Extended Version of Integrated Task and Motion Planning for Safe Legged Navigation in Partially Observable Environments

Abdulaziz Shamsah, *Student Member, IEEE,* Jonas Warnke, *Student Member, IEEE,*
Zhaoyuan Gu, *Student Member, IEEE,* and Ye Zhao, *Member, IEEE*

***Abstract*—This study proposes a hierarchically integrated framework for safe task and motion planning (TAMP) of bipedal locomotion in a partially observable environment with dynamic obstacles and uneven terrain. The high-level task planner employs linear temporal logic (LTL) for a reactive game synthesis between the robot and its environment and provides a formal guarantee on navigation safety and task completion. To address environmental partial observability, a belief abstraction is employed at the high-level navigation planner to estimate the dynamic obstacles' location when they are out of the robot's local field of view. Accordingly, a synthesized action planner sends a set of locomotion actions including walking step, step height, and heading angle change, to the middle-level motion planner, while incorporating safe locomotion specifications extracted from safety theorems based on a reduced-order model (ROM) of the locomotion process. The motion planner employs the ROM to design safety criteria and a sampling algorithm to generate non-periodic motion plans that accurately track high-level actions. To address external perturbations, this study also investigates safe sequential composition of the keyframe locomotion state and achieves robust transitions against external perturbations through reachability analysis. A set of ROM-based hyperparameters are finally interpolated to design whole-body locomotion gaits generated by trajectory optimization and validate the viable deployment of the ROM-based TAMP to the full-body trajectory generation for a 20-degrees-of-freedom Cassie bipedal robot designed by Agility Robotics. The proposed framework is validated by a set of scenarios in uneven, partially observable environments with dynamical obstacles.**

***Index Terms*—Task and motion planning, bipedal locomotion, temporal logic, sequential composition, safety.**

I. Introduction

ROBOTS are increasingly being deployed in real-world environments, with legged robots presenting superior versatility in complex workspaces. However, safe legged navigation in real-life workspaces still poses a challenge, particularly in a partially observable environment comprised of dynamic and possibly adversarial obstacles as seen in Fig. 1. While motion planning for bipedal systems in dynamic

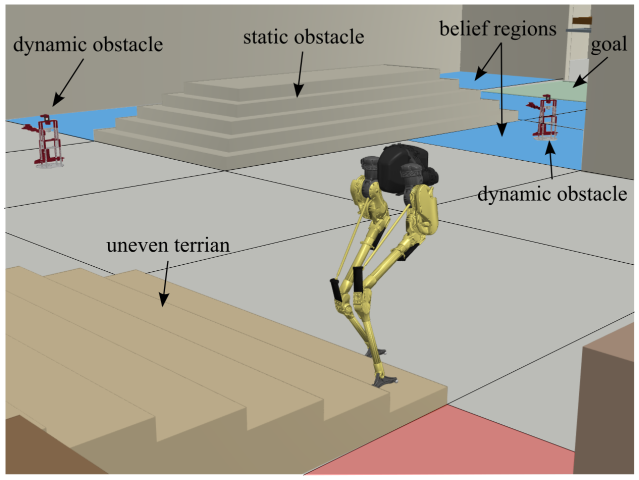


Fig. 1: A snapshot of the simulation environment for the proposed TAMP framework. The walking robot is deployed to accomplish safe navigation tasks. The environment contains static and dynamic obstacles, and uneven terrains.

environments has been widely studied [1]–[4], the proposed solutions often lack formal guarantees on simultaneous locomotion and navigation safety, with the exception of a recent work in [5]. Formal guarantees on safety and task completion in a complex environment has been gaining interest in recent years [6]–[9], however hierarchical planning frameworks with multi-level safety guarantees for underactuated legged robots remain lacking. An intrinsic challenge of such multi-level formal guarantees is how to guarantee viable execution of high-level commands for low-level, full-body motion generation that involves inherently complex bipedal dynamics.

This study proposes a hierarchically integrated task and motion planning (TAMP) framework as shown in Fig. 2 and provides multi-level formal safety guarantees on dynamic locomotion and navigation in dynamic and partially observable environments as shown in Fig. 1. A high-level temporal-logic-based task planner consists of global navigation and local action planners with formal guarantees on safety and task completion. A middle-level motion planner is comprised of a keyframe decision maker that determines the next locomotion keyframe, and a phase-space planner that generates safe center-of-mass (CoM) trajectories. Keyframe states of reduced-order models (ROMs) are used as interpolated hyperparameters of the whole-body motion planner for full-body trajectory optimization.

Our framework takes safety into account in each layer

This work was partially funded by the NSF grant # IIS-1924978, Georgia Tech Research Institute IRAD grant, and Georgia Tech Institute for Robotics and Intelligent Machines (IRIM) Seed Grant.

The authors are with the Laboratory for Intelligent Decision and Autonomous Robots, Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30313, USA {ashamsah3, jwarnke, zgu78, yzhao301}@gatech.edu
Abdulaziz Shamsah and Jonas Warnke equally contributed.

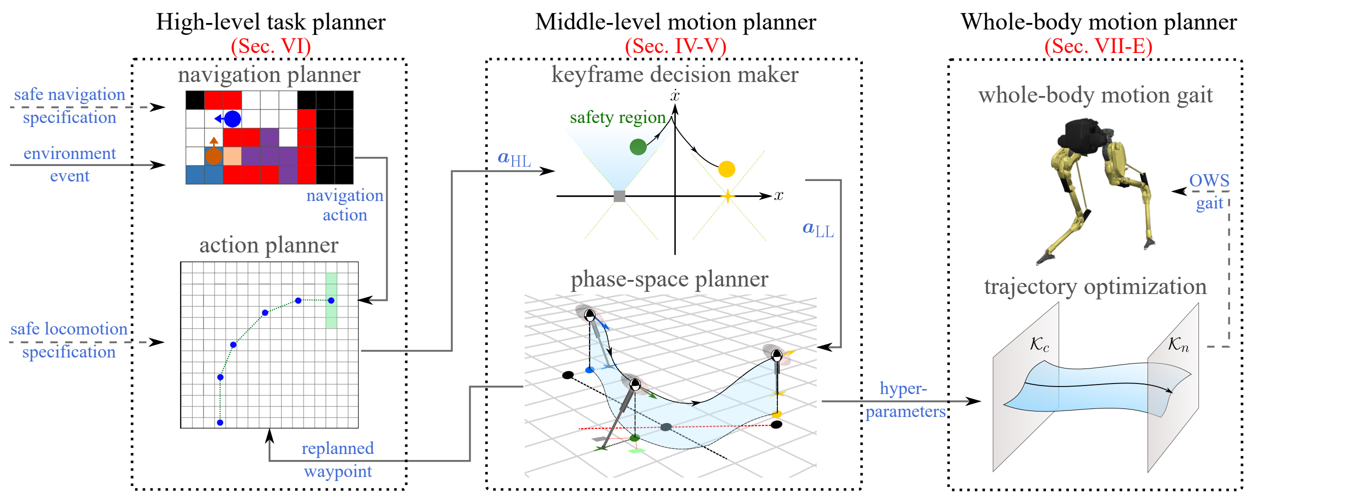


Fig. 2: Block diagram of the proposed TAMP framework. The high-level task planner employs a linear temporal logic approach to synthesize locomotion actions for navigation tasks. The middle-level motion planner generates a safe motion plan based on a ROM. Then the whole-body motion planner, generates full-body motion primitives for our Cassie bipedal robot by interpolating hyperparameters extracted from the ROM plan. The high-level task planner and the middle-level motion planner are integrated in an *online* fashion as shown by the solid black arrows. The dashed arrows represent *offline* computations.

of the hierarchical structure, and in how these layers are interconnected to achieve simultaneous safe locomotion and navigation. At the middle-level motion planning layer, we leverage our previous Phase-Space Planning (PSP) method [10]–[12] to propose a set of ROM-based safety criteria, where we constrain allowable CoM apex velocity and turning angle to refine safety specifications in the high-level task planner. Therefore, the high-level task planner incorporates locomotion dynamics constraints into the LTL safety specifications thus guaranteeing safe execution of the high-level actions in the underlying motion planners. The middle-level motion planner also employs a sampling-based algorithm to select the optimal next CoM apex velocity to track the high-level actions. We integrate the task and the motion planners *online* for *correction-as-needed* execution, in the presence of external perturbations.

Guaranteeing safe navigation of legged robots in the presence of possibly adversarial obstacles becomes particularly challenging in partially observable environments. The range of a robot's sensor and occlusion caused by static obstacles give an adversarial agent a strategic advantage when trying to falsify the bipedal robot's safety guarantees by moving through non-visible regions of the environment. Our work is motivated by the surveillance game literature [13] to track possible non-visible dynamic obstacle locations via belief space planning. Belief space planning allows us to model the set of possible obstacle locations and track how this set evolves, reducing adversarial agents' strategic advantage and guaranteeing collision avoidance in a larger set of environments.

Robustness at the motion planning layer is of key importance, as continuous perturbations (e.g., CoM perturbations) can be naturally handled at this layer [14], unlike in the discretized high-level task planner which is unaware of locomotion dynamics. To this end, we formulate the locomotion gait in the lens of controllable regions [15] and sequential composition [16] where we sequentially compose controllable regions to robustly complete a walking step. We employ ROM-based backward reachability analysis to compute robust controllable regions, and synthesize appropriate controllers to safely reach the targeted state.

The main contributions of this study are as follows:

- Design a hierarchically integrated planning framework that provides formal safety guarantees simultaneously for the high-level task planner and middle-level ROM-based motion planner, which enables safe locomotion and navigation involving steering walking.
- Design safe sequential composition of controllable regions for robust locomotion in the presence of perturbations, and sampling-based keyframe decision maker for accurate waypoint tracking to facilitate middle-level navigation safety.
- Synthesize a LTL-based reactive navigation game for safe legged navigation and employ a belief abstraction method to expand navigation decisions in partially observable environments.
- Design a non-periodic whole-body-dynamics-consistent motion library through trajectory optimization for a 20 Degrees-of-Freedom (DoFs) Cassie bipedal system.

A conference version of the work presented in this paper was published in [12]. This work includes extensions toward formal guarantees on middle-level motion plans under external perturbations, sampling-based keyframe decision maker, non-deterministic high-level transitions for online replanning, joint-belief abstractions, and design of whole-body motion plans.

This paper is outlined as follows. Sec. II is a literature review of related work. Sec. III introduces the ROM-based locomotion planning and keyframe definitions. Then safety theorems for locomotion planning and reachability-based analysis for robustness against perturbations are introduced in Sec. IV. In Sec. V, we introduce our sampling-based keyframe decision maker algorithm for lateral tracking. Sec. VI outlines our LTL-based high-level task planner which guarantees safe navigation in a partially observable environments. The results of our integrated framework are shown in Sec. VII as well as our trajectory-optimization-based whole-body motion planner. In Sec. VIII, we discuss the limitations, and conclude in Sec. IX. Appendix includes supplementary mathematical proofs.

II. Related Work

Motion planning in complex environments has been extensively studied, with a spectrum of approaches in the literature [17]–[22]. Reactive methods for motion planning with formal guarantees are widely studied through the lens of artificial potential functions [23], with more recent extensions [7], [8], to name a few. While the work in [7] provides convergence guarantees and obstacle avoidance, it is limited to environments with convex obstacles. Along the same line of research, [8] extends the work in [7] to geometrically complicated unknown environments. However, the approach detailed in [8] is restricted to static obstacles, and has only been demonstrated on a fully actuated particle, or a simple unicycle model for a quadruped. Whereas the framework we present here is able to generate safe locomotion plans reacting to environmental events that include multiple, possibly adversarial, dynamic obstacles with formal guarantees on task completion and safety. Moreover, our framework generates whole-body motion plans for a 20-DOF underactuated bipedal robot Cassie [24].

The concept of capture point [25], and its extension to N-step capturability [26], represent a powerful method to characterize and achieve locomotion stability against external disturbances by computing N-step capture regions, where the robot can take N walking steps to reach and come to a stop. An extension to capture regions was introduced in [15] as controllable regions. The difference of controllable regions from the capture regions lies in that the target robot state is not necessarily coming to a stop. The work in [15] is closely related to our work presented here, where we use controllable regions and viability theory [27] to guarantee safety and to achieve non-periodic walking gaits. A resounding difference is that our framework not only focuses on stability, but uses linear temporal logic (LTL) to synthesize the high-level task planner for making decisions on locomotion navigation and obstacle avoidance. In [26], the foot placements are controlled to achieve balancing safety or reach a target CoM velocity as in [15]. On the other hand, in our framework the foot location decision is determined by not only the low-level locomotion dynamics but also the high-level navigation goals of completing tasks and avoiding environmental obstacles. Since the foot placement is fixed from the high-level planner, our work focuses on manipulating the CoM apex velocities of the robot, to track waypoints determined by a specific high-level navigation task as well as maintaining locomotion balance.

Formal synthesis methods have been well established to guarantee high-level robot behaviors in dynamic environments [28]–[30]. Collision-free navigation in the presence of dynamic obstacles has been achieved via multiple approaches such as local collision avoidance controllers in [31], incrementally expanding a motion tree in sampling-based approaches [32], and Velocity Obstacle Sets generated by obstacle reachability analysis in [33]. Collision avoidance and task completion become more challenging to guarantee when the environment is only partially observable as such an environment has a strategic advantage in being adversarial. Navigating through partially known maps with performance guarantees has been achieved through exploring [34], updating the discrete abstraction, and re-synthesizing a controller at runtime in [35]. To avoid the computational costs of online re-synthesis, others have proposed patching a modified local controller into an existing global controller when unmodeled non-reachable cells, i.e. static obstacles, are discovered at runtime [36], [37]. The authors in [35] have proposed a satisfaction metric of specification to meet the specification as closely as possible when run-time discovered environment constraints render the specification unsatisfiable. Lastly, the work of [38] proposes a receding horizon planning method for efficient synthesis of short-horizon plans. As unmodeled obstacles appear in the planning horizon, a goal generator recomputes a path to a satisfying state. These approaches above are better suited for guaranteeing successful navigation and collision avoidance in environments that are uncertain only with respect to static obstacles as they can not reason about when and where a dynamic obstacle may appear.

Collision avoidance with dynamic obstacles in partially observable environments has been achieved through approaches such as POMDPs [39], Probabalistic velocity obstacle modeling [40], and object occlusion cost metrics [41]. The authors in [42] guarantee passive motion safety by avoiding braking Inevitable Collision States (ICS) at all times via a braking ICS-checking algorithm. While these solutions provide collision avoidance guarantees, they assume dynamic obstacles could appear at any time and result in an overly conservative strategies. Our method investigates belief-space planning to provide the controller additional information on when and where dynamic obstacles may appear in the robot's visible range to inform the synthesized strategy if navigation actions are guaranteed to be safe, even when static obstacles occlude the robot's view adjacent environment locations. We have devised a variant of the approach in [13] to explicitly track a belief of which non-visible environment locations are obstacle free, reducing the conservativeness of a guaranteed collision-free strategy. This belief tracking method is then integrated into our hierarchical TAMP framework.

III. Preliminaries

This section will introduce a phase-space planning (PSP) approach for CoM trajectory generation based on a reduced-order model (ROM) [10], [11]. Starting with a derivation of the dynamics of a Prismatic Inverted Pendulum Model (PIPM), and then define the locomotion keyframe state, a discretized feature state of our PSP approach, used as a connection between the high-level planner and the middle-level motion planner. Consequently, we define keyframe-based transitions to achieve safe locomotion. This section builds the basis for the safe locomotion planning proposed in later sections.

A. Reduced-order Locomotion Planning

This subsection introduces a mathematical formulation of our ROM. As shown in Fig. 3, the CoM position $\boldsymbol{p}_{\rm com} = (x, y, z)^T$ is composed of the sagittal, lateral, and vertical positions. We denote the apex CoM position as $\boldsymbol{p}_{\rm apex} = (x_{\rm apex}, y_{\rm apex}, z_{\rm apex})^T$, the foot placement as $\boldsymbol{p}_{\rm foot} =$

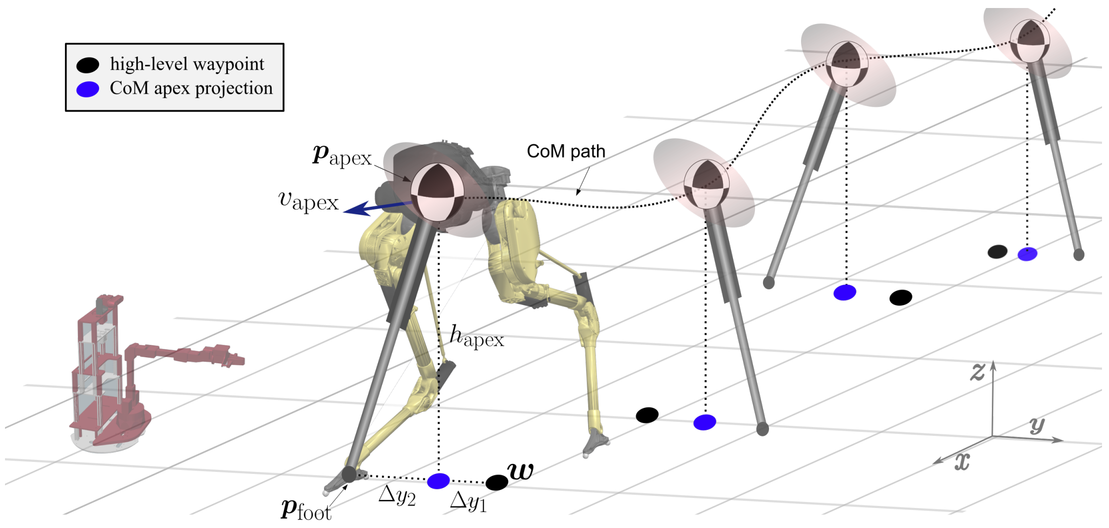


Fig. 3: Reduced-order modeling of our Cassie robot as a 3D prismatic inverted pendulum model with all of its mass concentrated on its CoM and a telescopic leg to comply to the varying CoM height. The CoM motion follows a parameterized CoM path depending on keyframe states. Δy_1 is the relative lateral distance between lateral CoM apex position and the high-level waypoint $\boldsymbol{w}$, and Δy_2 is the lateral distance between the CoM lateral apex position and the lateral foot placement.

$(x_{\text{foot}}, y_{\text{foot}}, z_{\text{foot}})^T$, and h_{apex} is the relative apex CoM height with respect to the stance foot height. v_{apex} denotes the CoM velocity at $\boldsymbol{p}_{\text{apex}}$. Δy_1 is the lateral distance between CoM and the high-level waypoint $\boldsymbol{w}$[1] at apex. $\Delta y_2 := y_{\text{apex}} - y_{\text{foot}}$ denotes the lateral CoM-to-foot distance at apex. This parameter will be used to determine the allowable steering angle in Sec. IV-A.

Prismatic Inverted Pendulum Model (PIPM) has been proposed for agile, non-periodic locomotion over rough terrain [11]. Here we reiterate for completeness the derivation of the centroidal momentum dynamics of this model. The single contact case using the moment balance equation along with linear force equilibrium is expressed as

$$(\boldsymbol{p}_{\text{com}} - \boldsymbol{p}_{\text{foot}}) \times (\boldsymbol{f}_{\text{com}} + m\boldsymbol{g}) = -\boldsymbol{\tau}_{\text{com}} \tag{1}$$

where $\boldsymbol{\tau}_{\text{com}}$ is the angular moments of the torso exerted on the CoM, and $\boldsymbol{g}$ is the gravitational vector. For nominal planning we set $\boldsymbol{\tau}_{\text{com}} = 0$. Formulating the dynamics in Eq. (1) for q^{th} walking step as a hybrid control system

$$\ddot{\boldsymbol{p}}_{\text{com},q} = \Phi(\boldsymbol{p}_{\text{com},q}, \boldsymbol{u}_q) = \begin{pmatrix} \omega_q^2(x - x_{\text{foot},q}) \\ \omega_q^2(y - y_{\text{foot},q}) \\ a\omega_q^2(x - x_{\text{foot},q}) \end{pmatrix} \tag{2}$$

where the asymptote slope $\omega_q = \sqrt{g/h_{\text{apex},q}}$. The hybrid control input is $\boldsymbol{u}_q = (\omega_q, \boldsymbol{p}_{\text{foot},q})$, with $\boldsymbol{p}_{\text{foot},q}$ being the discrete input[2]. The CoM motion is constrained within a piecewise linear surface parameterized by $h = a(x - x_{\text{foot}}) + h_{\text{apex}}$, where h denotes the CoM height from the stance foot height, the ROM becomes linear and an analytical solution exists. Detailed derivations are elaborated in Appendix I.

Summary of Phase-space Planning: In PSP, the sagittal planning takes precedence over the lateral planning. The decisions for the planning algorithm are primarily made in the sagittal phase-space, such as step length and CoM apex velocity, where we propagate the dynamics forward from the current apex state and backward from the next apex state until the two phase-space trajectories intersect. The intersection state defines the foot stance switching instant. On the other hand, the lateral phase-space parameters are searched for to adhere to the sagittal phase-space plan and have consistent timings between the sagittal and lateral plans. In this paper, we build on our previous PSP work [11], [12] to derive safety criteria for sagittal planning in order to achieve successful transitions between keyframe states in the presence of perturbations in Sec. IV. Moreover, we employ a sampling algorithm based on the lateral apex states to select the next sagittal apex velocity that allows the lateral dynamics to comply to high-level waypoint tracking in Sec. V.

[1]The high-level discrete representation of the robot location.

[2]Hereafter, we will ignore the subscript q for notation simplicity. We will instead use $\cdot_c$ and $\cdot_n$ denoting the current and next walking steps, respectively.

B. Locomotion Keyframe for 3D Navigation

PSP uses keyframe states for non-periodic dynamic locomotion planning [11]. Our study generalizes the keyframe definition in our previous work by introducing diverse navigation actions in 3D environments.

Definition III.1 (Locomotion keyframe state for 3D environment navigation)**.** *A keyframe state of our ROM is defined as* $\boldsymbol{k} = (d, \Delta\theta, \Delta z_{\text{foot}}, v_{\text{apex}}, z_{\text{apex}}) \in \mathcal{K}$, *where*

- $d := x_{\text{apex},n} - x_{\text{apex},c}$ *is the walking step length*[3]*;*
- $\Delta\theta := \theta_{\text{apex},n} - \theta_{\text{apex},c}$ *is the heading angle change at two consecutive CoM apex states;*
- $\Delta z_{\text{foot}} := z_{\text{foot},n} - z_{\text{foot},c}$ *is the height change for successive foot placements;*
- v_{apex} *is the CoM sagittal apex velocity;*
- z_{apex} *is the global CoM height at apex.*

The keyframe state above can be divided into two action sets: a high-level (HL) action ($\boldsymbol{a}_{\text{HL}}$) and a low-level (LL) action ($\boldsymbol{a}_{\text{LL}}$). The HL action includes $\boldsymbol{a}_{\text{HL}} = (d, \Delta\theta, \Delta z_{\text{foot}}) \in$

[3]In straight walking d represents the step length. However, during steering walking d is adjusted to reach the next waypoint on the new local coordinate.

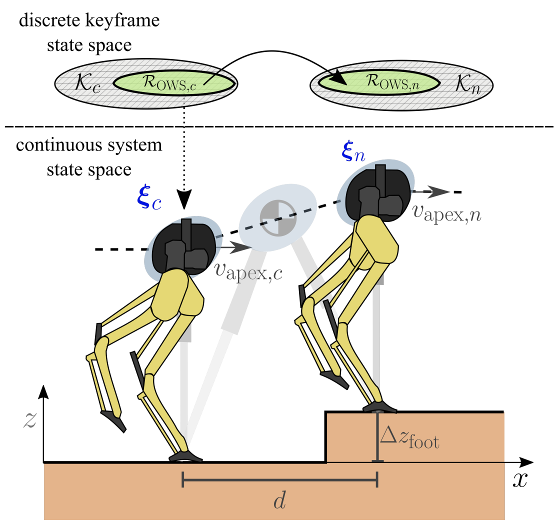

discrete keyframe state space
$\mathcal{K}_c$
$\mathcal{R}_{\mathrm{OWS},c}$
$\mathcal{R}_{\mathrm{OWS},n}$
$\mathcal{K}_n$
continuous system state space
$\boldsymbol{\xi}_c$
$\boldsymbol{\xi}_n$
$v_{\mathrm{apex},c}$
$v_{\mathrm{apex},n}$
z
Δz_{foot}
x
d

Fig. 4: Sagittal system state transition for One Walking Step (OWS) without heading angle change. OWS is shown as the transition between two consecutive apex states, $\boldsymbol{\xi}_c$ and $\boldsymbol{\xi}_n$. The system state transition in Def. III.4 is shown as the projection of keyframe state onto the system state $\boldsymbol{\xi}_c$, and d and Δz_{foot} are used to select the next foot placement $\boldsymbol{p}_{\mathrm{foot},n}$ in the hybrid control input $\boldsymbol{u}$. In the discrete keyframe state space we show the transition between two consecutive keyframe states, where $\mathcal{R}_{\mathrm{OWS},c}$ is the set of current viable keyframe states that allows a successful system state transition to the next viable keyframe state set $\mathcal{R}_{\mathrm{OWS},n}$.

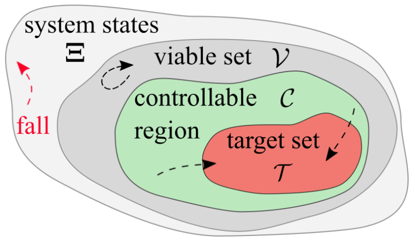


Fig. 5: Viable sets and controllable regions for the set of system states Ξ. Any state within the Viable set $\mathcal{V}$ (dark gray region), is guaranteed to remain inside $\mathcal{V}$ in finite time thus avoiding a fall. While, any state within the controllable region set $\mathcal{C}$ (green region) is guaranteed to reach the target set $\mathcal{T}$ (red region) in finite time given an appropriate control input. The red trajectory indicate an initial state that results in a fall. In general, the viable set $\mathcal{V}$ is computationally difficult to estimate (e.g., for the states composing an OWS but not reaching $\mathcal{T}$), and thus this study focuses on computing the controllable region $\mathcal{C}$ for safe sequential composition.

$\mathcal{A}_{\mathrm{HL}}$, which is determined by the navigation policy to be designed in the task planner. The parameters d, $\Delta\theta$, and Δz_{foot} are expressed in the Cartesian space as the high-level waypoints $\boldsymbol{w}$. On the other hand, the LL action is $\boldsymbol{a}_{\mathrm{LL}} = (v_{\mathrm{apex}}, z_{\mathrm{apex}}) \in \mathcal{A}_{\mathrm{LL}}$, which is determined in the middle-level motion planner. The keyframe parameters are sent from the high-level task planner to the middle-level motion planner *online* as shown in Fig. 2.

C. Keyframe Transition

We now aim to formulate locomotion transition definitions in terms of the locomotion keyframe state $\boldsymbol{k}$ and describe the connection between the descritzed keyframe state and the continuous dynamics of our reduced-order system introduced in Sec. III-A. We will first define locomotion safety.

Definition III.2 (Locomotion safety)**.** *Safety for a locomotion process is defined as a formally-guaranteed successful transition between consecutive locomotion keyframe states $\boldsymbol{k} \in \mathcal{K}$ while the robot maintains its balance, i.e., avoids a fall.*

Note that, the keyframe state $\boldsymbol{k}$ includes high-level actions $\boldsymbol{a}_{\mathrm{HL}}$ so the control is implicit in the *Locomotion Safety*. Based on our keyframe definition in Def. III.1, we define One Walking Step (OWS) as the transition between two consecutive keyframe states as shown in Fig. 4. Therefore, we define the set of viable keyframe states for OWS as follows.

Definition III.3 (Viable keyframe set for one walking step)**.** *$\mathcal{R}_{\mathrm{OWS}}$ is the set of keyframe states $\mathcal{K}$ that results in a viable transition to the next desired keyframe state through the continuous PIPM dynamics in Eq. (2), thus achieving locomotion safety for OWS.*

The transition between keyframes is hybrid since it includes a continuous progression of the system states under the PIPM dynamics in Eq. (2), followed by a discrete foot contact switch. In this study, we aim to provide formal guarantees that the selected keyframe states are within $\mathcal{R}_{\mathrm{OWS}}$. Since quantifying $\mathcal{R}_{\mathrm{OWS}}$ is computationally intractable due to its high dimensionality, we propose a set of safety theorems to quantify the viable region when $\mathcal{R}_{\mathrm{OWS}}$ is projected onto a reduced dimensional parameter space, which is selected as

$$\text{Sagittal CoM system state: } \boldsymbol{\xi} = (x, \dot{x}) \in \Xi$$

The current discrete keyframe state $\boldsymbol{k}_c \in \mathcal{K}$ corresponds to (i) the continuous system state at apex ($\boldsymbol{\xi}_c$) and (ii) the PSP hybrid control input $\boldsymbol{u}$ at the CoM apex in both straight and steering walking scenarios, where the apex state in the keyframe Def. III.1 is the system state at the CoM apex[4], and the step length and step height are used to calculate $\boldsymbol{p}_{\mathrm{foot}}$ in the hybrid control input $\boldsymbol{u}$. An illustration of these variables is shown in Fig. 4. The desired next system state is always selected to be an apex state $\boldsymbol{\xi}_n = (d, v_{\mathrm{apex},n})$, where $d \in \boldsymbol{a}_{\mathrm{HL}}$ is determined by the high-level planner and $v_{\mathrm{apex},n}$ is determined by the keyframe decision maker as detailed in Sec. V. Therefore, we can define a system state transition.

Definition III.4 (System state transition $\boldsymbol{\xi}_n = T(\boldsymbol{k}_c)$)**.** *$T$ is a system state transition that takes the projection of the current keyframe state onto the continuous system state at apex and hybrid control input ($\boldsymbol{\xi}_c.\boldsymbol{u}$), to the desired next system state $\boldsymbol{\xi}_n$ through PSP of the centroidal dynamics Φ in Eq. (2).*

In Fig. 4, we illustrate the system state transition for OWS. By projecting $\boldsymbol{k}_c$ state onto the $\boldsymbol{\xi}_c$, the centroidal dynamics in Eq. (2) allows the system state to reach $\boldsymbol{\xi}_n$ based on $\boldsymbol{u}$.

IV. Safe Locomotion Planning

This section will propose a set of safety theorems based on PSP for the ROM that allows us to select a safe next keyframe states under nominal conditions. Then in Sec. IV-B we define and compute controllable regions under bounded state disturbance by reachability analysis for a set of keyframe transitions which satisfy the safety theorems introduced in Sec. IV-A. Within this controllable region, any state is guaranteed to reach

[4] $\boldsymbol{\xi}_c = \boldsymbol{\xi}_{\mathrm{apex},c}$ only when $\Delta\theta = 0$, otherwise $\boldsymbol{\xi}_c$ is a non-apex state as can be seen in Fig. 6(a).

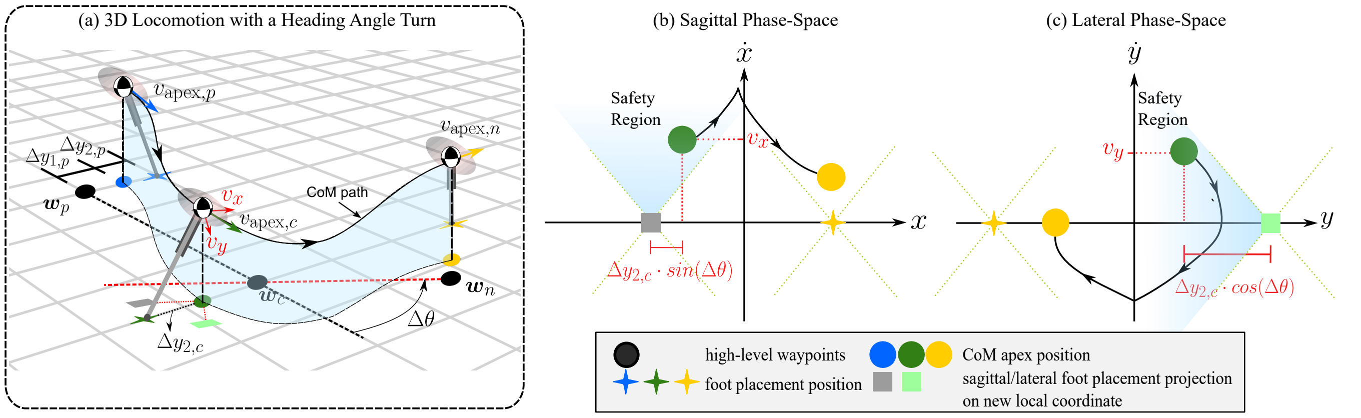


Fig. 6: Phase-space safety region for steering walking: (a) shows three consecutive keyframes with a heading angle change ($\Delta\theta$) between the current keyframe and the next keyframe. The CoM trajectory and its projection on the sagittal-lateral plane is represented by the blue surface. The direction change introduces a new local coordinate, where the dashed black line is the sagittal coordinate before the turn, and the dashed red line is the sagittal coordinate after the turn. Subfigures (b) and (c) show the sagittal and lateral phase-space plots respectively, both satisfying the safety criteria proposed in Theorem IV.2. The CoM apex state in the original coordinate becomes non-apex in the new coordinate (due to the coordinate change). The subscripts p, c and n denote the previous, current, and next walking steps, respectively.

a target set ($\mathcal{T}$) in finite time given a feasible control sequence as shown in Fig. 5. Accordingly, we sequentially compose consecutive controllable regions through our hybrid control input to guarantee locomotion safety and correctness. In addition to the safety criteria defined only for sagittal locomotion, safety for lateral tracking of the high-level waypoint is equally important for 3D locomotion. Therefore, Sec. V presents a sampling-based algorithm that adjusts the sagittal phase-space plan to allow for lateral waypoint tracking.

A. Locomotion Safety Criteria

In this subsection, we propose safe locomotion criteria based on the PIPM introduced in Sec. III-A, and provide safety constraints for the locomotion keyframe state.

As a general principle of balancing safety, the sagittal CoM position should be able to cross the sagittal apex with a positive CoM velocity while the lateral CoM velocity should be able to reach the zero lateral velocity at the next apex. Ruling out the fall situations provides us the bounds of balancing safety regions. First, we study the constraints between apex velocities of two consecutive walking steps and propose the following theorems and corollaries.

Theorem IV.1. *For safety-guaranteed straight walking, given d and ω, the apex velocity for two consecutive walking steps ought to satisfy the following velocity constraint:*

$$-\omega^2 d^2 \leq \underbrace{v_{\text{apex},n}^2 - v_{\text{apex},c}^2}_{\substack{\textit{apex velocity square difference}\\ \textit{for two consecutive steps}}} \leq \omega^2 d^2 \tag{3}$$

where $d^2 = (x_{\text{apex},n} - x_{\text{apex},c})(x_{\text{apex},c} + x_{\text{apex},n} - 2x_{\text{foot},c})$. Notably, d is equal to the step length in Def. III.1, i.e., $d = x_{\text{apex},n} - x_{\text{apex},c}$, during a straight walking where $x_{\text{apex},c} = x_{\text{foot},c}$. The proof of this criterion can be seen in Appendix II.

Another consideration for safety is to limit the maximum allowable velocity of the CoM. Since the maximum velocity occurs at the foot switching instant, we explicitly enforce an upper velocity bound to this switching velocity v_{switch} to avoid over-accelerated motions, which can be further magnified by the ground impact dynamics in the real system. Through the analytical solution of the ROM in Appendix I, we solve for $v_{\text{switch}} = \Psi(v_{\text{apex},c}, v_{\text{apex},n}, d)$. Therefore, we set an upper bound on v_{switch}, i.e., $v_{\text{switch}} \leq v_{\text{max}}$.

Similar to Theorem IV.1, v_{switch} provides a nonlinear relationship between sagittal apex velocities for two consecutive apex states. Combining the boundary conditions in Theorem IV.1 and the limit of v_{switch} allows us to quantify the viable region of $v_{\text{apex},n}$ given $v_{\text{apex},c}$, d, and ω.

The steering case requires a more restrictive criterion. A fall will occur when the turning angle $\Delta\theta$ is too large such that $v_{\text{apex},c}$ in the new local coordinate after the turn is out of a safety range such that either the lateral CoM velocity cannot reach zero at the next apex or the sagittal CoM can not climb over the next apex.

Theorem IV.2. *For safety-guaranteed steering walking, the current sagittal CoM apex velocity $v_{\text{apex},c}$ in the original local coordinate should be bounded by*

$$\Delta y_{2,c} \cdot \omega \cdot \tan\Delta\theta \leq v_{\text{apex},c} \leq \frac{\Delta y_{2,c} \cdot \omega}{\tan\Delta\theta} \tag{4}$$

The proof of this theorem is shown in Appendix III. This theorem provides a bound on the heading angle change $\Delta\theta$ given the current apex velocity $v_{\text{apex},c}$. Fig. 6 shows a steering walking trajectory and phase-space plot that satisfy Theorem IV.2. Namely, the CoM location in the sagittal and lateral phase-space in the new local coordinate after the turn should not cross the asymptote line of the shaded safety region in Fig. 6. This criterion is specific to steering walking, as the heading change ($\Delta\theta$) introduces a new local frame and yields the current state $\boldsymbol{\xi}_c$ to no longer be an apex in the new coordinate. As such, it has non-apex sagittal and lateral components, i.e., $v_{y,c} \neq 0$, and $x_{\text{apex},c} \neq x_{\text{foot},c}$.[5]

Corollary IV.3. *For steering walking in Theorem IV.2, given d, $\Delta\theta$, $\Delta y_{2,c}$ and ω, two consecutive apex velocities ought to satisfy the following velocity constraint:*

$$-\omega^2 d^2 \leq v_{\text{apex},n}^2 - (v_{\text{apex},c}\cos\Delta\theta)^2 \leq \omega^2 d_+^2 \tag{5}$$

[5]In this study, we use $\dot{x}$ and v exchangeably to represent the CoM velocity.

where $d_+^2 = d^2 + 2\Delta y_{2,c} d \sin \Delta\theta$.

Corollary IV.4. *For steering walking in Theorem IV.2, similarly, given* d, $\Delta\theta$, $\Delta y_{2,c}$, *and* ω, *two consecutive apex velocities ought to satisfy the following velocity constraints,*

$$-\omega^2 d^2 \leq v_{\text{apex},n}^2 - (v_{\text{apex},c} \cos \Delta\theta)^2 \leq \omega^2 d_-^2 \tag{6}$$

where $d_-^2 = d^2 - 2\Delta y_{2,c} d \sin \Delta\theta$. *Note that, parameters* $v_{\text{apex},n}$, d, *and* $\Delta\theta$ *in Eqs. (3)-(6) are the keyframe states.*

The aforementioned safety theorems provide quantifiable bounds on the next keyframe selection that leads to viable transitions under governance of nominal disturbance-free PIPM dynamics. The next section will focus on definitions based on controllable regions and sequential composition to provide guarantees on the safe progression of the continuous system states $\boldsymbol{\xi}$ adhering to Theorems IV.1-IV.2 under bounded disturbances $\tilde{\Xi}$.

B. Controllable Regions and Sequential Composition

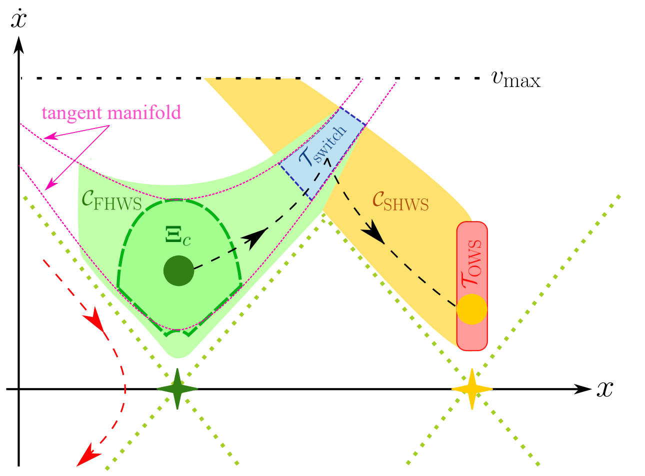
First let us decompose OWS into two half steps at the instant when the hybrid control input $\boldsymbol{u}$ switches. Therefore, the First Half Walking Step (FHWS) starts from $\boldsymbol{\xi}_c$ until the foot contact switching state $\boldsymbol{\xi}_{\text{switch}}$, the Second Half Walking Step (SHWS) starts with $\boldsymbol{\xi}_{\text{switch}}$ until $\boldsymbol{\xi}_n$. We start by defining controllable regions of FHWS and SHWS.

Definition IV.1 (Controllable region of FHWS)**.** *Given* $\tilde{\Xi}$, $\mathcal{U}$ *and* $\mathcal{T}_{\text{switch}}$, *a controllable region of FHWS is defined as* $\mathcal{C}_{\text{FHWS}} := \{\boldsymbol{\xi} | \dot{\boldsymbol{\xi}}(t) = \Phi(\boldsymbol{\xi}(t) + \tilde{\boldsymbol{\xi}}, \boldsymbol{u}(t)), \boldsymbol{u}(t) \in \mathcal{U}, \text{such that } \exists \boldsymbol{\xi}(t_{\text{FHWS}}) \in \mathcal{T}_{\text{switch}},\ t_{\text{FHWS}} \text{ is finite}\}$, *where* $\tilde{\boldsymbol{\xi}} \in \tilde{\Xi}$ *represents a bounded state disturbance.*

Definition IV.2 (Controllable region of SHWS)**.** *Given* $\tilde{\Xi}$, $\mathcal{U}$ *and* $\mathcal{T}_{\text{OWS}}$, *a controllable region of SHWS is defined as* $\mathcal{C}_{\text{SHWS}} := \{\boldsymbol{\xi} | \dot{\boldsymbol{\xi}}(t) = \Phi(\boldsymbol{\xi}(t) + \tilde{\boldsymbol{\xi}}, \boldsymbol{u}(t)), \boldsymbol{u}(t) \in \mathcal{U}, \text{such that } \exists \boldsymbol{\xi}(t_{\text{SHWS}}) \in \mathcal{T}_{\text{OWS}},\ t_{\text{SHWS}} \text{ is finite}\}$, *where* $\tilde{\boldsymbol{\xi}} \in \tilde{\Xi}$ *represents a bounded state disturbance.*

Numerical computation of the controllable region for OWS is achievable given Ξ_c, $\mathcal{T}_{\text{OWS}}$, $\mathcal{U}$, and a bounded disturbance $\tilde{\Xi}$ through backward dynamic propagation using ROCS [43]. Given the backward propagation nature of ROCS, $\mathcal{C}_{\text{SHWS}}$ is computed first starting from $\mathcal{T}_{\text{OWS}}$, which is selected to be the set of desired $\boldsymbol{\xi}_n$ at the next apex. Then we set $\mathcal{T}_{\text{switch}}$ to be all the states of $\mathcal{C}_{\text{SHWS}}$ that are within the tangent manifolds of FHWS, where the tangent manifolds represent the nominal phase-space trajectory given $v_{\text{apex,min}}$, $v_{\text{apex,max}}$ and $x_{\text{foot},c}$ (see Fig. 7) [11], [44]. Then we compute $\mathcal{C}_{\text{FHWS}}$ with $\mathcal{T}_{\text{switch}}$ being the target set. Note that $\mathcal{T}_{\text{switch}}$ represents the set of system states that a foot contact transition can occur at. Moreover, ROCS also allows us to synthesize a controller $\boldsymbol{u}(t) \in \mathcal{U}$ that guarantees that the system state $\boldsymbol{\xi}$ reaches the target set $\mathcal{T}_{\text{OWS}}$ as long as the system state remains within the controllable region. The details of the controllable regions in Defs. IV.1-IV.2 are shown in Fig. 7.

Sequentially composing the controllable regions defined in Defs. IV.1-IV.2, i.e, $\mathcal{T}_{\text{switch}} \neq \emptyset$, affords a guarantee on safe task completion for OWS. Controllable regions have a



$\dot{x}$
$v_{\max}$
tangent manifold
$\mathcal{T}_{\text{switch}}$
$\mathcal{C}_{\text{FHWS}}$
Ξ_c
$\mathcal{C}_{\text{SHWS}}$
$\mathcal{T}_{\text{OWS}}$
x

Fig. 7: Projection of the controllable regions for OWS on the sagittal phase-space based on Defs. IV.1-IV.2. Given $\boldsymbol{\xi}_c$ (green circle) $\in \Xi_c$ (green dashed region) $\subset \mathcal{C}_{\text{FHWS}}$ (green region), the continuous system state is guaranteed to reach $\mathcal{T}_{\text{switch}}$ (blue region). $\mathcal{T}_{\text{switch}}$ is bounded by pink tangent manifolds and $\mathcal{C}_{\text{SHWS}}$. Switching from the current foot stance to the next foot stance (green and yellow stars respectively) at $\mathcal{T}_{\text{switch}}$ guarantees that the continuous system state will reach $\boldsymbol{\xi}_n$ (yellow circle) $\in \mathcal{T}_{\text{OWS}}$ (red region). The red dashed arrow shows a system state outside of the controllable region and results in a fall.

"funnel"-type geometry that is guaranteed to reach a target set, and correct switching between such funnels ultimately leads to the target set $\mathcal{T}_{\text{OWS}}$ as seen in Fig. 8. A projection of the composition of the controllable regions on the sagittal phase-space satisfying Theorem IV.5 can be seen in Fig. 7.

The controllable regions in Def. IV.1-IV.2 depend on not only the state of the system, but also the high-level keyframe state $\boldsymbol{k}$ as it determines the target set $\mathcal{T}$ as well as the foot placement in the hybrid control input $\boldsymbol{p}_{\text{foot}} \in \boldsymbol{u}$. This further provides safety guarantees within our layered framework.

Theorem IV.5. *The controllable regions for OWS are sequentially and safely composable, i.e.,* $\mathcal{C}_{\text{FHWS}} \cap \mathcal{C}_{\text{SHWS}} = \mathcal{T}_{\text{switch}} \neq \emptyset$, *if the locomotion safety defined in Def. III.2 is satisfied, i.e., the PSP obeying (i) Theorem IV.1, (ii)* $v_{\text{switch}} \leq v_{\max}$, *and (iii) Theorem IV.2 generates a feasible CoM trajectory.*

Proof. To guarantee the feasibility of a nominal phase-space plan generated through forward and backward propagation of the PIPM dynamics, the following two conditions on $v_{\text{apex},n}$ need to be satisfied: (i) $\exists x_{\text{switch}}$ such that $x_{\text{apex},c} \leq x_{\text{switch}} \leq x_{\text{apex},n}$, which is guaranteed by *designing* $v_{\text{apex},n}$ that obeys Theorem IV.1 given a feasible d and current keyframe state $\boldsymbol{\xi}_c \in \Xi_c$; (ii) given a maximum CoM velocity threshold $v_{\max}$, $v_{\text{apex},n}$ is *chosen* such that $v_{\text{switch}} \leq v_{\max}$ through the forward and backward propagation. The designed $v_{\text{apex},n}$ meeting the two conditions above will guarantee feasible phase-space trajectories that safely compose the controllable regions of the two half walking steps, i.e., $\mathcal{C}_{\text{FHWS}} \cap \mathcal{C}_{\text{SHWS}} = \mathcal{T}_{\text{switch}} \neq \emptyset$. Even in the extreme case of that v_{switch} is highly close to $v_{\max}$, we still guarantee $\mathcal{T}_{\text{switch}} \neq \emptyset$ since part of the $\mathcal{T}_{\text{switch}}$ region is below the the normal phase-space trajectory and guaranteed to be feasible.

As for the turning case, given the designed $v_{\text{apex},n}$, condition (iii) constrains the maximally allowable heading angle change $\Delta\theta$ for the next walking step. This condition guar-

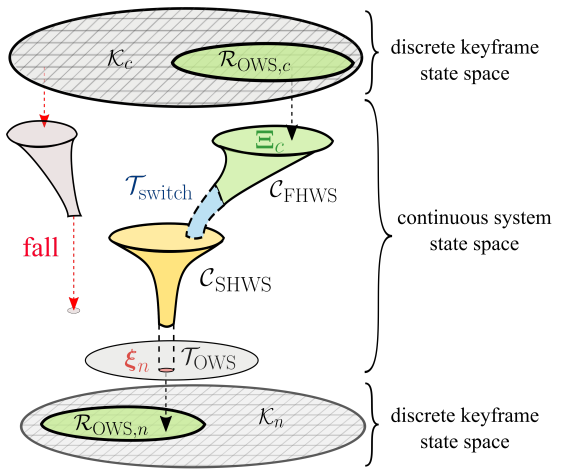

$\mathcal{K}_c$
$\mathcal{R}_{\text{OWS},c}$
discrete keyframe state space
Ξ_c
$\mathcal{T}_{\text{switch}}$
$\mathcal{C}_{\text{FHWS}}$
continuous system state space
fall
$\mathcal{C}_{\text{SHWS}}$
$\boldsymbol{\xi}_n$
$\mathcal{T}_{\text{OWS}}$
$\mathcal{R}_{\text{OWS},n}$
$\mathcal{K}_n$
discrete keyframe state space

Fig. 8: A conceptual illustration of the keyframe transition between two consecutive keyframes, where starting from the current viable keyframe state set $\mathcal{R}_{\text{OWS},c}$, can be projected to the continuous state space and the dynamics are then modeled as the hybrid control system in Eq. (2). Evolvement of the continuous state is formulated as a sequential composition of controllable regions for One Walking Step (OWS) as in Defs. IV.1-IV.2. The state of the system starts from Ξ_c in $\mathcal{C}_{\text{FHWS}}$ (green funnel). After a finite time the state reaches $\mathcal{T}_{\text{switch}}$ (dashed blue region), where the dynamics of the system switch from $\mathcal{C}_{\text{FHWS}}$ to $\mathcal{C}_{\text{SHWS}}$ (yellow funnel). Finally the state of the system reaches $\mathcal{T}_{\text{OWS}}$ (red region). The switch between $\mathcal{C}_{\text{FHWS}}$ and $\mathcal{C}_{\text{SHWS}}$ can occur at any instant within $\mathcal{T}_{\text{switch}}$ and the state would still be guaranteed to reach $\mathcal{T}_{\text{OWS}}$. Any $\boldsymbol{k}_c \in \mathcal{R}_{\text{OWS}}$ is guaranteed to reach $\mathcal{T}_{\text{OWS}}$ based on Def. III.3, while the states outside $\mathcal{R}_{\text{OWS}}$ are considered failed state as shown in the red arrows.

antees that the CoM state in the sagittal and lateral phase-space in the new local coordinate after the turn will not cross the asymptote line of the shaded safety region (see Fig. 6). As such, the controllable region $\mathcal{C}_{\text{FHWS}}$ centering around the nominal PSP trajectory will exist (in the space between the nominal PSP and the asymptote line) and interact wtih $\mathcal{C}_{\text{SHWS}}$ such that $\mathcal{T}_{\text{switch}} \neq \emptyset$. □

According to Theorem IV.5, we can determine a set of $v_{\text{apex},n}$, $\Delta\theta$, and d parameters[6] that satisfy conditions (i)-(iii) thus guaranteeing that the system state can reach the desired target set $\mathcal{T}_{\text{OWS}}$. The selected set are included as safe locomotion specifications in the high-level planner as shown in Fig. 2 and detailed in Sec. VI-C.

Corollary IV.6. *To realize safe walking for an arbitrary number of steps, (i) the target set of the current step is required to be a subset of the controllable region of the first half walking step for the next step, i.e., $\mathcal{T}_{\text{OWS},c} \subset \mathcal{C}_{\text{FHWS},n}$, and (ii) the applied perturbation during execution does not push the system state outside of the controllable regions.*

Fig. 8 conceptually shows how after a system state transition $\boldsymbol{\xi}_n = T(\boldsymbol{k}_c)$, $\boldsymbol{\xi}_n$ can be projected onto the viable keyframe set for the next OWS $\mathcal{R}_{\text{OWS},n}$ to guarantee the viability of next walking step.

V. Keyframe Decision-Making for Navigation Waypoint Tracking

In the previous section, we proposed safety theorems that guarantee locomotion safety. Now we shift our focus to another consideration for safe locomotion by ensuring tracking of the high-level waypoints. The lateral phase-space plan is determined based on the sagittal phase-space plan, as the contact switch timing in the lateral dynamics needs to obey that of the sagittal dynamics. Therefore, the lateral dynamics depend on sagittal apex velocities and sagittal step length. In our previous work [11], the lateral foot placement is solved through a Newton-Raphson search method, such that the lateral CoM velocity is equal to zero at the next CoM apex. While our previous method achieved stable walking and turning, it lacks the guarantee of accomplishing high-level navigation through tracking of the waypoints. Therefore, the lateral CoM motion may not track the desired waypoints. In [12], we propose a heuristic-based policy that restricts the allowable keyframe transitions to achieve waypoint tracking for specific locomotion plans. In this study, we extend our previous work by designing an algorithm that formally manipulates the sagittal phase-space plan to take into account high-level waypoint tracking. Particularly, we use Δy_1 and Δy_2 to track the lateral distance between the CoM at apex and the high-level waypoint as seen in Fig. 3. First, let's define viable ranges for Δy_1 and Δy_2.

Definition V.1 (Viable range for lateral-apex-CoM-to-waypoint distance Δy_1). $\mathcal{R}_{\Delta y_1} := \{\Delta y_1 | \Delta y_1 + \Delta y_2 \leq b_{\text{safety}}\}$, *where b_{safety} denotes the safety boundary around the waypoint.*

Definition V.2 (Viable range for lateral-apex-CoM-to-foot distance Δy_2). *Given the safety criterion for steering walking defined in Theorem IV.2, the viable range for lateral CoM-to-foot distance at apex is defined as* $\mathcal{R}_{\Delta y_2} := \{\Delta y_2 | v_{\text{apex,max}} \cdot \tan\Delta\theta/\omega \leq \Delta y_2 \leq (v_{\text{apex,min}})/(\omega \cdot \tan\Delta\theta)\}$.

$\mathcal{R}_{\Delta y_1}$ and $\mathcal{R}_{\Delta y_2}$ is defined as such to avoid the lateral drift of the robot's CoM and foot location from the high-level waypoint, and further avoid collisions with obstacles. Given Defs. V.1-V.2, we can track the high-level waypoint as follows.

Proposition V.1. *Viable lateral tracking of the high-level waypoint is guaranteed only if (i) Δy_2 and Δy_1 are bounded within their respective viable ranges, i.e., $\Delta y_1 \in \mathcal{R}_{\Delta y_1}$ and $\Delta y_2 \in \mathcal{R}_{\Delta y_2}$, and (ii) the sign of $(\Delta y_1 + \Delta y_2)$ alternates between two consecutive keyframes.*

Proposition V.1 requires that (i) the distance sign of the lateral foot stance position relative to the waypoint alternates between consecutive keyframes and (ii) the waypoints and CoM trajectory are bounded within the lateral foot placement width. An example of this trajectory is shown in Fig. 12.

The analytical solutions of $\Delta y_{1,n}$ and $\Delta y_{2,n}$ are highly nonlinear functions of multiple parameters including the step length d, heading angle change $\Delta\theta$, current and next apex velocities $v_{\text{apex},c}$, $v_{\text{apex},n}$ and the current lateral state of the system $\Delta y_{1,c}$ and $\Delta y_{2,c}$. Thus, it is difficult to quantitatively analyze the relationship between Δy_1, Δy_2 and other parameters aforementioned. Since $(d, \Delta\theta) \in \boldsymbol{a}_{\text{HL}}$ are determined by the navigation policy designed in the high-level task planner, and $v_{\text{apex},c}$, $\Delta y_{1,c}$ and $\Delta y_{2,c}$ are fixed from the previous step, we manipulate $v_{\text{apex},n}$ to adjust the sagittal phase-space

[6]Specific values of d, v_{apex}, and $\Delta\theta$ are included in Table. II, Sec. VII.

plan and subsequently the lateral phase-space plan through the updated walking step timing. To this end, we sample a set of equidistant values $v_{\text{apex},n} \in [v_{\text{apex,min}}, v_{\text{apex,max}}]$ and calculate a cost λ, which penalizes deviation of $\Delta y_{1,n}$ and $\Delta y_{2,n}$ from their respective desired values $\Delta y_{1,d} \in \mathcal{R}_{\Delta y_1}$ and $\Delta y_{2,d} \in \mathcal{R}_{\Delta y_2}$[7]. After the sampling, we set $v_{\text{apex},n}$ to the optimal next apex velocity $v_{\text{apex,opt}}$ that results in the minimum cost. This procedure is presented in Algorithm 1.

Algorithm 1: Optimal Next Apex Velocity Design for Lateral Waypoint Tracking

```
1  Input: $d$, $v_{\text{apex},c}$, $\Delta y_{1,c}$, $\Delta y_{2,c}$, and a velocity
     sampling increment $v_{\text{inc}}$;
2  Set: $v_{\text{apex},n} \leftarrow v_{\text{apex,min}}$, cost $\lambda \leftarrow \infty$, $\Delta y_{1,d}$ and
     $\Delta y_{2,d}$, and cost weights $c_1$ and $c_2$;
3  while $v_{\text{apex},n} \leq v_{\text{apex,max}}$ do
4      $t_{\text{FHWS}}$, $t_{\text{SHWS}} \leftarrow$ sagittal PSP with ($d$, $v_{\text{apex},c}$,
         $v_{\text{apex},n}$);
5      $\Delta y_{1,n}$, $\Delta y_{2,n} \leftarrow$ Newton-Raphson Search [11];
6      $\lambda_{\text{new}} = c_1|\Delta y_{1,d} - \Delta y_{1,n}| + c_2|\Delta y_{2,d} - \Delta y_{2,n}|$;
7      if $\lambda_{\text{new}} < \lambda$ then
8          $\lambda \leftarrow \lambda_{\text{new}}$;
9          $v_{\text{apex,opt}} \leftarrow v_{\text{apex},n}$
10     end
11     $v_{\text{apex},n} \leftarrow v_{\text{apex},n} + v_{\text{inc}}$
12 end
13 Output: $v_{\text{apex},n} = v_{\text{apex,opt}}$
```

Algorithm 1 is robust to different step lengths during straight walking, however waypoint tracking during a turning sequence is more complex. In extreme turning cases that Algorithm 1 fails to find an apex velocity that yields viable waypoint tracking in Proposition. V.1, we will propose an *online* replanning mechanism to adjust the waypoint (see Sec. VI-B).

VI. Task Planning via Belief Abstraction

This section will expound the high-level task planning structure, consisting of global navigation and local action planners that employ linear temporal logic (LTL) to achieve safe locomotion navigation in a partially observable environment with dynamic obstacles. Low-level locomotion dynamics constraints are encoded into LTL specifications to ensure that high-level actions can be successfully executed by the middle-level motion planner to maintain balancing safety.

Definition VI.1 (Navigation Safety)**.** *Navigation safety is defined as safe maneuvering in partially observable environments with uneven terrain while avoiding collisions with static and dynamic obstacles.*

To achieve safe navigation, the task planner evaluates observed environmental events at each walking step and commands a safe action set to the middle-level motion planner as shown in Fig. 2 while guaranteeing goal positions to be visited *in order* and *infinitely often*. In particular, we study a pick-up and drop-off task while guaranteeing static and dynamic obstacle collision avoidance.

[7]$\Delta y_{1,d}$ and $\Delta y_{2,d}$ are heuristically selected according to our Cassie robot's leg kinematics. Exact values of $\Delta y_{1,d}$ and $\Delta y_{2,d}$ are shown in Table. II in Sec.VII-B

We design our task planner using formal synthesis methods to ensure locomotion actions guarantee navigation safety and liveness, specifically we use General Reactivity of Rank 1 (GR(1)), a fragment of LTL. GR(1) allows us to design temporal logic formulas (φ) with atomic propositions (AP (φ)) that can either be True ($\varphi \vee \neg\varphi$) or False ($\neg\mathsf{True}$). With negation ($\neg$) and disjunction ($\vee$) one can also define the following operators: conjunction ($\wedge$), implication ($\Rightarrow$), and equivalence ($\Leftrightarrow$). Other temporal operators include "next" ($\bigcirc$), "eventually" ($\Diamond$), and "always" ($\Box$). Safety specifications capture how the system and environment may transition during one step of the synthesized controller's execution, while liveness specifications capture which transitions must happen *infinitely often*. Further details of GR(1) can be found in [45]. Our implementation uses the SLUGS reactive synthesis tool [46] to design specifications with Atomic Propositions (APs), natural numbers, and infix notation, which are automatically converted to ones using only APs.

The discrete abstraction granularity required to plan walking actions for each keyframe is too fine to synthesize plans for large environment navigation. Therefore, we have split the task planner into two layers: A high-level navigation planner that plays a navigation and collision avoidance game against the environment on a global coarse discrete abstraction, and an action planner that plays a local game on a fine abstraction of the local environment (corresponding to one coarse cell). The action planner generates action sets at each keyframe to progress through the local environment and achieve the desired coarse-cell transition after multiple walking steps.

A. Navigation Planner Design

A top-down projection of the navigation environment is discretized into a coarse two-dimensional grid as shown in Fig. 13. Each time the robot enters a new cell, the navigation planner evaluates the robot's discrete location ($l_{r,c} \in \mathcal{L}_{r,c}$) and heading ($h_{r,c} \in \mathcal{H}_{r,c}$) on the coarse grid, as well as the dynamic obstacle's location ($l_o \in \mathcal{L}_o$), and determines a desired navigation action ($n_a \in \mathcal{N}_a$). The planner can choose for the robot to stop, or to transition to any reachable safe adjacent cell. $\mathcal{L}_{r,c}$ and $\mathcal{L}_o$ denote sets of all coarse cells the robot and dynamic obstacle can occupy, while $\mathcal{H}_{r,c}$ represents the four cardinal directions in which the robot can travel on the coarse abstraction. The dynamic obstacle moves under the following assumptions: (a) it will not attempt to collide with the robot when the robot is standing still, (b) it's maximum speed only allows it to transition to an adjacent coarse cell during one turn of the navigation game, and (c) it will eventually move out of the way to allow the robot to pass. Assumption (c) prevents a deadlock [47]. Static obstacle locations are encoded as safety specifications. Given these assumptions, the task planner in Section VI-D will guarantee that the walking robot can achieve a specific navigation goal.

B. Action Planner Design

The local environment, i.e., one coarse cell, is further abstracted into a fine discretization. At each walking step, the action planner evaluates the robot's state in the environment ($\boldsymbol{e}_{\rm HL}$)[8] consisting of the discrete waypoint location ($l_{r,f} \in \mathcal{L}_{r,f}$) and heading ($h_{r,f} \in \mathcal{H}_{r,f}$) on the fine grid, as well as the robots current stance foot index ($i_{\rm st}$), and determines an appropriate action set ($\boldsymbol{a}_{\rm HL}$) defined in Def. III.1. The action planner generates a sequence of locomotion actions guaranteeing that the robot eventually transitions to the next desired coarse cell while ensuring all action sets are safe and achievable based on $\boldsymbol{e}_{\rm HL}$ and $\boldsymbol{a}_{\rm HL}$. Note that, the fine abstraction also models the terrain height for each fine-level cell, allowing the action planner to choose the correct step height $\Delta z_{\rm foot}$ for each keyframe transition.

During locomotion, the nominal robot state transitions are deterministically modeled within the action planner based on the current game state and system action, however, the nominal transition is not guaranteed. To account for this, we model additional necessary nondeterministic transitions to handle the following cases:

- The robot location is far enough from the centroid of a cell that the same geometric cell transition puts the robot in a different cell at the next step than expected. This occurs because infinite number of continuous locations are captured in one discrete cell.
- Not all the robot states can be captured in the discrete abstraction, such as the robot CoM velocity, which, however, may still affect transitions.
- The robot may be perturbed externally while walking, altering the foot location at the next walking step.

We have encoded nondeterministic transitions, and associated transition flags (t_{nd}), to capture these cases into action planner's environment assumptions. This flag variable t_{nd} is encoded as a special automaton state that will be used to replan the foot location of the next walking step. An example of addressing a sagittal perturbation will be shown in Fig. 15 (c).

An example of modeled nondeterministic transitions can be seen in Fig. 9. The CoM trajectory sometimes imperfectly tracks the waypoints due to accumulated differences in the continuous keyframe state represented by the same discrete state $\boldsymbol{e}_{\rm HL}$. The reduced-order motion planner identifies when the waypoint needs to be shifted from the lateral case and informs the action planner, which verifies the updated waypoint is allowed by the non-deterministic transition model and continues planning from the new waypoint.

C. Encoding Low-level Dynamics Constraints into High-level Planner Specifications

To ensure the action planner only commands safe and feasible actions, we must take into account the underlying *Locomotion Safety*. This is achieved by capturing low-level constraints in the high-level planner specifications. Action

[8]We use the symbol $\boldsymbol{e}_{\rm HL}$ to represent the robot state, since this symbol represents the second player in the game, i.e., the environment player.

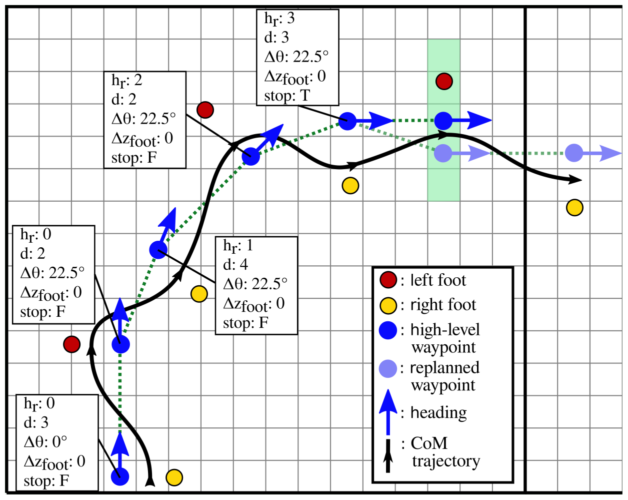


Fig. 9: Illustration of fine-level steering walking within one coarse cell. Discrete actions are planned at each keyframe allowing the robot to traverse the fine grid toward the next coarse cell. The waypoint transitions nondeterministically following the turn. A set of locomotion keyframe decisions are also annotated.

planner state transition limitations based on straight walking step length constraints in Theorem IV.1, and kinematic constraints from the Cassie leg, are directly encoded in the action planner specifications. *Locomotion safety* is guaranteed when the combination of apex velocity, heading angle change, and foot placement meets Theorems IV.1-IV.2. These constraints are not able to be directly captured as the action planner does not reason about CoM velocity and the dynamic equations of motion can not be encoded in symbolic specifications. Instead, they are captured by generating a library of permissible turning sequences based on discrete robot states that are known to meet the above constraints (see Table. II). For example, given $\omega = 3.15$ rad/s, $\Delta y_{1,c} = 0.14$ m (equals to $\Delta y_{1,d}$ in Algorithm 1), and an allowable $v_{\rm apex}$ range $[0.2, 0.7]$ m/s, Theorem IV.2 results in $\Delta\theta \leq 24.40°$. Any turning angle larger than this value will results in a high-level action that is not executable by the middle-level motion planner. Thus we choose $\Delta\theta = 22.5°$ such that we can complete a $90°$ turn in 4 consecutive walking steps. A safe turning sequence can be seen in Fig. 9.

To ensure that collision avoidance in the abstract game translates to collision-free locomotion in the continuous domain, we guarantee the location $l_{r,f}$ stays far enough away from any obstacles. Algorithm. 1 ensures that the distance between $l_{r,f}$ and the robot's desired foot placement does not exceed $b_{\rm safety}$ as detailed in Sec. V. The action planner guarantees $l_{r,f}$ is never in a cell that is less than a distance $b_{\rm safety}$ away from the neighboring coarse cell that may contain static or dynamic obstacles via safety specifications. The planner guarantees this distance even after non-deterministic sagittal and lateral transitions, ensuring collision avoidance.

D. Task Planner Synthesis

A navigation game structure is proposed by including robot actions in the tuple $\mathcal{G} := (\mathcal{S}, s^{\rm init}, \mathcal{T}_N)$ with

- $\mathcal{S} = \mathcal{L}_{r,c} \times \mathcal{L}_o \times \mathcal{H}_{r,c} \times \mathcal{N}_a$ is the augmented state;
- $s^{\rm init} = (l_{r,c}^{\rm init}, l_o^{\rm init}, h_{r,c}^{\rm init}, n_a^{\rm init})$ is the initial state;

- $\mathcal{T}_N \subseteq \mathcal{S} \times \mathcal{S}$ is a transition relation describing the possible moves of the robot and the obstacle.

To synthesize the transition system $\mathcal{T}_N$, we define the rules for the possible successor state locations which will be further expressed in the form of LTL specifications ψ. The successor location of the robot is based on its current state and action $succ_r(l_{r,c}, h_{r,c}, n_a) = \{l'_{r,c} \in \mathcal{L}_{r,c} | \exists l'_o, h'_{r,c}((l_{r,c}, l_o, h_{r,c}, n_a), (l'_{r,c}, l'_o, h'_{r,c}, n'_a)) \in \mathcal{T}_N\}$. We define the set of possible successor robot actions at the next step as $succ_{n_a}(n_a, l_{r,c}, l'_{r,c}, l_o, l'_o, h_{r,c}, h'_{r,c}) = \{n'_a \in \mathcal{N}_a | ((l_{r,c}, l_o, h_{r,c}, n_a), (l'_{r,c}, l'_o, h'_{r,c}, n'_a)) \in \mathcal{T}_N\}$. We define the set of successor locations of the obstacle. $succ_o(l_{r,c}, l_o, n_a) = \{l'_o \in \mathcal{L}_o | \exists l'_{r,c}, h'_{r,c}.((l_{r,c}, l_o, h_{r,c}, n_a), (l'_{r,c}, l'_o, h'_{r,c}, n'_a)) \in \mathcal{T}_N\}$. Later we will use a belief abstraction inspired by [13] to solve our synthesis in a partially observable environment.

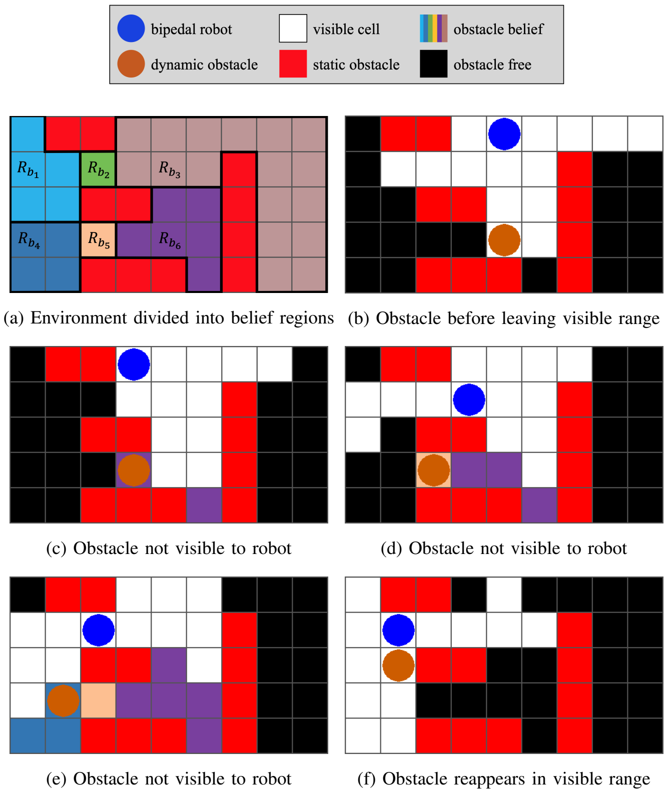
The task planner models the robot and environment interplay as a two-player game. The robot action is Player 1 while the possibly adversarial obstacle is Player 2. The synthesized strategy guarantees that the robot will always win the game by solving the following reactive problem.

Reactive synthesis problem: Given a transition system $\mathcal{T}_N$ and linear temporal logic specifications ψ, synthesize a winning strategy for the robot such that only correct decisions are generated in the sense that the executions satisfy ψ.

The action planner is synthesized using the same game structure as the navigation planner, with possible states and actions corresponding to Section VI-B. Nondeterministic robot location transitions are captured in the robot successor function $succ_{r,f}(l_{r,f}, h_{r,f}, \boldsymbol{a}_{\text{HL}}) = \{l'_{r,f} \in \mathcal{L}_{r,f}, h'_{r,f} \in \mathcal{H}_{r,f} | ((l_{r,f}, h_{r,f}, \boldsymbol{a}_{\text{HL}}), (l'_{r,f}, h'_{r,f}, \boldsymbol{a}'_{\text{HL}})) \in \mathcal{T}_A\}$, where $\mathcal{T}_A$ is the transition relation in the action planner. Compared to the transition relation $\mathcal{T}_N$, $\mathcal{T}_A$ does not have the obstacle location l_o but includes locomotion actions $\boldsymbol{a}_{\text{HL}}$. Given the current robot state and action, $succ_{r,f}$ provides a set of possible locations at the next turn in the game. Obstacle avoidance is taken care of in the navigation game the obstacle location $\mathcal{L}_o$ and successor function $succ_o$ are not needed for action planner synthesis. Since reactive synthesis is used for both navigation and action planners, and the action planner guarantees the robot transition in the navigation game, the correctness of this hierarchical task planner is guaranteed.

E. Belief Space Planning in A Partially Observable Environment

The navigation planner above synthesizes a safe game strategy that is always winning but only in a fully observable environment. We relax this assumption by assigning the robot a visible range only within which the robot can accurately identify a dynamic obstacle's location. To reason about where an out-of-sight obstacle could be, we devise an abstract belief set construction method based on the work in [13]. Using this belief abstraction, we explicitly track the possible discrete locations of a dynamic obstacle, rather than assuming it could be in any non-visible cell. The abstraction is designed by partitioning regions of the environment into sets of discrete belief regions (R_b) and constructing a powerset of these regions ($\mathcal{P}(R_b)$). We choose smaller partitions around static obstacles that may block the robot's view as this allows the planner to guarantee collision-free navigation for a longer horizon like the scenario depicted in Fig. 10. We index each set in $\mathcal{P}(R_b)$ to represent a belief state $b_o \in \mathcal{B}_o$ that captures non-visible regions potentially with a dynamic obstacle.



bipedal robot
visible cell
obstacle belief
dynamic obstacle
static obstacle
obstacle free
R_{b_1}
R_{b_2}
R_{b_3}
R_{b_4}
R_{b_5}
R_{b_6}
(a) Environment divided into belief regions
(b) Obstacle before leaving visible range
(c) Obstacle not visible to robot
(d) Obstacle not visible to robot
(e) Obstacle not visible to robot
(f) Obstacle reappears in visible range

Fig. 10: Simulation showing how the navigation planner's belief evolves when the dynamic obstacle leaves the visible range for several turns. 6 colored belief regions are shown, as well as the robot (blue circle), the dynamic obstacle (orange circle) and static obstacles (red cells). Black cells represent non-visible cells believed to be obstacle free while white cells are visible. The planner believes the obstacle could be in any colored cell depicted, and can therefore reason where the obstacle could and could not reappear, allowing the planner to determine which navigation actions are safe.

The fully observable navigation game structure is modified to generate a partially observable belief-based navigation game with an updated state $\mathcal{S}_{\text{belief}}$ and transition system $\mathcal{T}_{\text{belief}}$ In addition to the obstacle location $l_o \in \mathcal{L}_o$, $\mathcal{S}_{\text{belief}}$ captures the robot's belief of the obstacle $b_o \in \mathcal{B}_o$. A visibility function $vis : \mathcal{S}_{\text{belief}} \to \mathbb{B}$ is added such that it maps the state $(l_{r,c}, l_o)$ to the Boolean as True if and only if l_o is a location in the visible range of $l_{r,c}$. We do not need to modify $succ_{n_a}$ since the dynamic obstacle only affects the possible one-step robot action if it is in the visible range. $succ_r$ also remains the same as the relationship between the robot's actions and its state is not changed by the belief. The set of possible successor beliefs of the obstacle location, b'_o, is defined as $succ_{b_o} = \{b'_o \in \mathcal{B}_o | ((l_{r,c}, b_o, l_o), (l'_{r,c}, b'_o, l_o)) \in \mathcal{T}_{\text{belief}}\}$ where b'_o indexes $\emptyset$ when $vis(l_{r,c}, l'_o) = \text{True}$ and b'_o indexes a nonempty set in $\mathcal{P}(R_b)$ when $vis(l_{r,c}, l'_o) = \text{False}$.

Four classes of belief transitions, shown in Fig. 10, are defined for accurate and meaningful belief tracking:

- Visible to visible: as in the fully observable case, the obstacle may transition to any adjacent visible cell.

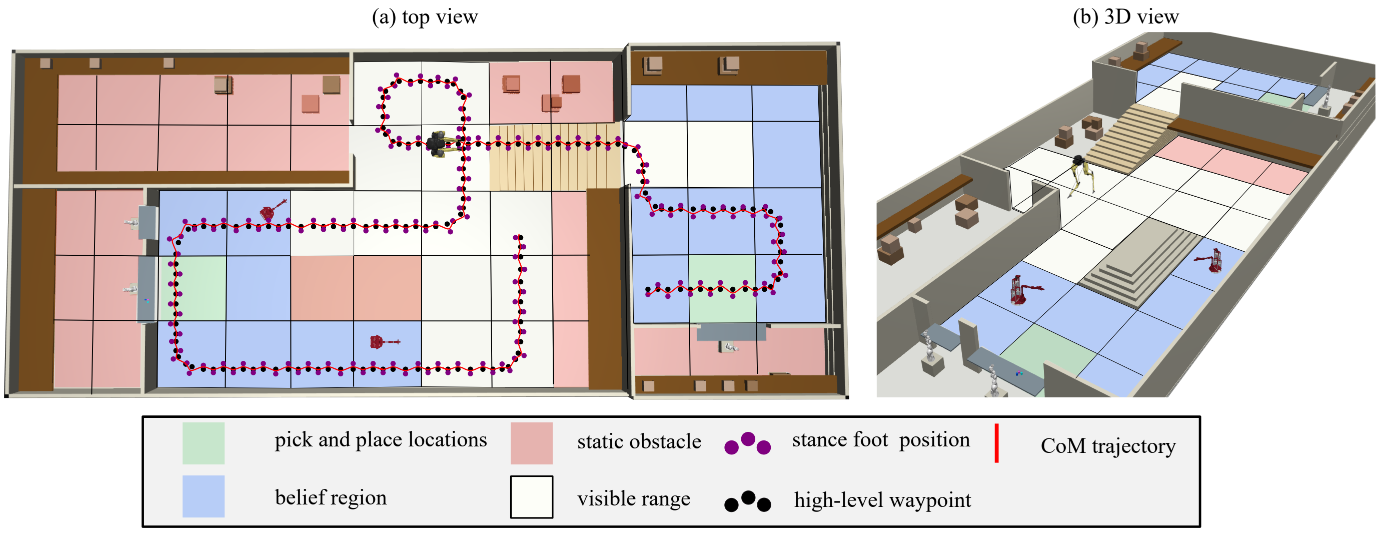


Fig. 11: 3D simulation of the Cassie robot dynamically navigating in the partially observable environment while avoiding collisions with two mobile robots that are treated as dynamic obstacles. Cassie's task is to move between designated initial and goal locations for package delivery. Trajectories of Cassie CoM, foot placements as well as environment coarse-level cell abstraction are shown in subfigure (a). Subfigure (b) shows a 3D view of the tested environment.

- Visible to belief: the belief state represents the set of regions containing non-visible cells adjacent to the obstacle's previous visible location.
- Belief to belief: the obstacle could be in any non-visible (or newly visible[9]) cell represented by the current belief state, the next belief state represents the current belief plus the belief regions the dynamic obstacle could have entered given its limited motion capability.
- Belief to visible: similar to the previous case, the current obstacle may be in any non-visible or newly visible cell represented by the planner's belief, and may move to any adjacent cell, which defines the visible cells it could appear in at the next time step.

This method of belief tracking guarantees that all real transitions the obstacle can make during its turn are captured in the planner's belief. When the obstacle enters cells in a new belief region, the planner believes it could be anywhere in that region, therefore the belief is an over-approximation of possible obstacle locations. We guarantee that the obstacle is within the regions captured by the belief state, therefore we can guarantee that the obstacle can only appear in a visible cell when there is a modeled transition from the current belief state to that cell. Since both the action planner and the allowable navigation actions remain the same for the partially observable game, the game captures the same safety guarantees, but allow for a larger set of navigation options than would be possible without tracking the belief of the dynamic obstacle's location.

F. Belief Tracking of Multiple Obstacles

Our task planner is extensible to environments with multiple dynamic obstacles. It is possible to directly add any number of additional obstacles and their associated beliefs to the navigation planning game, however, the synthesis has polynomial time complexity. To improve computational tractability, we merge all non-visible obstacles' believed states into one combined belief region. Reasoning about a combined belief region still allows the planner to guarantee collision-free navigation without the complexity of tracking each obstacle individually.

To model a combined belief state we separate the obstacles' state from it's belief. Each obstacle's state is either a visible cell on the grid, or and index representing the obstacle is not visible ($l_{o,i,c} \in L_{o,i,c} | L_{o,i,c} = \mathcal{L}_o + \mathcal{I}_{nv}$). The joint belief state consists of the powerset of belief regions, including the empty set when all obstacles are visible. ($b_{oj} \in \mathcal{B} | \mathcal{B} = \mathcal{P}(R_b)$).

We generate a new multi-obstacle game structure $\mathcal{G}_{\text{combined-belief}} := (\mathcal{S}_{\text{belief}}, s^{\text{init}}_{\text{belief}}, \mathcal{T}_{\text{belief}}, vis)$ with

- $\mathcal{S}_{\text{belief}} = \mathcal{L}_{r,c} \times \mathcal{L}_{o,i,c} \times \mathcal{B}_o \times \mathcal{H}_{r,c} \times \mathcal{N}_a$;
- $s^{\text{init}}_{\text{belief}} = (l^{\text{init}}_{r,c}, l^{\text{init}}_{o,i,c}, \{b^{\text{init}}_o\}, h^{\text{init}}_{r,c}, n^{\text{init}}_a)$ is the initial location of the obstacle known a priori;
- $\mathcal{T}_{\text{belief}} \subseteq \mathcal{S}_{\text{belief}} \times \mathcal{S}_{\text{belief}}$ are possible transitions where $((l_{r,c}, l_{o,i,c}, b_o, h_{r,c}, n_a), (l'_{r,c}, l'_{o,i,c}, b'_o, h'_{r,c}, n'_a)) \in \mathcal{T}_{\text{belief}}$;
- $vis : \mathcal{S}_{\text{belief}} \to \mathbb{B}$ is a visibility function that maps the state $(l_{r,c}, l_{o,i})$ to the boolean as True iff $l_{o,i}$ is a real location in the visible range of $l_{r,c}$.

This game requires new specifications that govern $succ_{o,i}(l_{r,c}, l_{o,i,c}, b)$ and $succ_b(l_{r,c}, l_{o,i,c}, b)$, the allowable successor obstacle state and joint belief state, all other successor functions remain the same. Even though the belief can represent multiple obstacles, the possible belief-to-belief transitions are the same as when the belief state represents a single obstacle. The key specifications to be changed are those governing $succ_{b_o}$ when an obstacle enters or exits the visible range. These changes can be made in the specifications defining the successor belief state $succ_{b_o}$.

VII. Results

This result section evaluates the performance of (i) the high-level task planner by assessing its task completion, collision avoidance, and safe action execution; (ii) the middle-level

[9]Due to the turn-based nature of the game, an obstacle may be in the robot's visible range after the robot makes a move, but the obstacle may move from this newly visible cell before the robot reevaluates its new visible range.

motion planner by employing our designed keyframe decision maker to choose proper keyframe states and generating safe locomotion trajectories; (iii) recoverability against perturbations using the reachability-based synthesized controller; and (iv) the feasibility of using the reduced-order motion plan to generate whole-body motion trajectories. The results are simulated using the Drake toolbox [48], and the open-source code can be found here `https://github.com/GTLIDAR/safe-nav-locomotion.git`. A video of the simulations is `https://youtu.be/w-SrjuUb078`.

A. LTL Task Planning Implementation

The task planner is evaluated in an environment with multiple static and dynamic obstacles, and two rooms with different ground heights connected by a set of stairs as seen in Fig. 11. To generate the navigation planning abstraction, the environment is discretized into a 10×5 coarse grid, with a 2.7×2.7 m^2 cell size. $\mathcal{L}_{r,c}$ is the set of all accessible discrete cells, $\mathcal{H}_{r,c}$ is the set of cardinal directions, and $\mathcal{N}_a$ is a set of navigation actions in those cardinal directions (N, E, S, W). Each coarse cell is further discretized into a finer 26×26 grid for local action planning. We model the possible actions as step length $d \in \{\mathsf{small1}, \mathsf{small2}, \mathsf{medium1}, \mathsf{medium2}, \mathsf{large1}, \mathsf{large2}\}$, heading change $\Delta\theta \in \{\mathsf{left}, \mathsf{none}, \mathsf{right}\}$), and step height $\Delta z_{\text{foot}} \in \{z_{\text{down2}}, z_{\text{down1}}, z_{\text{flat}}, z_{\text{up1}}, z_{\text{up2}}\}$. The possible heading changes $\Delta\theta \in \{-22.5°, 0°, 22.5°\}$, are constrained by the minimum number of steps needed to make a 90° turn, and the maximum allowable heading angle change that results in viable keyframe transitions as defined in Theorem IV.2. We choose $\Delta\theta = \pm 22.5°$ so that a 90° turn can be completed in four steps as shown in Fig. 9. Completing the turn in fewer steps is not feasible as it would overly constrain v_{apex}, as can be seen in Fig. 6(b). Due to the allowable heading change of $\pm 22.5°$, $\mathcal{H}_{r,f}$ contains a discrete representation of the 16 possible headings the robot could have.

A set of specifications is designed to describe the allowable successor locations and actions in the transition system. Here, we only show a few specifications as examples:

$$\begin{aligned}
&\square\big((h_{r,f} = \mathcal{H}_{r,c} \wedge ((i_{\text{st}} = \mathsf{left} \wedge \Delta\theta = \mathsf{right}) \\
&\vee (i_{\text{st}} = \mathsf{right} \wedge \Delta\theta = \mathsf{left})) \Rightarrow \bigcirc(d = \mathsf{medium2})\big), \quad (7)\\
&\square\big((h_{r,f} = \mathcal{H}_{r,c} \wedge ((i_{\text{st}} = \mathsf{left} \wedge \Delta\theta = \mathsf{left}) \\
&\vee (i_{\text{st}} = \mathsf{right} \wedge \Delta\theta = \mathsf{right})) \Rightarrow \bigcirc(d = \mathsf{small2})\big), \quad (8)
\end{aligned}$$

which govern the allowable step length during the first step of a turning process.

Both navigation and action planners are constructed by combining environment assumptions and system specifications generated by the successor functions described in Sections VI-D and VI-F into a transition system and using the LTL synthesis tool SLUGS to generate a winning strategy. Synthesis occurs offline, and the winning strategy is efficiently encoded in a binary decision diagram (BDD) [49] which can be accessed online by interfacing the controller directly with SLUGS. At each turn of the game, the controller computes the new abstracted environment state and passes it to SLUGS which returns the corresponding system action.

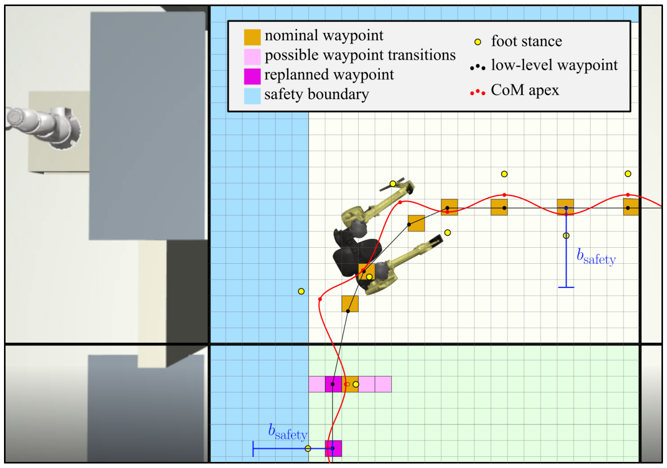


Fig. 12: Illustration of *online* updating the high-level waypoint to maintain lateral tracking at the middle-level motion planner. The high-level waypoint is also required to keep a safe distance away from the adjacent coarse cell to avoid collisions with static or dynamic obstacles. In this run, we set the safety boundary to be 6 fine cells as shown in light blue.

TABLE I: Successful motion plan results for the pick and place task

Steps	**turns**	**waypoint correction**
200	9	4
260	17	12
500	29	22

B. Nominal Online Planning for A Pick and Place Task

The middle-level motion planner is able to generate CoM trajectories of the ROM for a pick and place task infinitely often that include traversing stairs, steering, stopping, and avoiding dynamic obstacles. The keyframe decision maker, detailed in Sec. V, selects the optimal next keyframe for waypoint tracking. The action planner interfaces with the middle-level motion planner *online* to pass the action set for the next keyframe. In the case when the keyframe decision maker cannot satisfy the lateral tracking of high-level waypoints in Proposition V.1, a new non-deterministic transition from the action planner is selected based on the modified lateral phase-space plan *online*. The action planner receives the updated waypoint which allows the planner to chose the correct transition to the next game state. Our simulation shows that the robot successfully traverses uneven terrain to complete its navigation goals while steering away from dynamic obstacles when they appear in the robot's visible range. The robot's navigation trajectory is shown in Fig. 11. The tracking results for multiple plans with different obstacle paths are detailed in Table. I using PSP parameters given in Table. II. Waypoint correction only occurs in the last step of a turning sequence due to the complexity of lateral tracking during steering scenarios. 12 out of 260 steps[10] result in alternative discrete state transitions in the lateral direction, all of which were seamlessly handled by the action planner as shown in Fig. 12. This result show that the integration of the high-level planner and the middle-level motion planner in an *online* fashion allows for successful and safe TAMP.

[10]the step count refers to the number of high-level actions received by the middle-level motion planner, which includes stopping actions

TABLE II: Nominal PSP parameters values

parameter	value	parameter	value
$v_{\text{apex,min}}$	0.20 m/s	$v_{\text{apex,max}}$	0.70 m/s
h_{apex}	0.985 m	Δz_{foot}	$\{0, \pm 0.1, \pm 0.2\}$ m
$\Delta y_{1,d}$	0.10 m (0.0 m for steering)	$\Delta y_{2,d}$	0.14 m
c_1	1.0 (7 for steering)	c_2	1.0 (4 for steering)
$\Delta\theta$	$\{0°, \pm 22.5°\}$	b_{safety}	0.52 m
d	$\{0.21, 0.28, 0.31, 0.38, 0.42, 0.43, 0.47, 0.52\}$ m	v_{inc}	0.01 m/s

C. Belief Space Planning

The belief abstraction in the navigation planner is successful in tracking and bounding nonvisible obstacles as can be seen in Fig. 10. The tracked belief enables the robot to navigate around static obstacles while guaranteeing that the dynamic obstacles are not in the immediate non-visible vicinity. Fig. 13 depicts a snapshot of a simulation where the robot must navigate around such an obstacle to reach its goal states. The grid-world environment is abstracted into 6 distinct belief regions resulting in 64 possible belief states. A successful strategy can be synthesized only when using a belief abstraction. Without explicitly tracking possible non-visible obstacle locations, the task planner believes the obstacle could be in any non-visible cell when it is out of sight, including the adjacent visible cell in the next turn of the game. That means the planner can not guarantee collision avoidance and is not able to synthesize a strategy that would allow the robot to advance. Fig. 13b depicts a potential collision that could occur in pink. This comparison underlines the significance of the belief abstraction approach.

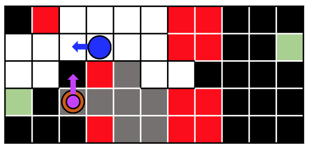
The belief abstraction provides additional information for deciding long-horizon navigation actions beyond guaranteeing immediate collision avoidance. In the simulation shown in Fig. 11, it is challenging to navigate around the vision occluding static obstacles at the lower-level (including the walls and a multi-stair platform). The synthesized strategy reacts to the additional information about the dynamic obstacle provided by belief tracking in three distinct ways. Based on the belief, the robot either (i) continues on the most direct route to the goal location; (ii) loops around to the right and positions itself to be able to go around either side of the static obstacle; or (iii) stops and waits until the dynamic obstacle disappears (see the result in the simulation video). The planner can choose any of these three strategies as long as all safety specifications are met. This nondeterministic mechanism offers the task planner flexibility in choosing safe navigation actions.

Generating global navigation task planners for two dynamic obstacles using a joint belief abstraction requires only 40% of the synthesis time as that of independently tracking the belief state of each obstacle. Specifically, synthesizing a strategy for the scene in Fig. 11 with two dynamic obstacles took 34 mins using joint belief tracking and 85 mins when individually tracking the belief of each obstacle.

D. Safe Recoverability and Replanning

The proposed sequential composition of controllable regions and reachability analysis in Sec. IV-B allows our middle-level motion planner to be robust against perturbations exerted on the CoM in the sagittal space. Given a keyframe transition for OWS, the synthesized controller is able to guarantee that the CoM state reaches the targeted state within OWS, thus successfully completing a OWS safely. In Fig. 14(b) we show the composition of controllable regions for multiple walking step and demonstrate that the CoM trajectory is recoverable when employing the synthesized controller. Table. III shows the success rate for randomly generated keyframe transitions, where the step length is $d_1 = 0.312$ m, $d_3 = 0.416$ m and $d_3 = 0.52$ m. The data is generated using ROCS [43] with 1000 runs for each desired keyframe transition, a randomly selected $\boldsymbol{\xi}_c \in \Xi_c$ and the applied disturbance bound $\tilde{\Xi}_{\text{applied}}$ is uniformly distributed within $[-2, 2]$ m for CoM position and $[-5, 5]$ m/s for CoM velocity. The controllable regions are synthesized with state space granularity of (0.002 m, 0.004 m/s), a control input $\omega \in [2.8, 3.5]$ rad/s with a granularity of 0.02 rad/s, and the added noise bound at synthesis $\tilde{\Xi}_{\text{synthesis}}$ is uniformly distributed within $[-0.01, 0.01]$ m for CoM position and $[-0.02, 0.02]$ m/s for CoM velocity. In Fig. 14(a), we show 15 successful random keyframe transitions where $v_{\text{apex},n} = [0.45, 0.7]$ m/s and $d = 0.415$ m.

Large perturbations can push the system state outside of the controllable regions and the synthesized controller cannot recover to $\mathcal{T}_{\text{switch}}$. To safely recover from such large perturbations, we employ a variant of the capture point formulation


(a) With explicit belief tracking

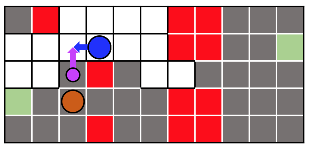
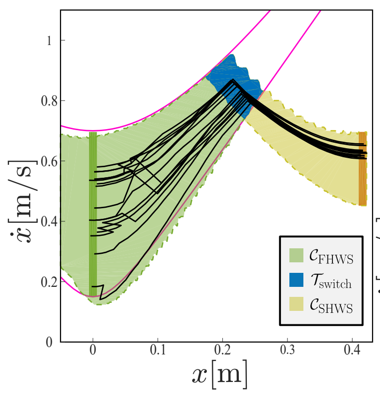

(b) No explicit belief tracking

Fig. 13: A snapshot of the coarse-level navigation grid during a simulation where the robot (blue circle) is going between the two goal states (green cells), while avoiding a static obstacle (red cells) and a dynamic obstacle (orange circle). White cells are visible while grey and black cells are non-visible. Gray cells represent the planner's belief of potential obstacle locations. The closest distance the obstacle could be to the robot, as believed by the planner, is depicted by the pink circle.

(a)



(b)

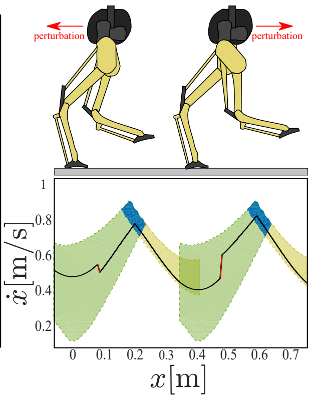


Fig. 14: Results of OWS robust PSP. (a) shows a 15 random keyframe transitions with bounded disturbances, where $\mathcal{T}_{\text{OWS}} = (0.416 \text{ m}, [0.45, 0.7] \text{ m/s})$. (b) Composition of controllable regions of OWS. Here, we demonstrate that the synthesized controller is able to handle the perturbed CoM trajectory, shown as a black solid line, inside the superimposed controllable regions and successfully complete multiple steps when controllable regions are composed as proposed in Corollary IV.6.

TABLE III: Success rate of perturbed OWS transitions

$v_{\text{apex},n}$ margin	Success Rate		
	d_1	d_2	d_2
$[0.2, 0.45]$ m/s	90.2%	91.6%	92.5%
$[0.45, 0.7]$ m/s	91.8%	92.2%	93.6%

[11], [50] to redesign the next foot position $x_{\text{foot},n}$ while maintaining the desired $v_{\text{apex},n}$ via the following formula:

$$x_{\text{foot},n} = x_{\text{switch}} + \frac{1}{\omega}(\dot{x}^2_{\text{switch,dist}} + v^2_{\text{apex},n})^{1/2} \qquad (9)$$

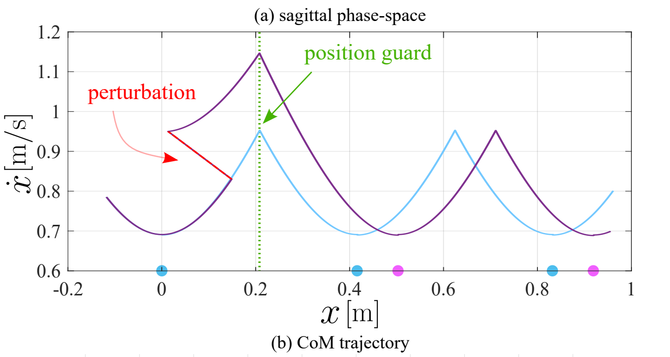
where x_{switch} is determined analytically based on the nominal transition, and $\dot{x}_{\text{switch,dist}}$ is the post-disturbance sagittal CoM velocity at switch instant and computed through a position guard $x = x_{\text{switch}}$ shown as the vertical dashed line in Fig.15 (a). The nominal foot position is determined by the high-level waypoint. In case that the new foot location lands in a different fine cell, the *online* integration mechanism between the high-level and middle-level will update the action planner for a new waypoint location as shown in Fig. 15(b) and (c). The action planner reacts to to the perturbation by replanning d and $\Delta\theta$ in a_{HL}, which further induces a waypoint change at the next walking step. In particular, the nondeterministic transition flag $t_{nd} = \{\text{nominal}, \text{forward}, \text{backward}\}$ indicates the perturbation direction. The automata shown in fig. 15 (c) is a fragment from the larger action planner consisting of 21447 nodes. The navigation planner automaton has 20545 nodes. Online resynthesis of these planner automata is computationally intractable, and thus we incorporate the nondeterministic transition flag t_{nd} into the automaton offline synthesis and employ them online for action replanning.

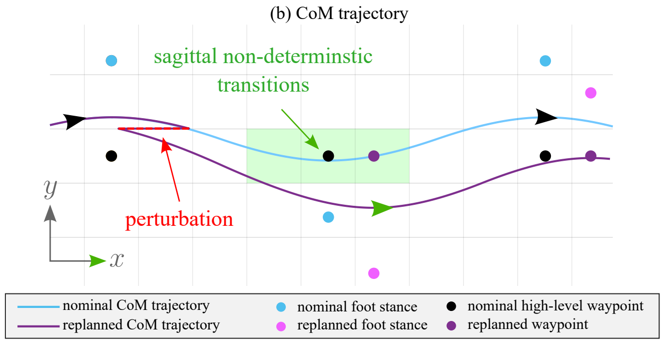
E. Whole-body Non-periodic Trajectory Generation

To validate the feasibility and effectiveness of the synthesized phase-space trajectory on the bipedal robot Cassie, we exploit one-walking-step trajectory optimization (OWS-TO) to generate full-body motions that meet strict constraints of the keyframe hyperparameters in Fig. 2.

The selected keyframe transitions in the phase-space plan allow non-periodic gaits for rough terrain traversability. A subset of keyframe state $\boldsymbol{k}$ is selected as hyperparameters: the step length d, sagittal apex velocities at the current and next walking step $v_{\text{apex},c}$, $v_{\text{apex},n}$, and the heading change $\Delta\theta$. We select viable ranges of these hyperparameters based on Cassie's physical feasibility. These hyperparameter ranges are discretized to create a finite set of combinatorial gait parameters. These parameter combinations are enforced as boundary conditions in the TO, and produce non-periodic motion primitives between keyframes. We choose $d \in [0.2, 0.6]$ m with an 0.1 m increment, $v_{\text{apex}} \in [0.2, 0.5]$ m/s, and $\Delta\theta \in [-30°, 0°, 30°]$. The possible actions that the high-level planner (Sec. VI-D) can take are bounded within the range of the discretized hyperparameters, so we can always take the action to interpolate the motion primitives in the offline generated library to obtain a full-body motion.

The OWS-TO solves the optimal full-body trajectory by minimizing the sum of square of control efforts (i.e., joint torques) along a trajectory that obeys physical constraints. We represent the trajectory with a finite number of discrete





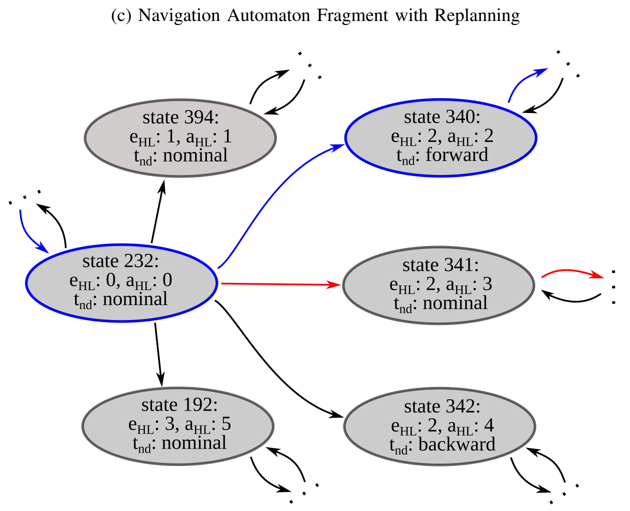

(c) Navigation Automaton Fragment with Replanning
state 394: e_{HL}: 1, a_{HL}: 1 t_{nd}: nominal
state 340: e_{HL}: 2, a_{HL}: 2 t_{nd}: forward
state 232: e_{HL}: 0, a_{HL}: 0 t_{nd}: nominal
state 341: e_{HL}: 2, a_{HL}: 3 t_{nd}: nominal
state 192: e_{HL}: 3, a_{HL}: 5 t_{nd}: nominal
state 342: e_{HL}: 2, a_{HL}: 4 t_{nd}: backward

Fig. 15: Safe recovery from a large perturbation. (a) shows the sagittal phase-space plan, where a position guard is used to determine a safe replanned foot location to recover from the perturbation. (b) shows the CoM trajectory in Cartesian space and the *online* integration of the high-level action planner and the middle-level PSP for a waypoint modification. (c) shows a fragment of the synthesized action planner automaton capturing modeled nondeterministic transitions (with the associated flag t_{nd}). For each next state of the environment (e_{HL}), there is a set of game states corresponding to all possible t_{nd}. Blue transitions capture the replanned execution when the robot CoM is perturbed forward while red transitions depict a nominal execution without any perturbation. Numerical values for e_{HL} and a_{HL} index distinct environment state and robot action sets in the algorithm implementation.

node points with equal time intervals. Each node contains state variable (full-body joint position and velocity $(q, \dot{q})$) and control variables (full-body joint torque τ), which are used to construct the cost and constraint functions.

Due to the inherent hybrid nature of legged systems, the formulated TO comprises $D = 2$ stance phases separated by a foot contact switch. The TO problem is formulated using the direct collocation method [51], which places collocation points in the middle of each two nodes to enforce the dynamics constraints. In our implementation, each phase is composed of

$N_j = 11$ nodes. We transcribe an initial trajectory into a nonlinear program (NLP):

$$\begin{aligned}
\underset{X}{\arg\min} \quad & \sum_{j=1}^{D}\sum_{i=0}^{N_j} \Omega_j \cdot \mathcal{J}_j(\tau_i^j) && (10)\\
\text{s.t.} \quad & H_j(q_i^j)\ddot{q}_i^j + V_j(q_i^j, \dot{q}_i^j) = \tau_i, && \text{(dynamics)}\\
& \dot{q}_0^{j+1} = \Delta_j(\dot{q}_{N_j}^j), && \text{(reset map)}\\
& f_{c,z} \geq 0, \ |f_{c,xy}| \leq \mu f_{c,z}, && \text{(friction)}\\
& C_j^{\text{path}}(q_i^j, \dot{q}_i^j, \tau_i^j) \leq 0, && \text{(path)}\\
& C_j^{\text{bound}}(q_i^j, \dot{q}_i^j) = 0 && \text{(hyperparameters)}
\end{aligned}$$

The NLP above describes the problem for solving the optimal state-control decision variable $X^* = \{q_i^{j*}, \dot{q}_i^{j*}, \tau_i^{j*}\}$. By minimizing the cost of pseudo energy $\mathcal{J}_j = ||\tau_i^j||^2$ with weights Ω_j while enforcing the physical constraints, we get continuous trajectory by interpolate between the optimal decision variables. The discretized full-body dynamics are enforced, which contains H and V denoting the inertia, bias force matrices of the equation of motion. The state after the contact switch $\dot{q}_0^{j+1} = \Delta_j(\dot{q}_{N_j}^j)$ is mapped from the state before the switch. We adopt the linear complementarity constraints for non-slippery ground contact. The path constraints also enforce joint and torque limits of the robot system. As for the boundary constraint of each OWS, the sagittal apex velocity, heading angle change, and step length follow the desired combinations of keyframe hyperparameters.

We use the fast robot optimization and simulation toolkit (FROST) [52] to construct the NLP and exploit the IPOPT [53] solver to offline find a set of 585 OWS motion primitives in Matlab. The TO stopping criteria of the constraints are on the order of 10^{-3}. We use the Intel i7-8700 CPU that takes 236 minutes to create the whole motion primitive set. The final trajectory are parameterized as a piecewise polynomial using the state and control decision variables. For each OWS phase-space trajectory, we extract the hyperparameters at the keyframe and generate the corresponding full-body trajectory through interpolation between the motion primitives in the set. We show Cassie's full-body CoM trajectory in Fig. 16 for a continuous straight walking and a 90° turning in 4 steps. We evaluate the discrepancy between the CoM trajectory from the phase-space plan and that estimated from the generated full-body motion, using the root-mean-square (RMS) distance metric. The RMS between the trajectories are 0.011 m for straight walking and 0.017 m for the four-steps turning. For the CoM trajectory solved by the full-body TO, there is a vertical velocity jump at the foot transition instant (see the red trajectory in Fig. 16(a)-(b)). This is induced by the impact dynamics with ground contact. The proposed TO method validates the feasibility of deploying the reduced-order phase-space trajectories onto Cassie hardware.

VIII. Discussion and limitations

Belief tracking expands the guaranteed safe navigation actions available to the navigation planner. Merging the belief of multiple dynamic obstacles into one abstract state captures less information than individual obstacle tracking by design. This

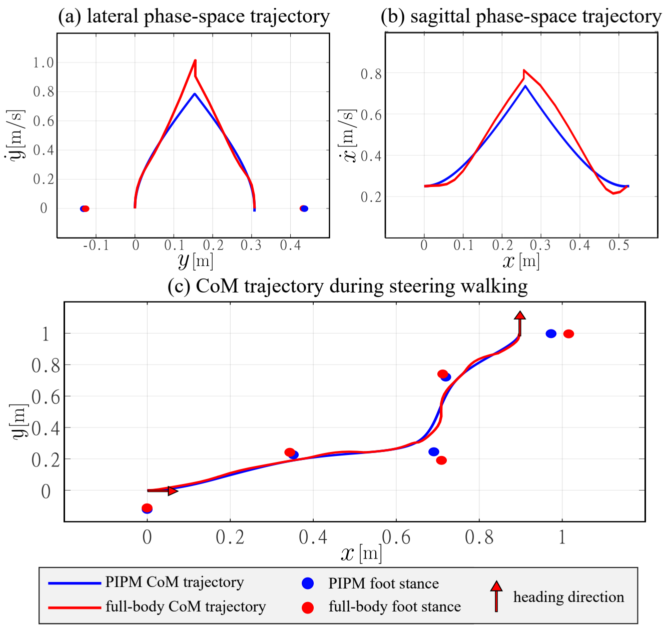


Fig. 16: Comparison of the phase-space CoM trajectory with that estimated from the full-body motion generated from TO. (a) and (b) show the OWS of continuous straight walking. (c) illustrates a four-step turning sequence. The full-body motion, matching the phase-space trajectory, can successfully maneuver the robot to execute a safe turn.

reduces computational complexity while providing the same guarantees of capturing dynamic obstacle locations. One path to enhance the proposed framework in the future is to model small obstacles in the action planner so that an entire coarse cell containing such obstacles are still accessible to the robot in the navigation game. Additionally, the library of turning sequences can be expanded to incorporate more aggressive navigation decisions while obeying the safety criteria proposed in Sec. IV-A and to include fine-level obstacle avoidance maneuvers.

Our proposed safe PSP demonstrates successful execution of high-level actions under nominal conditions for a large number of walking steps as detailed in Table. I. While the framework still lacks formal guarantee on successful lateral tracking of the high-level waypoints for *infinite* number of steps or under extremely large perturbations, our results show empirical guarantees afforded by the integration of the formal navigation and obstacle avoidance guarantees in the high-level task planner in Sec. VI, locomotion safety guarantees in Sec. IV-A, and the online replanning algorithm for waypoint tracking in Sec. V. Such empirical guarantees are shown in Figs. 11-12 and Table. I. Moreover, in real-life deployment of our framework on Cassie, *correction-as-needed* strategies can be explored for specific tasks, such as stopping motions at pick and place locations, therefore robot state deviation will not accumulate during *infinite* walking step scenarios.

The success rate of completing OWS safely under perturbation highly depends on various factors such as the state space granularity, robot actuation capability, environmental uncertainties, and the locomotion phase when the perturbation is applied. Comprehensive analysis of the success rate with respect to these factors is beyond the scope of this study. In the future, we plan to design more advanced safety criteria addressing adversarial pedestrian in the environment [5], [54], [55] and contact uncertainties from terrain [56]. Moreover, the reachability analysis is based on the ROM dynamics, therefore a discrepancy will be induced when taking into account Cassie's full-body dynamics.

In Sec. VII-E, we validate the feasibility of the reduced-order motion plans to generate dynamics-consistent whole-body motions. Our results in Fig. 16 show that the whole-body motion of Cassie is able to successfully track the reduced-order motion plan. This is the first yet significant step towards deployment of our proposed framework on Cassie hardware and remains a motivation for future work.

IX. Conclusion

Long-horizon and formally-guaranteed safe TAMP in complex environments with dynamic obstacles has long been a challenging problem, specifically for underactuated bipedal systems. On the other hand, symbolic planners are powerful in providing formal guarantees on safety and task completion in complex environments. For this reason, integrating high-level formal methods and low-level safe motion planning ought to be explored by the locomotion community to attain formally safe TAMP for real-life tasks. The way we address this problem is through multi-level safety in a hierarchically integrated planning framework.

Our proposed TAMP framework seamlessly integrates low-level locomotion safety specifications into a formal high-level LTL synthesis, to guarantee safe execution of the high-level commands. The middle-level motion planner generates non-period motion plans that accurately execute the safe high-level actions. Our high-level planner employs a belief abstraction to address the partial observability of a large environment and guarantees safe navigation. We also investigate robustness against external perturbation through safe sequential composition of keyframe states to achieve robust locomotion transitions. The generated CoM motion plans are also validated to be viable for generating full-body locomotion gaits of the 20-degree-of-freedom Cassie bipedal robot.

Appendix I
Analytical Solution for PIPM Dynamics

When the CoM motion is constrained within a piece-wise linear surface parameterized by $h = a(x - x_{\text{foot}}) + h_{\text{apex}}$, the reduced-order model becomes linear and an analytical solution exists:

$$x(t) = Ae^{\omega t} + Be^{-\omega t} + x_{\text{foot}} \quad (11)$$
$$\dot{x}(t) = \omega(Ae^{\omega t} - Be^{-\omega t}) \quad (12)$$

where $\omega = \sqrt{\frac{g}{h_{\text{apex}}}}$, $A = \frac{1}{2}((x_0 - x_{\text{foot}}) + \frac{\dot{x}_0}{\omega})$, $B = \frac{1}{2}((x_0 - x_{\text{foot}}) - \frac{\dot{x}_0}{\omega})$. manipulate Eq. (11)-(12) gives

$$x + \frac{\dot{x}}{\omega} - x_{\text{foot}} = 2Ae^{\omega t} \quad (13)$$

which renders

$$t = \frac{1}{\omega}\log(\frac{x + \frac{\dot{x}}{\omega} - x_{\text{foot}}}{2A}) \quad (14)$$

To find the dynamics, $\dot{x} = f(x)$, which will lead to the switching state solution, remove the t term by plugging Eq. (14) into Eq. (11).

$$\frac{1}{2}(x - \frac{\dot{x}}{\omega} - x_{\text{foot}}) = \frac{2AB}{x + \frac{\dot{x}}{\omega} - x_{\text{foot}}} \quad (15)$$

$$(x - x_{\text{foot}})^2 - (\frac{\dot{x}}{\omega})^2 = 4AB \quad (16)$$

which yields

$$\dot{x} = \pm\sqrt{\omega^2((x - x_{\text{foot}})^2 - (x_0 - x_{\text{foot}})^2) + \dot{x}_0^2} \quad (17)$$

If the apex height is constant, then ω is constant. According to the constrain that sagittal velocity should be continuous, the saggital switching position is obtained by

$$x_{\text{switch}} = \frac{1}{2}(\frac{C}{x_{\text{foot},n} - x_{\text{foot},c}} + (x_{\text{foot},c} + x_{\text{foot},n})) \quad (18)$$

where

$$\begin{aligned} C =& (x_{\text{apex},c} - x_{\text{foot},c})^2 - (x_{\text{apex},n} - x_{\text{foot},n})^2 \\ &+ \frac{v_{\text{apex},n}^2 - v_{\text{apex},c}^2}{\omega^2} \end{aligned} \quad (19)$$

Appendix II
Proof of Theorem IV.1

Proof. First, the sagittal switching position can be obtained from the analytical solution in Appendix I:

$$x_{\text{switch}} = \frac{1}{2}(\frac{C}{x_{\text{foot},n} - x_{\text{foot},c}} + (x_{\text{foot},c} + x_{\text{foot},n})) \quad (20)$$

where $C = (x_{\text{apex},c} - x_{\text{foot},c})^2 - (x_{\text{apex},n} - x_{\text{foot},n})^2 + (\dot{x}_{\text{apex},n}^2 - \dot{x}_{\text{apex},c}^2)/\omega^2$. This walking step switching position is required to stay between the two consecutive CoM apex positions, i.e.,

$$x_{\text{apex},c} \leq x_{\text{switch}} \leq x_{\text{apex},n} \quad (21)$$

which introduces the sagittal apex velocity constraints for two consecutive keyframes as follows.

$$\begin{aligned} &\omega^2(x_{\text{apex},n} - x_{\text{apex},c})(x_{\text{apex},c} + x_{\text{apex},n} - 2x_{\text{foot},n}) \\ &\quad \leq v_{\text{apex},n}^2 - v_{\text{apex},c}^2 \leq \\ &\omega^2(x_{\text{apex},n} - x_{\text{apex},c})(x_{\text{apex},c} + x_{\text{apex},n} - 2x_{\text{foot},c}) \end{aligned} \quad (22)$$

Given this bounded difference between two consecutive CoM apex velocity squares, the corresponding safe criterion for straight walking can be expressed as Eq. (3). □

Appendix III
Proof of Theorem IV.2

Proof. First, for the sagittal phase-space, the sagittal velocity is required to be above the asymptote:

$$\dot{x}_{\text{apex},c} \geq \omega \cdot x_{\text{foot},c} \quad (23)$$

Initiating a heading angle change introduces a new local sagittal coordinates as seen in Fig. 6. Therefore Eq. (23) becomes

$$v_{\text{apex},c} \cdot \cos\Delta\theta \geq \omega \cdot \Delta y_{2,c} \cdot \sin\Delta\theta \quad (24)$$

As for the lateral phase-space, the lateral velocity is required to be below the asymptote in the new coordinate as follows

$$v_{\text{apex},c} \cdot \sin \Delta\theta \leq \omega \cdot \Delta y_{2,c} \cdot \cos \Delta\theta \tag{25}$$

Combining Eqs. (24-25) results in the steering safety criterion in Eq. (4). □

Acknowledgment

The authors would like to express our gratefulness to Suda Bharadwaj and Ufuk Topcu for their discussions on belief abstraction, Yinan Li and Jun Liu for their assistance in the reachability controller implementation, and Jialin Li for his early-stage help in setting up our Cassie robot simulation.

References

[1] N. Bohórquez, A. Sherikov, D. Dimitrov, and P.-B. Wieber, "Safe navigation strategies for a biped robot walking in a crowd," in *IEEE-RAS International Conference on Humanoid Robots*, 2016.

[2] A. Pajon and P.-B. Wieber, "Safe 3d bipedal walking through linear mpc with 3d capturability," in *International Conference on Robotics and Automation*. IEEE, 2019, pp. 1404–1409.

[3] M. S. Motahar, S. Veer, and I. Poulakakis, "Composing limit cycles for motion planning of 3d bipedal walkers," in *IEEE 55th conference on decision and control*, 2016, pp. 6368–6374.

[4] J. Luo, Y. Su, L. Ruan, Y. Zhao, D. Kim, L. Sentis, and C. Fu, "Robust bipedal locomotion based on a hierarchical control structure." *Robotica*, vol. 37, no. 10, pp. 1750–1767, 2019.

[5] N. Scianca, P. Ferrari, D. De Simone, L. Lanari, and G. Oriolo, "A behavior-based framework for safe deployment of humanoid robots," *Autonomous Robots*, pp. 1–22, 2021.

[6] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scltl motion planning for mobility-on-demand," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1481–1488.

[7] O. Arslan and D. E. Koditschek, "Sensor-based reactive navigation in unknown convex sphere worlds," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 196–223, 2019.

[8] V. Vasilopoulos, G. Pavlakos, S. L. Bowman, J. D. Caporale, K. Daniilidis, G. J. Pappas, and D. E. Koditschek, "Reactive semantic planning in unexplored semantic environments using deep perceptual feedback," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4455–4462, 2020.

[9] Y. Chen, J. Tumova, A. Ulusoy, and C. Belta, "Temporal logic robot control based on automata learning of environmental dynamics," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 547–565, 2013.

[10] 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*, 2012, pp. 726–733.

[11] 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.

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

[13] S. Bharadwaj, R. Dimitrova, and U. Topcu, "Synthesis of surveillance strategies via belief abstraction," in *IEEE Conference on Decision and Control*, 2018, pp. 4159–4166.

[14] S. Heim and A. Spröwitz, "Beyond basins of attraction: Quantifying robustness of natural dynamics," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 939–952, 2019.

[15] P. Zaytsev, W. Wolfslag, and A. Ruina, "The boundaries of walking stability: Viability and controllability of simple models," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 336–352, 2018.

[16] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *The International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

[17] S. P. Chinchali, S. C. Livingston, M. Chen, and M. Pavone, "Multi-objective optimal control for proactive decision making with temporal logic models," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1490–1512, 2019.

[18] S. Teng, Y. Gong, J. W. Grizzle, and M. Ghaffari, "Toward safety-aware informative motion planning for legged robots," *arXiv preprint arXiv:2103.14252*, 2021.

[19] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[20] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 2182–2189, 2017.

[21] K. V. Alwala and M. Mukadam, "Joint sampling and trajectory optimization over graphs for online motion planning," *arXiv preprint arXiv:2011.07171*, 2020.

[22] 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*. IEEE, 2020.

[23] E. Rimon, "Exact robot navigation using artificial potential functions," Ph.D. dissertation, Yale University, 1990.

[24] A. Robotics, "Cassie simulators." [Online]. Available: http://www.agilityrobotics.com/sims/,2018

[25] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *6th IEEE-RAS international conference on humanoid robots*, 2006, pp. 200–207.

[26] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The international journal of robotics research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[27] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre, *Viability theory: new directions*. Springer Science & Business Media, 2011.

[28] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[29] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE transactions on robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[30] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 2020–2025.

[31] J. A. DeCastro, J. Alonso-Mora, V. Raman, D. Rus, and H. Kress-Gazit, "Collision-free reactive mission and motion planning for multi-robot systems," in *Robotics research*. Springer, 2018, pp. 459–476.

[32] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI communications*, vol. 29, no. 1, pp. 151–162, 2016.

[33] A. Wu and J. P. How, "Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles," *Autonomous robots*, vol. 32, no. 3, pp. 227–242, 2012.

[34] S. Sarid, B. Xu, and H. Kress-Gazit, "Guaranteeing high-level behaviors while exploring partially known maps," 07 2012.

[35] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal motion planning for hybrid systems in partially unknown environments," in *the 16th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC. Association for Computing Machinery, 2013, p. 353–362.

[36] S. C. Livingston, R. M. Murray, and J. W. Burdick, "Backtracking temporal logic synthesis for uncertain environments," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 5163–5170.

[37] S. C. Livingston, P. Prabhakar, A. B. Jose, and R. M. Murray, "Patching task-level robot controllers based on a local μ-calculus formula," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 4588–4595.

[38] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *ACM International Conference on Hybrid Systems: Computation and Control*, 2010.

[39] S. Ragi and E. K. Chong, "Uav path planning in a dynamic environment via partially observable markov decision process," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 4, pp. 2397–2412, 2013.

[40] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 1610–1616.

[41] W. Chung, S. Kim, M. Choi, J. Choi, H. Kim, C.-b. Moon, and J.-B. Song, "Safe navigation of a mobile robot considering visibility of environment," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 3941–3950, 2009.

[42] S. Bouraine, T. Fraichard, and H. Salhi, “Provably safe navigation for mobile robots with limited field-of-views in dynamic environments,” *Autonomous Robots*, vol. 32, no. 3, pp. 267–283, 2012.

[43] Y. Li and J. Liu, “Rocs: A robustly complete control synthesis tool for nonlinear dynamical systems,” in *the 21st International Conference on Hybrid Systems: Computation and Control*, 2018, pp. 130–135.

[44] 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,” *arXiv preprint arXiv:1811.04333*, 2018.

[45] N. Piterman, A. Pnueli, and Y. Sa’ar, “Synthesis of reactive(1) designs,” in *Verification, Model Checking, and Abstract Interpretation*. Springer, 2006, pp. 364–380.

[46] R. Ehlers and V. Raman, “Slugs: Extensible gr (1) synthesis,” in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 333–339.

[47] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit, “Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles,” *Autonomous Robots*, vol. 42, no. 4, pp. 801–824, 2018.

[48] R. Tedrake and the Drake Development Team, “Drake: Model-based design and verification for robotics,” 2019. [Online]. Available: https://drake.mit.edu

[49] S. B. Akers, “Binary decision diagrams,” *IEEE Transactions on computers*, vol. 27, no. 06, pp. 509–516, 1978.

[50] J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schäffer, and G. Hirzinger, “Bipedal walking control based on capture point dynamics,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4420–4427.

[51] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition*, 2nd ed. Society for Industrial and Applied Mathematics, 2010. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898718577

[52] A. Hereid and A. D. Ames, “Frost$*$: Fast robot optimization and simulation toolkit,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 719–726.

[53] A. Wächter and L. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical programming*, vol. 106, pp. 25–57, 03 2006.

[54] W. Zhi, T. Lai, L. Ott, and F. Ramos, “Anticipatory navigation in crowds by probabilistic prediction of pedestrian future movements,” in *IEEE International Conference on Robotics and Automation*, 2021, pp. 8459–8464.

[55] K. Majd, S. Yaghoubi, T. Yamaguchi, B. Hoxha, D. Prokhorov, and G. Fainekos, “Safe navigation in human occupied environments using sampling and control barrier functions,” *arXiv preprint arXiv:2105.01204*, 2021.

[56] L. Drnach and Y. Zhao, “Robust trajectory optimization over uncertain terrain with stochastic complementarity,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1168–1175, 2021.