Reactive Locomotion Decision-Making and Robust Motion Planning for Real-Time Perturbation Recovery

Zhaoyuan Gu, Nathan Boyd, and Ye Zhao

***Abstract*— In this paper, we examine the problem of push recovery for bipedal robot locomotion and present a reactive decision-making and robust planning framework for locomotion resilient to external perturbations. Rejecting perturbations is an essential capability of bipedal robots and has been widely studied in the locomotion literature. However, adversarial disturbances and aggressive turning can lead to negative lateral step width (i.e., crossed-leg scenarios) with unstable motions and self-collision risks. These motion planning problems are computationally difficult and have not been explored under a hierarchically integrated task and motion planning method. We explore a planning and decision-making framework that closely ties linear-temporal-logic-based reactive synthesis with trajectory optimization incorporating the robot's full-body dynamics, kinematics, and leg collision avoidance constraints. Between the high-level discrete symbolic decision-making and the low-level continuous motion planning, behavior trees serve as a reactive interface to handle perturbations occurring at any time of the locomotion process. Our experimental results show the efficacy of our method in generating resilient recovery behaviors in response to diverse perturbations from any direction with bounded magnitudes.**

I. Introduction

As legged robots are increasingly deployed in complex environments, the need for robots to accomplish tasks through symbolic planning and decision-making becomes more apparent. Although locomotion robustness has been extensively explored at the motion planning level, resilience to uncertainties and external disturbances at the task planning level has been largely overlooked. Hierarchically integrated task and motion planning (TAMP) is capable of handling logical and whole-body dynamics objectives simultaneously. Unexpected errors or even failures at the lower-level can lead to expensive re-planning at the higher task planning level. On the other hand, high-level discrete task plans can result in infeasible low-level motion plans. With these cascading effects, novel TAMP methods are imperative to make robust locomotion decisions resilient to environmental perturbations and enable robots to efficiently recompute plans at both task and motion planning levels.

At the motion planning level, push recovery of bipedal locomotion has been extensively studied in previous works and inspired by human locomotion biomechanics [1], [2]. Various strategies such as hip, ankle, and foot placement strategies are proposed to handle external perturbations [3]–[5]. However, many of these push recovery strategies employ reduced-order models (RoMs) such as inverted pendulum or centroidal momentum models, making it difficult to guarantee leg self-collision avoidance. This challenge arises from solving full-leg kinematic constraints in these RoMs. It is a strong assumption to state that a robot will never be in close contact with itself in highly dynamic locomotion. Liu et al. [6] demonstrated a complete control framework that considers self-collision under various disturbances, but does not consider more complicated multi-step or non-periodic recoveries. Reactive approaches for high dimensional robots have also been explored [7]–[9], which rely on a distance metric to generate safe repulsive motions, but can lead to significant motion plan discrepancies. Behavior libraries have also been used to generate robust real-time walking in unstructured or constrained environments [10], [11]. In addition, few motion planning strategies incorporate higher-level task planning.

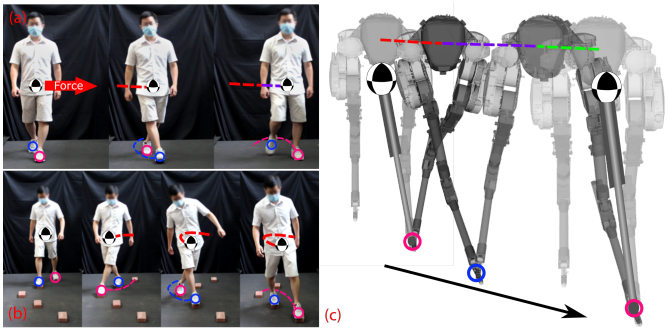

Fig. 1: a) Human is forced to cross legs to recover from an external disturbance. b) Human must execute leg crossing to traverse stepping stones. c) An illustration of recovery motion of bipedal robot Cassie.

In high-level task planning, reactivity is critical to account for environmental changes at runtime. Temporal-logic-based reactive synthesis [12]–[14] has been widely explored to find strategies that generate formally-guaranteed safe and provably correct robot actions in response to environmental events. However, this method has been under-explored for dynamic locomotion problems until recent years. Recent works [15]–[17] adopted linear temporal logic (LTL) to synthesize reactive locomotion navigation plans over rough terrains. However, the feasibility of executing synthesized task plans on high degree-of-freedom legged robots is unexplored. To address this challenge, this study leverages trajectory optimization (TO) as the low-level motion planner to verify the synthesized task feasibility.

Behavior Trees (BTs), as graphical mathematical models,

The authors are with the Laboratory for Intelligent Decision and Autonomous Robots, Woodruff School of Mechanical Engineering, Georgia Institute of Technology. {zgu78, nboyd31, yzhao301}@gatech.edu

This work was funded by the NSF grant # IIS-1924978 and Georgia Tech Institute for Robotics and Intelligent Machines Seed Grant.

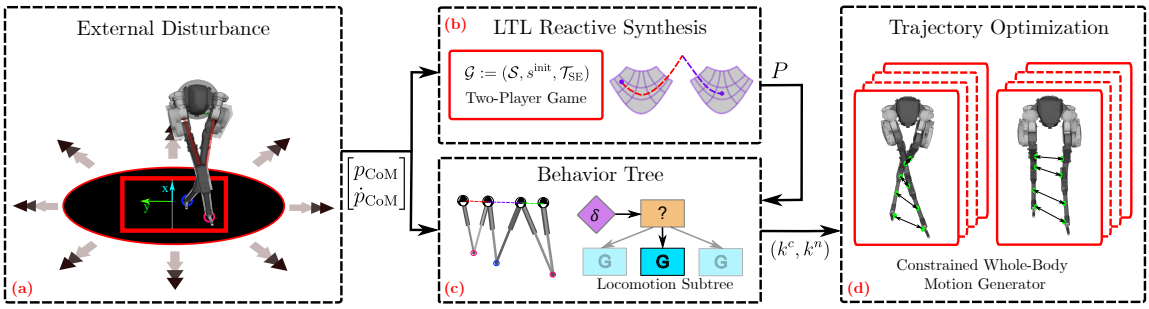

Fig. 2: Block diagram of the proposed framework. a) Experiments of Cassie disturbed during stable walking; b) The high-level task planner synthesis, employing an LTL two-player game; c) The BTs act as a middle layer that reactively execute subtrees based on real-time environmental disturbances; d) A whole-body motion planner is used to generate feasible motions and refine LTL specifications ψ. The high-level task planner and the phase-space planner are integrated in an *online* fashion as shown by the solid black arrows.

have been widely explored to schedule autonomous tasks and handle unexpected environmental changes [18], [19]. Their reactive and modular structure can authorize multiple behavioral plans and achieve fault-tolerant task executions [20], [21]. [22] devised finite state machine (FSM) controllers for unexpected terrain height variation, but relied on large handmade state machines. Intuitively speaking, BTs can be viewed as a feature-rich, acyclic version of FSM for complex behavior execution. Recent LTL-based reactive synthesis work [15] proposed reactive TAMP in combination with robust reachability analysis for dynamic maneuvers and disturbance rejection, but only accounts for perturbations applied at specific instances. Formal methods handling perturbation at any locomotion phase require further investigation. The BTs naturally handle the continuous environmental perturbations by designing actions online to amend the synthesized discrete automaton.

This study addresses the push recovery problem for legged robots subject to external perturbations that can happen anytime. We propose a combined TAMP framework composed of hierarchical planning layers operating at different temporal and spatial scales (Fig. 2). First, the LTL planning designs safety-guaranteed decisions on keyframe states, including center of mass (CoM) state or foot placements, in response to the keyframe perturbations. When perturbations occur at non-keyframe instants, analytical Riemannian manifolds are used to recalculate a new keyframe transition online for the current walking step. BTs are integrated to allow updated keyframes to be any continuous value within the allowable range, instead of a finite set of discrete values quantified in the LTL-based planner. Finally, full-body legged motions are generated using kinodynamic-aware TO for non-periodic multi-step locomotion with self-collision constraints.

The core contributions of this paper are summarized as:

- We present a hierarchically integrated LTL-BT TAMP framework for dynamic locomotion that reacts to environmental changes or new temporal logic specifications
- Our LTL-BT planner is capable of reacting to continuous environmental perturbations for resilient task execution.
- We employ Riemannian manifolds to quantify locomotion keyframe robustness margins and design robust transitions enabled by the reactive task planner.
- We propose a collision-aware, kinodynamic TO that generates collision-free and non-periodic full-body motions and use this TO to refine feasibility specifications in reactive synthesis.

II. Planning Methods

This section details the symbolic decision-making and motion planning framework (Fig. 2). Our hierarchical reactive framework is composed of (i) LTL-level reactive synthesis handling perturbations at keyframe instants, (ii) BT for robust execution of one walking step (OWS) between keyframe instances, (iii) full-body motion primitive generation from kinodynamic-aware TO.

A. Keyframe-based Non-periodic Locomotion

To define a multi-step walking motion for bipedal robot walking, we separate the entire trajectory into multiple OWS phases that start and end at a keyframe state [23]. The i^{th} OWS cycle can be represented by a discrete keyframe transition pair (k^i, k^{i+1}). The keyframe contains the sagittal and lateral CoM apex state, as well as the stance foot index (Sec. II-B). Given two consecutive keyframe states in the sagittal plane, forward and backward numerical integration is used to solve for the contact switching time of OWS. The lateral keyframe transition is determined once the time (t_1 and t_2 for the first-half and second-half OWS phases, respectively) is found, which gives the CoM state $(p_{\text{switch}}, \dot{p}_{\text{switch}})$ at contact switch instant. The next foot placement can be computed in the lateral direction from the current keyframe with the analytical solution:

$$p_{\text{foot}} = p_{\text{switch}} + \frac{(e^{2\omega_{\text{asym}} t_2} - 1)\dot{p}_{\text{switch}}}{(e^{2\omega_{\text{asym}} t_2} + 1)\omega_{\text{asym}}} \tag{1}$$

where the asymptote slope $\omega_{\text{asym}} = \sqrt{g/h_{\text{apex}}}$ and h_{apex} is the relative apex CoM height with respect to the stance foot height. g is the gravity constant. Detailed derivations can be referred in the work of [23].

Compared to periodic walking, where the robot repeats the same motion pattern periodically, keyframe-based walking allows for non-periodic walking that better accommodates rough terrain and environment disturbances. A keyframe decision maker is thus necessary to design safe and feasible keyframe transitions at runtime.

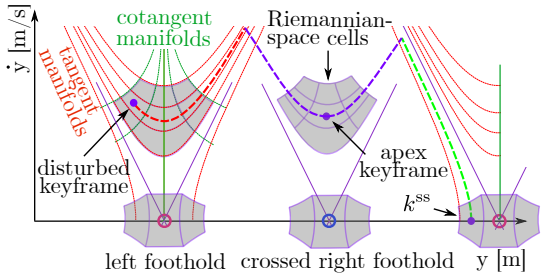

Fig. 3: An illustration of a phase-space Riemannian partition and non-deterministic lateral keyframe transition for disturbance recovery.

B. LTL Specifications for Push Recovery

As the complexity of locomotion tasks increases, making safe decisions on keyframe states to recovery from push becomes intricate. To address this challenge, we employ reactive synthesis, which is built upon task specifications and an abstraction of dynamical systems [13], [24]. The tasks are represented by LTL specifications, which describe temporal and logical relations of the system properties. The abstraction (i.e., transition system) is a discrete description of the system and environment dynamics. An LTL formula operates over atomic propositions (APs) that can be True ($\varphi \vee \neg\varphi$) or False ($\neg\mathsf{True}$). The formulas use logical symbols of negation ($\neg$), disjunction ($\vee$), and conjunction ($\wedge$). Temporal operators such as "next" ($\bigcirc$), "eventually" ($\Diamond$), and "always" ($\Box$) are used as extensions to the propositional logic. Detailed LTL semantics are omitted due to space limit and can be found in [25].

To formally guarantee locomotion task completion under environmental disturbances, we adopt the General Reactivity of Rank 1 (GR(1)) [26], a fragment of LTL. GR(1) provides correct-by-construction guarantees of the realizability of LTL specifications. Provided a transition system $\mathcal{T}_{\mathrm{SE}}$ and LTL specification ψ, the reactive synthesis problem aims for a winning strategy for the robot system such that the execution path satisfies ψ [17]. If the specification is realizable, an automaton will be constructed and provide correct transitions for any environmental actions obeying the assumptions.

Definition 2.1 (Riemannian partition): The transition system discretizes the continuous robot state space (i.e., robot's CoM phase-space near the apex state) into Riemannian partitions defined as:

$$\begin{aligned}\mathcal{R} &\coloneqq \mathcal{R}_{\mathrm{position}} \cup \mathcal{R}_{\mathrm{velocity}} \\ &= \{r_{p,n}, r_{p,z}, r_{p,p}\} \cup \{r_{v,z}, r_{v,s}, r_{v,m}, r_{v,f}\}\end{aligned}$$

where the elements in $\mathcal{R}_{\mathrm{position}}$ define the relative position (negative, zero, positive) of CoM with respect to the stance foot frame, and $\mathcal{R}_{\mathrm{velocity}}$ defines the CoM apex velocity (zero, slow, medium, fast). Riemannian partitions are defined for both sagittal and lateral phase-space, each constitutes 12 cells.

Fig. 3 shows a disturbed keyframe state $(r_{p,n}, r_{v,m})$, which represents a negative position and medium velocity. A keyframe state whose CoM velocity is zero in sagittal axis is noted as $(r_p)_s = (r_{p,z})_s$. The Riemannian partitions use the analytical manifolds of CoM dynamics derived from the Prismatic Inverted Pendulum Model (PIPM) and align the robustness margin sets with the locomotion dynamics. More details will be introduced in Sec. II-E.

Definition 2.2 (Locomotion keyframe): A keyframe $\mathcal{K}$ is defined as a system apex state composed of the sagittal partition $\mathcal{R}_s$, the lateral partition $\mathcal{R}_l$, as well as the stance foot index set $\mathcal{F}_{\mathrm{st}} = \{\mathsf{left}, \mathsf{right}\}$ (used to identify the leg crossing or wider lateral step strategies).

$$\mathcal{K} \coloneqq \mathcal{R}_s \cup \mathcal{R}_l \cup \mathcal{F}_{\mathrm{st}}.$$

The system takes actions $a_{\mathrm{sys}} \in \mathcal{A}_{\mathrm{sys}} \subseteq \mathcal{R}_s \cup \mathcal{R}_l \cup \mathcal{L} \cup \mathcal{W}$ to decide the next keyframe state k^n. $\mathcal{L} = \{\mathsf{small}, \mathsf{medium}, \mathsf{large}\}$ and $\mathcal{W} = \{\mathsf{small}, \mathsf{medium}, \mathsf{large}\}$ represent the step length and width. $l \in \mathcal{L}$ and $w \in \mathcal{W}$ are the distances between the current and the next foot placements projected on sagittal and lateral axis. Note that $\mathcal{R}_s, \mathcal{R}_l$ are the relative CoM apex state in foot frame while $\mathcal{L}, \mathcal{W}$ are the global distances between footholds. Together, they completely define a transition action.

The environment state is represented by a perturbation set $p_{\mathrm{env}} \in \mathcal{P}_{\mathrm{env}} \coloneqq \mathcal{R}_s \cup \mathcal{R}_l \cup \{\emptyset\}$ that pushes the system to a specific Riemannian cell center. In the task planner, we assume that the environment action is a perturbation only applied at a keyframe instant. The perturbation induces a CoM position and velocity jump after applying an external force to the robot's pelvis frame. The environment can also choose to not perturb, i.e., $p_{\mathrm{env}} = \emptyset$. The system action $\mathcal{A}_{\mathrm{sys}}$ and environment action $\mathcal{P}_{\mathrm{env}}$ together decides the next apex keyframe state $k^n = \mathcal{T}_{\mathrm{SE}}(k^c, a_{\mathrm{sys}}, p_{\mathrm{env}})$. Both actions are a part of the automaton state $\mathcal{S}$.

Definition 2.3 (Steady state keyframe): A special set of keyframes are defined as steady state keyframes $k^{\mathrm{ss}} \in \mathcal{K}^{\mathrm{ss}}$ during perturbation-free walking.

$$\mathcal{K}^{\mathrm{ss}} = \{k^{\mathrm{ss}} | k^{\mathrm{ss}} = \big((r_{p,z}, r_{v,\cdot})_s, (r_{p,\cdot}, r_{v,z})_l, f_{\mathrm{st}}\big)\}$$

where $(r_{p,z}, r_{v,\cdot})_s$ means that the sagittal CoM apex position is on top of the nominal foot placement and can take any allowable sagittal velocities, while $(r_{p,\cdot}, r_{v,z})_l$ means that the lateral CoM apex position can take any values and the apex velocity has to be zero $r_{v,z}$ (see Fig. 3).

Let the system start from a steady state $k^{\mathrm{ss}}_{\mathrm{sys}} = ((r_{p,z}, r_{v,m})_s, (r_{p,z}, r_{v,z})_l, \mathsf{right})$. We have

$$\begin{aligned}s^{\mathrm{init}} &= (k^{\mathrm{init}}, a^{\mathrm{init}}_{\mathrm{sys}}, p^{\mathrm{init}}_{\mathrm{env}}) \\ &= \big(k^{\mathrm{ss}}, ((r_{p,z}, r_{v,m})_s, (r_{p,z}, r_{v,z})_l), \emptyset\big)\end{aligned}$$

The robot chooses to maintain stable walking so long as there is no perturbation from the environment. In the presence of perturbations, the keyframe state returns to a steady state within two steps:

$$\Box\big(k = \neg k^{\mathrm{ss}} \Rightarrow (\bigcirc k = k^{\mathrm{ss}}) \vee (\bigcirc \bigcirc k = k^{\mathrm{ss}})\big)$$

The feasibility of transitions must be verified by the low-level full-body TO (Sec. II-F). Certain high-level transitions should be removed due to infeasible full-body kinematics and dynamics constraints. In this way, we define a set of TO-refined task specifications. For example, after the TO refinement, we obtain all full-body-dynamics-feasible

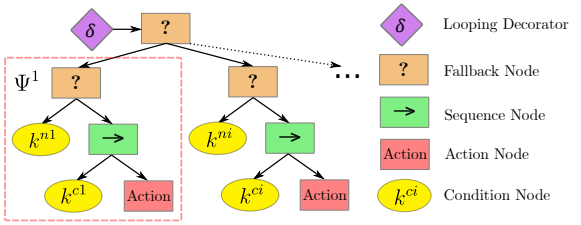

Fig. 4: An illustration of the PABT structure. The PABT groups a set of locomotion subtrees Ψ^i. Each subtree is a fallback tree that encodes a keyframe transition $(k^{c,i}, k^{n,i})$ and a Riemannian recalculation action.

transitions offline and encode TO-refined specifications. An example of TO-refined specification can be:

$$\begin{aligned}\square\big(&k = ((r_{p,z}, r_{v,m})_s, (r_{p,z}, r_{v,m})_l, \mathsf{right}) \Rightarrow \\ &a = ((r_{p,z}, r_{v,m})_s, (r_{p,z}, r_{v,s})_l, \mathsf{small}, \mathsf{small}) \\ &\quad\ldots \\ &\vee a = ((r_{p,z}, r_{v,m})_s, (r_{p,z}, r_{v,m})_l, \mathsf{small}, \mathsf{medium})\big)\end{aligned}$$

In the presence of perturbations, recovering to a steady state k^{ss} requires the next keyframe k^n to decrease the lateral apex velocity and minimizes the sagittal apex deviation from its normal value. For example, a current medium apex velocity indicates the next apex velocity is either medium, small or zero: $\square\big(r_v = r_{v,m} \Rightarrow (\bigcirc r_v = r_{v,m} \vee r_{v,s} \vee r_{v,z})\big)$. A smaller step width $w \in \mathcal{W}$ will be chosen rather than larger ones. $\square\big((w = \mathsf{small} \vee w = \mathsf{large}) \Rightarrow (\bigcirc w = \mathsf{small})\big)$.

For the recovery motion execution not to be interrupted, we assume the environment perturbation happens at most once per two steps: $\square\big(p_{\text{env}} = \neg\emptyset \Rightarrow (\bigcirc p_{\text{env}} = \emptyset)\big)$

C. Task Planner Synthesis

Given the LTL specifications above, the task planner models the robot system and the environment interplay as a two-player game. We construct the keyframe transition game structure in the form of a tuple $\mathcal{G} := (\mathcal{S}, s^{\text{init}}, \mathcal{T}_{\text{SE}})$ with:

- $\mathcal{S} = \mathcal{K} \times \mathcal{A}_{\text{sys}} \times \mathcal{P}_{\text{env}}$ is the possible automaton state of the transition system,
- $s^{\text{init}} = (k^{\text{init}}, a^{\text{init}}_{\text{sys}}, p^{\text{init}}_{\text{env}})$ is the initial automaton state and
- $\mathcal{T}_{\text{SE}} \subseteq \mathcal{S} \times \mathcal{S}$ is a transition describing the possible moves of the robot system and antagonist environment.

In the extreme case where the disturbance is towards the stance leg (see Fig. 1), the foot placement of the swing leg would naturally move closer to the stance foothold location or require a crossed-leg motion. A minimum of two steps is required to recover, during which self-collision poses a challenge for making safe decisions. The TO-refined transition specifications guarantee that the task planner makes dynamics-informed decisions on keyframe transitions. By construction, the TO indicates that all the constraints on the full-body motion are fulfilled and a keyframe transition is feasible.

At each keyframe instant, the decision maker uses the estimated current system keyframe state k^c and plans a sequence of transitions until the final state $k^f = k^{\text{ss}}$. The action roll out produces an action plan $P = \{k^c, \ldots, k^f\}$.

Algorithm 1: Keyframe Decision Making and PABT Execution

Input: PABT Ψ, Decision Maker DM, current $time$;
Set: status = success;
while *status* == success **do**
  $k^c, time$ = StateEstimation();
  if $time$ == keyframe_instant **then**
    $P = DM(k^c)$;
    for (k^c, k^n) in P **do**
      Ψ^c = LocomotionSubtree(k^c, k^n);
      Ψ.Insert(Ψ^c);
    end
  end
  /* PABT Riemannian Recalculation */
  $status$ = Ψ.Tick();
  $(k^c, k^n)'$ = Ψ.GetModifiedTransition();
end
Output: updated PABT Ψ, modified keyframe transition $(k^c, k^n)'$;

D. Behavior-Tree-Based Dynamic Replanning

To address continuous perturbations at non-keyframe instants, we propose a perturbation-aware behavior tree (PABT) that online modifies the desired keyframe transition $(k^{c,d}, k^{n,d})$ received from the high-level reactive synthesis. The PABT complements the reactive synthesis by locally modifying the keyframe transitions, given the real-time captured CoM state $(p_{\text{CoM}}, \dot{p}_{\text{CoM}})$.

The PABT groups a set of locomotion subtrees $\Psi = \bigcup_i \Psi^i$. Each Ψ^i encodes a pair of the current-to-next keyframe states $(k^{c,i}, k^{n,i})$. These pairs are represented as condition nodes in the locomotion subtrees (Fig. 4). The locomotion subtrees are fallback BTs that execute their action nodes when the desired keyframe transition from the high-level matches their condition nodes. For instance, the pre-condition nodes check if the desired transition $k^{c,d}$ matches with their keyframe condition $k^{c,i}$, the same for the post-condition nodes.

The PABT modifies its keyframe transitions locally to handle non-keyframe perturbations. After the modification, the desired keyframe transition remains feasible despite the CoM state deviation. The action node A^i can also be a keyframe recalculation procedure. Here we use the recovery strategy [23] to perform a Riemannian recalculation, which resolves the motion plan when the CoM state is perturbed off from the nominal manifold. Specifically, the PABT uses a position guard strategy (Sec. II-E) to recalculate the keyframe state. It is worth noting that the modified keyframe state may not be a Riemannian cell center or even end up in a different cell, which is suitable for continuous perturbation recovery. The LTL keyframes are always planned from the Riemannian cell center, but local BT modifications will handle the discrepancy.

The PABT grows as the new action plan P is commanded from the task planner. The PABT constructs new subtrees Ψ^c that represent the transitions (k^c, k^n) from P. The new

subtrees are inserted under the root node as new behaviors. A tick of the PABT will trigger the corresponding subtree that matches the subtree conditions. The PABT expansion and execution process is illustrated in Alg. 1.

E. Riemannian Robustness Margin Design

To quantify robustness margin, we use the Riemannian distance metric proposed in [23] to measure the deviation of CoM state from the nominal CoM manifolds in the CoM phase-space. This Riemannian metric discretizes the phase-space with tangent and cotangent locomotion manifolds, instead of using naïve Euclidean-type discretization. The tangent and cotangent manifolds comply with the PIPM locomotion dynamics and provide an intuitive trajectory re-calculation strategy for CoM deviation. Detailed derivations of Riemannian manifolds can be found in the work of [23].

We use the position guard strategy to recalculate the next CoM apex state. Assuming the CoM state jumps to $(p'_{\text{CoM}}, \dot{p}'_{\text{CoM}})$ on a new tangent manifold σ', the recalculated next CoM apex state is:

$$(p_{\text{apex}}, \dot{p}_{\text{apex}}) = (p_{\text{foot}}, \sqrt{\frac{\dot{p}'^2_{\text{CoM}} \pm \sqrt{\dot{p}'^4_{\text{CoM}} - 4\omega^2_{\text{asym}}\sigma'}}{2}}) \tag{2}$$

Note that the $(p_{\text{apex}}, \dot{p}_{\text{apex}})$ corresponds to the next keyframe k^n at the LTL level. The motion primitive set interpolates a full-body motion that connects the current CoM state to the updated next keyframe.

F. Collision-Aware Kinodynamic Trajectory Optimization

The task planner and PABTs generate keyframe transitions robust to perturbations. However, mapping the transitions to whole-body trajectories in real-time often poses a challenge due to the curse of dimensionality. To address this, we use TO to create a set of motion primitives offline. The TO generates desired motions that satisfy the physical constraints while minimizing the trajectory cost [27]–[29]. The TO is also used as a verification to check the feasibility of high-level keyframe transitions. The nonlinear program (NLP) of TO is formulated as:

$$\begin{aligned}
\arg\min_{X} \quad & \sum_{j=1}^{D}\sum_{i=0}^{N_j} \Omega_j \cdot \mathcal{L}_j(x_i^j, u_i^j) && (3)\\
\text{s.t.} \quad & H_j(x_i^j)\dot{x}_i + V_j(x_i^j) + G_j(x_i^j) = u_i, && \text{(dynamics)}\\
& x_0^{j+1} = \Delta_j(x_{N_j}^j), && \text{(reset map)}\\
& \lambda_{c,z} \geq 0,\ |\lambda_{c,xy}| \leq \mu\lambda_{c,z}, && \text{(friction)}\\
& C_j^{\text{kin}}(x_i^j) \leq 0, && \text{(kinematics)}\\
& C_j^{\text{col}}(x_i^j) \leq 0, && \text{(self-collision)}\\
& C_j^{\text{key}}(x_i^j, u_i^j) = 0 && \text{(keyframe boundary)}
\end{aligned}$$

where the domains include $D = 2$ continuous single stance phases and 1 velocity reset map. Each single stance contains N_j nodes, which represents a state-control pair $n_i^j = (x_i^j, u_i^j)$ at the i^{th} instant; the state represents the $x = [q; \dot{q}]$ with q denoting the robot generalized coordinate states. The NLP above solves the optimal state-control trajectory $X^* = \{n_i^{j*}\}$ by minimizing the pseudo energy $\mathcal{L}(\cdot) = ||u_i||^2$ with weights Ω_j while enforcing the physical constraints of the robot.

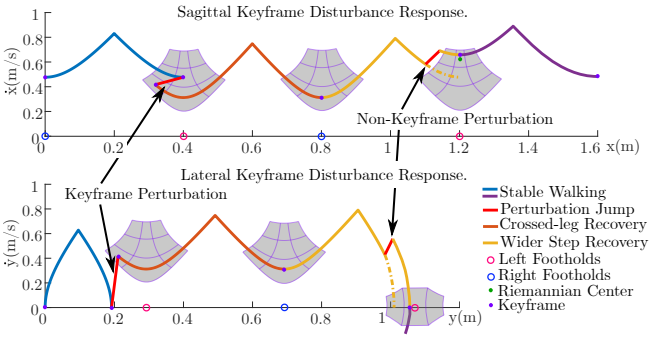

Fig. 5: Lateral and sagittal responses to diagonal disturbances at keyframe and non-keyframe instants while walking at 0.5 m/s apex velocity. Each color represents a single step generated by the LTL-BT.

The physical constraints shape the resultant trajectory. The dynamics constraint is enforced between node points using Hermite-Simpson collocation. H, V, and G denote the inertia, coriolis, and gravity matrices of the robot's rigid body dynamics. $x_0^{j+1} = \Delta_j(x_{N_j}^j)$ maps the velocity jump between two consecutive stance modes. The horizontal contact forces $\lambda_{c,xy}$ are bounded by a linearized friction cone. The kinematics constraints $C_j^{\text{kin}}(x_i^j)$ ensure that the joint angles, foot positions, and CoM trajectories are bounded. M geometric point pairs $(\boldsymbol{g}_l^m, \boldsymbol{g}_r^m)$ on two legs are selected as self-collision constraints (see Fig. 2d). The signed distances are evaluated at each geometric point pair using forward kinematics $FK_{\boldsymbol{g}^m}(x_i)$ for all $m \in M$, $i \in N_j$.

$$\begin{aligned}
\boldsymbol{d}^m(x_i) &= FK_{\boldsymbol{g}_l^m}(x_i) - FK_{\boldsymbol{g}_r^m}(x_i),\\
C_j^{\text{col}}(x_i) &= ||\boldsymbol{d}_{\min}^m||_2^2 - ||\boldsymbol{d}^m(x_i)||_2^2.
\end{aligned} \tag{4}$$

The keyframe transition (k^c, k^n) commanded from the task planner is enforced as a boundary condition for the foot placement, the apex CoM position and velocity.

III. Results

To demonstrate the robustness of the proposed methods, we have tested various scenarios in simulation using Matlab Simulink with a bipedal robot, Cassie. The kinematics and dynamics functions were generated using the Fast Robot Optimization and Simulation Toolkit (FROST) [28]. The NLP solver IPOPT [30] solved the TO problems detailed in Eq. 3. Our framework, together with a virtual constraint controller [11], ran at a rate of 2kHz online. Impulse forces were measured through near discontinuous changes in CoM velocity. As for the task planner, the SLUGS reactive synthesis toolbox [31] was used to design LTL specifications with APs and synthesize the keyframe-based automaton.

For our crossed-leg experimentation, we used the 9 partitions with non-zero apex velocities for $\mathcal{R}_s^c$, $\mathcal{R}_l^c$ and $\mathcal{R}_s^n$, respectively. For each (r_s^c, r_l^c, r_s^n) pair, we used phase-space planning to find the next allowable r_l^n. This Riemannian abstraction provided $9 \times 9 \times 9 = 729$ possible crossed-leg transitions prior to the full-body TO. We evaluated the feasible transitions and automatically generated specifications

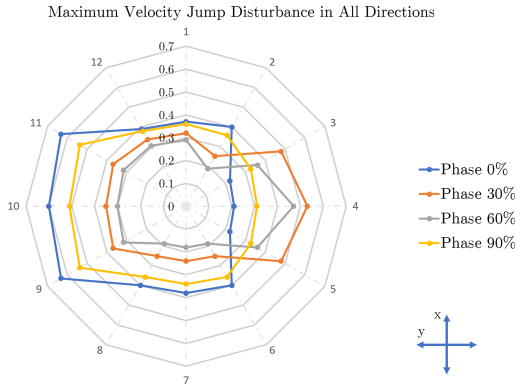

Fig. 6: Maximum allowable velocity change exerted on the CoM for a single step at 30° increments. Values on the left half resulted in single wider step recoveries and values on the right half require crossed-leg maneuvers.

into structured SLUGS files. These specifications encoded the high-level decisions of feasible keyframe transitions. For stable walking and wider step recovery scenario, the lateral r_l^s and r_l^n were the bottom three Riemannian partitions.

We evaluated the performance of our framework through multiple push recovery studies. As shown in Fig. 5, the system was capable of composing multiple OWS trajectories according to the reactive synthesis plan. The robot was firstly disturbed to the non-apex velocity $(0.4, 0.4)$ m/s at keyframe instant. The keyframe decision maker planned a two-step recovery strategy that took one crossed-leg step and one succeeding wider step to come back to a steady state k^{ss}. Disturbances at non-keyframe states required the robot to recalculate a new CoM trajectory to an updated keyframe state. The PABT locally modified the desired keyframe transition and allowed the transitions to start and terminate in non-Riemannian-cell-centers. The reactive synthesis could still update the keyframe transitions as long as the CoM state was inside the Riemannian robustness bound (the grey areas in Fig. 5). This preserved the notion of continuous recovery rather than that only from a finite set of discrete keyframe transitions.

In Fig. 6, we compared the maximum impulse velocity changes the system can recover from in 12 directions during OWS. The robot walked sagittally (positive x direction) at 0.5 m/s apex velocity. After the perturbation, it was allowed to recover using up to two steps. When the push direction was lateral left (positive y), the robot would take a wider step to come back at k^{ss}; otherwise, when the push direction was lateral right, the robot needs to adopt the crossed-leg maneuvers. The perturbations are applied at 4 different phases, with phase $\phi = 0\%$ and 90% closer to keyframe states (boundary phases), and $\phi = 30\%$ and 60% closer to the contact switch phase (50%). The result shows that the phases close to keyframes were better at absorbing large left perturbations. Closer to the contact switch phase, the right side push is handled better due to the increased lateral velocity halfway through the step.

We conducted an experiment to study the recovery success rate with 100 trials in 4 directions (Fig. 8). Diagonal disturbances were applied at $45°$ from the front to the right.

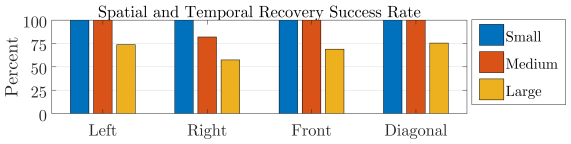

Fig. 7: Success rate of recovery motion when disturbance happens anytime during OWS at multiple directions. Three disturbances are used with a) small 0.1 m/s b) medium 0.2 m/s and c) large 0.3 m/s disturbances

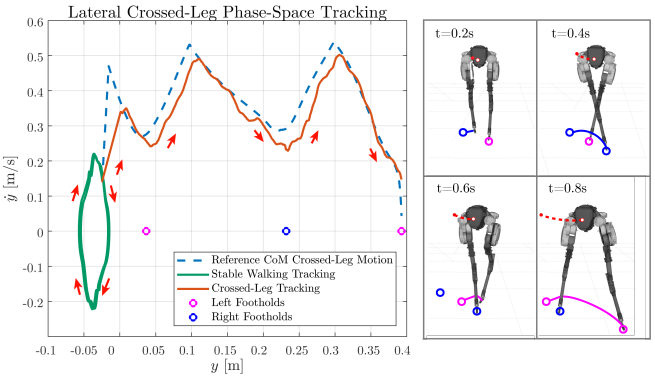

Fig. 8: Tracking controller results for Cassie executing a 0.4 m/s laterally disturbed leg crossing maneuver from a 0.5 m/s stable forward walking.

For each trial, the robot took the same was disturbed with 3 instantaneous velocity jumps of $0.1, 0.2, 0.3$ m/s. The perturbations for each trial were spaced evenly ($\phi = 1\%$) for the entire phase duration. Failures primarily occurred at the point of maximum velocity for the stance phase (right stance: $\phi \leq 10\%$ and $\phi \geq 90\%$, left stance: $40\% \leq \phi \leq 60\%$). Similar trends were seen in the maximum velocity disturbances Fig. 6.

Finally, the tracking performance for the system was evaluated for ± 0.4 m/s lateral disturbances while the left leg was in stance, during a stable walking with 0.5 m/s apex velocity. The positive disturbance forced a two-step crossed-leg recovery due to the stance feet (Fig. 8) and has a RMS tracking error of 0.0084 m and 0.0593 m/s. For negative lateral disturbances, the system stabilized within 1 wide step, similar to methods like capture point [3] with a RMS tracking error of 0.0039 m and 0.0363 m/s in lateral phase-space.

IV. Conclusion and Future Work

In this paper, we presented a locomotion framework for reactive disturbance rejection at the symbolic decision-making and continuous motion planning level. We combined reactive synthesis with BTs to demonstrate safe, continuous, disturbance rejection capabilities. At the low level, the TO generates full-body locomotion trajectories and refines feasible keyframe specifications in the reactive synthesis to fill the gap between the high-level decisions making and the low-level full-body motion planning.

For future work, we plan to explore multiple directions. We will quantify environment disturbances in the real world by evaluating results on the Computer Assisted Rehabilitation ENvironment (CAREN) system [32] and outdoors. The framework can also be generalized to handle more environmental events for either navigation collision avoidance or collaborative tasks with other robots and humans.

REFERENCES

[1] Benjamin Stephens. *Push recovery control for force-controlled humanoid robots*. Carnegie Mellon University, 2011.

[2] Pierre-brice Wieber. Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In *IEEE-RAS International Conference on Humanoid Robots*, pages 137–142, 2006.

[3] Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *IEEE-RAS international conference on humanoid robots*, pages 200–207. IEEE, 2006.

[4] Seung-Joon Yi, Byoung-Tak Zhang, Dennis Hong, and Daniel D Lee. Online learning of a full body push recovery controller for omnidirectional walking. In *IEEE-RAS International Conference on Humanoid Robots*, pages 1–6. IEEE, 2011.

[5] Milad Shafiee, Giulio Romualdi, Stefano Dafarra, Francisco Javier Andrade Chavez, and Daniele Pucci. Online dcm trajectory generation for push recovery of torque-controlled humanoid robots, 2019.

[6] Chengju Liu, Jing Ning, Kang An, and Qijun Chen. Active balance of humanoid movement based on dynamic task-prior system. *International Journal of Advanced Robotic Systems*, 14(3):1729881417710793, 2017.

[7] Chengxu Zhou, Cheng Fang, Xin Wang, Zhibin Li, and Nikos Tsagarakis. A generic optimization-based framework for reactive collision avoidance in bipedal locomotion. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1026–1033, 2016.

[8] Arne-Christoph Hildebrandt, Robert Wittmann, Daniel Wahrmann, Alexander Ewald, and Thomas Buschmann. Real-time 3d collision avoidance for biped robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4184–4190, 2014.

[9] Alexander Dietrich, Thomas Wimböck, Holger Täubig, Alin Albu-Schäffer, and Gerd Hirzinger. Extensions to reactive self-collision avoidance for torque and position controlled humanoids. In *IEEE International Conference on Robotics and Automation*, pages 3455–3462, 2011.

[10] Quan Nguyen, Xingye Da, J. Grizzle, and K. Sreenath. Dynamic walking on stepping stones with gait library and control barrier functions. In *WAFR*, 2016.

[11] Yukai Gong, Ross Hartley, Xingye Da, Ayonga Hereid, Omar Harib, Jiunn-Kai Huang, and Jessy Grizzle. Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway, 2018.

[12] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics*, 25(6):1370–1381, 2009.

[13] Jun Liu, Necmiye Ozay, Ufuk Topcu, and Richard M Murray. Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Transactions on Automatic Control*, 58(7):1771–1785, 2013.

[14] Keliang He, Andrew M. Wells, Lydia E. Kavraki, and Moshe Y. Vardi. Efficient symbolic reactive synthesis for finite-horizon tasks. In *International Conference on Robotics and Automation (ICRA)*, pages 8993–8999, 2019.

[15] Ye Zhao, Yinan Li, Luis Sentis, Ufuk Topcu, and Jun Liu. Reactive task and motion planning for robust whole-body dynamic locomotion in constrained environments, 2018.

[16] Sutej Kulgod, Wentao Chen, Junda Huang, Ye Zhao, and Nikolay Atanasov. Temporal logic guided locomotion planning and control in cluttered environments. In *2020 American Control Conference (ACC)*, pages 5425–5432, 2020.

[17] Jonas Warnke, Abdulaziz Shamsah, Yingke Li, and Ye Zhao. Towards safe locomotion navigation in partially observable environments with uneven terrain. In *IEEE Conference on Decision and Control (CDC)*, pages 958–965, 2020.

[18] Alejandro Marzinotto, Michele Colledanchise, Christian Smith, and Petter Ögren. Towards a unified behavior trees framework for robot control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5420–5427. IEEE, 2014.

[19] Shen Li, Daehyung Park, Yoonchang Sung, Julie A Shah, and Nicholas Roy. Reactive task and motion planning under temporal logic specifications. *arXiv preprint arXiv:2103.14464*, 2021.

[20] Michele Colledanchise and Petter Ögren. *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.

[21] Matteo Iovino, Edvards Scukins, Jonathan Styrud, Petter Ögren, and Christian Smith. A survey of behavior trees in robotics and ai. *arXiv preprint arXiv:2005.05842*, 2020.

[22] Hae-Won Park, Alireza Ramezani, and J. W. Grizzle. A finite-state machine for accommodating unexpected large ground-height variations in bipedal robot walking. *IEEE Transactions on Robotics*, 29(2):331–345, 2013.

[23] Ye Zhao, Benito R Fernandez, and Luis Sentis. Robust optimal planning and control of non-periodic bipedal locomotion with a centroidal momentum model. *The International Journal of Robotics Research*, 36(11):1211–1242, 2017.

[24] Hadas Kress-Gazit, Tichakorn Wongpiromsarn, and Ufuk Topcu. Correct, reactive, high-level robot control. *IEEE Robotics & Automation Magazine*, 18(3):65–74, 2011.

[25] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[26] Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. In *Verification, Model Checking, and Abstract Interpretation*, pages 364–380. Springer, 2006.

[27] Anil Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135, 01 2010.

[28] Ayonga Hereid and Aaron D. Ames. Frost: Fast robot optimization and simulation toolkit. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 719–726, 2017.

[29] Mikhail Koptev, Nadia Figueroa, and Aude Billard. Real-time self-collision avoidance in joint space for humanoid robots. *IEEE Robotics and Automation Letters*, 6(2):1240–1247, 2021.

[30] Andreas Wächter and Lorenz Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 03 2006.

[31] V. Raman R. Ehlers. Slugs: Extensible gr(1) synthesis, 2016.

[32] Brad Isaacson, Thomas Swanson, and Paul Pasquina. The use of a computer-assisted research environment (caren) for enhancing wounded warrior rehabilitation regimens. *The journal of spinal cord medicine*, 36:296–299, 07 2013.