, or when the user chooses to give feedback of "no preference".
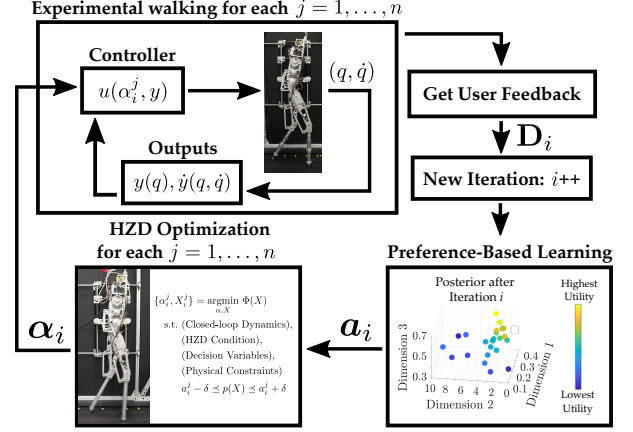

Fig. 2. The experimental procedure is illustrated in terms of each iteration i with n denoting the number of gaits compared in each iteration. The experiments presented in this work used $n = 2$. Using this notation, the set of n actions given to the HZD optimization is denoted: $\boldsymbol{a}_i = \{a_i^1, \dots, a_i^n\}$. The resulting n sets of Bézier coefficients given to the controller are denoted $\boldsymbol{\alpha}_i = \{\alpha_i^1, \dots, \alpha_i^n\}$.

Lastly, the algorithm updates its belief of a^* by modeling the posterior again using $\mathbf{D}_i$. Since obtaining the posterior over the entire search space $\mathbf{A}$ for high-dimensional action spaces has been shown to be computationally intractable [27], the posterior is only updated over $\mathbf{E}_i$:

$$\mathcal{P}(\boldsymbol{U}_{\mathbf{E}_i} | \mathbf{D}_i) \propto \mathcal{P}(\mathbf{D}_i | \boldsymbol{U}_{\mathbf{E}_i}) \mathcal{P}(\boldsymbol{U}_{\mathbf{E}_i}), \qquad (12)$$

which is approximated using the same procedure as for $\mathcal{P}(\boldsymbol{U}_{\mathbf{S}_i} | \mathbf{D}_{i-1})$ and applying the Laplace approximation to obtain the distribution $\mathcal{N}(\mu_{\mathbf{E}_i}, \Sigma_{\mathbf{E}_i})$. The algorithm's belief of the optimal action after iteration i is finally updated as:

$$a_i^* = \operatorname*{argmax}_{a \in \mathbf{E}_i} \mu_{\mathbf{E}_i}(a).$$

B. Changes to LINECOSPAR for use with a NLP

Three notable changes were made to the algorithm LINECOSPARNLP in comparison to LINECOSPAR. First, the LINECOSPARNLP selects $\mathbf{L}_i$ to intersect a_{i-1}^* as opposed to a_{i-2}^* which leverages more recent preference feedback. This change requires two posterior updates in each iteration but results in fewer required iterations. 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 $n > 1$ 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, otherwise known as user suggestions, is also added to the dataset $\mathbf{D}_i$ to improve sample-efficiency. However, these suggestions rely on understanding the mapping between a and $U(a)$; because this mapping is rarely well-understood for parameters of a nonlinear optimization problem, LINECOSPARNLP does not utilize coactive feedback.

IV. LEARNING TO WALK IN EXPERIMENTS

We experimentally deploy LINECOSPARNLP (open-source code: [35]) to tune the 5 essential constraints outlined in Table I on the planar bipedal robot, AMBER-3M [36]. This custom research platform has three interchangeable lower-limb configurations: flat-foot, point-foot, and spring-foot. We specifically selected this platform because of its engineering reliability [37], 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 motor drivers. The motor driver communication and the control logic run at ∼1kHz, each on a separate core.

A. Experimental Procedure

In the experiments, walking gaits are generated by the HZD-based 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 5th-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:

$$MCOT = \int_{t_0}^{t_f} \frac{P(t)}{mgv} dt, \qquad (13)$$

where $P(t) = \sum_{i=1}^{4} |u_i(t)\dot{q}_i^a(t)|$ is the 2-norm sum of power.

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 we empirically found that operators sometimes had difficulty remembering the details of more than two gaits at a time, leading to the most reliable preference feedback when $n = 2$. 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 static double-support configuration, starting the treadmill, and attempting to push the robot into the designed periodic orbit. If the resultant dynamics were not stable, 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, a preference was 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 disturbance
- Does not exhibit harsh noise (e.g. during impact)
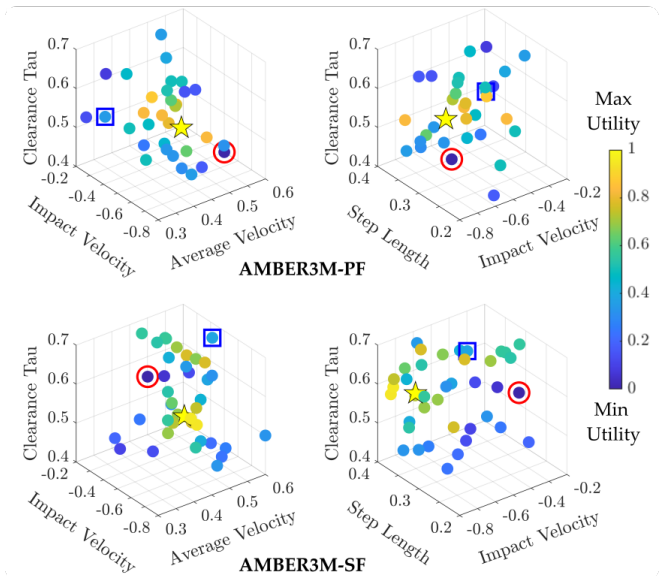- Is visually appealing (intuitive judgment from operator)

Fig. 3. The final obtained utilities for the 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 AMBER3M-PF and [0.4105, 0.5930, 0.0833, −0.7020, 0.3504] for AMBER3M-SF). The other two actions depicted in Fig. 4 are denoted with a red circle (worst gait) and a blue square (middle gait).

B. Procedure specific to AMBER3M-PF and AMBER3M-SF

In this work, we leverage two configurations of the robot: 1) the point-foot configuration, AMBER3M-PF (1.373 m, 21.3 kg); and 2) the spring-foot configuration, AMBER3M-SF (1.430 m, 23.5 kg) [36]. We first demonstrate the learning framework on AMBER3M-PF, with the corresponding rigid point-foot model used in the gait generation. To emphasize the scalability of our method, we repeat the exact procedure applied to AMBER3M-PF on AMBER3M-SF, but intentionally do not account for changes in the robot model and instead still generate gaits assuming the rigid-body model. Furthermore, we execute the gaits on hardware using the same controller with unmodified gains. Historically, robots with compliance are difficult to generate gaits for because of the resulting complexities which include: increased degrees of freedom of the system; the addition of a double support domain to the hybrid dynamics; and increased stiffness of the dynamics. Past success with compliant bipeds has relied on sophisticated models [38]. Therefore, the fact that our method yields stable walking despite the unmodeled compliance highlights it's effectiveness.

C. Results

A summary of the experimental results is illustrated in the supplementary video [39], with additional videos and material available at [40], and the final obtained posterior provided with the framework code in the repository [35].

The experiment with AMBER3M-PF was run for 30 iterations and sampled 27 unique gaits. The final posterior over the 27 executed actions is illustrated in the top row of Fig. 3. Since gaits quickly met the first criterion of being able to walk, preferences were mainly dictated based on the robustness and appearance of the experimental walking. The
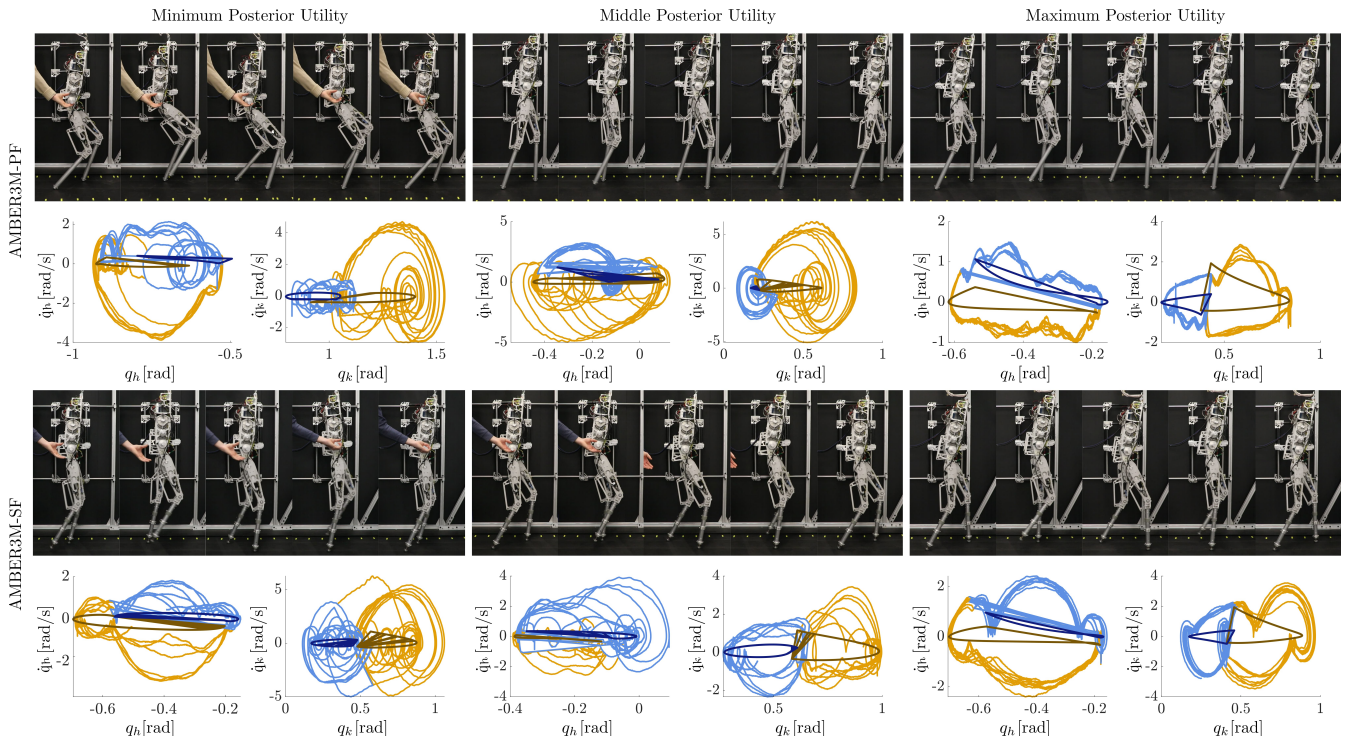
Fig. 4. Gait tiles with increasing posterior utility values from left to right are shown for the the rigid model (top) and spring model (bottom). The phase portraits of the hip (q_h) and knee (q_k) of the stance leg (blue) and swing leg (yellow) are shown below each corresponding gait, plotted over 10 seconds of data. The phase portraits clearly indicate that for both AMBER3M-PF and AMBER3M-SF the gaits evolved to be more experimentally robust.

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 Fig. 4 which illustrates the gaits corresponding to the minimum, a middle, and the maximum posterior utility; the iterations corresponding to when these gaits were first sampled is 1, 21, and 26, respectively. In Fig. 4, we note significantly lower velocity overshoot for all of the limbs and tighter tracking shown in the phase portraits for the gaits with higher posterior utility. It is also interesting to note the framework's success at improving the efficiency of the experimental walking: a latent property which is discernible to the human operator even though it is not immediately measured. This improvement is demonstrated by the MCOT values of the three gaits in Fig. 4: 0.74, 0.95, and 0.26 respectively.

When the procedure was repeated on AMBER3M-SF, many of the initial gaits were unable to walk due to the unmodeled compliance. Thus, gaits exhibiting periodic walking 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. Again, 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 Fig. 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 high-dimensional preference-based learning framework, LINECOSPARNLP (open-source code: [35]), specifically designed for use towards HZD-based gait generation. LINECOSPARNLP incorporates preference-based learning with an HZD optimization problem 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 the proposed 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 within 50 experimental iterations, with no requirement for simulation. Furthermore, LINECOSPARNLP achieves robust walking with unmodeled compliant legs, a challenging control task which historically relied on sophisticated models.

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 [41]. 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 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, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 876–891, 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] J. Reher, W.-L. Ma, and A. D. Ames, "Dynamic walking with compliance on a cassie bipedal robot," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 2589–2595.

[12] J. P. Reher, A. Hereid, S. Kolathaya, C. M. Hubicki, and A. D. Ames, "Algorithmic foundations of realizing multi-contact locomotion on the humanoid robot durus," in *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 400–415.

[13] 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.

[14] W.-L. Ma, K. A. Hamed, and A. D. Ames, "First steps towards full model based motion planning and control of quadrupeds: A hybrid zero dynamics approach," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5498–5503.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

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

[26] 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.

[27] 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," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

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

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

[35] M. Tucker, "Repository for LineCoSparNLP with learning results," https://github.com/maegant/ICRA2021-LearningHZD.

[36] 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.

[37] 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.

[38] 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.

[39] "Video of the experimental results." https://youtu.be/rLJ-m65F6C4.

[40] "Supplementary website featuring full-length experimental videos." https://maegant.github.io/ICRA2021-LearningHZD/.

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