Robust-Locomotion-by-Logic: Perturbation-Resilient Bipedal Locomotion via Signal Temporal Logic Guided Model Predictive Control

Zhaoyuan Gu, Yuntian Zhao, Yipu Chen, Rongming Guo, Jennifer K. Leestma, Gregory S. Sawicki, and Ye Zhao

***Abstract*—This study introduces a robust planning framework that utilizes a model predictive control (MPC) approach, enhanced by incorporating signal temporal logic (STL) specifications. This marks the first-ever study to apply STL-guided trajectory optimization for bipedal locomotion, specifically designed to handle both translational and orientational perturbations. Existing recovery strategies often struggle with reasoning complex task logic and evaluating locomotion robustness systematically, making them susceptible to failures caused by inappropriate recovery strategies or lack of robustness. To address these issues, we design an analytical robustness metric for bipedal locomotion and quantify this metric using STL specifications, which guide the generation of recovery trajectories to achieve maximum locomotion robustness. To enable safe and computational-efficient crossed-leg maneuver, we design data-driven self-leg-collision constraints that are 1000 times faster than the traditional inverse-kinematics-based approach. Our framework outperforms a state-of-the-art locomotion controller, a standard MPC without STL, and a linear-temporal-logic-based planner in a high-fidelity dynamic simulation, especially in scenarios involving crossed-leg maneuvers. Additionally, the Cassie bipedal robot achieves robust performance under horizontal and orientational perturbations such as those observed in ship motions. These environments are validated in simulations and deployed on hardware. Furthermore, our proposed method demonstrates versatility on stepping stones and terrain-agnostic features on inclined terrains.**

***Index Terms*—Signal temporal logic, Trajectory Optimization, Bipedal locomotion, Push recovery, Robustness quantification.**

I. Introduction

BIPEDAL robots possess superior physical capabilities to perform agile maneuvers, offering great potential in various outdoor applications that often involve complex terrain or environmental perturbations [1]–[3]. Existing studies have demonstrated impressive locomotion performance through the reactive regulation of angular momentum [4], [5] or the predictive control of foot placement [6], [7]. 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 a signal temporal logic (STL)-based trajectory optimization (TO) to offer multifaceted formal guarantees.

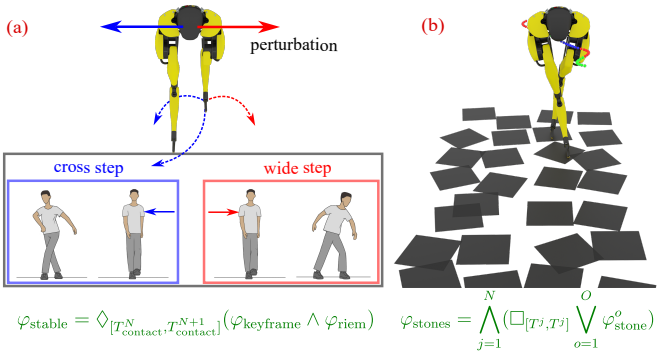

Fig. 1: Illustration of a bipedal walking robot synthesizing recovery maneuvers based on signal temporal logic specifications (green formulas) when subjected to unknown perturbations. (a) The robot takes wide steps or leg-crossing steps. (b) The robot selects stepping stones to traverse the challenging terrain.

In the domain of formal methods, STL [8], [9] is a mathematically precise language for defining specifications across various task objectives. Following these task specifications, a synthesized protocol ensures task completion by either providing a feasible plan or reporting infeasibility. Notably, STL admits *quantitative semantics* to assess the robustness of specification satisfaction. In this work, by integrating STL specifications into a TO, we solve optimal trajectories that ensure task completion with enhanced robustness against disturbances. As shown in Fig. 1, the STL-based TO achieves robust locomotion tasks under environmental perturbations by simultaneously (i) making decisions on the robot's actions (i.e., foot placements, center-of-mass apex state, and specific stepping stone to step on) and (ii) synthesizing corresponding continuous trajectories. To the best of the authors' knowledge, this work is the first study to leverage an STL-based TO for bipedal locomotion.

The proposed framework is shown in Fig. 2. As a core component of the framework, a model predictive controller (MPC) online executes the TO. In the TO, STL-based task specifications are encoded as an objective function to enhance task satisfaction and locomotion robustness. For bipedal locomotion, our task specifications are comprised of formally described objectives such as maintaining stability in a planning horizon and constraining foot placement regions. In addition to the STL specifications, the TO ensures safety against self-collision via a set of data-driven kinematic constraints. Solving the TO generates an optimal reduced-order plan that contains the center of mass (CoM) and swing-foot trajectories, including the optimized walking step durations. From these

Zhaoyuan Gu, Yuntian Zhao, Yipu Chen, Rongming Guo, and Ye Zhao are with the Laboratory for Intelligent Decision and Autonomous Robots, Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30313, USA.{zgu78, yzhao801, ychen3302, rguo61, yezhao}@gatech.edu

Jennifer K. Leestma and Gregory S. Sawicki are with the Physiology of Wearable Robotics Lab, Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30313, USA.{jleestma, gregory.sawicki}@gatech.edu

Corresponding author: Y. Zhao

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

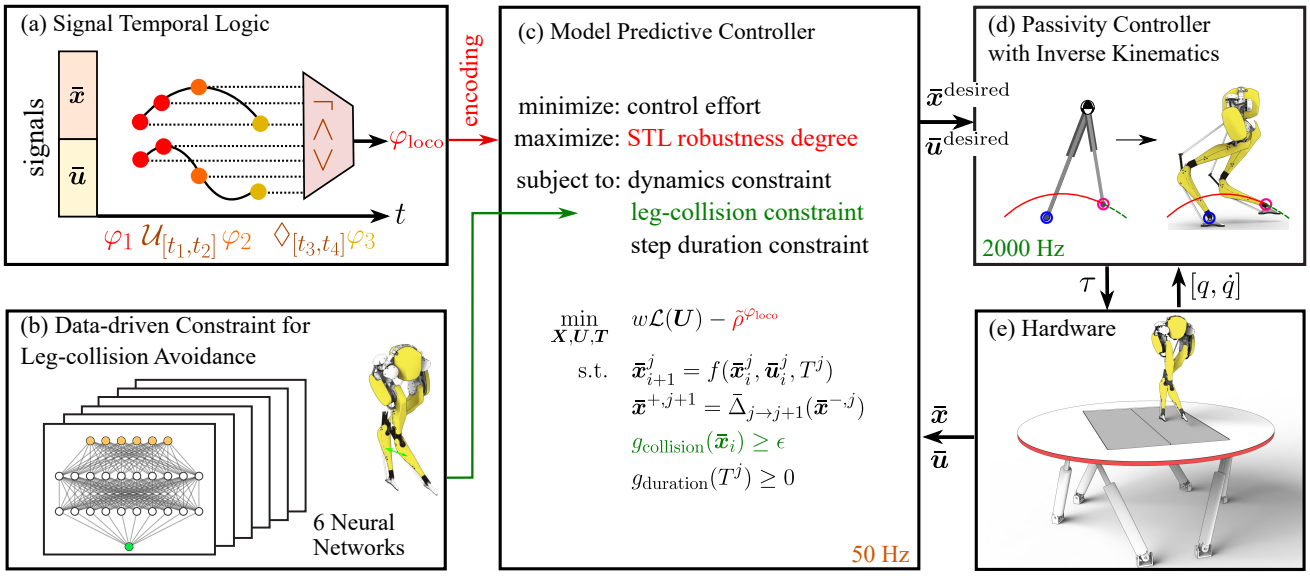

Fig. 2: Block diagram of the proposed STL-MPC framework. (a) The STL specification φ_{loco} specifies robust locomotion tasks such as recovery within two steps after perturbation. (b) A set of neuron-network-based constraints is trained with kinematics data to enforce the leg self-collision. (c) The proposed STL-MPC encodes the STL specification as a cost function and solves subsequent reduced-order keyframes at 50 Hz. (d) A whole-body passivity-based controller tracks the desired trajectory at 2000 Hz. (e) Hardware walking experiments on our bipedal robot Cassie.

trajectories, a full-body motion is derived through inverse kinematics and tracked by a passivity-based low-level controller. We thoroughly evaluate the proposed framework in simulations involving perturbations to both the robot's body and the terrain. Hardware experiments are conducted on a Computer-Aided Rehabilitation Environment (CAREN) [10] and a bump emulation (BumpEm) system [11].

This work is an evolved study from our previously published conference paper [12] with detailed mathematical formulation, extensive simulation, and hardware experiments. This work is also distinct from our previous study [13] in the following aspects. First, instead of a hierarchical task and motion planning (TAMP) framework using abstraction-based LTL [13], this study employs an integrated TO that encodes STL specifications. The LTL-based planner employs a hierarchy to decouple the discrete decision-making and continuous motion planning, which may induce an infeasibility issue when executing the high-level task plan at the low-level motion planner. STL stands out as it allows real-valued dense-time signals [14]. This property eliminates the mismatch between high-level discrete action sequences and low-level continuous motion plans. Specifically, the TO is solved in a MPC fashion with *continuous* foot trajectory updated at 50 Hz, whereas LTL makes decisions on *discrete* foot locations only once per walking step. This dense-time decision-making feature allows the TO to achieve reactive planning without the middle-level behavior trees proposed in [13], thus simplifying the LTL-based hierarchical framework.

Second, this proposed method integrates the STL task specification inside TO as an objective function, enabling the TO to provide a least-violating solution when the STL specification cannot be strictly satisfied. Conversely, the LTL-based planner in [13] enforces the specification satisfactions strictly and cannot handle large perturbations beyond a predefined bound.

Third, the proposed optimization adopts more accurate kinematics constraints and a faster online workflow to ensure leg self-collision avoidance. Instead of calculating the distances between a set of points on the robot [13], we approximate the leg geometry with capsules and train multi-layer perceptrons to capture the minimum distances between these capsule pairs.

We summarize our core contributions as follows:

- This work is the first study to incorporate STL-based formal methods into TO for legged locomotion. We design an STL task specification to achieve safe bipedal locomotion under perturbation.
- 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 optimized in the TO for robust locomotion.
- We design a rapid data-driven self-collision avoidance workflow to enable safe crossed-leg maneuvers. We integrate multi-layer perceptrons (MLPs) that approximate the distances between collision-prone body geometries as kinematic constraints in the TO.
- We conduct extensive experiments to demonstrate that our STL-based TO outperforms state-of-the-art methods from multiple perspectives: (i) Our framework utilizes crossed-leg maneuvers to achieve a more robust performance than a foot placement controller that uses an angular-momentum-based linear inverted pendulum (ALIP) [5]; (ii) Our STL-based TO outperforms the mixed-integer programming (MIP) encoding method in terms of computational speed; (iii) The STL-based TO shows a higher perturbation resistance capability compared with a standard MPC without STL and a linear-temporal-logic-based (LTL-based) planner [13]; (iv) Our framework exhibits remarkable generalizability across various challenging terrains, including stepping stones and dynamic moving surfaces with rotational perturbations.

This paper is organized as follows. Sec. II reviews related works on bipedal locomotion push recovery and formal methods. Sec. III introduces the system dynamics and the concept of keyframe-based locomotion. Sec. IV ourlines the fundamentals of STL. Sec. V presents our specification design for locomotion tasks, followed by the problem formulation of the proposed STL-based TO in Sec. VI. The experiment setup is detailed in Sec. VII. Simulation and hardware results are shown in Sec. VIII and Sec. IX, respectively. We discuss the limitation and future directions in Sec. X. Finally, we conclude the paper in Sec. XI.

II. Related Work

A. Planning and Control for Bipedal Push Recovery

Recovery from external perturbations has been a significant focus in bipedal robot locomotion [15], [16]. Various strategies such as hip, ankle, foot placement, and gait switching [17] have been explored to handle external perturbations [18], [19]. Notably, stepping strategies have shown superior performance in managing strong disturbances, by identifying robust step locations. For example, Englsberger et al. [20] develop a foot placement method based on the capture point that integrates center of mass (CoM) vertical motion and angular momentum. Feng et al. [21] apply differential dynamic programming for nominal trajectory design, complemented by a quadratic program (QP) to optimize the foot placement for tracking control. Xiong et al. [22] propose a method to modify the foot location, leveraging a hybrid reduced-order model (RoM) and its step-to-step dynamics. These varied stepping strategies highlight the importance of proactive planning and adaptive control in robust bipedal locomotion.

Our work focuses on the stepping strategy because the bipedal robot Cassie has limited centroidal momentum due to its small torso and relatively weak ankle actuation. Furthermore, our method has a significant emphasis on generating safe swing-leg trajectories and avoiding leg self-collisions in complex scenarios such as leg-crossing. Our approach is notably different from the work of Gibson et al. [23], where they formulate a model predictive controller (MPC) based on an angular momentum linear inverted pendulum (ALIP). Several major differences are worth noting. First, unlike ALIP-MPC, our method formally incorporates high-level task specifications. Second, our method allows for varying step durations that are better suited for disturbance recovery. Moreover, ALIP-MPC focuses on using terrain information to assist locomotion. Its performance is heavily dependent on the operator's ability to provide a real-time estimation of the terrain information. Our framework focuses on recovery from terrain perturbations. More importantly, our framework is terrain-perturbation-agnostic, meaning that it handles perturbed terrain without estimating environmental information. In this study, we will use ALIP-MPC as a baseline and demonstrate the enhanced blind walking performance of our approach.

While numerous studies have focused on the robustness of locomotion under *ad hoc* horizontal perturbations, such as ball hitting [24] or stick pushing [25], there is a notable research gap in understanding how systematic environmental characterizations, particularly omnidirectional perturbations, impact the performance of push recovery. [26], [27] has primarily explored the effects of terrain orientation and periodic height variations on locomotion stability. In this study, we explore omnidirectional perturbations, including orientational ones, using the Computer-Aided Rehabilitation Environment (CAREN) system [10] and comprehensively investigate the impact of these perturbations on locomotion robustness.

Formally quantifying robustness, defined as the system's tolerance to disturbances [28], is essential for formulating effective recovery strategies [29]. Previous research has largely focused on assessing the robustness based on a RoM, either by evaluating the deviation from a limit cycle [30] or a Poincaré map [31]. Built on top of a similar concept, our work extends the Riemannian robustness proposed by Zhao et al. [32], which quantifies the Riemannian distance between reduced-order trajectories within a CoM phase space. This approach is particularly advantageous, as the Riemannian distance is consistent with the inherent dynamics of inverted pendulum systems and offers a more intuitive metric for measuring the distance between two different CoM trajectories. By leveraging the concept of Riemannian robustness, we propose a novel robustness-aware trajectory optimization (TO) aimed at enhancing stability for bipedal push recovery.

B. Leg Self-Collision Avoidance

Push recovery methods of bipedal locomotion often employ RoMs [33], which do not fully capture the configurations and geometries of a robot's leg links. This limitation becomes critical in scenarios requiring self-collision avoidance. On the other hand, a computational challenge arises when taking into account full-body kinematic constraints online. To circumvent this challenge, heuristic constraints on foot placements, such as box constraints [21], [23], are commonly adopted. While these constraints simplify the computational process, they limit the range of collision-free motions and often rule out crossed-leg maneuvers that are physically feasible. This crossed-leg maneuver is an essential ingredient during highly dynamic locomotion or in extremely constrained environments such as stepping stones (see Fig. 1(b)).

To prevent self-leg-collision, Liu et al. [34] introduce a control framework that considers self-collision in the context of disturbances, but does not study advanced multi-step or non-periodic recovery strategies. Marew et al. [35] present a whole-body controller using Riemannian motion policies to avoid self-collisions and recover from disturbances using crossed-leg motions. Griffin et al. [25] adopt heuristic rules for selecting convex step regions to achieve crossed-leg motions. Khazoom et al. [6] propose a whole-body controller using control barrier functions (CBF) that prevent self-collision at the low-level tracking, but the controller can restrict the robot from reaching the desired step location due to the CBF constraints. To address the push recovery and self-collision avoidance problem simultaneously, we design neural-network-based constraints and integrate them into a TO. These constraints enable fast and accurate calculation of collision distances, which facilitate the implementation of our TO online in a MPC fashion.

C. Step Duration Adaptation

Step duration adaptation is gaining increasing attention in the locomotion community as it reveals the capability of improving the robustness of the stepping strategy [36], [37]. For instance, when a robot is perturbed towards a failure-prone state, a reduced step duration can rapidly reset the robot's state and stop an aggressive acceleration. However, identifying an optimal step duration presents a notable challenge because introducing step durations as decision variables in a TO typically results in a nonconvex problem. Consequently, step duration is often empirically specified in existing methods [23], [38]. However, such empirical methods limit the space of feasible solutions and often lead to conservative motions.

Recent advancements have focused on optimizing step duration by solving either a computationally expensive mixed-integer program (MIP) [39], [40] or a linear complementarity problem [41]. Alternative methods decouple the motion planning and duration adaptation, addressing them as two separate steps. For example, Griffin et al. [42] propose to first plan a swing-foot trajectory, and then adapt the duration of the planned trajectory separately. Insights from human data [43] indicate that the optimal recovery strategy alters durations for multiple walking steps.

To optimize the step duration over a multi-step horizon, numerous studies [44]–[46] manage to formulate and solve nonlinear programs (NLPs), despite their nonconvex property. [44] introduces hyperbolic step time variables in a nonlinear MPC (NMPC). However, the NMPC considers only discrete contact-switching instances, thus making it impossible to impose constraints during the continuous swing phase, such as self-collision avoidance constraints. [45] optimizes the time step intervals in an NLP that effectively modulates the step duration. To solve the optimal contact timing, we formulate our step duration adaptation problem for multiple walking steps using direct multiple-shooting, inspired by the work of [46].

D. Temporal-logic-based Formal Methods

Formal methods, such as temporal logic, are increasingly popular in robotics because of their ability to reason about both discrete actions and continuous motions [47]–[49]. In high-level task planning, reactivity is critical to account for environmental changes at runtime. To achieve reactive task planning, linear-temporal-logic-base (LTL-based) reactive synthesis [47], [50] has been widely explored. These methods synthesize automata that generate safe and provably correct robot actions in response to potentially adversarial environmental events. Recent works [51]–[54] adopt LTL to synthesize reactive legged navigation plans over rough terrains. Our earlier work [13] uses LTL to synthesize a safe automaton (i.e., decision-maker) for locomotion push recovery tasks.

Although the automaton-based method provides a formal guarantee of specification satisfaction, it requires a non-trivial abstraction (i.e., system discretization), which does not scale well to high-dimensional systems and is often limited to coarse cell discretization of the system state space [55]. In addition, unexpected changes or disturbances encountered at runtime between two consecutive discrete events can cause failures in the actions, and further pose risks to robot hardware [56].

Distinct from the conventional automaton-based approaches, signal temporal logic (STL) [57] is an abstraction-free method that can be formulated as an optimization for synthesizing safe and correct locomotion plans. For instance, [58] formulates an STL-based TO for high-dimensional nonlinear robotic manipulators. [59] uses STL to tackle the multi-drone reach-avoid problem. However, to the best of the authors' knowledge, no existing STL studies have focused on bipedal locomotion, such as a Cassie robot with 20 degrees of freedom, let alone the more challenging push recovery problem.

E. STL Specification Encoding Methods

STL specifications are encoded into an optimization problem in two major ways. A classic way to encode STL specifications is to introduce binary variables into the optimization problem [57], effectively constructing a MIP. However, these MIP-based synthesis methods [60], [61] are often computationally expensive due to the exponential complexity with respect to the number of binary variables involved, hampering the real-time performance of reactive planning for complex systems such as legged robots.

Another encoding approach [62], which excludes binary variables completely, leverages a smooth approximation of a task specification formula. This smooth approximation results in a NLP, wherein the STL specification is encoded as either an objective or a constraint. The benefit of this approach is that it exploits the efficiency of gradient-based optimization techniques to solve the synthesis problem, avoiding the difficulty of handling binary variables. In this study, we adopt the smooth encoding method and experimentally demonstrate its computational advantage over the MIP method, specifically in the context of bipedal locomotion.

III. System Dynamics and Riemannian Robustness

A. Hybrid Reduced-order Model for Bipedal Walking

In this study, we propose a new reduced-order model extending the traditional linear inverted pendulum model (LIPM) [33], [63] to model the center-of-mass (CoM) dynamics of a bipedal robot with its swing-foot position and velocity. The traditional LIPM has a point mass, denoted as the robot's CoM, and a mass-less telescopic stance leg that maintains the CoM height. The locomotion dynamics are hybrid due to discrete contact events. In between contacts, each walking step has the following continuous dynamics:

$$\dot{\boldsymbol{x}} = f^j(\boldsymbol{x}) + g^j(\boldsymbol{x})\boldsymbol{\tau},$$

where $\boldsymbol{x} \coloneqq [\boldsymbol{p}_{\mathrm{CoM}}; \boldsymbol{v}_{\mathrm{CoM}}]$ is the system state, $\boldsymbol{p}_{\mathrm{CoM}}, \boldsymbol{v}_{\mathrm{CoM}} \in \mathbb{R}^3$ are the position and velocity of the CoM in the local stance-foot frame, respectively, as shown in Fig. 3. The superscript j is the index of a walking step. $\boldsymbol{\tau}$ is the input torque about the CoM. Due to Cassie's relatively small torso that is maintained upright during walking, $\boldsymbol{\tau}$ is close to zero. Assuming $\boldsymbol{\tau} = 0$, the LIPM dynamics [33] with a constant height $p_{\mathrm{CoM},z}$ are

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

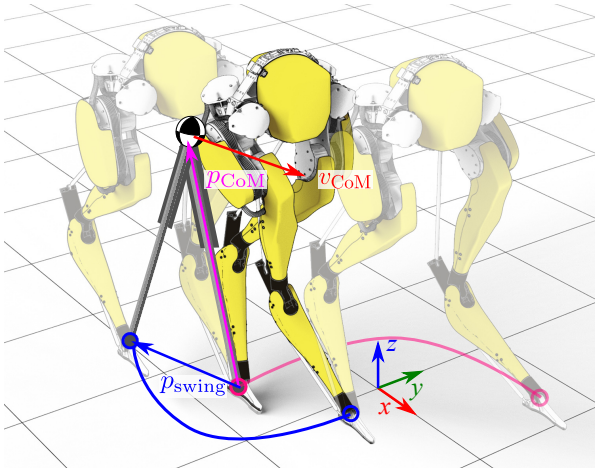

Fig. 3: Our reduced-order modeling of the Cassie robot as a 3D inverted pendulum model with all of its mass concentrated on its CoM. The massless telescopic leg maintains a constant CoM height. The swing foot position is included in the proposed model to deal with leg self-collision avoidance.

where $\omega = \sqrt{g/p_{\text{CoM},z}}$ and g is the gravity constant. The subscripts x and y mean the sagittal and lateral components of the vector.

Conventionally, when the robot foot contact switches, a *reset map* between two consecutive walking steps models the change between the pre-contact state $\boldsymbol{x}^-$ and the post-contact state $\boldsymbol{x}^+$: $\boldsymbol{x}^+ = \Delta_{j\to j+1}(\boldsymbol{x}^-)$. Given that the LIPM incorporates only a singular point mass, this study does not model the rigid-body impact dynamics associated with establishing and breaking contact. Also, we assume the velocity transition between walking steps is smooth, although a non-smooth version can be derived in a straightforward manner. Due to the use of a local coordinate, the LIPM state *reset map* follows the equations: $\boldsymbol{p}^+_{\text{CoM}} = \boldsymbol{p}^-_{\text{CoM}} - \boldsymbol{p}^-_{\text{swing}}, \boldsymbol{v}^+_{\text{CoM}} = \boldsymbol{v}^-_{\text{CoM}}$, where $\boldsymbol{p}^-_{\text{swing}} \in \mathbb{R}^3$ is the swing foot location before contact.

In this study, we design a variant of the traditional LIPM that incorporates the modeling of the swing-foot position and velocity. The swing leg is considered to be massless and therefore does not affect the CoM dynamics. As a result, the state vector is augmented as $\bar{\boldsymbol{x}} := [\boldsymbol{p}_{\text{CoM}}; \boldsymbol{v}_{\text{CoM}}; \boldsymbol{p}_{\text{swing}}]$, $\boldsymbol{p}_{\text{swing}} \in \mathbb{R}^3$. We then define the swing foot velocity $\dot{\boldsymbol{p}}_{\text{swing}}$ as the control input $\bar{\boldsymbol{u}} = \dot{\boldsymbol{p}}_{\text{swing}}$. As shown in Fig. 4, our model has a medium complexity that lies between the traditional LIPM and the full-order model. Such design comprises the advantages of both: it provides a fast and analytical solution for CoM dynamics while allowing full-body collision checking.

To find the numerical solution of the augmented LIPM dynamics, we use a second-order Taylor expansion of (1). The CoM state (i.e., position and velocity) can then be obtained via numerical integration (e.g., Euler integration). Moreover, we define $\boldsymbol{y} = [\bar{\boldsymbol{x}}; \bar{\boldsymbol{u}}] \in \mathbb{R}^{12}$ as the system output, which will be used for signal temporal logic (STL) definition in Sec. IV. 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. VI-C. The augmented state is estimated from the joint encoder and an IMU sensor in practice.

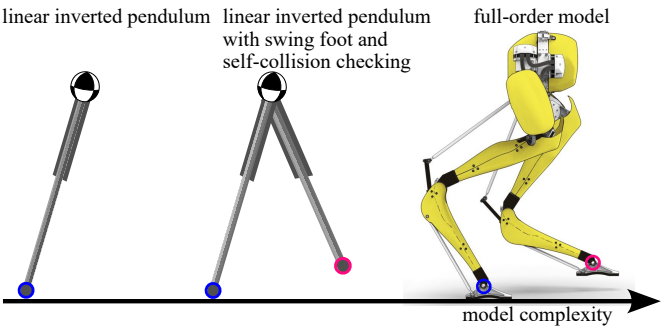

Fig. 4: Our model has a medium complexity that lies between the linear inverted pendulum model and the full-order model. It preserves the ability to reason about leg collision of the full-order model, yet maintains the simplicity of inverted-pendulum models that allow for online trajectory optimization.

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

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

The hybrid system transitions 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 position and velocity parameters above are expressed in a local coordinate attached to the stance foot. At a foot strike event, the swing foot transitions to become the new stance foot instantaneously, and all local position variables change accordingly. In the context of dynamic locomotion, the double-support contact phase is often short (approximately 40 ms in our experiments), which supports the assumption of the instantaneous contact switch in our study.

B. Keyframe-based Non-periodic Locomotion

In general, the bipedal locomotion process can be highly non-periodic due to rough terrain or unexpected environmental perturbations. To characterize this locomotion non-periodicity in the reduced-order state space of the robot, we adopt the concept of *keyframe* proposed in our previous work [32], [64]. As a critical locomotion state, the keyframe characterizes a non-periodic walking step in a reduced-order space.

Definition III.1 (Locomotion keyframe)**.** *Locomotion keyframe is defined as the robot's CoM state* $(\boldsymbol{p}_{\text{CoM}}, \boldsymbol{v}_{\text{CoM}})$ *at the apex, i.e., when the CoM is over the stance foot in the sagittal direction* ($p_{\text{CoM},x} = 0$)*, as shown in Fig. 9(a).*

A keyframe may not be defined for every walking step, where a walking step is defined as the smooth motion between two consecutive contact events (see Fig. 7). For a periodic walking gait, a keyframe state always exists in every walking step. However, this property does not always hold under external perturbations. For example, if the perturbation pushes the CoM to move backward, the robot loses the momentum to pass over the apex state (see Fig. 5(a, c)). Conversely, a forward perturbation of the CoM might result in bypassing the subsequent apex state entirely (see Fig. 5(b, d)). The absence of apex states during non-periodic locomotion poses

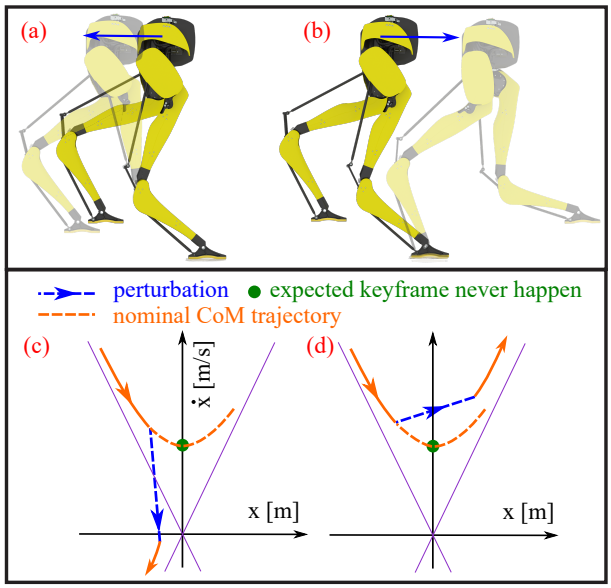

Fig. 5: This illustration depicts two scenarios where a keyframe is absent in the state space, due to perturbations in the backward and forward directions. We assume perturbation causes an instantaneous change to the CoM state, leading to subsequent non-periodic walking patterns.

challenges to the control synthesis in our previous phase-space planning method [13], which requires a keyframe for every walking step to make discrete locomotion decisions. To address this challenge, this study proposes a signal-temporal-logic-based optimization method. This method enables robust planning in the absence of a keyframe due to perturbations, provided that a keyframe is *eventually* established within the planning horizon (e.g., within two walking steps).

C. Robustness Quantification in the CoM State Space

The robustness measure is a crucial metric for quantifying locomotion stability and resilience to environmental perturbations. The robustness measure assesses the system's ability to tolerate perturbation-induced deviations from nominal states.

To quantify the robustness measure, we design a robust region centered around a nominal keyframe state in a Riemannian space, and will further integrate it as a cost function within the trajectory optimization (TO) in Sec. VI. As shown in Fig. 6, the Riemannian space [32] 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 Appendix A. Within the Riemannian space, we define a robust keyframe region that enables stable walking. This region is referred to as the Riemannian region.

Definition III.2 (Riemannian region)**.** *The Riemannian region $\mathcal{R}$ is the area centered around the nominal keyframe state $(\sigma_{\text{nom}}, \zeta_{\text{nom}})$.*

$$\begin{aligned}\mathcal{R}_x \coloneqq \{(p_{\text{CoM},x}, v_{\text{CoM},x}) | \sigma(p_{\text{CoM},x}, v_{\text{CoM},x}) \in \Sigma_x,\\ \zeta(p_{\text{CoM},x}, v_{\text{CoM},x}) \in \mathrm{Z}_x\}\\ \mathcal{R}_y \coloneqq \{(p_{\text{CoM},y}, v_{\text{CoM},y}) | \sigma(p_{\text{CoM},y}, v_{\text{CoM},y}) \in \Sigma_y,\\ \zeta(p_{\text{CoM},y}, v_{\text{CoM},y}) \in \mathrm{Z}_y\}\end{aligned}$$

where $\mathcal{R}_x$ and $\mathcal{R}_y$ define the Riemannian region $\mathcal{R}$ in the sagittal and lateral phase space, respectively. $\sigma(\cdot)$ and $\zeta(\cdot)$ are the tangent and cotangent manifolds. $\Sigma = [\sigma_{\text{nom}} - \delta\sigma, \sigma_{\text{nom}} + \delta\sigma]$ and $\mathrm{Z} = [\zeta_{\text{nom}} - \delta\zeta, \zeta_{\text{nom}} + \delta\zeta]$ are the range of manifold coefficients for σ and ζ, where $\delta\sigma, \delta\zeta$ are the predefined robustness margins.

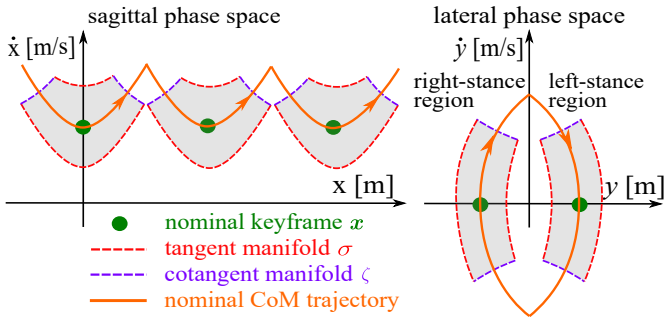

Fig. 6: An illustration of a phase-space Riemannian partition and a periodic walking trajectory.

The sagittal and lateral Riemannian regions in the phase space are illustrated in Fig. 6 as shaded areas. The bounds of these Riemannian regions are curved in the phase space and 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. 6. In this study, we leverage the Riemannian region to define the Riemannian robustness as a measure of locomotion robustness.

Definition III.3 (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, $\rho_{\text{riem}} \coloneqq \min_{l=1}^{8}(r_l(\boldsymbol{x}))$, where $r_l(\boldsymbol{x})$ is the signed distance to the l^{th} bound of the Riemannian regions, as illustrated in Fig. 9(c). We have a total of 8 bounds, as the sagittal and lateral Riemannian regions each have 4 bounds.*

The definition above indicates that a keyframe inside a Riemannian region has a positive robustness value, representing a stable walking step; conversely, a keyframe outside a Riemannian region gets a negative robustness value, indicating an unstable deviation from the nominal locomotion manifold. Furthermore, a keyframe positioned at the geometric center of a Riemannian region has the least deviation, i.e., the maximum robustness. In the next section, our goal is to leverage Riemannian robustness as an objective function and use STL-based TO to plan robust trajectories for locomotion push recovery. As a conceptual illustration, Fig. 7 shows one of such robust trajectories in the lateral phase space. A disturbed CoM state progresses through intermediate keyframes and eventually recovers to a stable keyframe.

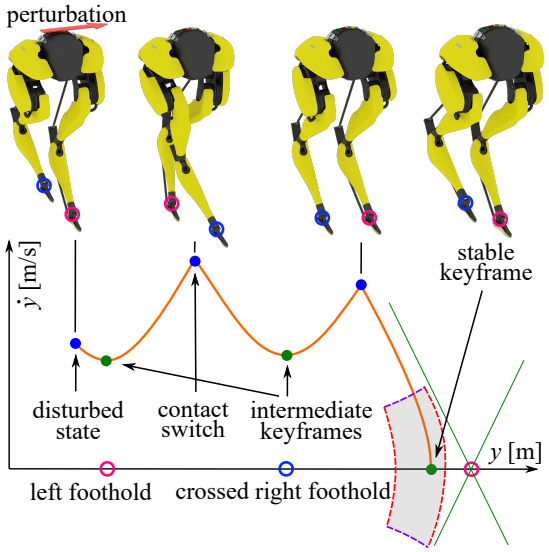

Fig. 7: An illustration of a phase-space Riemannian region and the lateral keyframe transition for disturbance recovery.

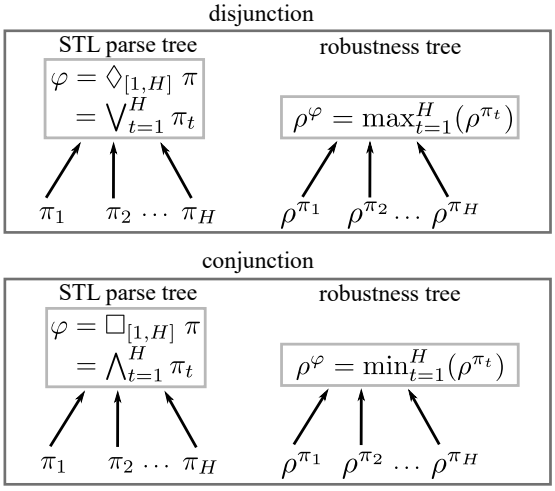

Fig. 8: Composition of an STL formula φ and its robustness degree ρ^{φ}. Each STL formula is represented in a tree whose child nodes combine in disjunction (top row) or in conjunction (bottom row). The corresponding robustness trees (on the right side) take the same structure, and combine with max and min functions, respectively.

IV. Preliminaries of Signal Temporal Logic

As conventional trajectory optimization (TO) methods primarily generate optimal robot motions based on state or control objectives, they may encounter challenges when incorporating task specifications encoded by logical objectives. For instance, the keyframe state (Def. III.1) is *always* achieved when the CoM state is at an apex. For another example, a robot is designed to *eventually* recover within a finite planning horizon after being perturbed.

Signal temporal logic (STL) provides a framework to express these logical objectives in the form of task specifications, which are incorporated in a TO as objectives. This specification-integrated TO effectively synthesizes control sequences that not only comply with task specifications but also enhance the locomotion robustness of the resulting trajectory. These unique features make STL an effective approach for tackling complex locomotion tasks that involve logical objectives, offering capabilities that cannot be easily achieved by traditional TO methods.

A. Signal Temporal Logic: Syntax and Robustness Degree

STL [8] 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 formula is defined with the following syntax:

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

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, a 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 represented 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. We denote a specific segment of a signal within the interval $[t_1, t_2]$ as $\boldsymbol{y}([t_1 : t_2])$. The STL semantics $(\boldsymbol{y}, t) \models \varphi$ indicates that the segment of the signal $\boldsymbol{y}([t : t_{\text{end}}])$ satisfies φ. The validity of 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 a unique capability of admitting *quantitative semantics*, which denotes the *robustness degree* [65] of how strongly a formula is satisfied by a signal [14]. A positive robustness value indicates satisfaction, and the magnitude represents the margin of robustness against disturbances. In a dynamic environment involving terrain perturbations, the robustness degree indicates a margin of locomotion maneuverability (i.e., the degree of adaptability) to react without violating robot task specifications [59]. When incorporating the robustness degree into TO, it also helps to generate a minimally specification-violating trajectory if the task specification cannot be satisfied strictly [60]. Table II shows the semantics of the robustness degree of STL.

TABLE II
ROBUSTNESS DEGREE SEMANTICS

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

In the implementation of STL specification, an STL formula is represented by logical combinations of its subformulas. Such logical combinations are organized in an STL parse tree [66], [67]. Two simple STL parse trees are shown in Fig. 8: the *eventually* operator $\Diamond$ integrates subformulas using disjunction, whereas the *always* operator $\Box$ combines subformulas with conjunction. A leaf node corresponds to a predicate π, and a parent node represents the logical combination of its subtrees. For complex specifications with cascaded STL logical operations, the corresponding STL parse tree has multiple layers and can be constructed based on STL semantics in Table I.

Correspondingly, each STL parse tree is associated with a robustness tree, inheriting the same tree structure. The robustness tree is different from the STL parse tree in two ways. First, STL parse tree nodes represent logical satisfactions whereas robustness tree nodes represent robustness degrees. Second, in the robustness tree, the combination of subtrees uses $\min$ and $\max$ functions, replacing the logical symbol $\wedge$ and $\vee$ in the STL parse tree, respectively. To represent the robustness tree, we employ a pre-order traversal approach, resulting in a vector $\boldsymbol{\rho}^{\varphi}(\boldsymbol{y}, t)$ that encapsulates all tree nodes. This pre-order traversal ensures that the robustness tree's root node is positioned as the first element of the vector, represented by the scalar $\rho^{\varphi}(\boldsymbol{y}, t)$. Later, this $\boldsymbol{\rho}^{\varphi}(\boldsymbol{y}, t)$ will be encoded as decision variables in our TO and $\rho^{\varphi}(\boldsymbol{y}, t)$ will be part of the cost function.

V. Problem Formulation

In this section, we study the synthesis of locomotion control using signal temporal logic (STL). Our objective is to plan a control sequence for a reduced-order bipedal walking system, ensuring locomotion resilience to environmental perturbations. To accomplish this, the synthesized control sequence must be both *correct* and *dynamically-feasible*. The notation of *correct* indicates that the continuous-time trajectory satisfies a given specification φ, and by *dynamically-feasible*, we mean that the trajectory satisfies the reduced-order dynamics of the bipedal system. In addition, we aim to enhance the *robustness degree* of the task specifications as detailed later in this section.

To address this problem, we use the *keyframe* concept described in Sec. III-B: the hybrid locomotion trajectory is segmented by contact-switching events into multiple walking steps, each parameterized by a keyframe state. The problem then boils down to planning a series of keyframe states. The sequence of keyframe states forms a hybrid trajectory satisfying the desired specifications. We introduce the specification design in the following subsection and the optimization approach for keyframe synthesis in the next section.

A. Specification Design for Perturbation-resilient Locomotion

This subsection elaborates on the design of the STL locomotion specification φ_{loco} and explains the stability guarantee it provides. We interpret locomotion stability as a *liveness* property in the sense that a keyframe with positive Riemannian robustness will *eventually* occur in the planning horizon.

Keyframe specification: To enforce properties on a keyframe, we first describe it using an STL formula $\varphi_{\text{keyframe}}$, checking whether or not a signal $\boldsymbol{y}(t)$ is a keyframe.

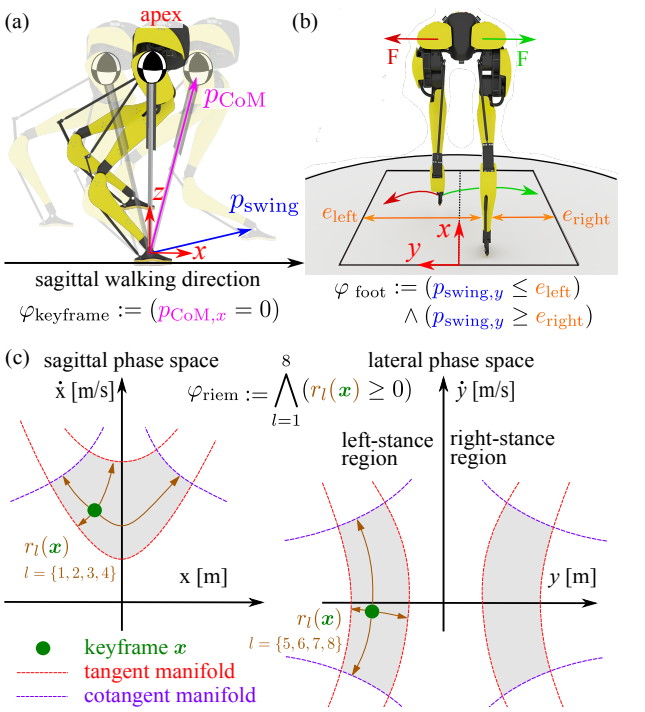

Fig. 9: Illustration of the locomotion specifications. (a) The highlighted state in the middle (i.e., the apex state) is the keyframe of a walking step. (b) Cassie's foot is specified to step inside the lateral bounds of the treadmill. (c) The grey areas are the predefined Riemannian regions in the sagittal and lateral phase spaces. The signed distances to the bounds of the Riemannian regions are indicated by the curved arrows.

Bipedal locomotion inherently has a fixed event sequence, alternating between keyframe and contact-switching events. However, the timing of these events is not fixed in non-periodic walking. For example, the keyframe event k^j must exist in the j^{th} walking step (between two contact events C^j and C^{j+1}), but the time of the keyframe occurrence can vary, depending on the center of mass (CoM) state within the walking step. Consequently, we cannot predetermine a certain timestep to be the keyframe state. This is a logical constraint where a traditional optimization constraint fail to model. On the other hand, STL is suitable to solve this issue by applying a logical formula by checking each timestep whether it is a keyframe.

According to the keyframe definition (Def. III.1), the keyframe state at apex happens when the CoM surpasses the foot contact in the sagittal direction, formally specified as an STL formula: $\varphi_{\text{keyframe}} := (\mu^{\pi}_{\text{CoM},x}(\boldsymbol{y}) = 0)$, where $\mu^{\pi}_{\text{CoM},x}(\boldsymbol{y}) = p_{\text{CoM},x}$. In the implementation, since the STL predicate does not support equality constraints, we use two inequality conditions, i.e., $\varphi_{\text{keyframe}} := (\mu^{\pi}_{\text{CoM},x}(\boldsymbol{y}) \leq 0) \wedge (\mu^{\pi}_{\text{CoM},x}(\boldsymbol{y}) \geq 0)$, to check whether or not $p_{\text{CoM},x} = 0$.

Riemannian robustness: The robustness of a walking step is measured as the minimum Riemannian distance between the keyframe and a set of bounds of the Riemannian region. To achieve robust locomotion, the task specification ensures that the signal $\boldsymbol{y}$ of a keyframe resides in the Riemannian region:

$$\varphi_{\text{riem}} = \bigwedge_{l=1}^{8} (r_l(\boldsymbol{y}) \geq 0) \tag{4}$$

where $r_l(\boldsymbol{y})$ is the riemannian distance from the keyframe to the l^{th} bound of the Riemannian region, as introduced in

Sec. III-C. Since our robustness is defined for both sagittal and lateral CoM state space and the Riemannian region in each state space has 4 bounds, we have $2 \times 4 = 8$ bounds in total. The φ_{riem} monitors whether or not the signal $\boldsymbol{y}$ is inside the Riemannian region, and its robustness degree $\rho^{\varphi_{\text{riem}}}$ evaluates the robustness of the keyframe in Riemannian space.

Locomotion stability: To achieve locomotion stability, we enforce a *liveness* property of the signal in the sense that a steady-state keyframe will *eventually* be achieved in the planning horizon, thus making sure the robot always recovers to a steady-state gait. This is represented in a stability specification: $\varphi_{\text{stable}} = \Diamond_{[T^N_{\text{contact}}, T^{N+1}_{\text{contact}}]}(\varphi_{\text{keyframe}} \wedge \varphi_{\text{riem}})$, where the T^N_{contact} and T^{N+1}_{contact} are the time for N^{th} and $N+1^{\text{th}}$ contact, and represent the time bounds of the last walking step in the planning horizon.

Stability is achieved if the keyframe in the last walking step falls inside the corresponding Riemannian robust region. In our study, we need to examine only the last walking step because in case stability is achieved in an earlier walking step within the horizon, the robot will continue periodic walking and the last walking step will trivially satisfy φ_{stable}.

Locomotion step width bound: For locomotion in a narrow space (e.g., a treadmill with limited width), we use the *safety* specification $\square\varphi_{\text{foot}}$ as a limit of the foothold to land inside of the treadmill's edges. The operator $\square$ without a time bound means the specification should hold for the entire planning horizon. We have $\varphi_{\text{foot}} = (\mu^{\pi}_{\text{left}}(\boldsymbol{y}) \geq 0) \wedge (\mu^{\pi}_{\text{right}}(\boldsymbol{y}) \geq 0)$, 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 right edge e_{right} of the treadmill.

STL formula composition: The compounded locomotion specification is $\varphi_{\text{loco}} = \varphi_{\text{stable}} \wedge (\square\varphi_{\text{foot}})$. The satisfaction of the specification φ_{loco} is equivalent to the robustness degree being positive:

$$(\boldsymbol{y}, t) \models \varphi_{\text{loco}} \Leftrightarrow \rho^{\varphi_{\text{loco}}}(\boldsymbol{y}, t) \geq 0. \tag{5}$$

In order to maximize the locomotion robustness, we use the robustness degree $\rho^{\varphi_{\text{loco}}}$ as an objective function in the trajectory optimization (TO) in the following section.

Remark. *The main contribution of our STL formulation is its effectiveness and simplicity, allowing fast online reactive planning and from this formulation emerges complex behaviors such as crossed-leg locomotion for perturbation recovery.*

B. Specification Encoding via Smooth Approximation

The STL specifications defined above are encoded into a gradient-based TO. To this end, this subsection introduces a smooth operator to allow a smooth gradient in the TO for efficient computation.

A traditional approach of encoding an STL formula in a TO problem employs the big-M formulation [14]. In this formulation, signal satisfactions are associated with binary variables via inequality constraints, i.e., True equals 1 and False equals 0. To guarantee the satisfaction of the STL formula, additional equality constraints assert the binary variables equal to 1. This encoding method introduces extra binary variables, thus forming a mixed-integer program (MIP), which has a complexity exponential to the number of binary variables.

This study adopts an alternative encoding technique using the smooth approximation of the robustness degree. The robustness degree $\rho^{\varphi}(\boldsymbol{y})$ is originally non-smooth because of the $\min$ and $\max$ operators in its expression. A non-smooth function can lead to zero gradients in the TO, causing ill-conditioned issues. The smooth-operator encoding method replaces the non-smooth operators with smooth approximated operators. Consequently, the new robustness degree $\tilde{\rho}^{\varphi}$ becomes a smooth nonlinear function that can be optimized via efficient nonlinear programming solvers.

Specifically, we replace the non-smooth $\min$ and $\max$ operators with their smooth counterpart $\widetilde{\min}$ and $\widetilde{\max}$.

$$\widetilde{\min}([\rho_1, ..., \rho_m]^T) = -\frac{1}{k_1}\log(\sum_{i=1}^{m} e^{-k_1\rho_i})$$

$$\widetilde{\max}([\rho_1, ..., \rho_m]^T) = \frac{\Sigma_{i=1}^{m}\rho_i e^{k_2\rho_i}}{\Sigma_{i=1}^{m} e^{k_2\rho_i}}$$

where $k_1, k_2 > 0$ are tunable parameters, $m \in \mathbb{Z}^+$ is the number of parameters in the $\min/\max$ operator, ρ is the robustness degree, and e is the Euler's number. The smooth operators have a property of under-approximation, meaning that the approximated robustness degree $\tilde{\rho}^{\varphi}$ is strictly smaller than ρ^{φ}. This specific choice of the smooth operator renders a property of *soundness*: $\tilde{\rho}^{\varphi}(\boldsymbol{y}, t) \geq 0 \Rightarrow \rho^{\varphi}(\boldsymbol{y}, t) \geq 0 \Leftrightarrow (\boldsymbol{y}, t) \models \varphi$. We refer the interested reader to [68] for further details.

VI. Model Predictive Control for Push Recovery

A. Optimization Formulation

A model predictive control (MPC) is formulated to solve a sequence of optimal states and controls (i.e., signals) that satisfy the specifications φ_{loco}, system dynamics, and kinematic constraints within a finite-time horizon $\mathbb{H}$. Specifically, this MPC simultaneously determines foot placements, swing foot trajectories, and step durations in both normal and perturbed walking situations. Solving foot placement and swing foot trajectory simultaneously in an integrated formulation reduces the potential infeasibility that can arise in a hierarchical optimization approach, where the foot placement is determined first and the swing foot trajectory is solved separately.

We formulate the MPC problem as a nonlinear program:

$$\min_{\boldsymbol{X},\boldsymbol{U},\boldsymbol{T}} \quad w\mathcal{L}(\boldsymbol{U}) - \tilde{\rho}^{\varphi_{\text{loco}}}(\boldsymbol{X}, \boldsymbol{U}) \tag{6}$$

$$\text{s.t.} \quad \bar{\boldsymbol{x}}^j_{i+1} = f(\bar{\boldsymbol{x}}^j_i, \bar{\boldsymbol{u}}^j_i, T^j), \qquad i \in \mathbb{H} \setminus \mathbb{S}, \quad j \in \mathbb{J} \tag{7}$$

$$\bar{\boldsymbol{x}}^{+,j+1} = \bar{\Delta}_{j\to j+1}(\bar{\boldsymbol{x}}^{-,j}), \qquad j \in \mathbb{J} \tag{8}$$

$$g_{\text{collision}}(\bar{\boldsymbol{x}}_i) \geq \epsilon, \qquad i \in \mathbb{H} \tag{9}$$

$$g_{\text{duration}}(T^j) \geq 0, \qquad j \in \mathbb{J} \tag{10}$$

$$h_{\text{initial}}(\bar{\boldsymbol{x}}_0) = 0, \tag{11}$$

$$h_{\text{transition}}(\bar{\boldsymbol{x}}_i) = 0, \qquad i \in \mathbb{S} \tag{12}$$

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. 10 gives an example

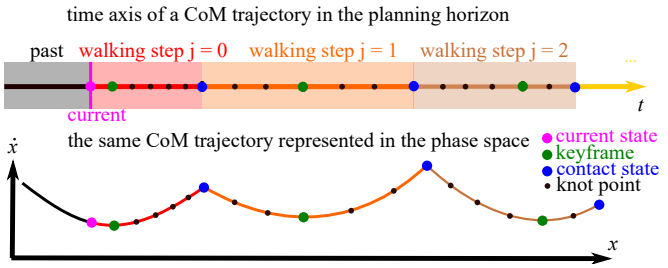

Fig. 10: The planning horizon starts from the current measured state (pink). An example of $N = 2$ walking steps and 7 knot points per walking step is illustrated. The time axis (top) and the phase space plot (bottom) represent the same CoM trajectory.

of a planning horizon with $N = 2$. $\mathbb{S}$ is the set of indices containing the time steps of all contact switch events, $\mathbb{S} \subset \mathbb{H}$. $\mathbb{J} = \{0, \dots, N\}$ is the set of walking step indices. The decision variables include $\boldsymbol{X} = \{\bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_M\}$, $\boldsymbol{U} = \{\bar{\boldsymbol{u}}_1, \dots, \bar{\boldsymbol{u}}_M\}$, and $\boldsymbol{T} = \{T^0, \dots, T^N\}$. $\boldsymbol{T}$ 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 [68]. 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. IV. For the MPC objective (6), the selection of w is a tradeoff between satisfying signal temporal logic (STL) specification and minimizing control effort. Specifically, a smaller w allows more aggressive control for rapid recovery from disturbance. In contrast, a larger w penalizes more on control effort but less on the center of mass (CoM) deviation.

To satisfy the linear inverted pendulum model (LIPM) dynamics (1) while adapting step durations $\boldsymbol{T}$, we use a second-order Taylor expansion to derive the approximated discrete dynamics (7). (8) defines the reset map from the foot-ground contact switch. (9) 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 $g_{\text{collision}}$ is a set of multilayer perceptrons (MLPs) learned from leg configuration data, as detailed in Sec. VI-C. (10) 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 [36]. (11) 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 $\mathbb{S}$.

B. Step Duration Adaptation

We adapt the durations of walking steps to enhance the perturbation recovery capability of the bipedal system.

For our MPC, we follow the direct multiple shooting method [46]. This method introduces additional decision variables $\boldsymbol{T}$ to represent the durations of the future $N+1$ walking steps. We clamp the step duration T^j within a time bound $[T_{\min}, T_{\max}]$ through the constraint (10) for the following reasons: (i) the upper bound $T_{\max}$ is useful to prevent a fall due to a slow stepping frequency in dynamic walking; (ii) the lower bound $T_{\min}$ is useful to avoid an exceedingly fast leg motion that is beyond the physical capability of the robot.

Note that, as the robot state approaches the contact event, the remaining duration T_{remain} of the current walking step keeps reducing, and declines its flexibility to change. To address this issue, we fix the current step duration T^0 to the latest solved solution while still solving the future N step durations once T_{remain} reduces to less than a threshold.

C. Data-driven Self-collision Avoidance Constraints

To calculate the collision status using only the reduced-order model, we adopt MLPs that learn the mapping from Cassie's LIPM state $[\boldsymbol{p}_{\text{CoM}}; \boldsymbol{p}_{\text{swing}}]$ to the shortest collision distances between leg geometry pairs that are under high risk of collision. These MLP constraints are particularly useful for planning crossed-leg motions. We encode the MLPs as constraints of the MPC to ensure collision-free trajectories.

The geometry of Cassie's leg can be approximated by three capsules attached to its shin, tarsus, and Achilles rod, respectively. According to Cassie's leg configurations, collision risks are high amongst the following 6 capsule pairs: 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). The remaining 3 pairs will not collide within the range of joint motions. Accordingly, 6 MLPs computing the shortest collision distances between these capsule pairs are generated. Each MLP consists of 2 hidden layers with 24 neurons implemented using PyTorch [69]. Instead of training one MLP to represent the minimum distance among all capsule pairs, we use separate MLPs for each pair because the former requires a larger network structure but yields worse accuracy.

We generate the training dataset such that each data point corresponds to a particular robot configuration, containing the feature (the LIPM state) and the label (six analytically-computed distances $[d_1, d_2, ..., d_6]$). A total of 10^6 configurations centering around the robot's standing configuration are collected in the dataset. The training takes less than 2 hours in total with an Intel® Core™ i7-12700H CPU. The MLPs achieved an accurate prediction performance, yielding a maximum absolute error of 0.03 m and an average absolute error of 0.002 m. The evaluation speed of the MLPs exceeds 1 MHz, as compared to less than 1 kHz for a traditional method requiring iterative inverse kinematics. The accuracy and speed of the MLPs facilitate the online execution of our MPC.

VII. Experimental Setup

We provide implementation details, parameter setup of our signal-temporal-logic-based model predictive controller (STL-MPC), and experimental configurations in this section.

Our STL implementation leverages STLPY [66], a Python-based STL library. STLPY facilitates the integration of signals into the MPC as decision variables, denoted by $\boldsymbol{y}(t) =$

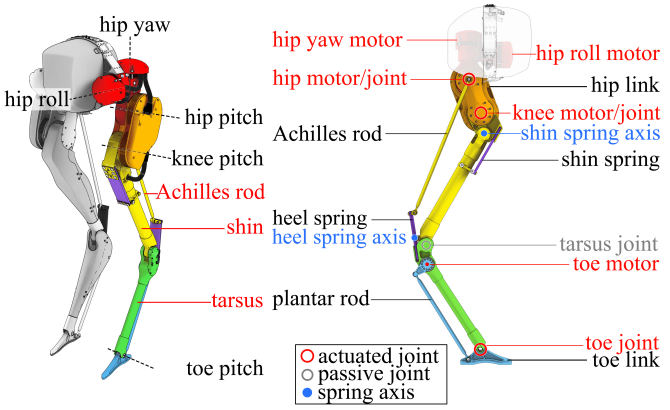

Fig. 11: The anatomy of Cassie's leg. The Achilles rod, shin, and tarsus are used for collision checking because they have a higher risk of collision.

$(\boldsymbol{x}(t), \boldsymbol{u}(t))$, which are optimized for specification satisfaction and robustness maximization.

The STL-MPC is formulated as a nonlinear program using pydrake [70] and solved by a sequential quadratic programming (SQP) algorithm using SNOPT [71]. The planning horizon includes a current walking step and $N = 2$ future walking steps, as shown in Fig. 10. Each walking step is assigned with 7 knot points, leading to a total of $M = 21$ time steps. To prioritize STL satisfaction, we set $w = 0.01$, indicating a small control effort penalty. The minimum distance between collision pairs $\epsilon = 0.03$ m. The walking step duration range in (10) is designed according to human biomechanics data [72]: $[T_{\min}, T_{\max}] = [0.3, 0.5]$ s. Here, $T_{\min}$ represents a physical limitation of how fast a step can be. Conversely, $T_{\max}$ reflects a design consideration, where a larger step duration implies a less dynamic movement. We choose $T_{\max} = 0.5$ s to encourage a higher stepping frequency and enable more dynamic behaviors against perturbations. The desired gait is a periodic motion with 0.4 s step duration and 0.6 m/s center of mass (CoM) apex velocity. The robot commanded under our controller is capable of walking up to 1 m/s [5]. We choose 0.6 m/s to have a better buffer against forward perturbation that increases the robot's forward velocity. The robot's yaw direction (i.e., heading direction) is regulated to maintain straight walking, via an estimated state from the IMU sensor.

Simulation experiments are conducted in Matlab Simulink [73], a high-fidelity simulator based on Cassie's full-body dynamics. Perturbations are introduced in the form of impulses, i.e., a magnitude of force applied for a short time period. Throughout the simulation, self-collision is actively checked via a daemon function considering full-body kinematics and the approximated capsule geometry of the robot. Note that this daemon collision checking function runs in the background of the simulator and is different from the data-driven self-collision constraints proposed in Sec. VIII-A. Additionally, the robot state is monitored at every simulation step to detect robot failures. Specifically, failures include scenarios when: (1) the robot falls; (2) any joint angle exceeds its predefined limit; (3) a collision is detected; and (4) the robot drifts too much in the lateral direction. Failures effectively reflect extreme perturbations that are beyond the robot's ability to recover.

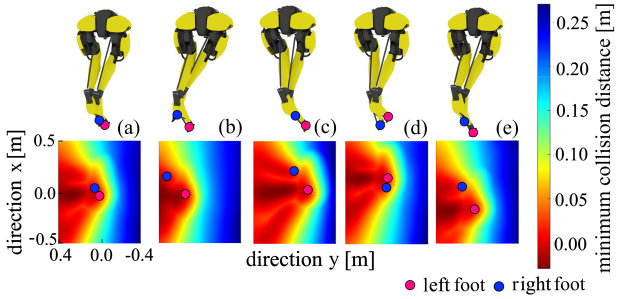

Fig. 12: Illustration of the self-collision landscape with left leg fixed at multiple configurations. The green points in the figure correspond to configurations in the second row. The collisions are highlighted in the red-circled area.

For all of our hardware tests, a high-level STL-MPC planner and a low-level controller operate simultaneously on two separate onboard computers. They communicate via a UDP protocol implemented in ROS2 [74]. During each planning cycle, the STL-MPC solves a trajectory optimization (TO) problem as described in Sec. VI and sends the solution to the low-level controller. The STL-MPC then reinitializes the same problem based on the latest state measurements. At the low level, a passivity-based controller [75] tracks the high-level trajectory. As a fail-safe mechanism to handle MPC failures, the low-level controller falls back to the previous plan. This controller also sends real-time estimates of the system state to the high-level planner. The high-level planner runs on an Intel i7-1260P CPU at 50 Hz for the task specification φ_{loco}, while the low-level controller runs on the other PC at 2 kHz.

VIII. Simulation Results

A. Leg Workspace Study for Collision Avoidance

We evaluate the allowable range of motion of the robot's swing leg under the set of trained collision constraints proposed in Sec. VI-C.

Given Cassie's bilateral symmetrical design, we designate Cassie's left leg as the stance leg and its right leg as the swing leg, without loss of generality. We fix Cassie's left foot at five distinct locations, including its nominal standing position at $[0, 0.1305, -0.9]$ m relative to the pelvis, and four other positions that are horizontally displaced by 0.1 m to the left $(+y)$, right $(-y)$, front $(+x)$, and back $(-x)$, respectively. Meanwhile, we move Cassie's right foot to scan continuously within a $1 \times 0.8\ m^2$ rectangle in the xy plane centered at $[0, -0.1305, -0.9]$ m relative to the pelvis, which is a nominal standing position for the right foot. For each configuration during the scanning process, a set of 6 collision distances is evaluated by the trained multilayer perceptrons (MLPs). The minimum distance from the set, $d_{\min} = \min(d_1, d_2, d_3, ..., d_6)$, as well as its associated geometry pair is recorded. At each configuration, one particular pair is exhibiting the highest risk of collision and effectively constraining the swing foot's range of motion. The resulting data is illustrated in Fig. 12 (a)-(e) as five 2D landscape plots. The red color indicates a small $d_{\min}$, which translates to a higher collision risk. We select a risky configuration from each of the five scenarios and show it along with the landscape.

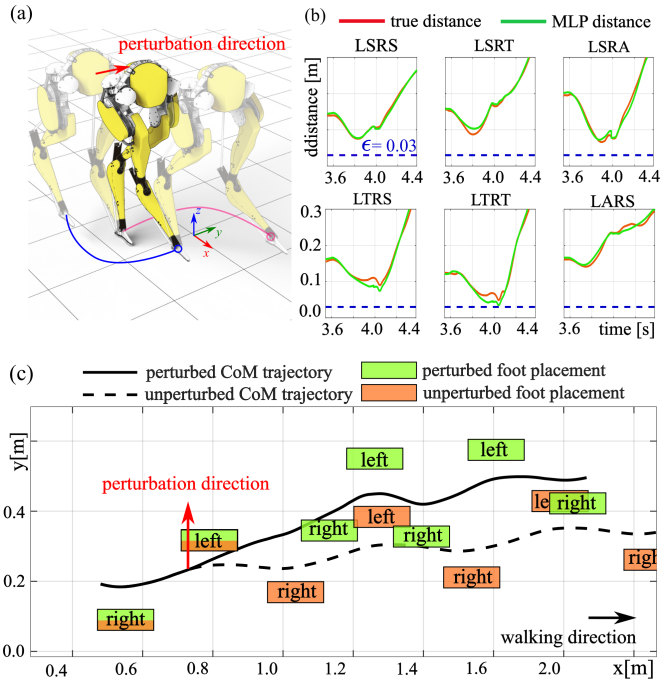

Fig. 13: (a) Snapshots of Cassie performing a crossed-leg maneuver for push recovery. (b) The MLP-approximated collision distances are accurate compared with the ground truth, and the planned leg trajectory is safe against the threshold $\epsilon = 0.03$. (c) An overhead view of the CoM trajectory and foot placements when a lateral perturbation induces a crossed-leg maneuver.

All the landscapes show a consistent trend of increasing collision risk as the right foot moves closer to the left foot, reflected by the dark red zones around the left foot. Additionally, all plots contain patterns of clustered red zones, suggesting that different collision pairs are active for each cluster. Notably, Fig. 12(c) depicts a larger expanse of the red zone than other landscapes. This observation is consistent with our expectation that an inward stance-leg configuration poses a higher collision risk and enforces more restrictive spatial constraints on the swing leg.

B. Self-collision Avoidance for Push Recovery

We demonstrate the planner's ability to avoid leg collisions in a critical push recovery setting, where a perturbation forces the robot to execute a crossed-leg maneuver. The perturbation is set up such that the robot's center of mass (CoM) is forcefully pushed towards the stance leg. In this circumstance, the only viable recovery strategy is to cross the leg. The model predictive control (MPC) with collision constraints generates a trajectory, as shown in Fig. 13(a), where the swing leg adeptly maneuvers around the stance leg and lands at a crossed-leg recovery point. The robot extricates (i.e., uncrosses) in the next step following a curved collision-free trajectory. Fig. 13(b) compares the ground truth and the multilayer perceptron (MLP)-approximated collision distances during the crossed-leg maneuver. The leg distances stay above a pre-specified threshold of ϵ, which protects against approximation and tracking error. An overhead view comparing perturbed and unperturbed CoM trajectories is shown in Fig. 13(c).

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

To encode signal temporal logic (STL) specifications into our trajectory optimization (TO) formulation, we adopt a smooth-operator method [62] 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 $\widetilde{\max}$, as detailed in Sec. V-B.

We benchmark the solving speed of the smooth-operator method [62] (ours) with a traditional mixed-integer program (MIP) method [57]. Similar to our smooth method, the MIP solves the signal $\boldsymbol{y}$ and durations of N walking steps. Our smooth-operator method formulates a nonlinear program (NLP) solved by SNOPT [71], while the MIP formulation is solved with Gurobi [76]. To make a fair comparison, we disable the nonlinear collision constraints in our NLP and have only linear constraints when evaluating both methods.

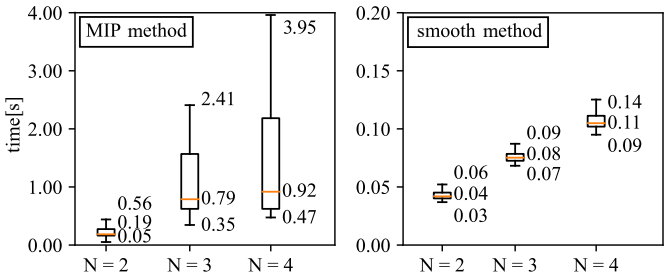

Fig. 14: A comparison of the traditional MIP method and our smooth method shows the planning time to solve trajectories for N-walking-step horizons. The smooth method is faster and more consistent over all horizons.

We compare the solving speed of both methods for an N-walking-step horizon, where $N = \{2, 3, 4\}$. The results are shown in Fig. 14. On average, the MIP method is 5–10 times slower than the smooth-operator method, and as the number of walking steps N increases, the performance difference becomes increasingly pronounced. This is due to the combinatorial complexity of the MIP. In contrast, the smooth-operator method exhibits solving times that are not only faster but also more consistent, as indicated by the narrower range between the minimum and maximum solving times, with each solving time typically falling within $\pm 50\%$ of the median time.

D. Stepping Stone Maneuvering

To demonstrate the signal-temporal-logic-based model predictive controller (STL-MPC)'s ability to handle a broad set of task specifications, we study locomotion in a stepping-stone scenario as shown in Fig. 15.

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. φ_{stones} specifies that the foot placement is within one of the stones. Each stone is a fixed-size rectangle generated at a random horizontal position and yaw orientation. For each rectangular stone, a stance foot $\boldsymbol{p}_{\text{stance}}$ is bounded inside its four edges, represented as a stone specification:

$$\varphi^o_{\text{stone}} = \bigwedge_{i=1}^{4} (\mu^o_i(\boldsymbol{p}_{\text{stance}}) \geq 0), \tag{13}$$

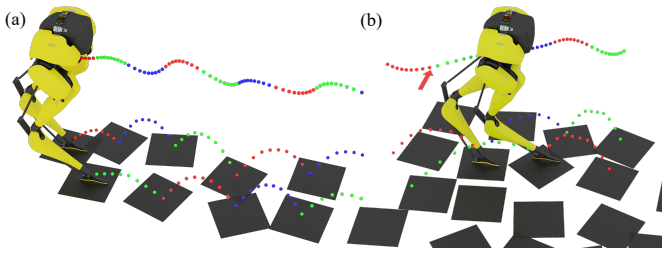

Fig. 15: Illustration of maneuvering over two stepping-stone scenarios. (a) STL-MPC solves dynamically feasible trajectories that satisfy an additional foot-on-stones specification. (b) STL-MPC successfully plans crossed-leg maneuvers to recover from perturbation.

where $o \in \{1, \dots, O\}$, $O \in \mathbb{Z}$ is the total number of stepping stones, and μ_i^o is the signed distance from the stance foot to the i^{th} edge of the o^{th} stone. For the MPC to plan a N-step trajectory, it needs to make decisions for N foothold locations:

$$\varphi_{\text{stones}} = \bigwedge_{j=1}^{N} (\square_{[T^j, T^j]} \bigvee_{o=1}^{O} \varphi_{\text{stone}}^o). \tag{14}$$

The augmented specification φ'_{loco} is the compound of the original locomotion specification φ_{loco} and the newly-added stepping stone specification:

$$\varphi'_{\text{loco}} = \varphi_{\text{loco}} \wedge \varphi_{\text{stones}}. \tag{15}$$

We test STL-MPC using φ'_{loco} in two scenarios, both with a series of stepping stones, as shown in Fig. 15. Given the initial position and the arrangement of the stepping stones, the MPC repeatedly solves a trajectory horizon and advances the global state by one walking step along the predicted trajectory. For the second scenario in Fig. 15(b), the STL-MPC demonstrates the ability to cross legs in response to a lateral perturbation. This result demonstrates our planner's capability to simultaneously satisfy complex task-level objectives (e.g., maintain balance) and respect physical requirements (e.g., step on stones).

E. Omnidirectional Perturbation Recovery

We examine the robustness of the STL-MPC framework through an ensemble of push-recovery tests conducted in simulation, where horizontal impulses are systematically applied to Cassie's pelvis. The impulses are exerted for a fixed duration of 0.1 s but vary in magnitude, direction, and timing. Specifically, the impulses have: 9 magnitudes evenly distributed between 80 N and 400 N; 12 directions evenly distributed between $0°$ and $330°$; and 4 locomotion phases indexed as a percentage $s = 0\%, 25\%, 50\%, 75\%$ through a full walking cycle. Collectively, this experimental design encompasses a total of 432 distinct trials. For a baseline comparison, the same perturbation procedure is applied to an angular-momentum-based reactive controller (ALIP controller) [5].

In Fig. 16, we compare the maximum allowable impulse the STL-MPC can withstand to that of the baseline ALIP controller. The STL-MPC (captured by the blue region) demonstrates superior perturbation recovery performance across the vast majority of directions and phases, as reflected by the blue region encompassing the red region for the ALIP controller. The improvement is particularly evident for directions around

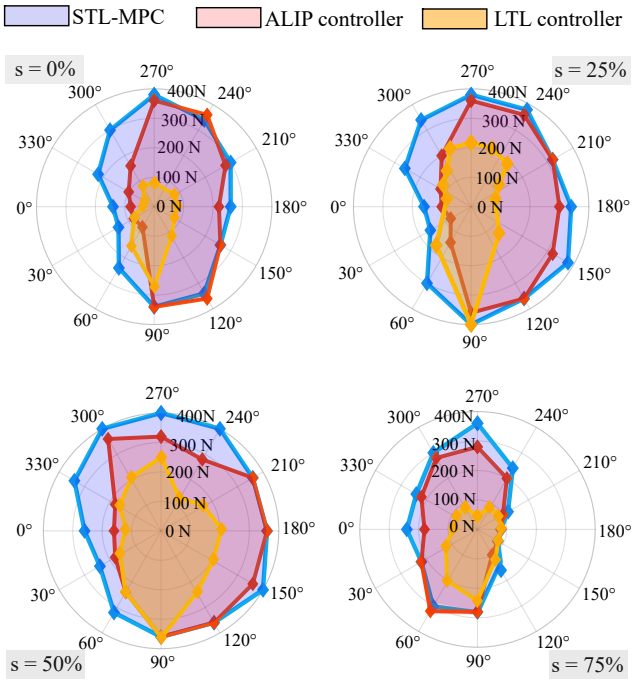

Fig. 16: The maximum allowable force exerted on the pelvis from which the robot can safely recover within two steps in all 12 directions. The perturbations happen at different phases s during a left leg stance. Values on the left half result in crossed-leg maneuvers, and values on the right half correspond to wide-step recoveries. We benchmark with an ALIP controller in Sec. VIII-B and an LTL-based controller in Sec.VIII-F.

$0°$, wherein crossed-leg maneuvers are induced for recovery. This superior performance is due to the STL-MPC's capability to generate safe crossed-leg behaviors via the self-collision avoidance constraints; the ALIP controller, on the other hand, cannot avoid self-leg collisions. For perturbations around $180°$, both frameworks generate wide side-steps for recovery and exhibit comparable performance.

F. Benchmarking with Two Baseline Planners

To highlight the superior performance of our method, this section compares the task planning performance between our STL-MPC, a linear-temporal-logic-based (LTL-based) planning framework [13], and a baseline MPC without STL.

To explore the maximum allowable force the LTL-based method can endure compared with our STL-MPC, we conduct the same omnidirectional perturbations applied in Sec. VIII-E and record the robot's keyframe state after perturbation. We mark the maximum allowable force for the LTL-based method in the yellow regions of Fig. 16. The limits for the LTL-based method are smaller than those of the STL-based method. Any force beyond the maximum allowable perturbation magnitude would cause an LTL planner failure. Compared to the LTL method, our STL-MPC can solve a recovery trajectory even when a perturbed state is outside the Riemannian region, which increases its robustness against perturbations.

In another benchmarking, we compare a traditional TO with our STL-guided TO. STL enables straightforward encoding of logical tasks, such as identifying a keyframe based on a certain system condition and specifying the keyframe to *eventually*

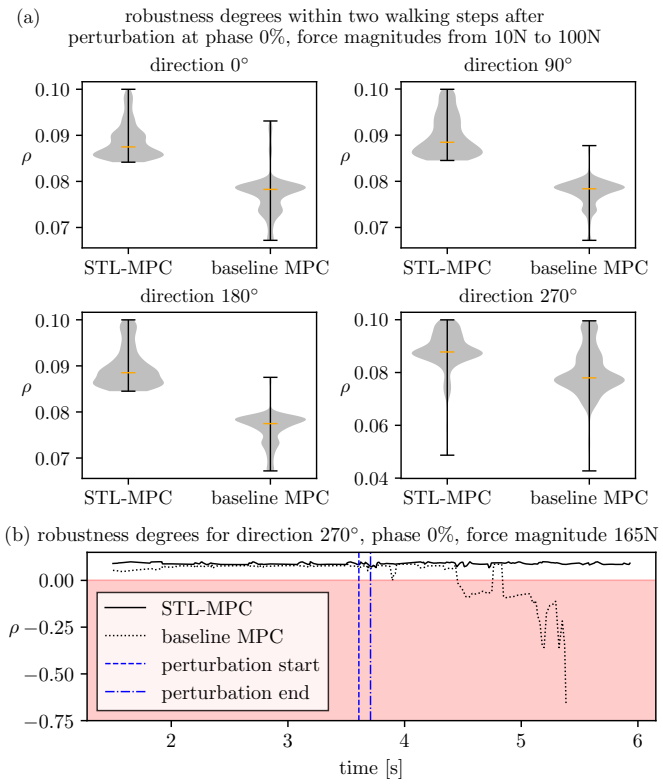

Fig. 17: Comparison of the robustness degrees between the solutions of our STL-MPC and a baseline MPC without STL. (a) Robustness degrees $\rho^{\varphi_{\text{loco}}}$ (a notation of ρ is used in the figure for simplicity) within two walking steps after the perturbation of different magnitudes are collected. Each subfigure shows the results with perturbations from a specific direction. The distribution of the robustness degrees of the two methods is shown in a violin plot with marked minimum, median, and maximum. STL-MPC offers solutions with higher robustness under all directions of perturbations. (b) A specific perturbation case where our STL-MPC recovers successfully while the baseline MPC fails. The region with negative robustness, which indicates a failure to satisfy the specification, has been marked in red.

reach a Riemannian region. Conversely, traditional methods cannot encode such logic-based specifications effectively.

To highlight the performance improvement from the logical task encoding, we implement a baseline MPC. Compared with the STL-MPC, the only difference is that the basedline MPC does not encode STL specifications. Instead, the baseline MPC simply constrains the CoM state at the end of the horizon (i.e., the contact switching time) to be within a predefined box-shaped region. In this experiment, we perturb the Cassie robot in four different directions at locomotion phase 0% for 0.1 s, employing 10 force magnitudes ranging from 10 N to 100 N. The performance of the two planners is assessed via robustness degrees $\rho^{\varphi_{\text{loco}}}$ of the solved trajectory within the two walking steps following the perturbation. The distribution of the collected robustness degrees for each perturbation direction is shown in Fig. 17. The baseline MPC demonstrates a lower robustness degree across all directions, indicating a reduced ability to recover from disturbance. Moreover, we identify scenarios that intuitively showcase the performance difference. For example, the perturbation of 0° with force magnitude 165 N applied at phase 0% leads to a failure of the baseline MPC, whereas STL-MPC manages to recover.

The performance improvement offered by the STL-MPC is

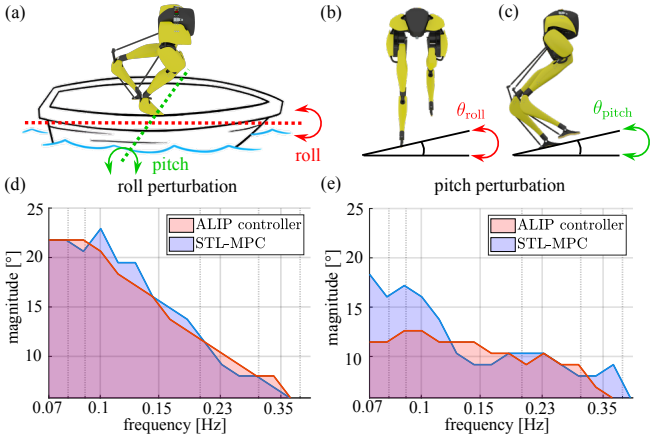

Fig. 18: "Bode Plot" of recovery capability w.r.t dynamic rigid ship surfaces. (a) illustrates the ship motions, where we oscillate the ship surfaces in pitch and roll axes. (b) and (c) shows a different view of the roll and pitch perturbations, respectively. (d) and (e) show the tolerable magnitudes and frequencies of STL-MPC compared with the ALIP baseline along the roll and pitch axis, respectively. Note that, the failure cases are not shown in the figure, and the frequency axis is in the logarithmic scale.

attributed to the direct encoding of the robustness metric into the cost function (6). Moreover, our STL specification enables a broad set of solutions because it allows a flexible keyframe that can be achieved at different time steps given different perturbation scenarios. Although removing the encoding of robustness metric marginally boosts the solving speed to 60 Hz, the standard MPC obtains a lower robustness degree and worse performance. Additionally, the baseline MPC constrains only the final contact-time state, leading to conservative solutions.

G. Orientation Perturbation of Dynamic Rigid Surfaces

To further evaluate the capability of our framework to handle more challenging terrain perturbations, we simulate dynamic rigid surface perturbations [26], [27], as commonly observed from ship motions in a marine environment. This case study allows us to demonstrate our framework's terrain-agnostic recovery ability. The study from [77] reveals the motion of a ship can be characterized by decoupled linear translations and rotational sinusoidal waves. We set up a dynamic rigid surface simulation in the aforementioned MATLAB Simulink and oscillate the ship surface in sinusoidal waves along roll and pitch axes, respectively (as shown in Fig. 18 (a)-(c)). The surface tilting angle follows $\theta_{\text{roll}} = A_{\text{roll}} \sin(2\pi f_{\text{roll}} t)$ and $\theta_{\text{pitch}} = A_{\text{pitch}} \sin(2\pi f_{\text{pitch}} t)$, where $A_{\text{roll}}, A_{\text{pitch}}$ are the magnitude, and $f_{\text{roll}}, f_{\text{pitch}}$ are the oscillation frequency. We combinatorially test 26 different values of A_{pitch}, evenly distributed between 6° and 30°, and 20 different frequencies f_{pitch}, logarithmically distributed between 0.07 Hz and 0.7 Hz. In total, we have 520 distinct oscillations along the pitch axis. The same setup is employed for the roll axis.

We show the simulation results in Fig. 18(d)-(e); we name it "Bode Plot" of recovery capability because we show the magnitude of the maximum allowable terrain angle w.r.t a range of frequencies in the logarithmic scale. The blue and red areas represent the tolerable magnitudes for our STL-MPC and the ALIP controller, respectively. Notably, our method has a larger tolerance to pitch oscillations. The failure cases in

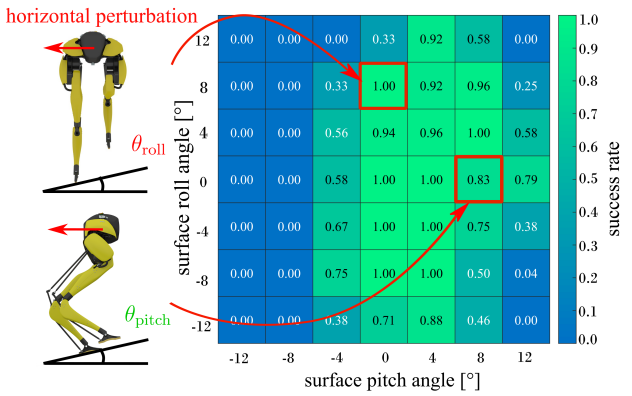

Fig. 19: The success rate matrix of the robot recovering from horizontal perturbations across various stationary surface inclinations. For each surface inclination, perturbations of 100 N are applied from 12 directions at the locomotion phase $s = 0\%$.

pitch oscillation are primarily attributed to the loss of contact when the terrain moves downward in the vertical direction. This case frequently happens when the robot moves far away from the rotation center of the terrain. According to our observations, our STL-MPC framework allows more variation in CoM height to adapt to the terrain height movement and firmly maintain ground contact. However, the ALIP controller is prone to losing contact and slipping. For the roll oscillations, our STL-MPC and the baseline ALIP controller achieve a similar level of performance. This is attributed to the proximity of the footholds to the rotation axis, which reduces the influence of roll oscillations on the CoM velocity, therefore not inducing significant lateral velocity shifts that can be decently handled by our STL-MPC. Instead, most failure cases are attributed to the slip from large tilting roll angles, and the slip does not leverage the advantage of our method. Therefore, both methods achieve a similar performance.

H. Horizontal Perturbation on Inclined Stationary Surfaces

To explore the robustness of our STL-MPC across various terrain orientations, we set up horizontal perturbation experiments on a stationary inclined platform. In each trial, the platform's pitch and roll angles $[\theta_{\text{pitch}}, \theta_{\text{roll}}]$ are fixed to specific values, and a horizontal perturbation is introduced upon stable walking. A total of 49 surface inclination scenarios are tested, with θ_{pitch} and θ_{roll} varying from $-12°$ to $12°$ in $4°$ increments. For each surface inclination, perturbations of 100 N from 12 directions (the same as ones used in Sec. VIII-E) are individually tested. The perturbation recovery success rate for each inclination setup is shown in Fig. 19 as a heat map.

As expected, there is a general trend of declining success rate as the platform's inclination increases. Moreover, the STL-MPC performs better in uphill scenarios than downhill ones, reflected by the cells of higher success rate with positive pitch angles. A downhill scenario poses a greater challenge due to the terrain-agnostic nature of the STL-MPC, which always plans the foot contact position to be at the same height as the stance foot. Consequently, there remains a gap between the actual swing foot height and the ground at the expected contact timing of each walking step, and transitioning to the next walking step requires the controller to blindly step down further to make a contact, potentially leading to kinematic reachability issues or instability due to delayed contact. Additionally, it is observed that the STL-MPC has a satisfactory robust performance to roll inclinations, attributed to its ability to initiate collision-free crossed-leg maneuvers. Overall, the STL-MPC demonstrates robust push recovery performance across a range of surface inclinations.

IX. Hardware Results

A. Omnidirectional Horizontal Perturbation on CAREN

Following the setup of the omnidirectional perturbation simulation in Sec. VIII-E, we conduct a comprehensive hardware experiment on the bipedal robot Cassie to demonstrate the robustness achieved by our signal-temporal-logic-based model predictive controller (STL-MPC). The experiment employs a Computer-Aided Rehabilitation Environment (CAREN) [10], as shown in Fig. 20. The CAREN system comprises a treadmill mounted in a 6-degree-of-freedom Stewart platform, enabling walking surface translations and rotations along all axes.

In our experiments, we systematically apply omnidirectional horizontal terrain perturbations (no terrain height variation). The hardware experiment incorporates a combination of perturbation features: three magnitudes (horizontal translations of 10, 15, and 20 cm), twelve directions, and four locomotion phases. The directions and phases are the same as those used in the simulation. To synchronize the timing of perturbations with Cassie's non-periodic walking phases, we employ a force plate underneath the treadmill to measure the vertical ground reaction forces and detect gait contact events, which are then used to estimate the phase of the walking cycle based on a nominal step duration.

Our STL-MPC successfully recovers from all perturbations with one exception. This outlier involves a perturbation at the 75% phase and $180°$ direction, with the maximum translation distance of 20 cm. This particular case is challenging because it requires a crossed-leg maneuver and a subsequent large wide step. This recovery sequence is infeasible within the treadmill's width, which is a limitation from the spatial layout of the CAREN instead of our framework. Among all successful trials, we show four perturbation recoveries in Fig. 20. These perturbations are applied in four directions at phase $s = 75\%$ of a walking step with a 15 cm translation magnitude.

To further analyze the perturbation response, we illustrate the CoM phase space plots, as depicted in Fig. 21. These plots superimpose the estimated center of mass (CoM) state of the hardware with the STL-MPC's planned CoM trajectory, showcasing both sagittal and lateral phase space for three distinct perturbation scenarios. The first plot shows the CoM's periodic motion in normal walking conditions, where no perturbation is applied. We then examine scenarios with perturbations applied at phase $s = 75\%$ of a walking step, specifically focusing on $0°$ and $180°$ directional perturbations. The $0°$ perturbation induces a crossed-leg maneuver, corresponding to the CAREN moving leftward (as shown in the third row of Fig. 20). The $180°$ perturbation causes the STL-MPC to choose a wide-step recovery strategy. Both scenarios reveal that recovery is achieved within two walking steps, as indicated by the CoM, which returns to a periodic orbit after two walking steps.

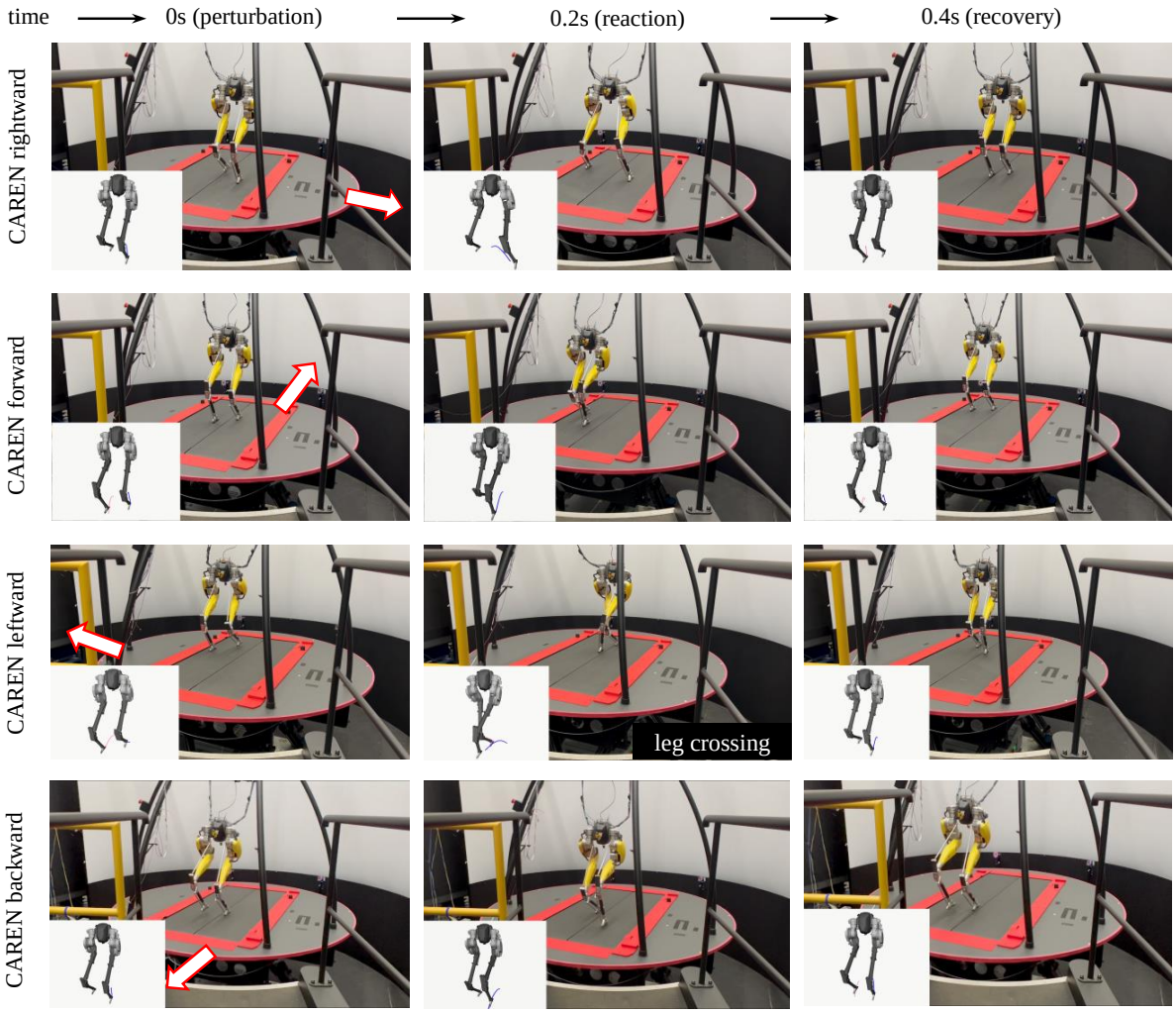

Fig. 20: We test our STL-MPC framework on the CAREN system. The Stewart platform provides perturbations at a controlled timing with a specified direction and magnitude. The Cassie robot recovers from perturbations in all four directions using diverse recovery strategies. For example, the leftward CAREN perturbation (third row) induces a crossed-leg maneuver. The phase-space trajectory of this crossed-leg maneuver is shown in Fig. 21(b).

A noteworthy observation from the 0° perturbation scenario is a sagittal velocity decrease, attributed not to the perturbation itself but to the required crossed-leg maneuver. To avoid collision between legs during the maneuver, the swing foot must be positioned significantly ahead of the stance foot. This foot placement selection compromises the sagittal velocity tracking, but it is necessary to maintain stability in the lateral direction. This scenario shows the complexity of executing such a maneuver and exemplifies the STL-MPC's planning capability to strategically prioritize subtasks (i.e., maintaining sagittal CoM velocity vs. maintaining lateral CoM stability) to achieve overall task success. In contrast, the wide step maneuver does not result in a sagittal velocity decrease, due to the less restrictive kinematic condition in the sagittal plane.

Our results show the STL-MPC framework's robustness against perturbations, particularly for those in lateral directions that induce leg-crossing behaviors.

B. Large Horizontal Perturbation on a BumpEm System

Given the CAREN system's limitation in generating substantial perturbations, we decide to further challenge the STL-MPC framework to explore its boundaries of robustness. To this end, we employ a powerful perturbation emulator, the bump emulation (*BumpEm*) [11], which allows us to generate large and accurate impulses from direct cable pulling. As shown in Fig. 22(a), the pulling direction is predetermined by wiring the cable through the pulley mounted on the handrail. Notably, perturbations from the CAREN and the BumpEm differ fundamentally: the BumpEm exerts forces at the robot's pelvis, whereas the CAREN generates terrain-based disturbances taking effect through the foot-ground contact. Although the BumpEm and CAREN experiments have different perturbation mechanisms aforementioned, we observe similar recovery patterns in both experiments: pulling the pelvis toward one direction induces a similar recovery strategy of moving the CAREN platform in the opposite direction.

We apply horizontal perturbations to Cassie's pelvis in 8 directions at the initial phase $s = 0\%$ of a walking cycle. To prevent over torque the pulling motor, we keep the pulling force small by setting a longer pulling duration of 0.2 s. For each direction, we increase the pulling force until the robot fails to maintain balance. The maximum tolerable force is

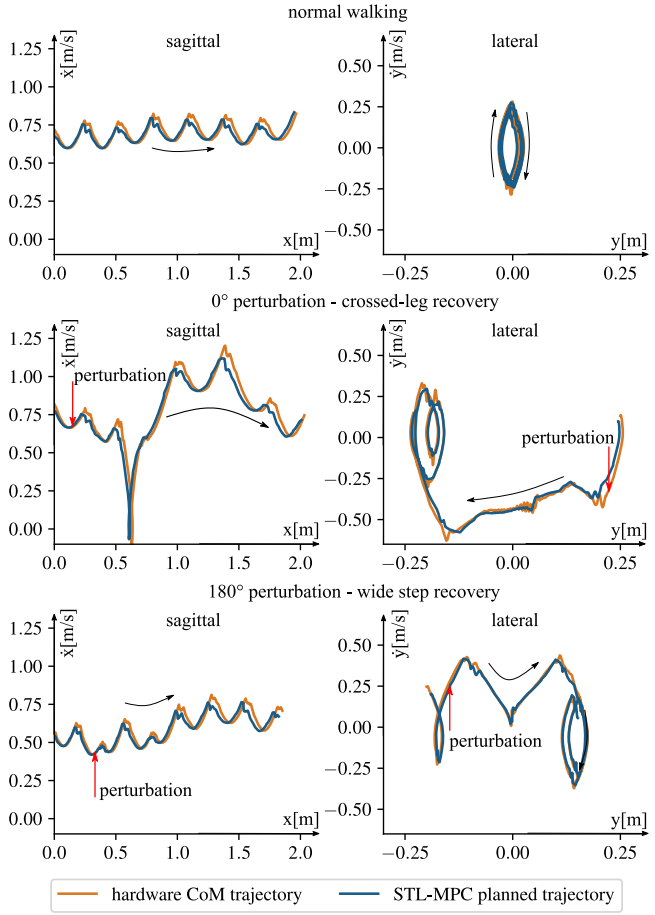

Fig. 21: The planned CoM trajectory by STL-MPC and the real hardware data. The figure shows the control result for 3 CAREN perturbation scenarios: normal walking, left perturbation, and right perturbation.

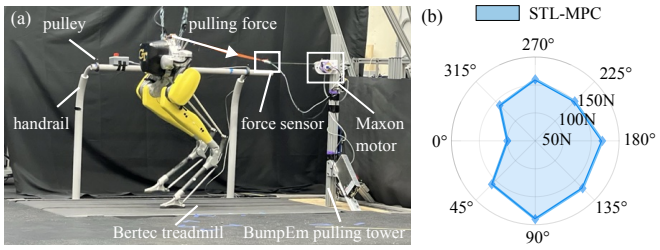

Fig. 22: (a) The experiment setup of the BumpEm. The BumpEm is a pulling tower that comprises a motor pulling the string attached to the Cassie robot. The pulling direction is configured via the routing around the pulley. The force on the string is accurately controlled using the feedback from the force sensor. Each impulse lasts for 0.2 s. (b) The maximum force the STL-MPC can resist for each direction.

illustrated in Fig. 22(b). The maximum tolerable impulses observed on hardware have similar values with the impulses used in simulation in Sec. VIII-E. In simulation, the forces are roughly double the magnitude compared to the ones used on hardware but are applied for half the duration. For BumpEm perturbations in the lateral direction, crossed-leg maneuvers are also observed when we perturb Cassie with 130 N from 0° direction, as shown in Fig. 23.

C. Orientational Perturbation on CAREN

This experiment demonstrates the resilience of our framework against continuous orientational perturbations. Leveraging the CAREN system, we simulate ship-like tilting motions that incorporate dynamic rotations along both the pitch and roll axes, similar to the ones applied in the ship motion simulation in Sec. VIII-G. During the experiment, the roll and pitch axes undergo smooth sinusoidal oscillations while the Cassie robot walks forward on the split-belt treadmill.

Remarkably, our framework withstands up to a 5° oscillation of 0.25 Hz around a single axis, i.e., either roll or pitch axis. Note that, 5° is the limit for safe CAREN operation. To introduce more challenging platform motions, we simultaneously rotate both axes, assigning distinct frequencies for each axis to avoid the synchronization of their sinusoidal waves and enable a more diverse set of motion patterns. Specifically, we set 0.25 Hz for the pitch axis and 0.16 Hz for the roll axis. As shown in Fig. 24(c), our framework is capable of withstanding dual-axis ship motions up to 4°.

The hardware experiments reveal that the robot frequently adopts crossed-leg recovery strategies to resist roll oscillations (Fig. 24(a, c)), underscoring the significance of this maneuver for enhancing the overall robustness. Furthermore, we observe a notable adaptation in step duration that correlates with changes in terrain pitch; the step duration decreases during ascents and increases as the robot descends, showcasing the terrain-agnostic locomotion capability of our STL-MPC.

X. Discussions

Although not detailed in this paper, our planning framework is inherently capable of adapting to rough terrain because both the center of mass (CoM) height and the swing foot height are part of the decision variables in the state vector $\bar{x}$. These decision variables can be optimized in the trajectory optimization (TO) if a terrain height map is known. Navigating rough terrain necessitates a few additional constraints. Specifically, the swing foot's height at the moment of contact will be adjusted to match the terrain's elevation. Also, the model predictive control (MPC) requires an additional kinematic reachability constraint between the CoM and the stance foot, accommodating variations in the CoM height.

For signal temporal logic (STL), the literature presents various methods of incorporating the robustness degree into a TO, as an objective function [58], [68], as a constraint [78], or both [62]. Incorporating the robustness degree as a constraint (5) provides a correct-by-construction property, i.e., the solution is guaranteed to satisfy the specification. However, this approach is more susceptible to numerical failures, especially in scenarios with large disturbances. Consequently, we choose to integrate the robustness degree into the objective function, which yields a satisfactory success rate of the numerical solve. This formulation is practically beneficial for hardware implementations. In large perturbation scenarios, our result generates a slightly violating solution, whereas the constrained formulation fails to solve.

To increase the solving efficiency of the TO, we exploit the warm start feature of the SNOPT solver. This feature allows for an initial guess for the decision variables, wherein we utilize the solution from the preceding problem as the initial guess for the subsequent one. Another solving speed boost

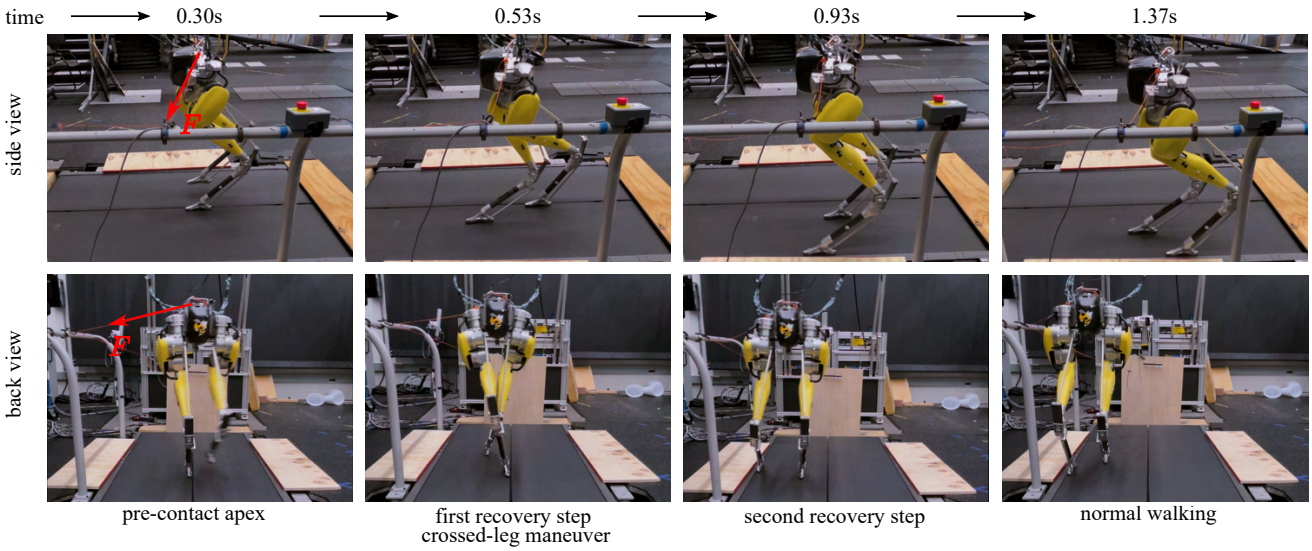

Fig. 23: Cassie recovers from a CoM perturbation applied by the BumpEm pulling tower. The perturbation is of 130 N magnitude at 180° direction. The STL-MPC planner reacts with a crossed-leg maneuver, followed by a wide step, and finally recovers to stable normal walking.

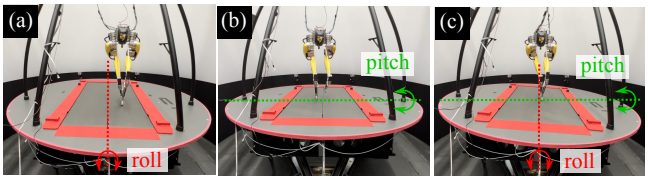

Fig. 24: (a) The roll oscillation up to 5°. (b) The pitch oscillation to 5°. (c) Oscillation along both the roll and pitch axes up to 4°.

is derived from employing the analytical gradients of multi-layer perceptrons. By supplying these analytical gradients, we obviate the need for the solver to approximate the gradient via finite differentiation. The analytical gradient is 100 times faster than the solver's built-in finite differentiation.

One of the limitations of our method lies in the proposed reduced-order model. Although the proposed model enhances the traditional linear inverted pendulum by incorporating the kinematic information of the swing foot, it does not consider the feasibility of the swing-leg dynamics. Therefore, the planner may generate a motion for the swing leg that exceeds the controller's tracking ability. To mitigate this issue, our TO is properly constrained for the swing foot. Namely, we penalize the swing foot displacement between two consecutive time steps to promote trajectory smoothness and impose reachability constraints on the swing foot position to enhance feasibility.

For our future work, we plan to explore diverse recovery strategies for further improvement of locomotion robustness. Research by [43] reveals that humans exhibit jumping behaviors (an agile CoM vertical motion) to counteract severe perturbations. This insight lays the groundwork for our future studies, where we will examine the feasibility of integrating such dynamic recovery strategies into our robotic systems.

XI. Conclusions

This study presented a novel approach to enhancing bipedal locomotion robustness through the integration of signal temporal logic (STL) into a model predictive control (MPC) framework. This framework increased the locomotion performance of Cassie by 81% in terms of maximum impulse tolerance during leg-crossing scenarios. Extensive simulation and hardware experiments verify the robustness of our framework against omnidirectional terrain perturbations. The achieved results demonstrate great potential to apply the proposed STL-guided MPC framework to other robotics fields such as navigation and whole-body loco-manipulation.

Appendix A
Analytical Manifolds for the Riemannian Region

The Riemannian manifolds in Def. III.2 are analytical solutions derived as the linear inverted pendulum model (LIPM) dynamics in (1). The center of mass (CoM) dynamics $\ddot{p}_{\mathrm{CoM,dir}} = \omega^2 p_{\mathrm{CoM,dir}}$, where $\mathrm{dir} = \{x, y\}$ for sagittal and lateral, respectively. We omit the direction dir in the following derivation. Solving the equation above, we derive an analytical solution: $p_{\mathrm{CoM}}(t) = p_{\mathrm{CoM}}(0)\cosh(\omega t) + \frac{1}{\omega} v_{\mathrm{CoM}}(0)\sinh(\omega t)$ and $v_{\mathrm{CoM}}(t) = \omega p_{\mathrm{CoM}}(0)\sinh(\omega t) + v_{\mathrm{CoM}}(0)\cosh(\omega t)$. The $p_{\mathrm{CoM}}(0)$ and $v_{\mathrm{CoM}}(0)$ are the initial CoM state of a nominal walking step, which is shown in Fig. 6, and we represent them as p_0 and v_0. $p_{\mathrm{CoM}}(t)$ and $v_{\mathrm{CoM}}(t)$ are the CoM state at time t, denoted as p_t and v_t.

The solution is represented as:

$$\begin{bmatrix} p_t \\ v_t \end{bmatrix} = \begin{bmatrix} p_0 & v_0/\omega \\ v_0 & \omega p_0 \end{bmatrix} \begin{bmatrix} \cosh(\omega t) \\ \sinh(\omega t) \end{bmatrix} \tag{16}$$

which implies:

$$\begin{bmatrix} \cosh(\omega t) \\ \sinh(\omega t) \end{bmatrix} = \frac{1}{\omega p_0^2 - v_0^2/\omega} \begin{bmatrix} \omega p_0 & v_0/\omega \\ -v_0 & p_0 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \end{bmatrix} \tag{17}$$

From $\cosh(x)^2 - \sinh(x)^2 = 1$, we have $(\omega p_0 p_t - v_0 v_t/\omega)^2 - (-v_0 p_t + p_0 v_t)^2 = (\omega p_0^2 - v_0^2/\omega)^2$. After organizing the terms to one side, we get:

$$0 = p_0^2(2v_0^2 - v_t^2 + \omega^2(p_t^2 - p_0^2)) - v_0^2 p_t^2 + v_0^2(v_t^2 - v_0^2)/\omega^2 =: \sigma \tag{18}$$

The nominal manifold (i.e., $\sigma = 0$) and its initial CoM state p_0, v_0 is a design choice. For our experiment, we choose the

nominal manifold based on a periodic walking gait with apex velocity 0.6 m/s, and p_0, v_0 as the CoM state at the contact-switching time. The σ indicates the deviation of the CoM state p_t, v_t from the nominal manifold.

For the derivation of the cotangent manifold, we use the property that tangent and cotangent manifolds are orthogonal. By taking the derivative of (18), we have: $d\sigma = \frac{\partial \sigma}{\partial p_t} dp_t + \frac{\partial \sigma}{\partial v_t} dv_t$, where $\frac{\partial \sigma}{\partial p_t} = 2p_t(\omega^2 p_0^2 - v_0^2)$ and $\frac{\partial \sigma}{\partial v_t} = -2v_t(p_0^2 - v_0^2/\omega^2)$. The normal vector of σ is calculated through its gradient $(2p_t(\omega^2 p_0^2 - v_0^2), -2v_t(p_0^2 - v_0^2/\omega^2))^T$, which is orthogonal to its tangent vector. Since ζ is orthogonal to σ, the normal vector of ζ is the tangent vector of σ:

$$d\zeta = \frac{\partial \zeta}{\partial p_t} dp_t + \frac{\partial \zeta}{\partial v_t} dv_t$$

where $\frac{\partial \zeta}{\partial p_t} = 2v_t(p_0^2 - v_0^2/\omega^2)$ and $\frac{\partial \zeta}{\partial v_t} = -2p_t(\omega^2 p_0^2 - v_0^2)$. Via the equations above, we further obtain:

$$\frac{dv_t}{dp_t} = -\frac{v_t}{\omega^2 p_t} \Rightarrow \omega^2 \int_{v_0}^{v_t} \frac{dv_t}{v_t} = -\int_{p_0}^{p_t} \frac{dp_t}{p_t}$$

Then we have

$$\ln(\frac{v_t}{v_0})^{\omega^2} + \ln(\frac{p_t}{p_0}) = 0 \Rightarrow (\frac{v_t}{v_0})^{\omega^2} \frac{p_t}{p_0} = 1$$

The cotangent manifold is defined as

$$\zeta := \zeta_0 (\frac{v_t}{v_0})^{\omega^2} \frac{p_t}{p_0}$$

where ζ_0 is a constant nonnegative scaling factor and ζ is the phase progression value. (p_0, v_0) is the initial condition at $\zeta = \zeta_0$.

REFERENCES

[1] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," in *Robotics: Science and Systems*, 2022.

[2] D. Crowley, J. Dao, H. Duan, K. Green, J. Hurst, and A. Fern, "Optimizing bipedal locomotion for the 100m dash with comparison to human running," in *IEEE International Conference on Robotics and Automation*, 2023, pp. 12 205–12 211.

[3] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the darpa robotics challenge," *Journal of Field Robotics*, vol. 32, no. 2, pp. 293–312, 2015.

[4] 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.

[5] 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.

[6] 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 International Conference on Humanoid Robots*, 2022, pp. 558–565.

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

[8] 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.

[9] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed Systems*, K. Chatterjee and T. A. Henzinger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 92–106.

[10] B. Isaacson, T. Swanson, and P. Pasquina, "The use of a computer-assisted research environment (caren) for enhancing wounded warrior rehabilitation regimens," *The journal of spinal cord medicine*, vol. 36, pp. 296–299, 07 2013.

[11] G. R. Tan, M. Raitor, and S. H. Collins, "Bump'em: an open-source, bump-emulation system for studying human balance and gait," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 9093–9099.

[12] Z. Gu, R. Guo, W. Yates, Y. Chen, Y. Zhao, and Y. Zhao, "Walking-by-logic: Signal temporal logic-guided model predictive control for bipedal locomotion resilient to external perturbations," in *International Conference on Robotics and Automation*, 2024.

[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] 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.

[15] B. J. Stephens, "Push recovery control for force-controlled humanoid robots," Ph.D. dissertation, Carnegie Mellon University, 2011.

[16] P.-b. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 137–142.

[17] T. Kamioka, H. Kaneko, M. Kuroda, C. Tanaka, S. Shirokura, M. Takeda, and T. Yoshiike, "Dynamic gait transition between walking, running and hopping for push recovery," in *IEEE-RAS International Conference on Humanoid Robotics*, 2017, pp. 1–8.

[18] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *IEEE-RAS international conference on humanoid robots*, 2006, pp. 200–207.

[19] M. Shafiee, G. Romualdi, S. Dafarra, F. J. A. Chavez, and D. Pucci, "Online dcm trajectory generation for push recovery of torque-controlled humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots*, 2019, pp. 671–678.

[20] J. Englsberger and C. Ott, "Integration of vertical com motion and angular momentum in an extended capture point tracking controller for bipedal walking," in *IEEE-RAS International Conference on Humanoid Robots*, 2012, pp. 183–189.

[21] S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim, "Robust dynamic walking using online foot step optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 5373–5378.

[22] X. Xiong and A. Ames, "3-d underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2405–2425, 2022.

[23] G. Gibson, O. Dosunmu-Ogunbi, Y. Gong, and J. Grizzle, "Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 6724–6731.

[24] H. Chen, Z. Hong, S. Yang, P. M. Wensing, and W. Zhang, "Quadruped capturability and push recovery via a switched-systems characterization of dynamic balance," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2111–2130, 2023.

[25] R. Griffin, J. Foster, S. Pasano, B. Shrewsbury, and S. Bertrand, "Reachability aware capture regions with time adjustment and cross-over for step recovery," in *IEEE-RAS 22nd International Conference on Humanoid Robots*, 2023, pp. 1–8.

[26] A. Iqbal, Y. Gao, and Y. Gu, "Provably stabilizing controllers for quadrupedal robot locomotion on dynamic rigid platforms," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 4, pp. 2035–2044, 2020.

[27] A. Iqbal, S. Veer, and Y. Gu, "Asymptotic stabilization of aperiodic trajectories of a hybrid-linear inverted pendulum walking on a vertically moving surface," in *American Control Conference*, 2023, pp. 3030–3035.

[28] J. E. Pratt, "Exploiting inherent robustness and natural dynamics in the control of bipedal walking robots," Ph.D. dissertation, MIT, 2000.

[29] C. O. Saglam and K. Byl, *Quantifying and Optimizing Robustness of Bipedal Walking Gaits on Rough Terrain*. Springer International Publishing, Jul. 2017, p. 235–251.

[30] H. Dai and R. Tedrake, "L2-gain optimization for robust bipedal walking on unknown terrain," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 3116–3123.

[31] M.-Y. Cheng and C.-S. Lin, "Measurement of robustness for biped locomotion using linearized poincare' map," in *IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, vol. 2, 1995, pp. 1321–1326 vol.2.

[32] 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.
[33] 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.
[34] C. Liu, J. Ning, K. An, and Q. Chen, "Active balance of humanoid movement based on dynamic task-prior system," *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, 2017.
[35] D. Marew, M. Lvovsky, S. Yu, S. Sessions, and D. Kim, "Integration of riemannian motion policy with whole-body control for collision-free legged locomotion," in *IEEE-RAS International Conference on Humanoid Robots*, 2023, pp. 1–8.
[36] 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.
[37] T. Egle, J. Englsberger, and C. Ott, "Step and timing adaptation during online dcm trajectory generation for robust humanoid walking with double support phases," in *IEEE-RAS International Conference on Humanoid Robots*, 2023, pp. 1–8.
[38] 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.
[39] A. Ibanez, P. Bidaud, and V. Padois, "Emergence of humanoid walking behaviors from mixed-integer model predictive control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4014–4021.
[40] M. R. O. A. Maximo, C. H. C. Ribeiro, and R. J. M. Afonso, "Mixed-integer programming for automatic walking step duration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 5399–5404.
[41] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.
[42] 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.
[43] J. K. Leestma, P. R. Golyski, C. R. Smith, G. S. Sawicki, and A. J. Young, "Linking whole-body angular momentum and step placement during perturbed walking," *Journal of Experimental Biology*, 2023.
[44] 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.
[45] J. Li and Q. Nguyen, "Dynamic walking of bipedal robots on uneven stepping stones via adaptive-frequency mpc," *IEEE Control Systems Letters*, vol. 7, pp. 1279–1284, 2023.
[46] S. Dafarra, S. Bertrand, R. J. Griffin, G. Metta, D. Pucci, and J. Pratt, "Non-linear trajectory optimization for large step-ups: Application to the humanoid robot atlas," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 3884–3891.
[47] 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.
[48] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, "Correct, reactive, high-level robot control," *IEEE Robotics and Automation Magazine*, vol. 18, no. 3, pp. 65–74, 2011.
[49] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI Communications*, vol. 29, pp. 151–162, 2016.
[50] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1771–1785, 2013.
[51] Y. Zhao, Y. Li, L. Sentis, U. Topcu, and J. Liu, "Reactive task and motion planning for robust whole-body dynamic locomotion in constrained environments," *The International Journal of Robotics Research*, vol. 41, no. 8, pp. 812–847, 2022.
[52] 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.
[53] A. Shamsah, Z. Gu, J. Warnke, S. Hutchinson, and Y. Zhao, "Integrated task and motion planning for safe legged navigation in partially observable environments," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4913–4934, 2023.
[54] J. Jiang, S. Coogan, and Y. Zhao, "Abstraction-based planning for uncertainty-aware legged navigation," *IEEE Open Journal of Control Systems*, 2023.
[55] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *International Conference on Robotics and Automation*, 2014, pp. 5319–5325.
[56] K. W. Wong, R. Ehlers, and H. Kress-Gazit, "Correct high-level robot behavior in environments with unexpected events," in *Robotics: Science and Systems*, 2014.
[57] 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 *IEEE Conference on Decision and Control*, 2014, pp. 81–87.
[58] V. Kurtz and H. Lin, "Trajectory optimization for high-dimensional nonlinear systems under stl specifications," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1429–1434, 2021.
[59] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, "Fly-by-logic: Control of multi-drone fleets with temporal logic objectives," in *International Conference on Cyber-Physical Systems*, 2018, pp. 186–197.
[60] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *Annual Allerton Conference on Communication, Control, and Computing*, 2015, pp. 772–779.
[61] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *CoRR*, vol. abs/2201.05247, 2022.
[62] 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.
[63] Y. Zhao and L. Sentis, "A three dimensional foot placement planner for locomotion in very rough terrains," in *IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2012, pp. 726–733.
[64] Y. Zhao, B. R. Fernandez, and L. Sentis, "Robust phase-space planning for agile legged locomotion over various terrain topologies." in *Robotics: Science and Systems*, vol. 12, 2016.
[65] 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.
[66] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Systems Letters*, vol. 6, pp. 2635–2640, 2022.
[67] K. Leung, N. Aréchiga, and M. Pavone, "Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," *The International Journal of Robotics Research*, vol. 42, no. 6, pp. 356–370, 2023.
[68] 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.
[69] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.
[70] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019.
[71] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM Rev.*, vol. 47, no. 1, p. 99–131, 2005.
[72] A. L. Hof, S. M. Vermerris, and W. A. Gjaltema, "Balance responses to lateral perturbations in human treadmill walking," *Journal of Experimental Biology*, vol. 213, no. 15, pp. 2655–2664, 08 2010.
[73] T. M. Inc., "Matlab simulink version: 10.3 (r2021a)," 2021.
[74] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.
[75] 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.
[76] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022.
[77] E. V. Lewis, *Principles of Naval Architecture (version 2nd revision.)*. Society of Naval Architects and Marine Engineers, Nov. 1989, vol. 3.
[78] R. Takano, H. Oyama, and M. Yamakita, "Continuous optimization-based task and motion planning with signal temporal logic specifications for sequential manipulation," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 8409–8415.