2024 IEEE International Conference on Robotics and Automation (ICRA)
May 13-17, 2024. Yokohama, Japan

Walking-by-Logic: Signal Temporal Logic-Guided Model Predictive Control for Bipedal Locomotion Resilient to External Perturbations

Zhaoyuan Gu, Rongming Guo, William Yates, Yipu Chen, Yuntian Zhao, and Ye Zhao

2024 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-8457-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICRA57147.2024.10610811

***Abstract*— This study proposes a novel planning framework based on a model predictive control formulation that incorporates signal temporal logic (STL) specifications for task completion guarantees and robustness quantification. This marks the first-ever study to apply STL-guided trajectory optimization for bipedal locomotion push recovery, where the robot experiences unexpected disturbances. Existing recovery strategies often struggle with complex task logic reasoning and locomotion robustness evaluation, making them susceptible to failures due to inappropriate recovery strategies or insufficient robustness. To address this issue, the STL-guided framework generates optimal and safe recovery trajectories that simultaneously satisfy the task specification and maximize the locomotion robustness. Our framework outperforms a state-of-the-art locomotion controller in a high-fidelity dynamic simulation, especially in scenarios involving crossed-leg maneuvers. Furthermore, it demonstrates versatility in tasks such as locomotion on stepping stones, where the robot must select from a set of disjointed footholds to maneuver successfully.**

I. Introduction

This study investigates signal temporal logic (STL) based formal methods for robust bipedal locomotion, with a specific focus on circumstances where a robot encounters environmental perturbations at unforeseen times.

Robust bipedal locomotion has been a long-standing challenge in the field of robotics. Existing studies have demonstrated impressive performance through the reactive regulation of angular momentum [1], [2] or the predictive control of foot placement [3], [4]. Diverging from these approaches, our research aims to provide formal guarantees on a robot's ability to recover from perturbations via temporal-logic-based formal control methods. To achieve this, our research centers around designing formal requirements (i.e., task specifications) for bipedal locomotion push recovery, and employing trajectory optimization (TO) that guarantees system robustness.

Formal methods for bipedal systems have gained significant attention in recent years [5], [6]. The prevailing approach in existing works often relies on abstraction-based methods such as linear temporal logic (LTL) [7] with relatively simple verification processes, which abstract complex continuous behaviors into discrete events and low-dimensional states. However, challenges arise when addressing continuous, high-dimensional systems like bipedal

The authors are with the Laboratory for Intelligent Decision and Autonomous Robots, Woodruff School of Mechanical Engineering, Georgia Institute of Technology. {`zgu78, rguo61, wyates7, ychen3302, yzhao801, yezhao`}`@gatech.edu`

This work was funded by the Office of Naval Research Award #N000142312223, NSF grants #IIS-1924978, #CMMI-2144309, #FRR-2328254.

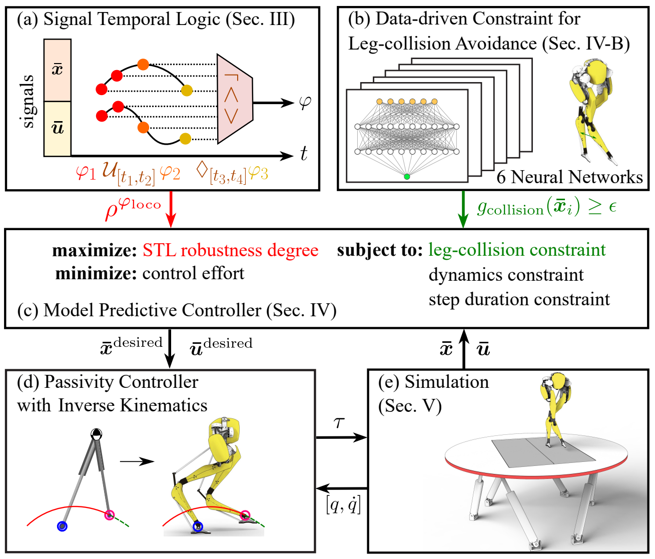
(a) Signal Temporal Logic (Sec. III)
signals
$\bar{x}$
$\bar{u}$
$\neg$ $\wedge$ $\vee$
φ
t
$\varphi_1 \, \mathcal{U}_{[t_1,t_2]} \varphi_2 \quad \lozenge_{[t_3,t_4]} \varphi_3$
(b) Data-driven Constraint for Leg-collision Avoidance (Sec. IV-B)
6 Neural Networks
$\rho^{\varphi_{\text{loco}}}$
$g_{\text{collision}}(\bar{x}_i) \geq \epsilon$
maximize: STL robustness degree
minimize: control effort
subject to: leg-collision constraint
dynamics constraint
step duration constraint
(c) Model Predictive Controller (Sec. IV)
$\bar{x}^{\text{desired}}$ $\bar{u}^{\text{desired}}$
$\bar{x}$ $\bar{u}$
(d) Passivity Controller with Inverse Kinematics
τ
$[q, \dot{q}]$
(e) Simulation (Sec. V)

Fig. 1: Block diagram of the proposed framework. (a) The signal temporal logic specification φ^{loco} specifies the locomotion task. (b) A set of data-driven kinematic constraints enforce the leg self-collision avoidance. (c) The model predictive control-based trajectory optimization solves a stable locomotion trajectory. (d) A whole-body controller tracks the desired trajectory. (e) Perturbed walking experiments on our bipedal robot Cassie.

robots. As a distinguished formal logic, STL [8] offers mathematical guarantees of specifications on dense-time, real-valued signals, making it suitable for reasoning about task logic correctness and quantifying robustness in complex robotic systems.

Self-collision avoidance is another crucial component for ensuring restabilization from disturbances, especially for scenarios involving crossed-leg maneuvers [3], [4], [9] where the distance between the robot's legs diminishes, as shown in Fig. 1(b). Several previous studies [2], [10] relied on inverted pendulum models to plan foot placements for recovery but often overlooked the risk of potential self-collisions during the execution of the foot placement plan. On the other hand, swing-leg trajectory planning that considers full-body kinematics and collision checking is prohibitively expensive for online computation.

In order to address these challenges, we design an optimization-based planning framework, illustrated in Fig. 1. As a core component of the model predictive controller (MPC) framework, we encode a series of STL specifications (e.g., stability and foot placements) as an objective function to enhance task satisfaction and locomotion robustness. Compared with traditional TO [11], [12] without formal specification encoding, our proposed STL-based TO has the capability of symbolic planning and reasoning to achieve more complex task requirements such as temporal sequencing order or timing constraints for task completion. Furthermore, this TO ensures safety against leg self-collision via a set of data-driven kinematic constraints. Solving the TO generates a reduced-order optimal plan that describes the center of mass (CoM) and swing-foot trajectories, including the walking-step durations. From this solved trajectory, a low-level controller derives a full-body motion through inverse kinematics and then uses a passivity-based technique for motion tracking. We summarize our core contributions as follows:

- This work represents the first-ever step towards incorporating STL-based formal methods into TO for dynamic legged locomotion. We design a series of STL task specifications that guide the planning of bipedal locomotion under perturbations.
- We propose a Riemannian robustness metric that evaluates the walking trajectory robustness based on reduced-order locomotion dynamics. The Riemannian robustness is seamlessly encoded as an STL specification and is therefore optimized in the TO for robust locomotion.
- We conduct extensive push recovery experiments with perturbations of varying magnitudes, directions, and timings. We compare the robustness of our framework with that of a foot placement controller baseline [2].

This work is distinct from our previous study [13] in the following aspects. (i) Instead of a hierarchical task and motion planning (TAMP) framework using abstraction-based LTL [13], this study employs an optimization-based MPC that integrates STL specifications to allow real-valued signals. This property eliminates the mismatch between high-level discrete action sequences and low-level continuous motion plans. (ii) The degree to which STL specifications are satisfied is quantifiable, enabling the MPC to provide a least-violating solution when the STL specification cannot be strictly satisfied. The LTL-based planner in [13], on the other hand, makes decisions only inside the robustness region, which is more vulnerable in real-system implementation.

II. Non-periodic Locomotion Modeling

A. Hybrid Reduced-Order Model for Bipedal Walking

We propose a new reduced-order model (ROM) that extends the traditional linear inverted pendulum model (LIPM) [14], [15]. The traditional LIPM features a point mass denoted as the center-of-mass (CoM), and a massless telescopic leg that maintains the CoM at a constant height. The LIPM has a system state $\boldsymbol{x} := [\boldsymbol{p}_{\text{CoM}}; \boldsymbol{v}_{\text{CoM}}]$, where $\boldsymbol{p}_{\text{CoM}} = [p_{\text{CoM},x};\ p_{\text{CoM},y};\ p_{\text{CoM},z}]$ and $\boldsymbol{v}_{\text{CoM}} = [v_{\text{CoM},x};\ v_{\text{CoM},y};\ v_{\text{CoM},z}]$ are the position and velocity of the CoM in the local stance-foot frame, as shown in Fig. 2(a). The LIPM dynamics are expressed as follows:

$$\begin{bmatrix}\ddot{p}_{\text{CoM},x}\\ \ddot{p}_{\text{CoM},y}\end{bmatrix} = \omega^2 \begin{bmatrix}p_{\text{CoM},x}\\ p_{\text{CoM},y}\end{bmatrix} \tag{1}$$

where $\omega = \sqrt{g/p_{\text{CoM},z}}$ and g is the acceleration due to gravity. The subscripts x and y indicate the sagittal and lateral components of a vector, respectively.

We design a variant of the traditional LIPM that additionally models the swing-foot position and velocity (Fig. 2(a)). In effect, the state vector is augmented as $\bar{\boldsymbol{x}} := [\boldsymbol{p}_{\text{CoM}}; \boldsymbol{v}_{\text{CoM}}; \boldsymbol{p}_{\text{swing}}], \boldsymbol{p}_{\text{swing}} \in \mathbb{R}^3$, and the control input $\bar{\boldsymbol{u}}$ sets the swing foot velocity $\dot{\boldsymbol{p}}_{\text{swing}}$. Moreover, we define $\boldsymbol{y} = [\bar{\boldsymbol{x}}; \bar{\boldsymbol{u}}] \in \mathbb{R}^{12}$ as the system output, which will be used in Sec. III for signal temporal logic (STL) definitions. Our addition of the swing-foot position $\boldsymbol{p}_{\text{swing}}$, together with $\boldsymbol{p}_{\text{CoM}}$, uniquely determines the leg configuration of the Cassie robot (e.g., via inverse kinematics), allowing us to plan a collision-free trajectory using only the ROM in Sec. IV-B. The augmented state is estimated from the joint encoder and IMU sensor in simulation.

At contact time, a *reset map* $\bar{\boldsymbol{x}}^+ = \bar{\Delta}_{j\to j+1}(\bar{\boldsymbol{x}}^-)$ uses the swing foot location to transition to the next walking step:

$$\begin{bmatrix}\boldsymbol{p}^+_{\text{CoM}}\\ \boldsymbol{v}^+_{\text{CoM}}\\ \boldsymbol{p}^+_{\text{swing}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{p}^-_{\text{CoM}} - \boldsymbol{p}^-_{\text{swing}}\\ \boldsymbol{v}^-_{\text{CoM}}\\ -\boldsymbol{p}^-_{\text{swing}}\end{bmatrix} \tag{2}$$

This occurs when the system state reaches the switching condition $\mathcal{S} := \{\bar{\boldsymbol{x}} | p_{\text{swing},z} = h_{\text{terrain}}\}$, where h_{terrain} is the terrain height. Note that the aforementioned position and velocity parameters are expressed in a local coordinate frame attached to the stance foot. The swing foot becomes the stance foot immediately after it touches the ground.

B. Keyframe-Based Non-Periodic Locomotion and Riemannian Robustness

To enable robust locomotion that adapts to unexpected perturbations or rough terrain, we employ the concept of *locomotion keyframe* [16]. A keyframe is a CoM apex state of a walking step. To quantify the robustness of a non-periodic walking step, we design a robust region centered around a nominal keyframe state in a Riemannian space. The Riemannian space [16] is a reparameterization of the Euclidean CoM phase space using tangent and cotangent locomotion manifolds, represented by a pair (σ, ζ). σ represents the tangent manifold along which the CoM dynamics evolve, while ζ represents the cotangent manifold orthogonal to σ. These manifolds can be derived analytically from the LIPM dynamics in (1); the detailed derivation is in [16]. Within the Riemannian space, we define a robust keyframe region that enables stable walking. This region is referred to as the Riemannian region.

Definition II.1 (Riemannian region)**.** *The Riemannian region $\mathcal{R}$ is the area centered around a nominal keyframe state $(\sigma_{\text{nom}}, \zeta_{\text{nom}})$: $\mathcal{R}_d := \{(p_{\text{CoM},d}, v_{\text{CoM},d}) \mid \sigma(p_{\text{CoM},d}, v_{\text{CoM},d}) \in \Sigma_d,\ \zeta(p_{\text{CoM},d}, v_{\text{CoM},d}) \in \mathrm{Z}_d\}$, where $d \in \{x, y\}$ indicates sagittal and lateral directions, respectively. $\Sigma_d = [\sigma_{\text{nom},d} - \delta\sigma_d, \sigma_{\text{nom},d} + \delta\sigma_d]$ and $\mathrm{Z}_d = [\zeta_{\text{nom},d} - \delta\zeta_d, \zeta_{\text{nom},d} + \delta\zeta_d]$ are the ranges of the manifold values for σ and ζ, where $\delta\sigma_d, \delta\zeta_d$ are robustness margins.*

The sagittal and lateral Riemannian regions in the phase space are illustrated in Fig. 2(b) as shaded areas. The bounds of these Riemannian regions are curved in the phase space

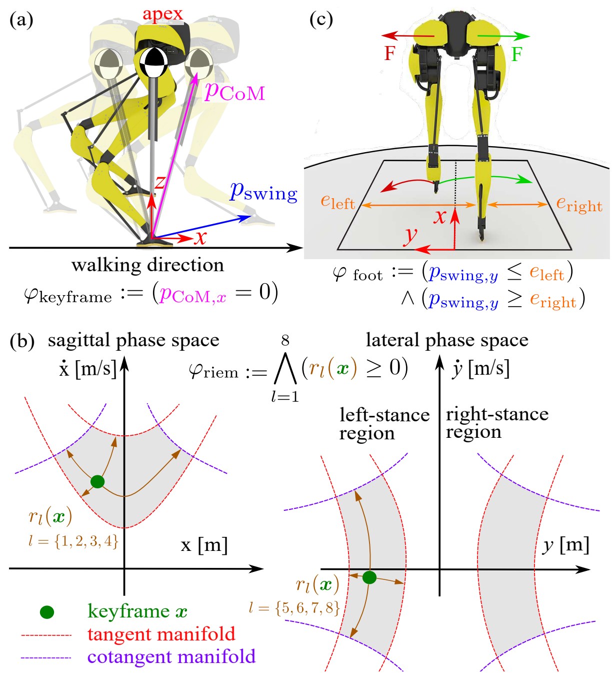

Fig. 2: Illustration of the locomotion specifications. (a) The highlighted state in the middle is the keyframe of a walking step. (b) The grey areas are the Riemannian regions in the sagittal and lateral phase spaces. The signed distances to the bounds of the Riemannian regions are indicated by the arrows. (c) Cassie's foot is specified to step inside the lateral bounds.

because they obey the LIPM locomotion dynamics. Notably, while two Riemannian regions exist in the lateral phase space, only one is active at any given time, corresponding with the stance leg labeled in Fig. 2(b).

Definition II.2 (Riemannian robustness)**.** *The Riemannian robustness ρ_{riem} is the minimum signed distance of an actual keyframe CoM state $\boldsymbol{x}$ to all the bounds of the Riemannian regions. Namely, $\rho_{\text{riem}} := \min_{l=1}^{8}(r_l(\boldsymbol{x}))$, where $r_l(\boldsymbol{x})$ is the signed distance to the l^{th} bound of the Riemannian regions, as illustrated in Fig. 2(b). We have a total of 8 bounds as the sagittal and lateral Riemannian regions each have 4 bounds.*

Riemannian robustness represents the locomotion robustness in the form of Riemannian regions. Any keyframe inside the Riemannian region has a positive robustness value, which indicates a stable walking step. In the next section, our goal is to leverage Riemannian robustness as an objective function and use STL-based optimization to plan robust trajectories for locomotion recovery.

III. Signal Temporal Logic and Task Specification for Locomotion

Signal temporal logic (STL) [17] uses logical symbols of negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$), as well as temporal operators such as eventually ($\Diamond$), always ($\Box$), and until ($\mathcal{U}$) to construct specifications. A specification is defined with the following syntax:

$$\begin{aligned} \varphi := \pi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \\ \Diamond_{[t_1,t_2]}\ \varphi \mid \Box_{[t_1,t_2]}\ \varphi \mid \varphi_1\ \mathcal{U}_{[t_1,t_2]}\ \varphi_2 \end{aligned} \tag{3}$$

where φ, φ_1, and φ_2 are STL specifications. $\pi := (\mu^\pi(\boldsymbol{y}) - c \geq 0)$ is a boolean predicate, where $\mu^\pi : \mathbb{R}^p \to \mathbb{R}$ is a vector-valued function, $c \in \mathbb{R}$, and the signal $\boldsymbol{y}(t) : \mathbb{R}_+ \to \mathbb{R}^p$ is a p-dimensional vector at time t. For a dynamical system, the signal $\boldsymbol{y}(t)$ is the system output (in our study, $\boldsymbol{y} = [\bar{\boldsymbol{x}}; \bar{\boldsymbol{u}}] \in \mathbb{R}^{12}$). The time bounds of an STL formula are denoted with t_1 and t_2, where $0 \leq t_1 \leq t_2 \leq t_{\text{end}}$ and t_{end} is the end of a planning horizon. The validity of an STL specification is inductively defined using the rules in Table I.

TABLE I
VALIDITY SEMANTICS OF SIGNAL TEMPORAL LOGIC

$(\boldsymbol{y},t) \models \pi$	$\Leftrightarrow$	$\mu^\pi(\boldsymbol{y}(t)) - c \geq 0$
$(\boldsymbol{y},t) \models \neg\varphi$	$\Leftrightarrow$	$(\boldsymbol{y},t) \not\models \varphi$
$(\boldsymbol{y},t) \models \varphi_1 \wedge \varphi_2$	$\Leftrightarrow$	$(\boldsymbol{y},t) \models \varphi_1 \wedge (\boldsymbol{y},t) \models \varphi_2$
$(\boldsymbol{y},t) \models \varphi_1 \vee \varphi_2$	$\Leftrightarrow$	$(\boldsymbol{y},t) \models \varphi_1 \vee (\boldsymbol{y},t) \models \varphi_2$
$(\boldsymbol{y},t) \models \Diamond_{[t_1,t_2]}\varphi$	$\Leftrightarrow$	$\exists t' \in [t+t_1, t+t_2], (\boldsymbol{y},t') \models \varphi$
$(\boldsymbol{y},t) \models \Box_{[t_1,t_2]}\varphi$	$\Leftrightarrow$	$\forall t' \in [t+t_1, t+t_2], (\boldsymbol{y},t') \models \varphi$
$(\boldsymbol{y},t) \models \varphi_1 \mathcal{U}_{[t_1,t_2]}\varphi_2$	$\Leftrightarrow$	$\exists t' \in [t+t_1, t+t_2], (\boldsymbol{y},t') \models \varphi_2 \wedge$ $\forall t'' \in [t+t_1, t'](\boldsymbol{y},t'') \models \varphi_1$

STL provides the capability of quantifying *robustness degree* [18] [19]. A positive robustness degree indicates specification satisfaction, and its magnitude represents the resilience to disturbances without violating this specification. When incorporated into trajectory optimization as a cost, the robustness degree allows for a minimally specification-violating trajectory if the task specification cannot be satisfied strictly [20]. Table II shows the semantics of the robustness degree.

TABLE II
ROBUSTNESS DEGREE SEMANTICS

$\rho^\pi(\boldsymbol{y},t)$	$=$	$\mu^\pi(\boldsymbol{y}(t)) - c$
$\rho^{\neg\varphi}(\boldsymbol{y},t)$	$=$	$-\rho^\varphi(\boldsymbol{y},t)$
$\rho^{\varphi_1\wedge\varphi_2}(\boldsymbol{y},t)$	$=$	$\min(\rho^{\varphi_1}(\boldsymbol{y},t), \rho^{\varphi_2}(\boldsymbol{y},t))$
$\rho^{\varphi_1\vee\varphi_2}(\boldsymbol{y},t)$	$=$	$\max(\rho^{\varphi_1}(\boldsymbol{y},t), \rho^{\varphi_2}(\boldsymbol{y},t))$
$\rho^{\Diamond_{[t_1,t_2]}\varphi}(\boldsymbol{y},t)$	$=$	$\max_{t' \in [t+t_1,t+t_2]}(\rho^\varphi(\boldsymbol{y},t'))$
$\rho^{\Box_{[t_1,t_2]}\varphi}(\boldsymbol{y},t)$	$=$	$\min_{t' \in [t+t_1,t+t_2]}(\rho^\varphi(\boldsymbol{y},t'))$
$\rho^{\varphi_1\mathcal{U}_{[t_1,t_2]}\varphi_2}(\boldsymbol{y},t)$	$=$	$\max_{t' \in [t+t_1,t+t_2]}(\min(\rho^{\varphi_2}(\boldsymbol{y},t'),$ $\min_{t'' \in [t+t_1,t']}(\rho^{\varphi_1}(\boldsymbol{y},t''))))$

The rest of this section introduces the locomotion specification φ_{loco}, designed to guarantee stable walking trajectories. We interpret locomotion stability as a *liveness* property in the sense that a keyframe with a positive Riemannian robustness will *eventually* occur in the planning horizon.

Keyframe specification $\varphi_{\text{keyframe}}$: To enforce properties on a keyframe, we first describe it using an STL formula $\varphi_{\text{keyframe}}$, checking whether or not a signal $\boldsymbol{y}$ is a keyframe. The keyframe occurs when the CoM is over the foot contact in the sagittal direction. Illustrated in Fig. 2(a), this definition is formally specified as $\varphi_{\text{keyframe}} := (\mu^\pi_{\text{CoM},x}(\boldsymbol{y}) = 0)$, where the predicate denotes the sagittal CoM position $\mu^\pi_{\text{CoM},x}(\boldsymbol{y}) = p_{\text{CoM},x}$.

Riemannian robustness φ_{riem}: A stable walking step has a keyframe with positive Riemannian robustness; i.e., the keyframe resides in the Riemannian region, as defined in Def. II.2. As shown in Fig. 2(b), we encode the Riemannian robustness specification φ_{riem} such that it is True when a CoM state $\boldsymbol{x}$ of a signal is inside the Riemannian region: $\varphi_{\text{riem}} := \bigwedge_{l=1}^{8}(r_l(\boldsymbol{x}) \geq 0)$, where $r_l(\boldsymbol{x})$ is the signed

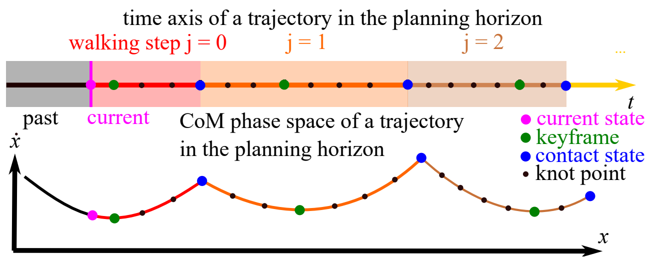

Fig. 3: The planning horizon starts from the current measured state (pink). An example of $N = 2$ walking steps and 8 knot points per walking step is illustrated for simplicity (our actual implementation has 10 knot points).

distance from $\boldsymbol{x}$ to the l^{th} bound of the Riemannian region in the Riemannian space.

Locomotion stability φ_{stable}: To encode this property using STL, we specify that the keyframe of the last walking step falls inside the corresponding Riemannian region. This stability property is encoded as $\varphi_{\text{stable}} := \Diamond_{[T^N_{\text{contact}}, T^{N+1}_{\text{contact}}]}(\varphi_{\text{keyframe}} \wedge \varphi_{\text{riem}})$, where T^N_{contact} and T^{N+1}_{contact} are the N^{th} and $N+1^{\text{th}}$ contact times and represent the time bounds of the last walking step in the planning horizon.

Swing foot bound φ_{foot}: For locomotion in a narrow space (e.g., a treadmill, as shown in Fig. 2(c)), we use a *safety* specification $\Box\varphi_{\text{foot}}$ to ensure the foothold lands inside of the treadmill's edges. The operator $\Box$ without a time bound means the specification should hold for the entire planning horizon. We define $\varphi_{\text{foot}} := (\mu^{\pi}_{\text{left}}(\boldsymbol{y}) \geq 0) \wedge (\mu^{\pi}_{\text{right}}(\boldsymbol{y}) \geq 0)$, where $\mu^{\pi}_{\text{left}} = -p_{\text{swing},y} + e_{\text{left}}$ and $\mu^{\pi}_{\text{right}} = p_{\text{swing},y} - e_{\text{right}}$ are the predicates for limiting the lateral foot location against the left edge e_{left} and the right edge e_{right} of the treadmill.

Overall locomotion specification φ_{loco}: The compounded locomotion specification is $\varphi_{\text{loco}} = \varphi_{\text{stable}} \wedge (\Box\varphi_{\text{foot}})$. Satisfying the specification φ_{loco} is equivalent to having a positive robustness degree: $(\boldsymbol{y}, t) \models \varphi_{\text{loco}} \Leftrightarrow \rho^{\varphi_{\text{loco}}}(\boldsymbol{y}, t) \geq 0$. In order to maximize the locomotion robustness, we use the robustness degree $\rho^{\varphi_{\text{loco}}}$ as an objective function in the trajectory optimization in the following section.

IV. Model Predictive Control for Push Recovery

A. Optimization Formulation

We design a model predictive controller (MPC) to solve a sequence of optimal states and controls (i.e., signals) that simultaneously satisfy specification φ_{loco}, system dynamics, and kinematic constraints within an N-step horizon.

The MPC functions as the primary motion planner of the framework and operates in both normal and perturbed locomotion conditions. Our MPC is formulated as the following nonlinear program:

$$\min_{\boldsymbol{X},\boldsymbol{U},\boldsymbol{T}} \quad w\mathcal{L}(\boldsymbol{U}) - \tilde{\rho}^{\varphi_{\text{loco}}}(\boldsymbol{X}, \boldsymbol{U}) \tag{4}$$

$$\text{s.t.} \quad \bar{\boldsymbol{x}}^j_{i+1} = f(\bar{\boldsymbol{x}}^j_i, \bar{\boldsymbol{u}}^j_i, T^j), \qquad i \in \mathbb{H} \setminus \mathbb{S}, \quad j \in \mathbb{J} \tag{5}$$

$$\bar{\boldsymbol{x}}^{+,j+1} = \bar{\Delta}_{j\rightarrow j+1}(\bar{\boldsymbol{x}}^{-,j}), \qquad j \in \mathbb{J} \tag{6}$$

$$g_{\text{collision}}(\bar{\boldsymbol{x}}_i) \geq \epsilon, \qquad i \in \mathbb{H} \tag{7}$$

$$g_{\text{duration}}(T^j) \geq 0, \qquad j \in \mathbb{J} \tag{8}$$

$$h_{\text{initial}}(\bar{\boldsymbol{x}}_0) = 0, h_{\text{transition}}(\bar{\boldsymbol{x}}_i) = 0, \qquad i \in \mathbb{S} \tag{9}$$

where $\mathbb{H}$ is a set of indices that includes all time steps in the horizon. We design $\mathbb{H}$ to span from the acquisition of the latest measured states till the end of the next N walking steps, with a total of M time steps. Fig. 3 illustrates a horizon with $N = 2$. $\mathbb{S}$ is the set of indices containing the time steps of all contact switch events, $\mathbb{S} \subset \mathbb{H}$. $\mathbb{J} = \{0, \ldots, N\}$ is the set of walking step indices. The decision variables include $\boldsymbol{X} = \{\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_M\}$, $\boldsymbol{U} = \{\bar{\boldsymbol{u}}_1, \ldots, \bar{\boldsymbol{u}}_M\}$, and $\boldsymbol{T} = \{T^0, \ldots, T^N\}$. $\boldsymbol{T}$ is a vector defining the individual step durations for all walking steps.

$\mathcal{L}(\boldsymbol{U}) = \sum_{i=1}^{M} ||\bar{\boldsymbol{u}}_i||^2$ is a cost function penalizing the control with a weight coefficient w. The robustness degree $\tilde{\rho}^{\varphi_{\text{loco}}}(\boldsymbol{X}, \boldsymbol{U})$ represents the degree of satisfaction of the signal $(\boldsymbol{X}, \boldsymbol{U})$ with respect to the locomotion specification φ_{loco}. $\tilde{\rho}^{\varphi_{\text{loco}}}$ is a smooth approximation of $\rho^{\varphi_{\text{loco}}}$ using smooth operators [21]. The exact, non-smooth version $\rho^{\varphi_{\text{loco}}}$ has discontinuous gradients, which can cause the optimization problem to be ill-conditioned. Maximizing $\tilde{\rho}^{\varphi_{\text{loco}}}(\boldsymbol{X}, \boldsymbol{U})$ encourages the keyframe towards the center of the Riemannian region, as discussed in Sec. III. The selection of w is a tradeoff between enhancing STL robustness and minimizing control effort. We choose a small $w = 0.01$ to promote the use of aggressive control for rapid disturbance recovery. Furthermore, we integrate $\tilde{\rho}^{\varphi_{\text{loco}}}$ as an objective function instead of a constraint to allow minimally-violating solutions and improve the TO-solving success rate in the presence of large perturbations.

To satisfy the LIPM dynamics (1) while adapting step durations $\boldsymbol{T}$, we use a second-order Taylor expansion to derive the approximated discrete dynamics (5). (6) represents the reset map (2) from the foot-ground contact switch. (7) represents a set of self-collision avoidance constraints, which ensures a collision-free swing-foot trajectory. The threshold ϵ is the minimum allowable distance for collision avoidance. The function $g_{\text{collision}}$ is a set of multilayer perceptrons (MLPs) learned from leg configuration data, as detailed in Sec. IV-B. (8) clamps step durations $\boldsymbol{T}$ within a feasible range. By allowing variations in step durations, we enhance the perturbation recovery capability of the bipedal system [22]. (9) are the equality constraints of the MPC: h_{initial} denotes the initial state constraint; $h_{\text{transition}}$ is the guard function posing kinematic constraints between the swing foot height and the terrain height, $p_{\text{swing},z} = h_{\text{terrain}}$, for walking step transitions at contact-switching indices in $\mathbb{S}$.

Upon the successful completion of an MPC optimization, the solution is immediately sent to the low-level passivity-based controller [23] for tracking and execution. The MPC then reinitializes the problem based on the latest state.

B. Data-Driven Self-Collision Avoidance Constraints

We design a set of multilayer perceptrons (MLPs) to incorporate rapid self-collision avoidance (SCA) into the MPC. Specifically, each MLP approximates the mapping from the reduced-order LIPM state to the collision distance between a particular geometry pair on Cassie that has high collision risk.

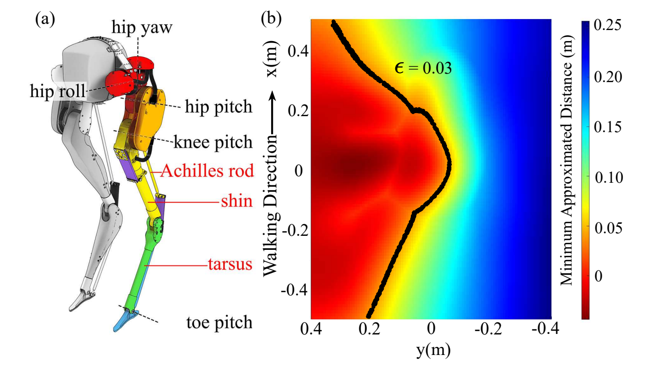

Fig. 4: (a) The robot kinematic anatomy for collision pair definitions. (b) The MLP prediction of the minimum distance between Cassie's two legs with the left foot affixed to $(0, 0)$ and the right foot moving in the xy plane.

According to Cassie's kinematic configuration depicted in Fig. 4(a), such collision-prone geometry pairs include: left shin to right shin (LSRS), left shin to right tarsus (LSRT), left shin to right Achilles rod (LSRA), left tarsus to right shin (LTRS), left tarsus to right tarsus (LTRT), and left Achilles rod to right shin (LARS). As a result, a total of 6 MLPs are trained and then encoded as constraints in the MPC to ensure collision-free trajectories.

A dataset with 10^6 entries is generated through extensive exploration of leg configurations. The collision distances of each configuration are automatically calculated by a built-in MATLAB function `checkCollision` [24] using a capsule collision model of Cassie. Each MLP consists of 2 hidden layers of 24 neurons. Upon back-propagation training in PyTorch [25], the MLPs achieve an accurate prediction with an average absolute error of 0.002 m, and an impressive evaluation speed of over 100 kHz, compared to 1 kHz using full-body inverse kinematics for collision checking.

We illustrate the effectiveness of the MLPs through kinematic analysis of the collision-free range of motion of Cassie's swing leg. Specifically, we consider a representative crossed-leg scenario where Cassie's left foot is designated as the stance foot and affixed directly beneath its pelvis. We move Cassie's right leg within the xy plane at the same height as the stance foot while recording the minimum value among all 6 MLP-approximated distances. The result is plotted as a heat map in Fig. 4(b), where the coordinate indicates the location of the right swing foot with respect to the pelvis. As expected, the plot reveals a trend of decreasing distance as the swing foot approaches the stance foot. A contour line drawn at $\epsilon = 0.03$ m indicates the MLP-enforced boundary between collision-free and collision-prone regions for foot placement. The collision-prone region to the left of the plane exhibits a cluster of red zones, each indicating a different active collision pair.

V. Results

A. Self-Collision Avoidance during Leg Crossing

We demonstrate the ability of the signal temporal logic-based model predictive controller (STL-MPC) to avoid leg collisions in a critical push recovery setting, where a perturbation forces the robot to execute a crossed-leg maneuver.

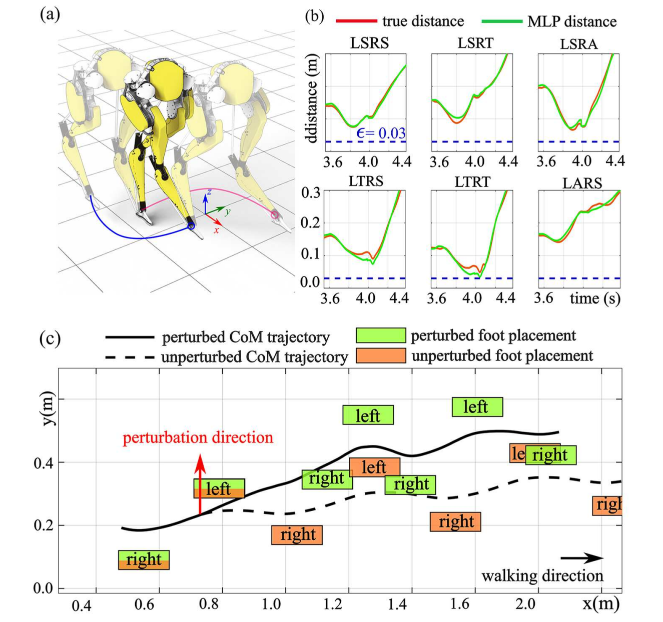

Fig. 5: (a) Snapshots of Cassie performing a crossed-leg maneuver for push recovery. (b) The MLP-approximated collision distances are accurate compared with the ground truth, and the planned leg trajectory is safe against the threshold $\epsilon = 0.03$ m. (c) An overhead view of the CoM trajectory and foot placements when a lateral perturbation induces a crossed-leg maneuver.

The MPC with collision constraints generates a trajectory shown in Fig. 5(a), where the swing leg adeptly maneuvers around the stance leg and lands at a safe crossed-leg recovery point. Similarly, the robot extricates itself from the crossed-leg state in the subsequent step, following a curved trajectory that actively avoids self-collisions. Fig. 5(b) shows that the multilayer perceptron (MLP)-approximated collision distances are accurate and that the planned trajectory is safe against the threshold ϵ. An overhead view comparing the perturbed and unperturbed trajectories is shown in Fig. 5(c).

B. Comprehensive, Omnidirectional Perturbation Recovery

We examine the robustness of the STL-MPC framework through an ensemble of push-recovery tests conducted in a high-fidelity Simulink simulation with virtual joint limits enforced and self-collision checked. For each experimental trial, a horizontal perturbation force is applied to Cassie's pelvis for a fixed duration of 0.1 s. Across the trials, the forces are systematically varied in 9 magnitudes evenly distributed between 80 N and 400 N; 12 directions evenly distributed between 0° and 330°; and 4 locomotion phases at a percentage s through a walking step, where s = 0%, 25%, 50%, 75%. Collectively, this experimental design encompasses a total of 432 distinct scenarios. For a baseline comparison, the same procedure is applied to an angular-momentum-based reactive controller (ALIP controller) [2].

In Fig. 6, we compare the maximum allowable force the STL-MPC can withstand to that of the baseline ALIP controller. The STL-MPC demonstrates superior perturbation recovery performance across the vast majority of directions and phases, as reflected by the blue region encompassing the red region. The improvement is particularly evident for

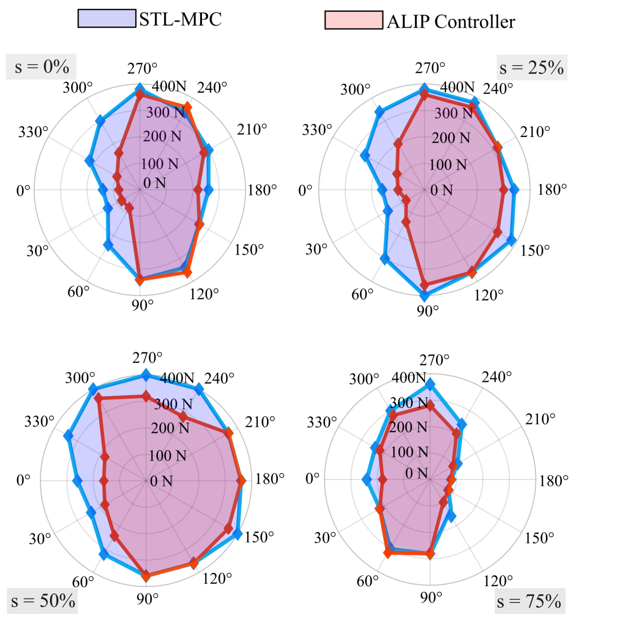

Fig. 6: The maximum allowable force exerted on the pelvis from which the robot can safely recover within two steps in all 12 directions. The perturbations happen at different phases s during a left leg stance. Values on the left half result in crossed-leg maneuvers, and values on the right half correspond to wide-step recoveries.

directions around $0°$, wherein crossed-leg maneuvers are induced for recovery, and active self-collision avoidance plays a critical role. This highlights the STL-MPC's capability to generate safe crossed-leg behaviors, thereby significantly enhancing its robustness against more challenging lateral perturbations. On the other hand, for perturbations between 180°, both frameworks exhibit comparable performance, generating wide side-steps for recovery. Note that we use $N = 2$ walking steps as the MPC horizon, as existing studies [26]–[28] indicate that a two-step motion is sufficient for recovery to a periodic orbit.

C. Stepping Stone Maneuvering

To demonstrate the STL-MPC's ability to handle a broad set of task specifications, we study locomotion in a stepping-stone scenario as shown in Fig. 7. To restrict the foot location to the stepping stones, we augment the locomotion specification φ_{loco} with an additional specification φ_{stones} that encodes stepping stone locations. For each rectangular stone, the presence of a stance foot $\boldsymbol{p}_{\text{stance}}$ inside its four edges is specified as $\varphi^o_{\text{stone}} = \bigwedge_{i=1}^4 (\mu^o_i(\boldsymbol{p}_{\text{stance}}) \geq 0)$, where $o \in \{1, \ldots, O\}$, $O \in \mathbb{Z}$ is the total number of stepping stones, and μ^o_i is the signed distance from the stance foot to the i^{th} edge of the o^{th} stone. Then the combined foot location specification for N walking steps is:

$$\varphi_{\text{stones}} = \bigwedge_{j=1}^{N} (\Box_{[T^j, T^j]} \bigvee_{o=1}^{O} \varphi^o_{\text{stone}})$$

The augmented specification is the compound of the original locomotion specification φ_{loco} and the newly-added stepping stone specification: $\varphi'_{\text{loco}} = \varphi_{\text{loco}} \wedge \varphi_{\text{stones}}$.

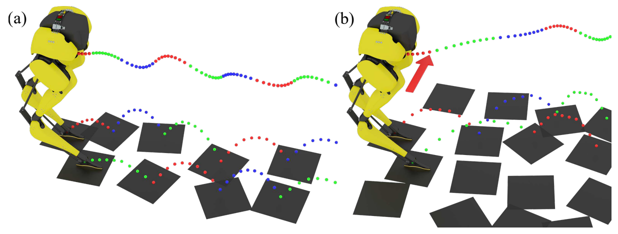

Fig. 7: Illustration of maneuvering over two stepping-stone scenarios. (a) STL-MPC solves dynamically feasible trajectories that satisfy an additional foot-on-stones specification. (b) STL-MPC successfully plans crossed-leg maneuvers to recover from perturbation.

We test STL-MPC using φ'_{loco} in two scenarios. The first scenario has stepping stones generated at ground level with random offsets and yaw rotations, as shown in Fig. 7(a). The STL-MPC advances Cassie forward successfully. In the second scenario, the STL-MPC demonstrates the ability to cross legs in response to a lateral perturbation in Fig. 7(b).

D. Computation Speed Comparison between Smooth Encoding Method and Mixed-Integer Program

To encode the robustness degree (as discussed in Sec. III) of STL specifications into our gradient-based trajectory optimization (TO) formulation, we adopt a smooth-operator method [29] that allows a smooth gradient for efficient computation. Specifically, we replace the non-smooth $\min$ and $\max$ operators in the robustness degree (as defined in Table II) with their smooth counterpart $\widetilde{\min}$ and $\widetilde{\max}$.

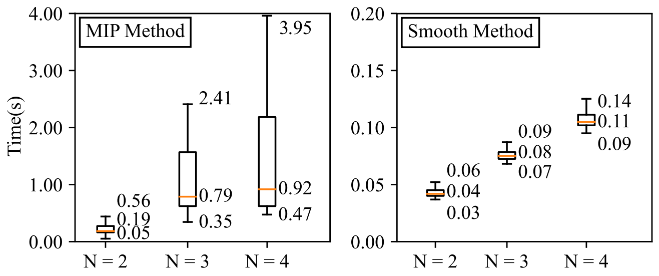

Fig. 8: A comparison of the traditional MIP method and our smooth method shows the planning time to solve trajectories for N-walking-step horizons. The smooth method is faster and more consistent over all horizons.

We benchmark the solving speed of the smooth method with the traditional mixed-integer programming (MIP) method [8]. The smooth method demonstrates a faster and more consistent solving speed, and its time consumption is nearer to linear with respect to the walking steps N.

VI. Conclusion

This study presents a model predictive controller using signal temporal logic (STL) for bipedal locomotion push recovery. Our main contribution is the design of STL specifications that quantify the locomotion robustness and guarantee stable walking. Our framework increased Cassie's impulse tolerance by 81% in critical crossed-leg scenarios. Further research will be focused on hardware verification and extensions to rough, dynamic terrain.

REFERENCES

[1] C. Khazoom and S. Kim, "Humanoid arm motion planning for improved disturbance recovery using model hierarchy predictive control," in *International Conference on Robotics and Automation*, 2022, pp. 6607–6613.

[2] Y. Gong and J. W. Grizzle, "One-step ahead prediction of angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-inspired controller," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 2832–2838.

[3] C. Khazoom, D. Gonzalez-Diaz, Y. Ding, and S. Kim, "Humanoid self-collision avoidance using whole-body control with control barrier functions," in *IEEE-RAS 21st International Conference on Humanoid Robots*, 2022, pp. 558–565.

[4] D. Marew, M. Lvovsky, S. Yu, S. Sessions, and D. Kim, "Riemannian motion policy for robust balance control in dynamic legged locomotion," 2023.

[5] S. Kulgod, W. Chen, J. Huang, Y. Zhao, and N. Atanasov, "Temporal logic guided locomotion planning and control in cluttered environments," in *American Control Conference*, 2020, pp. 5425–5432.

[6] J. Warnke, A. Shamsah, Y. Li, and Y. Zhao, "Towards safe locomotion navigation in partially observable environments with uneven terrain," in *IEEE Conference on Decision and Control*, 2020, pp. 958–965.

[7] H. Kress-Gazit, M. Lahijanian, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 211–236, 2018.

[8] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[9] R. Griffin, J. Foster, S. Fasano, B. Shrewsbury, and S. Bertrand, "Reachability aware capture regions with time adjustment and cross-over for step recovery," 2023.

[10] R. J. Griffin, G. Wiedebach, S. Bertrand, A. Leonessa, and J. Pratt, "Walking stabilization using step timing and location adjustment on the humanoid robot, atlas," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 667–673.

[11] S. Xin, R. Orsolino, and N. Tsagarakis, "Online relative footstep optimization for legged robots dynamic walking using discrete-time model predictive control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 513–520.

[12] F. M. Smaldone, N. Scianca, L. Lanari, and G. Oriolo, "Feasibility-driven step timing adaptation for robust mpc-based gait generation in humanoids," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1582–1589, 2021.

[13] Z. Gu, N. Boyd, and Y. Zhao, "Reactive locomotion decision-making and robust motion planning for real-time perturbation recovery," in *International Conference on Robotics and Automation*, 2022, pp. 1896–1902.

[14] 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 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2001, pp. 239–246 vol.1.

[15] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2003, pp. 1620–1626.

[16] Y. Zhao, B. R. Fernandez, and L. Sentis, "Robust optimal planning and control of non-periodic bipedal locomotion with a centroidal momentum model," *The International Journal of Robotics Research*, vol. 36, no. 11, pp. 1211–1242, 2017.

[17] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 152–166.

[18] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[19] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 115–140, 2019.

[20] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *53rd Annual Allerton Conference on Communication, Control, and Computing*, 2015, pp. 772–779.

[21] Y. Gilpin, V. Kurtz, and H. Lin, "A smooth robustness measure of signal temporal logic for symbolic control," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 241–246, 2021.

[22] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti, "Walking control based on step timing adaptation," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 629–643, 2020.

[23] H. Sadeghian, C. Ott, G. Garofalo, and G. Cheng, "Passivity-based control of underactuated biped robots within hybrid zero dynamics approach," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 4096–4101.

[24] T. M. Inc., "Robotic systems toolbox version: R2021b," Natick, Massachusetts, United States, 2022.

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[26] P. Zaytsev, S. J. Hasaneini, and A. Ruina, "Two steps is enough: No need to plan far ahead for walking balance," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 6295–6300.

[27] T. Koolen, T. de Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[28] J. Ding, C. Zhou, Z. Guo, X. Xiao, and N. Tsagarakis, "Versatile reactive bipedal locomotion planning through hierarchical optimization," in *International Conference on Robotics and Automation*, 2019, pp. 256–262.

[29] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *IEEE Conference on Control Technology and Applications*, 2017, pp. 1235–1240.