Physically-Feasible Reactive Synthesis for Terrain-Adaptive Locomotion

Anonymous Author(s)

***Abstract*—We present an integrated planning framework for quadrupedal locomotion over dynamically changing, unforeseen terrains. Existing methods often depend on heuristics for real-time foothold selection-limiting robustness and adaptability-or rely on computationally intensive trajectory optimization across complex terrains and long horizons. In contrast, our approach combines reactive synthesis for generating correct-by-construction symbolic-level controllers with mixed-integer convex programming (MICP) for dynamic and physically feasible footstep planning during each symbolic transition. To reduce the reliance on costly MICP solves and accommodate specifications that may be violated due to physical infeasibility, we adopt a symbolic repair mechanism that selectively generates only the required symbolic transitions. During execution, real-time MICP replanning based on actual terrain data, combined with runtime symbolic repair and delay-aware coordination, enables seamless bridging between offline synthesis and online operation. Through extensive simulation and hardware experiments, we validate the framework's ability to identify missing locomotion skills and respond effectively in safety-critical environments, including scattered stepping stones and rebar scenarios. Code and experimental videos are available at https://synthesis-micp.github.io/.**

***Index Terms*—Legged Robots, Optimization and Optimal Control, Formal Methods in Robotics and Automation**

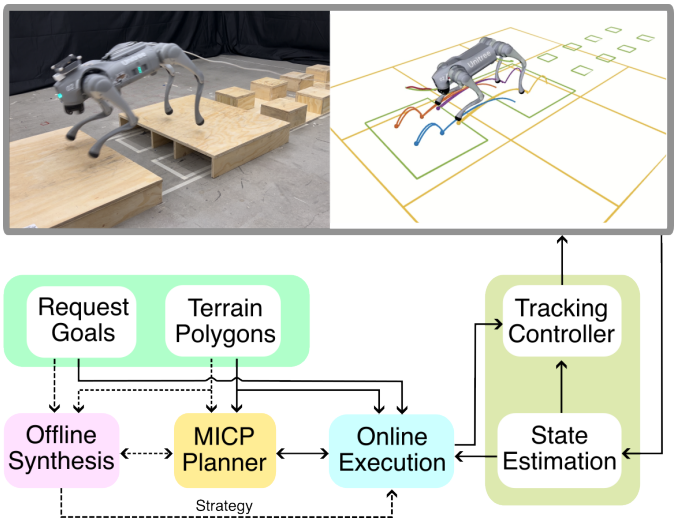


Fig. 1. System architecture overview. The solid lines indicate online communication, while dashed lines represent offline processes. The request goals in the offline phase are user-defined, while those in the online phase are provided by a global planner based on the global goal.

I. Introduction

TERRAIN-adaptive locomotion is essential for enhancing the mobility of legged robots and moving beyond blind walking strategies [1]–[3]. Many existing methods achieve terrain adaptability by adjusting footholds around a nominal trajectory in real time [4], [5]. To further boost traversability, some approaches have explored variable gait patterns. In [6]–[8], jumping motions are precomputed offline and triggered heuristically upon detecting obstacles. Other methods embed gait switching—such as transitions from walking to jumping—within simplified dynamics models during trajectory sampling [9], or rely on pre-trained classifiers to assess feasibility [10]. However, despite the importance of safety in high-risk settings such as disaster zones and construction sites, relatively few works address formal safety guarantees for dynamic locomotion [11], [12] that fully account for the robot's physical limitations.

Mixed-integer programming (MIP)-based methods [13], [14] model both contact state and contact surface selection for each leg and timestep as binary decision variables. This allows them to explicitly capture the interaction between terrain geometry and the robot's kinematic and dynamic constraints [15]. When dynamics and constraints are suitably relaxed, the resulting convex approximation (MICP) provides a global certificate of feasibility upon convergence [16], making it a powerful tool for formally assessing locomotion viability. However, the computational demands of MIP-based approaches scale poorly with increased terrain complexity and longer planning horizons, posing a major hurdle for real-time applications. To mitigate this, various simplifications have been proposed, such as fixing contact sequences [17], [18], constraining timing [19]–[21], or limiting the number of footsteps [22]. Despite these efforts, scalability remains a challenge in cluttered environments with various terrain segments and features.

To simplify navigation across complex terrains, one effective strategy is to decompose long-horizon, global tasks into a sequence of localized, robot-centric environments centered around intermediate waypoints [4]. Despite this simplification, the associated MIP formulation still faces computational challenges, as it must account for all nearby terrain elements. To address this, we discretize the local environment and constrain each MIP to solving only a single discrete transition at a time. These discrete transitions are then composed using a Linear Temporal Logic (LTL)-based reactive synthesis framework [23], which enables us to systematically construct a correct-by-construction strategy that guides the robot through local decisions toward achieving its intermediate goals.

In this work, we present an integrated planning framework that unifies reactive synthesis with MIP-based optimization to enable legged robots to respond safely and efficiently to dynamic and unpredictable terrain conditions. We abstract both the continuous robot states and local environmental context

into symbolic representations, allowing the robot to make high-level decisions in real time for negotiating complex obstacles such as large gaps or cluttered terrain (see Fig. 1). Each symbolic transition is validated through an MICP to ensure the physical feasibility of the locomotion process. This architecture not only connects high-level symbolic reasoning with low-level dynamic feasibility, but also mitigates the computational challenges of solving long-horizon MICP problems.

By leveraging guidance from the LTL-based symbolic planner, each MICP instance only needs to handle short horizons and a limited set of terrain features, which substantially reduces computational load. Another noteworthy feature of our planning framework is its two-stage hierarchy: in the offline synthesis stage, we employ *gait-free* MICP formulations that jointly optimize over contact mode and foothold placement to precompute a diverse set of locomotion skills for symbolic transitions. We leverage a high-level manager and symbolic repair proposed in [24] to minimize the number of calls to the costly *gait-free* MICP, enabling efficient synthesis of terrain-adaptive locomotion strategies. During online deployment, we switch to lightweight *gait-fixed* MICP formulations using these precomputed gaits, enabling real-time generation of physically consistent motions with real-world terrain data. If a symbolic transition is no longer possible or the current terrains and goals were not considered offline, we perform runtime repair to generate new robot skills and synthesize an updated robot strategy, allowing continuous traversal on unexpected terrains. Importantly, the proposed framework remains flexible and modular—it can be driven by commands from a higher-level planner [9], [10], [25]–[28] or directly from a user, and it easily accommodates new behaviors by expanding the symbolic transition set.

Although this manuscript adopts a similar framework to [24], which combines reactive synthesis with MICP, it substantially extends the previous work with a focus on practical deployment. First, we enhance the online execution workflow by addressing inevitable delays in MICP solving through coordinated scheduling between the strategy automaton and the tracking controller. Second, we introduce online retargeting of the robot's desired pose at each transition to ensure feasibility under the actual terrain configuration. Finally, beyond the simulation results reported in [24], we validate the full framework through hardware experiments and benchmark its performance against both a heuristics-based footstep planner and a pure MIP planner. We also include an extension study incorporating yaw orientation to further demonstrate the generalizability of the proposed framework.

The main contributions of this work are summarized as follows:

- We propose a general, full-stack planning and control framework that combines reactive synthesis with MICP for terrain-adaptive locomotion.
- We improve the online execution pipeline by coordinating the reactive synthesis strategy automaton with the tracking controller to handle delays from MICP solving, and by retargeting the robot's desired pose online to remain feasible with respect to real-world terrain.
- We validate the full framework on hardware, and benchmark its performance against both a heuristics-based footstep planner and a pure MIP planner.

II. Related Work

A. Hierarchical Planning for Terrain-Adaptive Locomotion

Planning for terrain-adaptive locomotion can be boiled down into solving contact sequence and timing (equivalently gait), location (equivalently foothold), and robot motion (e.g., Center of Mass (CoM) or base pose), given the chosen kinematics/dynamics model and the perceived terrain information. Hierarchical approaches first separately or simultaneously address part of the above problems and then solve for the rest, which allows for higher computational efficiency. Kinematic footstep planning focuses on the first two problems, neglecting the robot dynamics through graph-based approaches [4], [29]–[33] or optimization-based approaches [15], [20], [34], [35]. Although non-fixed gait/contact plans can be generated for very rough terrains [36]–[38], conservative quasi-static motions are generated for challenging terrains due to the kinematic simplification during footstep planning.

Given the recent advances in optimization-based approaches, notably Model Predictive Control (MPC), another focus of the literature is on the local foothold adaptation and the integration of robot dynamics for dynamic locomotion. Instantaneous foothold adaptation around a nominal location, with a fixed and cyclic gait, is performed based on heuristic search [7], [39], [40] or learning approaches [41]–[46]. In [5], [47], [48], the base pose and foothold are optimized jointly, demonstrating impressive terrain traversing capabilities on rough terrains. More recently, MPC has been employed within hierarchically integrated planning frameworks for legged navigation over rough terrain, incorporating explicit terrain uncertainty quantification [49]–[51].

However, the above dynamic locomotion works consider a fixed gait pattern that usually prohibits achieving versatile behaviors, such as jumping. In [6]–[8], jumping motions are generated offline and triggered through heuristics. In [9], [10], [26], RRT-Connect is used for rapid kino-dynamic locomotion planning, and the switch between normal walking and jumping is either embedded inside a reduced-order model during sampling [9] or decided by a pre-trained feasibility classifier [10]. Beyond limited gait modes, other works focus on online gait planning and/or foothold selection and deploy MPC to track the plan, which can be further categorized into model-free [52]–[56] and model-based methods [57]–[59].

B. Simultaneous Planning via Optimization

Planning for terrain-adaptive locomotion problem can also be formulated as a combinatorial optimization problem that simultaneously optimizes for gait, foothold, and robot dynamics. One approach is to incorporate rigid [60] or smoothed [61], [62] complementarity constraints, which accurately model the physical contact interaction behavior but remains computationally inefficient due to the non-smoothness and highly nonlinearity. Promising online contact-implicit MPC results

are reported in [63]–[65] but still limited to low-dimensional system or static environment.

Another way of encoding discrete contact decisions is to treat both the contact state and the contact plane selection for each leg at each timestep as binary variables [13], [14], [66]. The underlying problem can be formulated as Mixed Integer Program (MIP). Convex approximations of nonlinear dynamics, usually centroidal dynamics, and constraints are typically required to transform the MIP into a more tractable Mixed Integer Convex Program (MICP), albeit with an increased number of binary variables. A global certificate for the approximated convex problem exists upon convergence [16], providing an ideal means to determine the feasibility in our proposed method. To relieve the computational burden due to the introduction of many binary variables, the works of [17], [19], [21] assume the contact sequence and timing are chosen *a prior*, and only optimize the contact plane selection. Aceituno-Cabezas et al. [22] optimize the contact sequence within a certain gait cycle that fixes the number of footsteps. However, it is inevitable that the problem complexity grows exponentially when the number of discrete contact options increases along with the time horizon and terrain features. Our work aims to alleviate computational burden through a two-stage approach. In the offline phase, expensive MICPs that optimize both contact state and foothold selection are solved to generate necessary locomotion gaits. In the online phase, efficient MICPs with adaptive and predefined gaits are used to produce dynamically feasible motions and footholds. Additionally, guided by a synthesis-based task planner, each MICP in our work considers shorter time horizons and fewer terrain features compared to solving a single large MICP problem.

C. Task and Motion Planning for Contact-Rich Planning

Task and Motion Planning (TAMP) formally defines symbolic-level tasks and searches through a graph of predefined motion primitives that enable feasible symbolic transitions [67], [68]. For more complex physical contact resoning, Toussaint et al. propose Logic Geometric Programming (LGP) [69] and embed the high-level logic representation into the low-level motion planner, demonstrating contact-rich tool-use behaviors [70] after defining abundant action primitives. Building on this, a broader range of motions has been showcased, including versatile manipulation [71], [72] and loco-manipulation tasks [73]. In a similar vein, works such as [74]–[81] integrate graph-search or sampling-based methods with optimization-based approaches in a holistic manner, abstracting contact modes within a discrete domain. However, the combinatorial nature of these problems often leads to poor scalability due to the explosion of contact modes. Deploying such approaches online in dynamic environments remains an open challenge. From a different perspective, we abstract a smaller set of task-level contact modes as locomotion gaits over a specific time horizon and distance, allowing the MICP to solve for details such as contact locations and robot motion during execution. This shifts the focus to selecting locomotion gaits within a local environment while ensuring safety and completeness through synthesis-based approaches.

Unlike LGP that solves an integrated TAMP problem, synthesis-based approaches using Linear Temporal Logic (LTL) operate primarily at the task level, emphasizing safety and completeness guarantees within the task domain [82]. These methods typically synthesize a sequence of task-level actions, which are then executed by a motion planner. Recently, LTL has been applied to safe locomotion tasks, employing distinct task-level abstractions on topological maps of terrain [83]–[85] or locomotion keyframes for reduced-order models [11], [12], [86]. Reactive synthesis [87], widely applied to mobile robots [88], has been employed to enable prompt decision-making in response to more complex environments [12] or external perturbations [86]. Notably, Zhao et al. [11] formulate the terrain-adaptive locomotion problem as a sequential decision-making task, selecting appropriate locomotion modes with predefined contact and locomotion keyframes to accommodate dynamically changing terrains. However, the locomotion mode selection is explicitly encoded in the LTL specification using expert knowledge, lacking a physically feasibility guarantee. In this work, we aim to combine the strengths of reactive synthesis and MICP, utilizing the global certificate of MICP as a feasibility checker for terrain-adaptive locomotion.

D. Physically-Feasible Reactive Synthesis

While reactive synthesis can promptly and safely respond to dynamic environments, it typically does not assess physical feasibility when being deployed on complex robotic systems. To bridge the gap between discrete abstraction and continuous system dynamics, various strategies have been proposed. Studies in [89], [90] focus on automatically synthesizing controllers in the continuous domain based on high-level specifications. Another approach involves incorporating dynamics directly into the reactive synthesis process by abstracting physical systems, including nonlinear [91], switched [92], and hybrid systems [93] with manageable model complexity. However, for legged robots with multiple contacts navigating dynamic terrains, these physically feasible reactive synthesis approaches remain computationally intractable.

Recent efforts [94], [95] focus on symbolic repair for unrealizable task specifications, defining symbolic skills with preconditions and postconditions, similar to TAMP, to identify missing skills that make the specification realizable. These works introduce iterative feasibility checks based on symbolic repair suggestions, emphasizing alignment between symbolic task specifications and physical reality. Building upon this, Meng et al. [96] extend this type of approach to online symbolic repair. Inspired by these works, Zhou et al. [24] adopt a similar mechanism for terrain-adaptive locomotion. Specifically, they abstract the robot and terrain states in a local environment and define physically feasible skills by solving MICP. They leverage high-level manager and symbolic repair to efficiently identify missing but necessary skills, reducing the need for frequent calls to the computationally expensive gait-free MICP. Our work significantly completes [24] through seamless integration with a low-level tracking control module, enabling systematic benchmarking, and demonstrating real-world hardware deployment.

III. Preliminaries

Consider a robot equipped with sensors navigating an environment toward a designated global goal $g_{\text{global}} \in \mathbb{R}^2$. This task can be broken into a sequence of local navigation subtasks, where the robot moves toward intermediate local request goals $\mathcal{G}_{\text{local}}$, determined by a global planner based on the nearby segmented terrain polygons $\mathcal{P}$. We illustrate an instance of such a local task setup, following the same example introduced in [24].

Example 1. *Consider a 2D grid world with nine cells (in yellow) in Fig. 2. The robot is required to move from an initial cell (e.g. the center cell) to a desired goal cell indicated by the star. Each cell is assigned a terrain type, such as Dense or Sparse stepping stones.*

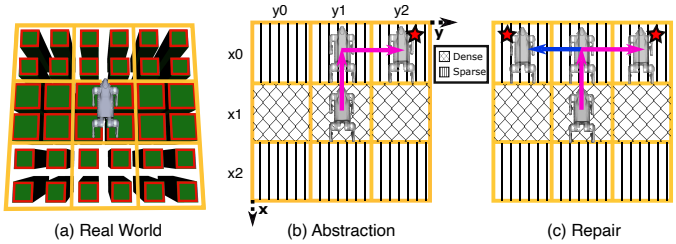


Fig. 2. Terrain abstraction and skill definition. (a) Top-down view of the real-world terrain before abstraction. The red polygons denote the segmented terrain polygons. (b) Abstraction of the terrain and robot's skills (pink) moving from one location to another. (c) Repair process to find a new skill (blue).

A. Abstractions

Given a set of terrain polygons $\mathcal{P}$, such as those in Fig. 2(a), we abstract them into a 2D grid of dimension $n \times m$. For each axis, we label every position with a symbol, producing two sets: $\mathcal{X} \coloneqq \{x_0, \ldots, x_{n-1}\}$ and $\mathcal{Y} \coloneqq \{y_0, \ldots, y_{m-1}\}$.

We introduce a collection of atomic propositions AP, partitioned into input variables $\mathcal{I}$ and output variables $\mathcal{O}$, which capture the symbolic world state and robot actions. The inputs $\mathcal{I}$ include robot position inputs $\mathcal{I}_{\text{robot}}$, request inputs $\mathcal{I}_{\text{req}}$, and terrain inputs $\mathcal{I}_{\text{terrain}}$, with $\mathcal{I} = \mathcal{I}_{\text{robot}} \cup \mathcal{I}_{\text{req}} \cup \mathcal{I}_{\text{terrain}}$. The robot inputs are defined as $\mathcal{I}_{\text{robot}} \coloneqq \{\pi_x \mid x \in \mathcal{X}\} \cup \{\pi_y \mid y \in \mathcal{Y}\}$ to represent the robot's location. Request inputs are given by $\mathcal{I}_{\text{req}} \coloneqq \{\pi_x^{\text{req}} \mid x \in \mathcal{X}\} \cup \{\pi_y^{\text{req}} \mid y \in \mathcal{Y}\}$ to represent the desired goal cell. The terrain inputs are defined as $\mathcal{I}_{\text{terrain}} \coloneqq \{n_x^y \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$, which describe the terrain type associated with each grid cell. We assume a finite set of terrain types of size n_t, defined according to their physical characteristics. For $z \in \mathbb{N}$, we denote $[z] \coloneqq \{0, \ldots, z-1\}$. Accordingly, each terrain input $n_x^y \in \mathcal{I}_{\text{terrain}}$ is an integer variable taking values from $[n_t]$, which we further translate into Boolean propositions following the approach of [97]. The robot state abstraction can also be extended to incorporate orientation, such as the yaw angle of the floating base (see Sec. IX-F).

An input state $\sigma_{\mathcal{I}}$ is a mapping $\sigma_{\mathcal{I}} : \mathcal{I} \to \{\texttt{True}, \texttt{False}\} \cup [n_t]$. Because the robot and its goal occupy only a single grid cell each, exactly one π_x, one π_y, one π_x^{req}, and one π_y^{req} must evaluate to `True` for any valid input state. We denote projections of $\sigma_{\mathcal{I}}$ as robot states $\sigma_{\text{robot}} : \mathcal{I}_{\text{robot}} \to \{\texttt{True}, \texttt{False}\}$, request states $\sigma_{\text{req}} : \mathcal{I}_{\text{req}} \to \{\texttt{True}, \texttt{False}\}$, and terrain states $\sigma_{\text{terrain}} : \mathcal{I}_{\text{terrain}} \to [n_t]$. The complete sets of such states are denoted by $\Sigma_{\mathcal{I}}$, Σ_{robot}, Σ_{req}, and Σ_{terrain}, respectively.

We introduce a grounding function to associate symbolic input states with their physical interpretations. Let the physical state space be $\mathcal{Z} \subseteq \mathbb{R}^n$, which encodes the physical world, including the robot's position, orientation, and terrain attributes. The grounding function $\mathrm{G} : \Sigma_{\mathcal{I}} \to 2^{\mathcal{Z}}$ maps each input state $\sigma_{\mathcal{I}}$ to the corresponding set of physical states in $\mathcal{Z}$.

For robot states $\sigma_{\text{robot}} \in \Sigma_{\text{robot}}$, we define $\mathrm{G}(\sigma_{\text{robot}}) \coloneqq \{z \in \mathcal{Z} \mid \exists \pi_x, \pi_y \in \mathcal{I}_{\text{robot}}.\sigma_{\text{robot}}(\pi_x) \wedge \sigma_{\text{robot}}(\pi_y) \wedge \text{robot\_pose}(z) \in \text{cell}(x, y)\}$. Similarly, for request states $\sigma_{\text{req}} \in \Sigma_{\text{req}}$, we set $\mathrm{G}(\sigma_{\text{req}}) \coloneqq \{z \in \mathcal{Z} \mid \exists \pi_x^{\text{req}}, \pi_y^{\text{req}} \in \mathcal{I}_{\text{req}}.\sigma_{\text{req}}(\pi_x^{\text{req}}) \wedge \sigma_{\text{req}}(\pi_y^{\text{req}}) \wedge \text{request\_goal}(z) \in \text{cell}(x, y)\}$. For terrain states $\sigma_{\text{terrain}} \in \Sigma_{\text{terrain}}$, the grounding $\mathrm{G}(\sigma_{\text{terrain}})$ is defined as the set of physical states whose terrain abstractions are consistent with σ_{terrain}. Given an input state $\sigma_{\mathcal{I}} \in \Sigma_{\mathcal{I}}$ with projections σ_{robot}, σ_{req}, and σ_{terrain}, we combine them as $\mathrm{G}(\sigma_{\mathcal{I}}) \coloneqq \mathrm{G}(\sigma_{\text{robot}}) \cap \mathrm{G}(\sigma_{\text{req}}) \cap \mathrm{G}(\sigma_{\text{terrain}})$. Finally, we introduce an inverse grounding function $\mathrm{G}^{-1} : \mathcal{Z} \to \Sigma_{\mathcal{I}}$ that maps each physical state $z \in \mathcal{Z}$ back to its corresponding symbolic input state.

In Example 1, the inputs are defined as $\mathcal{I}_{\text{robot}} \coloneqq \{\pi_{x_0}, \pi_{x_1}, \pi_{x_2}, \pi_{y_0}, \pi_{y_1}, \pi_{y_2}\}$, $\mathcal{I}_{\text{req}} \coloneqq \{\pi^{\text{req}} \mid \pi \in \mathcal{I}_{\text{robot}}\}$, and $\mathcal{I}_{\text{terrain}} \coloneqq \{n_{x_i}^{y_j} \mid i, j \in 0, 1, 2\}$, where $n_{x_i}^{y_j} = 1$ if the grid cell (x_i, y_j) is classified as Dense, and $n_{x_i}^{y_j} = 0$ if it is Sparse. The input state shown in Fig. 2(b) is $\sigma_{\mathcal{I}} : \pi_{x_1} \mapsto \texttt{True}, \pi_{y_1} \mapsto \texttt{True}, \pi_{x_0}^{\text{req}} \mapsto \texttt{True}, \pi_{y_2}^{\text{req}} \mapsto \texttt{True}, n_{x_1}^{y_j} \mapsto 1$ for $j \in \{0, 1, 2\}$. For brevity, throughout this paper we omit Boolean propositions set to `False` and integer propositions with value 0 when describing input states.

The output propositions $\mathcal{O}$ represent skills that enable the robot to move between grid cells under specific terrain conditions. A skill $o \in \mathcal{O}$ is defined by its preconditions $\Sigma_o^{\text{pre}} \subseteq \Sigma_{\text{robot}} \times \Sigma_{\text{terrain}}$, which specify the robot and terrain states from which it can be executed, and its postconditions $\Sigma_o^{\text{post}} \subseteq \Sigma_{\text{robot}}$, which describe the resulting robot state after execution. In Example 1, the skill o_0 moves the robot from the middle cell to the upper cell (Fig. 2b). The precondition of o_0 is $\Sigma_{o_0}^{\text{pre}} = \{(\sigma_{\text{robot}}, \sigma_{\text{terrain}}) : \pi_{x_1} \mapsto \texttt{True}, \pi_{y_1} \mapsto \texttt{True}, n_{x_1}^{y_0} \mapsto 1, n_{x_1}^{y_1} \mapsto 1, n_{x_1}^{y_2} \mapsto 1\}$, while the postcondition is $\Sigma_{o_0}^{\text{post}} = \{\sigma_{\text{robot}} : \pi_{x_0} \mapsto \texttt{True}, \pi_{y_1} \mapsto \texttt{True}\}$. Each symbolic skill is also paired with a continuous-level locomotion gait, such as a one-second trotting gait L, which physically realizes the symbolic transition (Sec. VI-A).

B. Specifications

We adopt the Generalized Reactivity (1) (GR(1)) fragment of linear temporal logic (LTL) [23] to encode task specifications. An LTL formula φ follows the grammar $\varphi \coloneqq \pi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \square\varphi \mid \lozenge\varphi$, where $\pi \in AP$ denotes an atomic proposition, $\neg$ and $\wedge$ are Boolean operators, and $\bigcirc$ ("next"), $\square$ ("always"), and $\lozenge$ ("eventually") are temporal operators. For a comprehensive introduction to LTL, we refer the reader to [98].

A GR(1) specification is written in the form $\varphi = \varphi_{\text{e}} \to \varphi_{\text{s}}$, where $\varphi_{\text{e}} = \varphi_{\text{e}}^{\text{i}} \wedge \varphi_{\text{e}}^{\text{t}} \wedge \varphi_{\text{e}}^{\text{g}}$ encodes the assumptions about the (possibly adversarial) environment, and $\varphi_{\text{s}} = \varphi_{\text{s}}^{\text{i}} \wedge \varphi_{\text{s}}^{\text{t}} \wedge \varphi_{\text{s}}^{\text{g}}$ represents the guarantees that define the system's behavior.

For $\alpha \in \{e, s\}$, the components $\varphi^{\mathrm{i}}_{\alpha}, \varphi^{\mathrm{t}}_{\alpha}, \varphi^{\mathrm{g}}_{\alpha}$ describe the initial conditions, safety requirements, and liveness objectives, respectively. In the context of symbolic repair (Sec. VI-C), we further partition the safety constraints $(\varphi^{\mathrm{t}}_{\mathrm{e}}, \varphi^{\mathrm{t}}_{\mathrm{s}})$ into two parts: skill-related constraints $(\varphi^{\mathrm{t,skill}}_{\mathrm{e}}, \varphi^{\mathrm{t,skill}}_{\mathrm{s}})$ and hard constraints $(\varphi^{\mathrm{t,hard}}_{\mathrm{e}}, \varphi^{\mathrm{t,hard}}_{\mathrm{s}})$. The skill constraints encode the preconditions and postconditions of skills (see Sec. VI-B) and are subject to modification by the repair procedure, whereas the hard constraints remain immutable.

In practice, GR(1) specifications often become quite large because they must capture detailed information about both terrain and task constraints. At runtime, however, the exact terrain and request states are already available. This allows us to apply a technique known as partial evaluation originally defined in [24], where known Boolean propositions are directly substituted with their truth values. By doing so, we reduce the overall state space, leading to a smaller and more computationally efficient specification for synthesis.

Definition 1. Given an LTL formula φ and two subsets $S^{\texttt{True}}, S^{\texttt{False}} \subseteq AP$, $S^{\texttt{True}} \cap S^{\texttt{False}} = \emptyset$, we define the *partial evaluation* of φ over $S^{\texttt{True}}, S^{\texttt{False}}$, as $\varphi[S^{\texttt{True}}, S^{\texttt{False}}]$, where we substitute propositions $\pi \in AP$ in φ with `True` if $\pi \in S^{\texttt{True}}$ and with `False` if $\pi \in S^{\texttt{False}}$.

IV. Problem Statement

Problem 1. Given (i) a global goal g_{global}, (ii) a set of possible local request goals $\mathcal{G}_{\text{local}}$ (iii) a set of predefined terrain polygons $\mathcal{P}_{\text{pre}}$ from user's prior knowledge, and (iv) a set of online terrain polygons $\mathcal{P}_{\text{on}}$ perceived during execution that may differ from the predefined ones, (v) a set of predefined locomotion gaits $\mathcal{L}$; generate controls that enable the robot to navigate the environment and reach the goal.

V. Approach Summary

To address Problem 1 efficiently, we manage complexity on both symbolic and physical levels. At the symbolic level, we employ reactive synthesis to break down the local navigation task into smaller subproblems, which can be reused across different scenarios. Each subproblem involves planning a short-horizon symbolic transition and is accompanied by solving a MICP to certify its physical feasibility.

As shown in Fig. 1, our overall framework is composed of three main modules: offline synthesis (Sec.VI), online execution (Sec.VII), and low-level tracking control (Sec.VIII). During offline synthesis, we generate a diverse set of locomotion skills and their associated gaits for various predefined terrain-task combinations. Symbolic repair is applied to discover additional necessary transitions that may not be captured initially.

At runtime, we invoke a symbolic repair mechanism to synthesize new transitions when the robot encounters unexpected terrain conditions or request goals not seen during offline planning. This allows the framework to remain flexible and adaptive to dynamically evolving environments. Additionally, the system re-targets the desired pose during each transition to reflect current terrain constraints and coordinates between the planner and tracking controller to handle potential delays in MICP solving without compromising execution continuity.

VI. Offline Synthesis

The offline synthesis module builds a strategy that allows the robot to reach local request goals under predefined terrain conditions. As shown in Fig. 3(a), the module takes in local request goals $\mathcal{G}_{\text{local}}$, terrain polygons $\mathcal{P}_{\text{pre}}$ from prior maps, and a set of locomotion gaits $\mathcal{L}$ specified by the user. We first apply the inverse grounding function G^{-1} to discretize the local environment, deriving a set of candidate request states $\Sigma^{\text{poss}}_{\text{req}} \subseteq \Sigma_{\text{req}}$ from the local request goals and mapping the predefined terrain polygons into possible terrain states $\Sigma^{\text{poss}}_{\text{terrain}} \subseteq \Sigma_{\text{terrain}}$. Afterward, we evaluate the feasibility of each candidate skill using a gait-fixed MICP formulated with the provided locomotion gaits (Sec. VI-A), and record all feasible transitions as symbolic skills in the specification (Sec. VI-B). To improve efficiency, a high-level manager then generates partial evaluations of the encoded specifications. Whenever one of these partial evaluations is deemed unrealizable, a repair mechanism is invoked: it proposes additional symbolic skills, whose feasibility is validated through gait-free MICP. This process ensures that the specification ultimately becomes realizable (Sec. VI-C).

A. Locomotion Gait and Feasibility Checking via MICP

We assume each locomotion gait L is characterized by a contact sequence $\mathcal{G}$ and its associated time durations T, formally written as $L = \mathcal{M}(\mathcal{G}, T)$. Each skill is tied to a distinct locomotion gait, which dictates how the robot executes motion at the continuous level. As illustrated in Fig. 3(a), since the offline phase has access to the complete set of request states $\Sigma^{\text{poss}}_{\text{req}}$ and terrain states $\Sigma^{\text{poss}}_{\text{terrain}}$, we can systematically enumerate all possible symbolic skills that enable navigation across the different local environments. The feasibility of each skill is then evaluated through gait-fixed MICP using the given locomotion gaits. In most examples shown in this paper, we assume the robot is only allowed to move horizontally and vertically by one cell and we show diagonal movements and heading angle change in Sec. IX-F.

Then the next critical step is to find whether there exists a locomotion gait for each skill to be physically feasible. We leverage mixed-integer convex programming (MICP) to check the physical feasibility given a set of predefined locomotion gaits, and only use the feasible one as robot skills to synthesize robot strategies. The MICP problem takes in the centroidal states $\mathcal{I}_{\text{robot}}$ from the precondition and postcondition of the skill as its initial and final conditions, which are located at the center of each abstracted cell. A set of homogeneous, predefined terrain polygons are considered in the MICP as steppable regions corresponding to the terrain states $\mathcal{I}_{\text{terrain}}$ involved in the precondition. Lastly, the contact information $\mathcal{G}$ and T is provided by the selected locomotion gait L. Fig. 4 shows the maneuver of the quadruped robot Go2 after solving the MICP problem corresponding to skill a_0 in Example 1 with a one-second trotting locomotion gait. The generic MICP

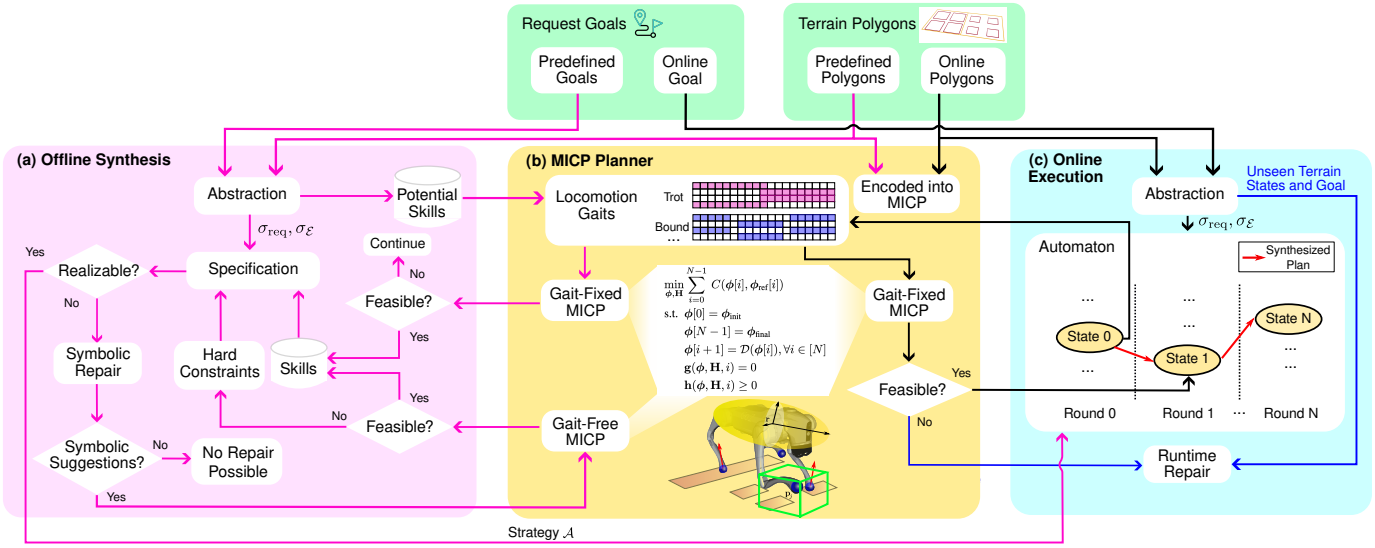


Fig. 3. System overview. During offline synthesis (pink arrows), an initial set of locomotion gaits is provided, and symbolic skills are iteratively generated by solving the MICP. When the task specifications are unrealizable, a symbolic repair is triggered to seek missing skills. During the online execution (black arrows), MICP is solved again taking online terrain segments and the symbolic state only advances when a solution is found. A runtime repair (blue arrows) is initiated if a solving failure occurs, or an unseen terrain or request goal is encountered.

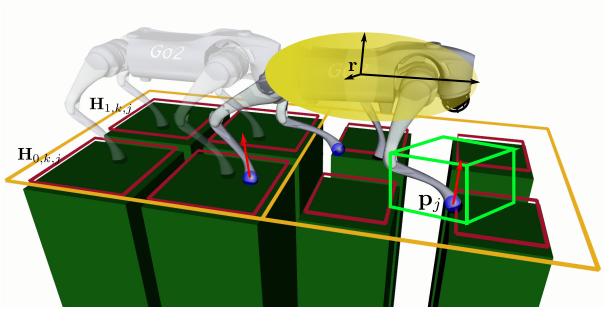


Fig. 4. Demonstration of decision variables and polygons when solving MICP for skill a_0 in Example 1 using a trotting locomotion gait.

problem considered in this work can formulated as:

$$\min_{\boldsymbol{\phi},\mathbf{H}} \sum_{i=0}^{N-1} C(\boldsymbol{\phi}[i], \boldsymbol{\phi}_{\text{ref}}[i])$$

$$\text{s.t. } \boldsymbol{\phi}[0] = \boldsymbol{\phi}_{\text{init}} \tag{1a}$$
$$\boldsymbol{\phi}[N-1] = \boldsymbol{\phi}_{\text{final}} \tag{1b}$$
$$\boldsymbol{\phi}[i+1] = \mathcal{D}(\boldsymbol{\phi}[i]), \forall i \in [N] \tag{1c}$$
$$\mathbf{g}(\boldsymbol{\phi}, \mathbf{H}, i) = 0 \tag{1d}$$
$$\mathbf{h}(\boldsymbol{\phi}, \mathbf{H}, i) \geq 0 \tag{1e}$$

where the decision variables $\boldsymbol{\phi}$ denote the continuous variables and $\mathbf{H}$ indicate the binary variables across the entire N timesteps. For any natural number $n \in \mathbb{N}$, we denote the set $\{0, \ldots, n-1\}$ as $[n]$. The general equality constraints $\mathbf{g}(\cdot)$ and inequality constraints $\mathbf{h}(\cdot)$ are incorporated and activated at certain time stamp i. The cost function depends on both the continuous variables $\boldsymbol{\phi}$ and a reference trajectory $\boldsymbol{\phi}_{\text{ref}}$.

The reference base position and angular trajectories are generated by linearly interpolating between the initial and final conditions. For scenarios with significant elevation changes, such as jumping onto higher terrain, an additional middle keyframe is introduced for the pitch angle to calculate the slope angle between the initial and final poses. The final reference pitch trajectory is then created by evenly interpolating between the initial, middle, and final poses. The reference EE trajectory relative to the base frame ${}^{\mathcal{B}}\mathbf{p}_j^{\text{ref}}$ remains constant and moves in sync with the reference base trajectory.

Given a specific locomotion gait, we first introduce the gait-fixed MICP formulation:

1) Continuous Decision Variables: The continuous decision variables $\boldsymbol{\phi}$ include base position $\mathbf{r}$, velocity $\dot{\mathbf{r}}$, acceleration $\ddot{\mathbf{r}}$, orientation $\boldsymbol{\theta}$ parameterized by Euler angle and its first and second-order derivatives $\dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}$, individual end-effector (EE) position $\mathbf{p}_j$, velocity $\dot{\mathbf{p}}_j$, and acceleration $\ddot{\mathbf{p}}_j$, and individual contact force $\mathbf{f}_j$ for the foot j with $j \in [n_f]$, where $n_f = 4$ denotes the index of feet. The compact form can be expressed as:

$$\boldsymbol{\phi} = [\mathbf{r}^\top, \dot{\mathbf{r}}^\top, \ddot{\mathbf{r}}^\top, \boldsymbol{\theta}^\top, \dot{\boldsymbol{\theta}}^\top, \ddot{\boldsymbol{\theta}}^\top, \mathbf{p}_j^\top, \dot{\mathbf{p}}_j^\top, \ddot{\mathbf{p}}_j^\top, \mathbf{f}_j^\top]^\top \tag{2}$$

2) System Dynamics: We encode the system dynamics as a simplified single rigid body in Eq. (3), using a double integrator to replace the original Euler equation to keep the dynamics constraint convex. The whole system is discretized through Backward Euler integration with Δt between each time step.

$$\begin{bmatrix} \mathbf{r}[i+1] \\ \dot{\mathbf{r}}[i+1] \\ m\ddot{\mathbf{r}}[i] \\ \boldsymbol{\theta}[i+1] \\ \dot{\boldsymbol{\theta}}[i+1] \end{bmatrix} = \begin{bmatrix} \mathbf{r}[i] + \Delta t \cdot \dot{\mathbf{r}}[i+1] \\ \dot{\mathbf{r}}[i] + \Delta t \cdot \ddot{\mathbf{r}}[i+1] \\ \sum_j \mathbf{f}_j[i] + m\mathbf{g} \\ \boldsymbol{\theta}[i] + \Delta t \cdot \dot{\boldsymbol{\theta}}[i+1] \\ \dot{\boldsymbol{\theta}}[i] + \Delta t \cdot \ddot{\boldsymbol{\theta}}[i+1] \end{bmatrix} \tag{3}$$

where m is the robot mass and g is the gravity term.

3) Cost Function: The cost function consists of tracking costs and regularization terms governed by diagonal matrices $\mathbf{Q}$ and $\mathbf{R}$.

$$\mathbf{C} = \delta\boldsymbol{\phi}_Q[i]^T\mathbf{Q}\,\delta\boldsymbol{\phi}_Q[i] + \boldsymbol{\phi}_R[i]^T\mathbf{R}\boldsymbol{\phi}_R[i] \tag{4}$$

where $\delta\boldsymbol{\phi}_Q$ and $\boldsymbol{\phi}_R$ are defined as:

$$\delta\boldsymbol{\phi}_Q = \begin{bmatrix} \mathbf{r} - \mathbf{r}_{\text{ref}} \\ \boldsymbol{\theta} - \boldsymbol{\theta}_{\text{ref}} \\ \mathbf{p}_j - \mathbf{p}_j^{\text{ref}} \end{bmatrix}, \boldsymbol{\phi}_R = \begin{bmatrix} \ddot{\mathbf{r}} \\ \ddot{\boldsymbol{\theta}} \\ \ddot{\mathbf{p}}_j \\ \mathbf{f}_j \end{bmatrix} \tag{5}$$

Tracking costs include deviation from the desired base and foot EE trajectories. The regularization term includes minimizing the base acceleration, Euler angle acceleration, EE acceleration, and contact forces to encourage motion smoothness. The weights for the tracking terms are defined in Table I.

4) Safe Region Constraint: To incorporate safe region constraints for selecting proper footholds, we introduce binary variables $\mathbf{H}_{r,k,j}$, with r, k, and j expressing the r^{th} convex region, k^{th} footstep, and j^{th} foot and $r \in [R], k \in [n_s]$. One footstep is defined as a full swing phase for a foot. We use n_s and R to represent the number of footsteps specified by the gait configuration and the number of convex terrain polygons to be considered. For example, Fig. 4 shows a case with eight polygons. Eqs. (6) - (7) restrict the robot's EE to stay within one of the convex polygons when the corresponding binary variable $\mathbf{H}_{r,k,j}$ is True. Each polygon is parameterized by an inequality constraint ($\mathbf{A}_r$ and $\mathbf{b}_r$) that defines multiple half-spaces, along with an equality constraint ($\mathbf{A}_{\text{eq},r}$ and $\mathbf{b}_{\text{eq},r}$) that ensures the foothold position lies on a 3D plane. We use $\mathcal{C}_{k,j}$ to denote the set of time steps indicating stance after the k^{th} footstep for the j^{th} foot and the safe region constraint only applies to the first stance time step represented as $\mathcal{C}_{k,j}[0]$. Note that, during the offline phase, the terrain polygons are homogeneous and predefined.

$$\forall r \in [R], k \in [n_s], j \in [n_f]$$
$$\mathbf{H}_{r,k,j} \Rightarrow \mathbf{A}_r\mathbf{p}_j[i] \leq \mathbf{b}_r, \forall i \in \mathcal{C}_{k,j}[0] \tag{6}$$
$$\mathbf{A}_{\text{eq},r}\mathbf{p}_j[i] = \mathbf{b}_{\text{eq},r}, \forall i \in \mathcal{C}_{k,j}[0] \tag{7}$$
$$\sum_{r=0}^{R-1} \mathbf{H}_{r,k,j} = 1 \tag{8}$$
$$\mathbf{H}_{r,k,j} \in \{0, 1\} \tag{9}$$

5) Frictional and Contact Constraints: Frictional constraints are defined in Eqs. (10) - (11), with $\mathbf{n}_r$ and $\mathcal{F}_r$ as the normal vector and friction cone of the r^{th} convex region corresponding to the x and y dimensions of the EE position $\mathbf{p}_j^{xy}$. Contact constraints in Eqs. (12) - (13) forces the EE velocity to be zero during stance phase and the contact force to be zero during the non-contact phase. $\mathcal{C}_j$ represents the set of all time steps indicating stance for the j^{th} foot.

TABLE I
TRACKING COST WEIGHTS

Cost Term	Weights
$\mathbf{r} - \mathbf{r}_{\text{ref}}$	(1000.0, 1000.0, 1000.0)
$\boldsymbol{\theta} - \boldsymbol{\theta}_{\text{ref}}$	(1000.0, 1000.0, 1000.0)
$\mathbf{p}_j - \mathbf{p}_j^{\text{ref}}$	(1000.0, 1000.0, 1000.0)
$\ddot{\mathbf{r}}$	(10.0, 10.0, 10.0)
$\ddot{\boldsymbol{\theta}}$	(10.0, 10.0, 10.0)
$\ddot{\mathbf{p}}_j$	(0.5, 0.5, 0.5)
$\mathbf{f}_j$	(0.1, 0.1, 0.1)

$$\forall r \in [R], k \in [n_s], j \in [n_f]$$
$$\mathbf{H}_{r,k,j} \Rightarrow \mathbf{f}_j[i] \cdot \mathbf{n}_r(\mathbf{p}_j^{xy}[i]) \geq 0,\ \forall i \in \mathcal{C}_{k,j} \tag{10}$$
$$\mathbf{f}_j[i] \in \mathcal{F}_r(\mu, \mathbf{n}_r, \mathbf{p}_j^{xy}[i]),\ \forall i \in \mathcal{C}_{k,j} \tag{11}$$
$$\dot{\mathbf{p}}_j[i] = 0,\ \forall i \in \mathcal{C}_j \tag{12}$$
$$\mathbf{f}_j[i] = 0,\ \forall i \notin \mathcal{C}_j \tag{13}$$

where μ denotes the friction coefficient.

6) Actuation Constraint: Since the joint angles are omitted in the single rigid body model, we use a fixed Jacobian $\mathbf{J}_j(\mathbf{q}_j^{\text{ref}})$ at a nominal joint pose $\mathbf{q}_j^{\text{ref}}$ for each leg to approximately consider the torque limit constraint in Eq. (14). In addition, since the translational and angular motions are decoupled, another actuation constraint on the angular acceleration is also added in Eq. (15).

$$\forall i \in [N], j \in [n_f]$$
$$\mathbf{J}_j(\mathbf{q}_j^{\text{ref}})^\top\mathbf{f}_j[i] \leq \boldsymbol{\tau}_{\max} \tag{14}$$
$$\mathbf{I}\ddot{\boldsymbol{\theta}}[i] \leq \boldsymbol{\tau}'_{\max} \tag{15}$$

where $\boldsymbol{\tau}_{\max}$ is the joint torque limit, $\boldsymbol{\tau}'_{\max}$ is the torque limit applied on the base, and $\mathbf{I}$ is the moment of inertia for the approximated single rigid body.

7) Kinematics Constraint: Lastly, the kinematics constraint in Eq. (16) strictly limits the possible EE movements to assure safety. Due to the convex nature of this MICP, we can determine the feasibility of a possible symbolic transition by checking if the optimal solution exists.

$$\mathbf{p}_j[i] \in \mathcal{R}_j({}^{\mathcal{B}}\mathbf{p}_j^{\text{ref}}, \mathbf{r}[i], \boldsymbol{\theta}_{\text{ref}}, \mathbf{p}_j^{\max}), \forall i \in [N], j \in [n_f] \tag{16}$$

where $\mathcal{R}_j$ is defined as a 3D box constraint around a nominal foot EE position based on the base position and orientation, and constrained by a maximum deviation $\mathbf{p}_j^{\max}$. Note that the reference orientation $\boldsymbol{\theta}_{\text{ref}}$ is used to avoid nonlinear constraint keep the problem convex.

B. Task Specification Encoding

After identifying a set of feasible robot skills, we encode them into the specification φ as part of the environment safety assumptions $\varphi_{\text{e}}^{\text{t}}$ and the system safety guarantees $\varphi_{\text{s}}^{\text{t}}$.

The environment-side skill assumptions $\varphi_{\text{e}}^{\text{t,skill}}$ capture the postconditions of each skill. These assumptions enforce that

whenever a skill's precondition holds and the skill is executed, at least one of its associated postconditions must also hold:

$$\varphi_{\text{e}}^{\text{t,skill}} \coloneqq \bigwedge_{o\in\mathcal{O}} \square((o \wedge \bigvee_{\sigma\in\Sigma_o^{\text{pre}}} \bigwedge_{\pi\in\mathcal{I}_{\text{robot}}\cup\mathcal{I}_{\text{terrain}}} \pi = \sigma(\pi)) \rightarrow \bigvee_{\sigma\in\Sigma_o^{\text{post}}} \bigwedge_{\pi\in\mathcal{I}_{\text{robot}}} \bigcirc(\pi = \sigma(\pi))) \tag{17}$$

In Example 1, the postcondition of skill o_0 is defined as

$$\square(o_0 \wedge \pi_{x_1} \wedge \pi_{y_1} \wedge n_{x_1}^{y_0} = 1 \wedge n_{x_1}^{y_1} = 1 \wedge n_{x_1}^{y_2} = 1 \wedge \ldots \rightarrow \bigcirc\pi_{x_0} \wedge \bigcirc\pi_{y_1} \wedge \neg\ldots) \tag{18}$$

The system-side skill guarantees $\varphi_{\text{s}}^{\text{t,skill}}$ encode the preconditions associated with each skill. These guarantees restrict when a skill may be executed by the robot. Specifically, a skill is permitted to run only if at least one of its preconditions is satisfied:

$$\varphi_{\text{s}}^{\text{t,skill}} \coloneqq \bigwedge_{o\in\mathcal{O}} \square(\neg(\bigvee_{\sigma\in\Sigma_o^{\text{pre}}} \bigwedge_{\pi\in\mathcal{I}_{\text{robot}}\cup\mathcal{I}_{\text{terrain}}} \bigcirc(\pi = \sigma(\pi))) \rightarrow \neg \bigcirc o) \tag{19}$$

In Example 1, the precondition of skill o_0 is defined as

$$\square\big(\neg(\bigcirc\pi_{x_1} \wedge \bigcirc\pi_{y_1} \wedge \bigcirc n_{x_1}^{y_0} = 1 \wedge \bigcirc n_{x_1}^{y_1} = 1 \wedge \bigcirc n_{x_1}^{y_2} = 1 \wedge \ldots) \rightarrow \neg \bigcirc o_0\big) \tag{20}$$

The hard constraints $\varphi_{\text{e}}^{\text{t,hard}}$ and $\varphi_{\text{s}}^{\text{t,hard}}$ are constraints that a repair cannot modify (see Sec. VI-C). Our system's hard constraints $\varphi_{\text{s}}^{\text{t,hard}}$ only allow the robot to execute one skill at a time: $\square\big(\neg(\bigcirc o \wedge \bigcirc o')\big)$, for any two different skills $o, o' \in \mathcal{O}$. Given that, we encode the following hard assumptions $\varphi_{\text{e}}^{\text{t,hard}}$:

(i) the uncontrollable terrain and request inputs cannot change during execution: $\square(\pi \leftrightarrow \bigcirc\pi)$, $\forall\pi \in \mathcal{I}_{\text{terrain}} \cup \mathcal{I}_{\text{req}}$, and

(ii) the robot inputs $\mathcal{I}_{\text{robot}}$ remain unchanged if no skill is executed: $\square\big(\bigwedge_{o\in\mathcal{O}} \neg o \rightarrow \bigwedge_{\pi\in\mathcal{I}_{\text{robot}}} (\pi \leftrightarrow \bigcirc\pi)\big)$.

C. High-level Manager and Symbolic Repair

Similar to [24], we design a high-level manager to efficiently synthesize controllers for encoded specifications under given sets of terrain and request states. The manager takes as input the specification φ, a predefined collection of terrain states $\Sigma_{\text{terrain}}^{\text{poss}} \subseteq \Sigma_{\text{terrain}}$, and a predefined set of request states $\Sigma_{\text{req}}^{\text{poss}} \subseteq \Sigma_{\text{req}}$. For each pair $(\sigma_{\text{terrain}}, \sigma_{\text{req}}) \in \Sigma_{\text{terrain}}^{\text{poss}} \times \Sigma_{\text{req}}^{\text{poss}}$, the manager first converts the integer-valued terrain state $\sigma_{\text{terrain}} : \mathcal{I}_{\text{terrain}} \rightarrow [n_t]$ into a Boolean representation $\sigma_{\text{terrain}}^{\mathcal{B}} : \mathcal{I}_{\text{terrain}}^{\mathcal{B}} \rightarrow \{\texttt{True}, \texttt{False}\}$, following the approach in [97]. We then overload $\sigma_{\text{terrain}}^{\mathcal{B}}$ and σ_{req} to denote the propositions that evaluate to `True`. Using this, the manager generates a partial evaluation $\varphi' \coloneqq \varphi[\sigma_{\text{terrain}}^{\mathcal{B}} \cup \sigma_{\text{req}}, \mathcal{I}_{\text{terrain}}^{\mathcal{B}} \cup \mathcal{I}_{\text{req}} \setminus \sigma_{\text{terrain}}^{\mathcal{B}} \cup \sigma_{\text{req}}]$ (see Definition 1). Skills whose preconditions are evaluated to `False` under the partial evaluation are discarded, which reduces the specification to robot inputs $\mathcal{I}_{\text{robot}}$ and a smaller skill set as outputs. This step prevents exponential growth of the synthesis procedure in the variable size. Finally, the manager synthesizes a strategy $\mathcal{A}_{\varphi'}$ for each reduced specification.

If φ' is unrealizable, this indicates that gait-fixed MICP feasibility checks (Sec. VI-A) failed to produce sufficient skills to reach the request under the given terrain. To address this, we rely on gait-free MICP (Sec. VI-D), which relaxes the fixed contact sequence and timing constraints while retaining the continuous dynamics. Since gait-free MICP is computationally demanding, we employ symbolic repair [95] to generate targeted symbolic suggestions that minimize unnecessary solves. Symbolic repair works by systematically modifying pre- and postconditions of existing skills to propose new candidates. These suggestions are then validated by gait-free MICP; if feasible, they are incorporated into the specification, making φ' realizable. An illustration of this process is shown in Fig. 2(c), where repair alters the postcondition of a skill from reaching (x_2, y_0) to instead target (x_0, y_0), thereby enabling the robot to satisfy the request.

D. Gait-Free MICP

We aim to implement the suggested skills by reconfiguring the contact sequence and timing to generate new locomotion gaits. This can be achieved by solving a gait-free version of MICP in Eq. (1). The gait-free MICP retains all continuous decision variables and constraints from the gait-fixed MICP but modifies the contact state-dependent constraints, as the contact sequence and timing are no longer fixed. Instead of using $\mathbf{H}_{r,k,j}$, we introduce a different set of binary variables $\mathbf{H}_{r,m,j}$ for defining the contact state for each foot at each time step explicitly. r and j still represent the r^{th} convex region and j^{th} foot, respectively, while $m \in [M]$ denotes the m^{th} time step, evenly discretized with Δt_m over the entire M time steps. We use $\mathcal{O}_{m,m+1}$ to denote the set of continuous time steps that span from the m^{th} to the $m+1^{\text{th}}$ binary time step. As a result, the following constraints are modified:

1) Safe Region Constraint: The safe region constraint is defined in Eq. (21) - (22) similar to the gait-fixed case in Eq. (6) - (7) when a binary variable $\mathbf{H}_{r,m,j}$ is activated. Different from Eq (8), the summation of the binary variables at m^{th} time step for the j^{th} foot is allowed to be zero to generate a swing phase as shown in Eq. (23).

$$\forall r \in [R], m \in [M], j \in [n_f]$$

$$\mathbf{H}_{r,m,j} \Rightarrow \mathbf{A}_r \mathbf{p}_j[i] \leq \mathbf{b}_r, \forall i \in \mathcal{O}_{m,m+1} \tag{21}$$

$$\mathbf{A}_{\text{eq},r} \mathbf{p}_j[i] = \mathbf{b}_{\text{eq},r}, \forall i \in \mathcal{O}_{m,m+1} \tag{22}$$

$$\sum_{r=0}^{R-1} \mathbf{H}_{r,m,j} \leq 1 \tag{23}$$

$$\mathbf{H}_{r,m,j} \in \{0, 1\} \tag{24}$$

2) Frictional and Contact Constraints: Different from the gait-fixed case in Eq. (10) - (13), the activation of frictional and

contact constraints is purely decided by the binary variables.

$$\forall r \in [R], m \in [M], j \in [n_f]$$

$$\mathbf{H}_{r,m,j} \;\Rightarrow\; \mathbf{f}_j[i] \cdot \mathbf{n}_r(\mathbf{p}_j^{xy}[i]) \geq 0, \forall i \in \mathcal{O}_{m,m+1} \tag{25}$$

$$\mathbf{f}_j[i] \in \mathcal{F}_r(\mu, \mathbf{n}_r, \mathbf{p}_j^{xy}[i]), \forall i \in \mathcal{O}_{m,m+1} \tag{26}$$

$$\sum_{r=0}^{R-1} \mathbf{H}_{r,m,j} = 1 \;\Rightarrow\; \dot{\mathbf{p}}_j[i] = 0, \forall i \in \mathcal{O}_{m,m+1} \tag{27}$$

$$\sum_{r=0}^{R-1} \mathbf{H}_{r,m,j} = 0 \;\Rightarrow\; \mathbf{f}_j[i] = 0, \forall i \in \mathcal{O}_{m,m+1} \tag{28}$$

Compared to gait-fixed MICP, gait-free MICP introduces more binary variables to determine contact states, resulting in longer computation times. As such, it is only triggered after performing symbolic repair.

VII. Online Execution

Due to the inevitable disparity between offline synthesis and real-world online terrain conditions, such as variations in terrain shape and position, additional efforts are required to bridge this gap and prevent potential execution failures. The online execution module takes in terrain states $\sigma_{\text{terrain}} \subseteq \mathcal{I}_{\text{terrain}}$ after abstracting the terrain polygons online, a request state $\sigma_{\text{req}} \subseteq \mathcal{I}_{\text{req}}$, and a strategy $\mathcal{A}$ from the offline synthesis module (Sec. VI). This module then executes the strategy automaton $\mathcal{A}$ by solving a MICP problem online again with the actual perceived terrain polygons, potentially different from the ones abstracted offline, to generate a reference trajectory for each transition (Sect. VII-A). This reference trajectory along with the corresponding contact information will be further tracked by a tracking controller in real time (Sec. VIII). As shown by the blue lines in Fig. 3(c), if the robot encounters an unseen terrain state, we leverage online symbolic repair to create new skills that handle the unexpected terrain state and failure transition at runtime (Sec. VII-B). After reaching the request state σ_{req}, the robot resets the strategy with a new terrain state, request goal states, and repeats the execution until reaching the final goal state.

A. Strategy Automaton Execution and Online MICP Modifications

The red path in Fig. 3(c) represents the automaton online execution process. Before transitioning to the new symbolic state, an online gait-fixed MICP is solved according to the skill provided by the automaton. The automaton advances only if the MICP successfully finds a solution. Slightly different from the gait-fixed MICP formulation during the offline phase, the online gait-fixed MICP is executed given accurate terrain information from a terrain segmentation module. For example, a skill transitioning from flat terrain to high terrain, as shown in Fig. 5(a), uses predefined, homogeneous terrain polygons for feasibility check during offline synthesis. Given that, Fig. 5(b) illustrates how the terrain polygons detected online for the same skill may differ from the offline ones in both position and geometry. In addition, the following modifications are highlighted when attempting to transition between two symbolic states in the automaton.

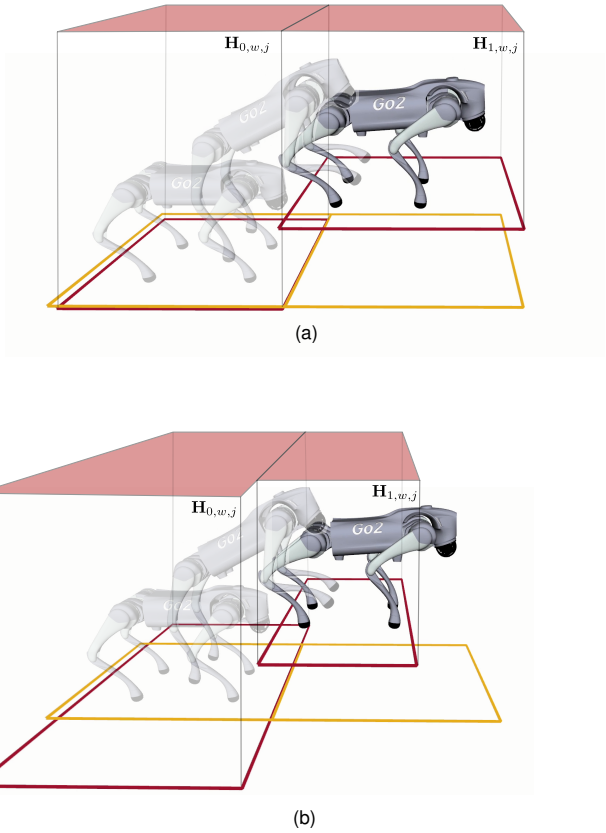


Fig. 5. Demonstrations of (a) offline MICP for a skill transitioning from a flat terrain to a high terrain with predefined polygons; (b) online MICP for the same skill but with online terrain polygons.

1) Robot Pose Re-Targeting: Since the final targeting condition significantly influences the reference trajectory and the feasibility of the MICP problem, we evaluate the final condition by solving a kinematic feasibility problem (a simplified MICP in Eq. (1)) before solving the online MICP. Compared with the gait-fixed MICP, the dynamics equation (3) is replaced by enforcing the center of mass (CoM) to lie inside a convex support polygon with all feet in stance to ensure static stability. In addition, a hard constraint is added to ensure that the modified robot pose stays within a certain threshold compared with the original desired pose. The cost function is simplified to stay as close as possible to the original desired pose. The higher-order terms of body position, orientation, and end-effectors are excluded from the decision variables. All the other constraints not involving those higher-order terms remain the same. Once the kinematic feasibility problem is solved successfully, the online MICP is solved using the modified re-targeted final condition and reference trajectory.

2) Collision Avoidance and Swing Foot Constraints: Since the trajectory generated by the online gait-fixed MICP is directly sent to a tracking control module later, high-quality end-effector (EE) trajectories are crucial to ensure collision avoidance and sufficient swing foot clearance from the terrain. To achieve this, we first introduce more binary variables $\mathbf{H}_{s,w,j}$ for defining the collision-free region selection for the j^{th} foot at each time step. Specifically, $s \in [S]$ represents the s^{th} convex collison-free region and $w \in [W]$ denotes the w^{th}

time step, evenly discretized with Δt_w over the entire W time steps. Fig. 5(b) demonstrates two collision-free regions.

$$\forall s \in [S], w \in [W], j \in [n_f]$$
$$\mathbf{H}_{s,w,j} \Rightarrow \mathbf{A}_s \mathbf{p}_j[i] \leq \mathbf{b}_s, \forall i \in \mathcal{C}_{w,w+1} \tag{29}$$
$$\sum_{s=0}^{S-1} \mathbf{H}_{s,w,j} = 1 \tag{30}$$
$$\mathbf{H}_{s,w,j} \in \{0,1\} \tag{31}$$

where $\mathcal{C}_{w,w+1}$ denotes the set of continuous time steps that span from the w^{th} to the $w+1^{\text{th}}$ binary time step.

To enable larger swing foot clearance from the terrain in case of non-trivial tracking errors, we also add constraints to force the swing foot height above a user-defined threshold h_{swing}, given the fixed contact timing.

Note that the collision avoidance and swing foot constraints are also incorporated during the offline MICP as part of the feasibility checking rules. However, we highlight it here due to its critical impact on the tracking performance.

B. Runtime Repair

The runtime repair procedure takes as input a terrain state $\sigma_{\text{terrain}} \in \Sigma_{\text{terrain}}$, a request state $\sigma_{\text{req}} \in \Sigma_{\text{req}}$, and a Boolean formula $\varphi_{\text{disallow}}$ that encodes disallowed transitions identified at runtime. We first update the system's hard constraint φ_s^t in the specification φ (see Sec. VI-B) to $\varphi_s^t \wedge \square\varphi_{\text{disallow}}$, ensuring that the specification reflects these newly discovered restrictions. Next, the high-level manager constructs a partial evaluation of φ over σ_{terrain} and σ_{req}. Symbolic repair is then applied to generate additional robot skills and synthesize a new strategy that accommodates the current terrain and request states, following the approach in Sec. VI-C.

VIII. Tracking Control

The tracking control module adopts an MPC-Whole-Body-Control (WBC) hierarchy, ensuring precise end-effector (EE) and centroidal momentum tracking. To smoothly and efficiently execute the strategy considering the potential computational delay in solving MICP, a harmonic coordination module between the strategy automaton roll-out and the tracking module is designed (Sec. VIII-B).

A. Tracking Control Module

1) Nonlinear Model Predictive Control (NMPC): The NMPC operates independently from the symbolic planning module, tracking reference trajectories generated by the online MICP using a more accurate dynamics model. In this work, the NMPC accounts for the robot's centroidal dynamics and kinematics, similar to approaches in [99]–[101], presenting a more accurate model than the simplified angular dynamics in the MICP. An analytical inverse kinematics solution based on the MICP output provides the full joint state reference trajectory for the NMPC. In addition to standard frictional and contact constraints and base tracking costs, an additional cost function for EE reference tracking from the MICP is included to enhance precision. The NMPC optimization problem is formulated as a Sequential Quadratic Program (SQP) and solved using the OCS2 library [102], with a time horizon of one second and 100 knot points.

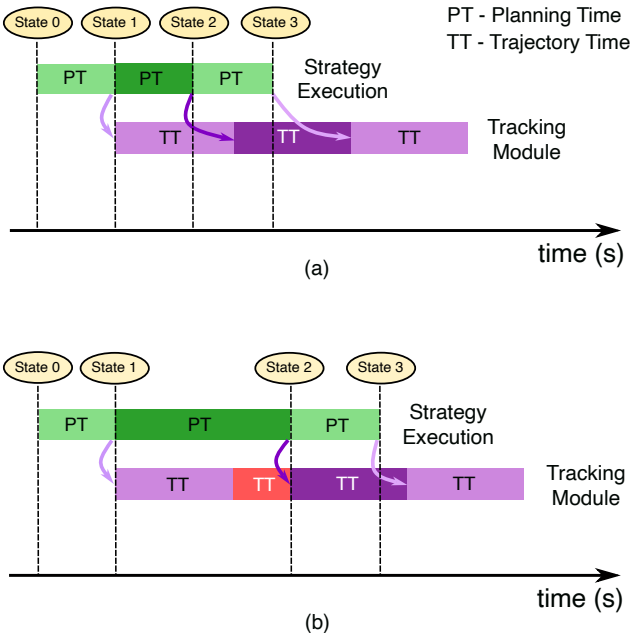


Fig. 6. Coordination between strategy execution and tracking control module. (a) The MICP solving time (denoted by PT) for transitioning from State 1 to State 2 is shorter than the time horizon of the previous reference trajectory (denoted by TT), allowing the seamless appending for the new trajectory. (b) The MICP solving time exceeds the time horizon of the previous reference trajectory, requiring the robot to come to a stop (red) and wait for the new trajectory whenever available.

2) Whole-Body Control (WBC): While NMPC operates at the centroidal level to generate dynamically feasible trajectories, WBC ensures precise execution by resolving full-body state control, including joint torques, accelerations, and contact forces. The whole-body controller tracks the NMPC trajectory by solving a weighted quadratic program (QP) [103] at 500 Hz. To improve the performance of agile motions such as leaping and jumping, centroidal momentum tracking [104] is included. This MPC-WBC hierarchy allows the system to handle both centroidal-dynamics-level and full-body-dynamics-level control in a coordinated manner, ensuring robustness in real-world deployment especially for highly dynamic motions such as jumping.

3) State Estimation: The state estimator employs a linear Kalman filter similar to that in [105], with the addition of OptiTrack motion capture (mocap) measurements fused alongside IMU data and joint encoder readings to achieve accurate body position estimation. For agile motions such as jumping, we observed improved base-height estimation by assigning higher noise values to the mocap feedback during aerial phases, due to its limited 120 Hz update rate.

B. Coordination between Strategy Execution and Tracking Module

Due to the potential delays in solving MICP during the strategy automaton roll-out, a harmonic coordination between the strategy execution and tracking control module is essential.

The tracking module's reference trajectory is updated dynamically alongside the strategy execution process, as shown in Fig. 6. The overall process can be summarized as follows:

1) Strategy Execution: The strategy automaton execution thread begins by solving the online gait-fixed MICP, starting from State 0 in the automaton. Once the MICP solution is obtained, it immediately sends the reference trajectories and corresponding contact information to the tracking control module. The automaton then progresses to the next transition, using the final condition in the previous trajectory segment as the new initial condition. In Fig. 6, the colored blocks in the strategy execution timeline indicate the time taken for each MICP solve during the automaton roll-out. PT and TT denote the planning time and trajectory time, respectively.

2) Tracking Control Module: The MPC-WBC tracking control module receives time-indexed reference trajectories, which can be updated in real-time. After completing the tracking of the current reference trajectory, the robot returns to a stance phase, ready to accept new reference trajectories. As depicted in Fig. 6, the colored blocks in the tracking control module represent the time horizon of each reference trajectory sent by the strategy execution thread. Once the initial reference trajectory is generated, the MPC-WBC thread begins tracking it using more accurate dynamics models, as described in Sec.VIII.

3) Delay Handling: For more complex scenarios involving a larger number of terrain polygons, the MICP solving time may increase and exceed the time horizon of the generated reference trajectory. Fig. 6(a) and (b) illustrate two cases of MICP solving times: Case (a): The solving time for transitioning from automaton State 1 to State 2 is shorter than the time horizon of the previous reference trajectory. In this case, the new reference trajectory is seamlessly appended to the existing one, and the robot continues tracking the previous trajectory. Case (b): The solving time exceeds the time horizon of the previous reference trajectory, and the robot has already entered a stance phase (red block) when the MICP solution is obtained. In this case, the robot waits until the newly generated reference trajectory is sent based on the current system time (red block in Fig. 6 (b)).

IX. Simulation

We present examples of maneuvering across diverse environments in a Gazebo simulation to demonstrate the framework's efficacy, scalability, and generalizability as in [24]. In addition, benchmarking experiments are conducted to further compare our proposed framework with pure MIP approaches.

A. Robot Setup

As shown in Fig. 7, we verify our algorithms on two separate robot platforms, Unitree Go2 and SkyMul Chotu (a modified Unitree Go1 robot dedicated for rebar tying tasks). The SkyMul Chotu is equipped with a rebar gun mounted on its head and customized "cross"-shaped feet designed for reliable standing on rebars. Note that in simulation, this foot shape is simplified to a regular round foot to avoid the complexity of simulating contact with irregular surfaces.

Fig. 7. Hardware platforms used for experiments: Unitree Go2 and SkyMul Chotu (equipped with a rebar gun and "cross"-shaped feet).

TABLE II
Robot Parameters

Parameter	Unitree Go2	SkyMul Chotu
m (kg)	15	20
$\boldsymbol{\tau}_{\max}$ (N · m)	(23.5, 23.5, 45.4)	(23.5, 23.5, 33.5)
$\mathbf{q}_j^{\text{ref}}$ (rad)	(0.0, 0.72, -1.44)	(0.0, 0.72, -1.44)
${}^{\mathcal{B}}\mathbf{p}_j^{\text{ref}}$ (m)	(0.1805, 0.1308, -0.29)	(0.2118, 0.210, -0.30)
$\mathbf{p}_j^{\max}$ (m)	(0.15, 0.1, 0.15)	(0.15, 0.1, 0.15)
$\mathbf{I}$ (kg · m^2)	diag(0.152, 0.369, 0.388)	diag(0.396, 0.915, 1.107)

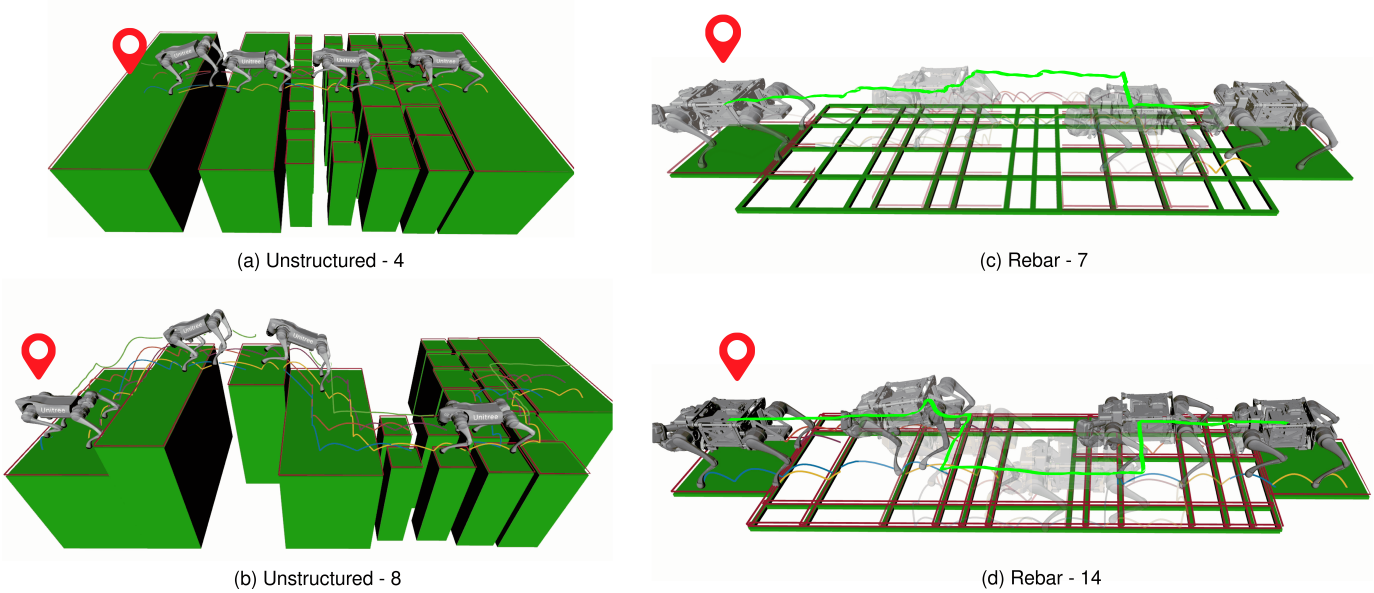
In our planner, we model both cases as point contact. The parameters for each robot are defined in Table II and used in our implementation, including the robot mass m, torque limit $\boldsymbol{\tau}_{\max}$ and the reference joint position $\mathbf{q}_j^{\text{ref}}$ for a single leg with three joints, the reference foot position relative to the base frame ${}^{\mathcal{B}}\mathbf{p}_j^{\text{ref}}$ and the maximum deviation of the foot position $\mathbf{p}_j^{\max}$ for the front left leg ($j = 0$), and the moment of inertia of the approximated single rigid body $\mathbf{I}$.

B. Simulation Environment Setup

To demonstrate the generalizability of our proposed framework, we evaluate it across two scenarios involving different terrain types, safe stepping regions, and robotic platforms—Unitree Go2 and SkyMul Chotu. Obstacles are modeled as a distinct terrain class, and obstacle avoidance is encoded as hard constraints within $\varphi_{\text{s}}^{\text{t}}$. The requested waypoints are assumed to be free of obstacles. We test both 3×3 and 5×5 grid abstractions. The 5×5 grid enables a longer planning horizon but introduces higher decision complexity. For discretization, we use a cell size of 0.8 m in unstructured terrains and 0.6 m for the rebar case. These values can be adjusted based on the requirements of specific deployment environments.

1) Unstructured Terrain: We abstract 8 terrain types—*flat terrain*, *high terrain*, *low terrain*, *dense stone*, *sparse stone*, *gap*, *high gap*, and *low gap*—based on classification criteria involving polygon heights, counts, and areas relative to the abstracted grid cells. A terrain is categorized as a *gap* when the ratio of its overlapping area with a grid cell falls below a specified threshold. We define two terrain configurations: one consisting of four selected terrain types, and another comprising all eight. These are illustrated in Figs. 8(a) and 8(b), respectively.

2) Rebar Terrain: Inspired by efforts to automate labor-intensive rebar tying tasks on construction sites [81], we



(a) Unstructured - 4

(c) Rebar - 7

(b) Unstructured - 8

(d) Rebar - 14

Fig. 8. Unstructured and rebar terrain scenarios (presented in [24] as well).

investigate a second case study in which a quadrupedal robot navigates a rebar mat. Each rebar is modeled as a rectangular polygon with a 3 cm width. Unlike the stepping stone scenario, this setup presents a denser distribution of potential footholds due to the higher rebar density. Notably, the robot's traversability depends on the directional sparsity of the rebars—whether aligned with or orthogonal to the robot's facing direction—within an acceptable tolerance.

We abstract and classify rebar types based on horizontal sparsity (perpendicular to the robot's facing direction) and vertical sparsity (parallel to it). Within each abstracted cell, the number and spacing of rebars are evaluated to classify the sparsity in each direction as *dense* (0.05–0.15 m), *sparse* (0.15–0.35 m), *extreme sparse* (above 0.35 m), *single*, or *none*. To reduce complexity, combinations deemed too challenging—such as *none* or *single* in both directions—are treated as *obstacle*. Based on this abstraction, we define two example configurations comprising 7 and 14 rebar terrain types. The 7-type configuration in Fig. 8(c) samples rebar sparsity randomly between 0.15 and 0.35 m, omitting *extreme sparse* regions. In contrast, the 14-type configuration in Fig. 8(d) includes sparsity ranging from 0.15 to 0.6 m, encompassing more complex and difficult combinations including *extreme sparse* types.

C. Offline Synthesis Results

We evaluate the offline synthesis module for both *Unstructured* and *Rebar Terrain* scenarios by initializing the skill set with two trotting gaits (2 s and 3 s). The inclusion of the longer-horizon trotting gait provides additional safer walking options from a practical standpoint. Prior to synthesis, we construct the set of possible terrain states $\Sigma^{\text{poss}}_{\text{terrain}}$—assumed to be provided by the user—by systematically sweeping a local grid over the entire terrain map, based on the best available prior knowledge. The set of request states $\Sigma^{\text{poss}}_{\text{req}}$ is defined as the top row of the local grid in the robot's facing direction, with the two corner cells also included to enable movement in diagonal directions.

We briefly summarize the key offline synthesis statistics previously reported in [24]. Specifically, the repair process was shown to successfully handle approximately 93.4% of all terrain and request state pairs $(\sigma_{\text{terrain}}, \sigma_{\text{req}}) \in \Sigma^{\text{poss}}_{\text{terrain}} \times \Sigma^{\text{poss}}_{\text{req}}$. The remaining 6.6% were unrepairable due to either unreachable request states caused by obstacles or physical infeasibility. Notably, the number of newly discovered skills was reduced by 71.6–97.6% compared to checking the full space of all possible skills, each requiring the solution of a costly gait-free MICP. On average, generating a new locomotion gait via gait-free MICP took 27.23 s, with a maximum time of 145.66 s. These results demonstrate the effectiveness of the offline repair strategy in minimizing expensive MICP solves. For a full breakdown of synthesis time, number of symbolic states, and repair outcomes, please refer to [24] (Table I).

D. Online Execution Results

Fig. 8 presents snapshots of Go2 and Chotu navigating various terrains during online execution. The robot is tasked with reaching a global waypoint using a naive global shortest-path planner. At each step, the local waypoint is selected as the closest non-obstacle cell in the local grid map pointing toward the global target.

In Fig. 8(a), the robot traverses a sequence of terrain types including *flat terrain*, *dense stepping stone*, *sparse stepping stone*, and *gap*. Fig. 8(b) showcases a more complex maneuver involving all eight terrain types, including elevation changes, where the robot adaptively selects suitable locomotion gaits. Similarly, Figs. 8(c) and (d) illustrate Chotu traversing rebar terrains with 7 and 14 defined rebar types, respectively. Notably, the robot utilizes newly discovered leaping gaits with short aerial phases to execute challenging transitions—such

TABLE III
ONLINE GAIT-FIXED MICP SOLVING TIME FOR SIMULATION

Scenario	Collision Avoidance	Binary Variables	Continuous Variables	Number of Polygons	Solve Time (s)
Unstructured - 4	No	52	6583.5	6.5	0.37
Unstructured - 8	Yes	256	5166	2	2.65
	No	69	7938	7.5	0.49
Rebar - 7	No	70	7938	7.875	1.11
Rebar - 14	No	58	8142.75	6.25	1.09

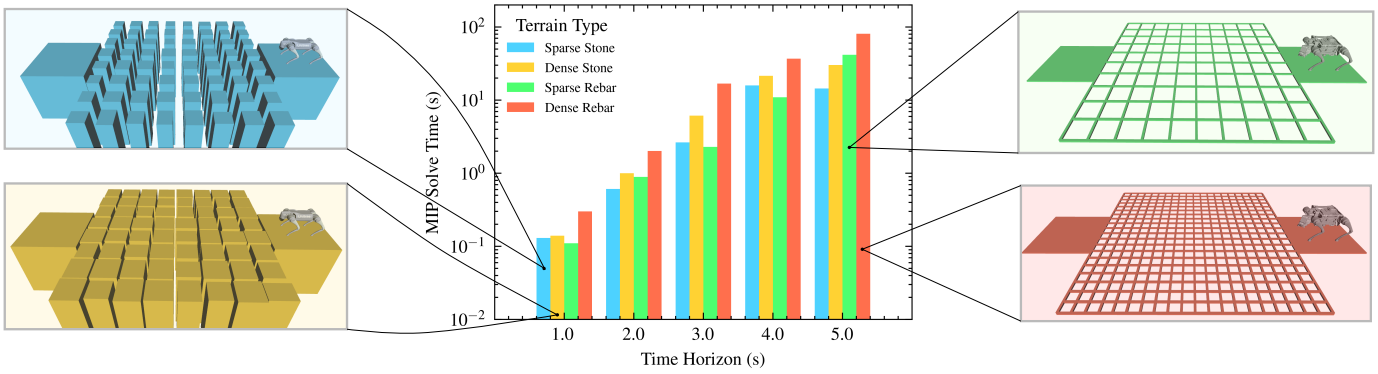


Fig. 9. MIP solve time benchmark. Four scenarios including sparse stepping stone (blue), dense stepping stone (yellow), sparse rebar (green), and dense rebar (red) are evaluated with time horizons ranging from 1 - 5 seconds.

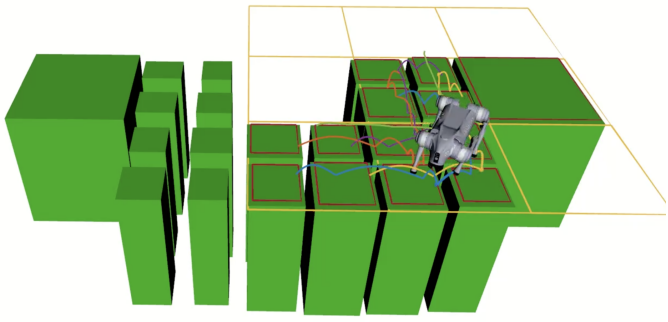


Fig. 10. Demonstration of the robot executing multiple turning behaviors to continuously progress toward the global waypoint on terrain arranged in a zig-zag pattern.

as moving from lower to higher elevations or crossing over gaps and *extreme sparse* rebar segments, as demonstrated in Figs. 8(b) and(d).

Table III summarizes the average number of decision variables, terrain polygons, and solve time for each scenario, computed across all step transitions in a full locomotion run with the online gait-fixed MICP. On average, the online gait-fixed MICP takes 430 ms to solve the unstructured terrain cases and 1.1 s for rebar cases when the collision avoidance is disabled. The rebar scenarios exhibit 155.8% longer solve times due to their overlapping, skewed rebar arrangement, and a larger number of terrain polygons. Additionally, enabling collision avoidance (necessary for *Unstructured - 8* case to handle the elevation change) increases the number of binary variables by 271.0%, increasing the solve time by 440.8% (highlighted in red).

E. Benchmark: Pure MIP Planner

To evaluate how the symbolic planning benefits the overall planning framework, we benchmark our method against pure MIP approaches, where we implement a planner to evaluate how computation time scales with the planning horizon and terrain complexity. Specifically, we examine five planning horizons ranging from 1 s to 5 s. In addition, we vary the terrain types by considering sparse stepping stones (0.15 m spacing), dense stepping stones (0.05 m spacing), sparse rebar (0.3 m spacing), and dense rebar (0.15 m spacing), as illustrated in Fig. 9 with specific colors.

The benchmarked planner is configured to solve only gait-fixed MICP; we omit gait-free MICP due to its significantly longer solve time in longer horizon setup that are more than 2 s. For a given planning horizon, the local planning region is linearly expanded to increase the traversable distance. The locomotion gait is repeated cyclically on a 1 s trotting pattern. During execution, the planner—similar to our proposed framework—receives a global waypoint and incrementally computes local waypoints, selecting the farthest reachable position within the local planning region.

As shown in Fig. 9, the histogram illustrates the average time to solve the MIP in different scenarios and time horizons. The results show that, within the same scenario, the MICP solve time increases almost exponentially with the planning horizon. Furthermore, dense rebar and stepping stone scenarios consistently require longer solve times than their sparse counterparts, due to the larger number of terrain segments involved in the optimization. In our proposed method, we primarily use 2 to 3 s gait horizons for each symbolic transition, striking a balance between computational efficiency and sufficient planning foresight. This choice avoids overly short myopic horizons, while still allowing for adaptive gait behaviors.

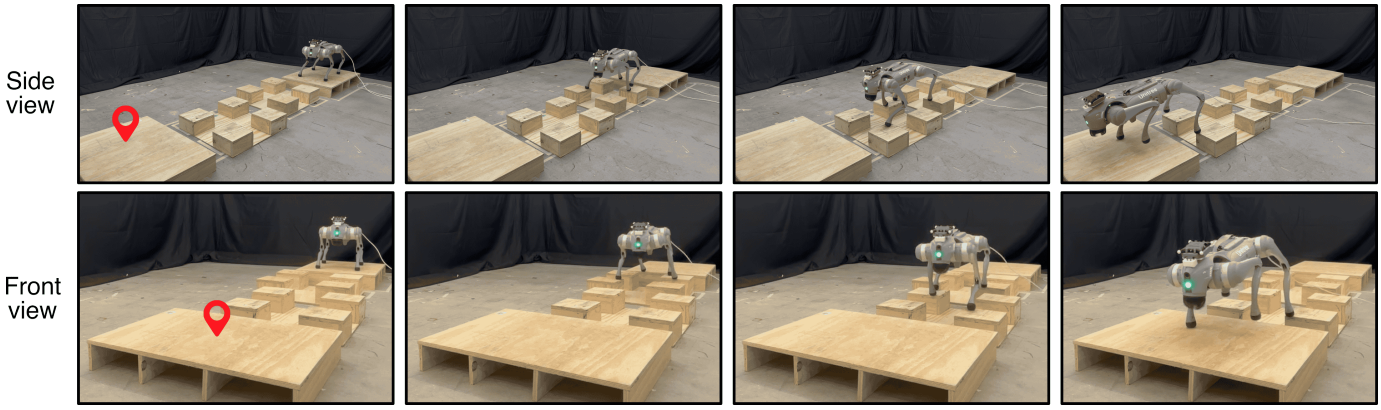


Fig. 11. Hardware experiment demonstration for the first unstructured terrain scenario with sparse stepping stones and flat terrains.

F. Extension: Locomotion with Yaw Command

In the previous examples, we assume that the yaw angle of the robot base is fixed to simplify the problem. However, the robot's physical capability to traverse terrains largely depends on the base orientation. For instance, performing a forward jump onto a high terrain is different from a sideway jump, in terms of both leg kinematics and dynamics. Therefore, to showcase the flexibility and generalizability of our framework, we provide an extension to the existing abstraction of the robot state to allow the orientation variation.

Specifically, the robot inputs are now defined as $\mathcal{I}_{\text{robot}} := \{\pi_x \mid x \in \mathcal{X}\} \cup \{\pi_y \mid y \in \mathcal{Y}\} \cup \{\pi_{yaw} \mid yaw \in \mathcal{W}\}$, where we define a new set of yaw angles $\mathcal{W} := \{-180°, -90°, 0°, 90°, 180°\}$. Note that, a finer discretization of the yaw angles can be defined as needed. Similarly, the request inputs are defined as $\mathcal{I}_{\text{req}} := \{\pi_x^{\text{req}} \mid x \in \mathcal{X}\} \cup \{\pi_y^{\text{req}} \mid y \in \mathcal{Y}\} \cup \{\pi_{yaw}^{\text{req}} \mid yaw \in \mathcal{W}\}$. The robot and the request input can be further divided into translational $\mathcal{I}_{\text{robot}}^{\text{trans}}, \mathcal{I}_{\text{req}}^{\text{trans}}$ and rotational parts $\mathcal{I}_{\text{robot}}^{\text{orien}}, \mathcal{I}_{\text{req}}^{\text{orien}}$, where the translational ones contain the x and y states and the orientational ones contain the yaw state.

The skill is further divided into translational and rotational skills. A translational skill $o_{\text{trans}} \in \mathcal{O}_{\text{trans}}$ consists of sets of preconditions $\Sigma_{o_{\text{trans}}}^{\text{pre}} \subseteq \Sigma_{\text{robot}} \times \Sigma_{\text{terrain}}$ that include the full robot state, and translational-only postconditions $\Sigma_{o_{\text{trans}}}^{\text{post}} \subseteq \Sigma_{\text{robot}}^{\text{trans}}$. A rotational skill in place $o_{\text{orien}} \in \mathcal{O}_{\text{orien}}$ consists of sets of rotational-only preconditions $\Sigma_{o_{\text{orien}}}^{\text{pre}} \subseteq \Sigma_{\text{robot}}^{\text{orien}}$ and postconditions $\Sigma_{o_{\text{orien}}}^{\text{post}} \subseteq \Sigma_{\text{robot}}^{\text{orien}}$. The terrain state can also be incorporated when generating rotational skills, particularly for terrains that pose challenges for turning. However, we omit this in our current implementation for simplicity. The specification encoding then follows the same procedure as described in Sec. VI-B, based on each skill's precondition and postcondition. The hard constraints remain unchanged, except that translational and rotational ones are defined separately to ensure the corresponding states remain unchanged when no skill is selected.

To demonstrate the resulting maneuver, we construct a scenario with multiple dense and sparse stepping stones arranged in an overall zig-zag pattern, requiring several turning behaviors. Although sideways movement skills are feasible, we manually exclude them before synthesis to highlight yaw adjustments controlled by the rotational skills. As shown in Fig. 10, the robot executes multiple turns to continuously progress towards the global waypoint. More complex coupled translational and rotational behaviors can also be defined by modifying the preconditions and postconditions associated with each skill.

X. Hardware Demonstrations

We demonstrate that our entire framework can be successfully deployed on hardware, highlighting two key aspects. First, we showcase the framework's capability to generate adaptive locomotion gaits across diverse terrain types, particularly focusing on agile leaping motions. Second, we illustrate the framework's assured safety through obstacle avoidance and selection of dynamically feasible gaits, benchmarked against a heuristic-based planner.

A. Hardware Experiment Setup

For hardware experiments, we set up similar real-world scenarios for both unstructured and rebar terrains. For the unstructured terrain scenario, we construct wooden blocks to form stepping stones, gaps, and flat regions, with each stepping stone elevated approximately 0.12 m above the ground. For the rebar scenario, we assemble a realistic rebar mat using fourteen $1/2$-inch $\times$ 4-foot rebars and six $1/2$-inch $\times$ 10-foot rebars. In the synthesis setup, obstacles are considered a distinct terrain type, consistent with the simulation. Both scenarios assume the maximum potential terrain types—8 for unstructured and 14 for rebar—though the real setups include fewer types. As demonstrated in the simulation, this does not increase complexity, as it only scales linearly with terrain-request pairs due to our high-level manager. For safety, we set abstraction cell sizes to 0.8 m for the unstructured terrain and 0.45 m for the rebar terrain.

B. Unstructured Terrain

As shown in Fig. 11, we first create a scenario with sparse stepping stones and flat terrain regions. The stepping stones are spaced between $0.18 - 0.23$ m apart. Both side and front

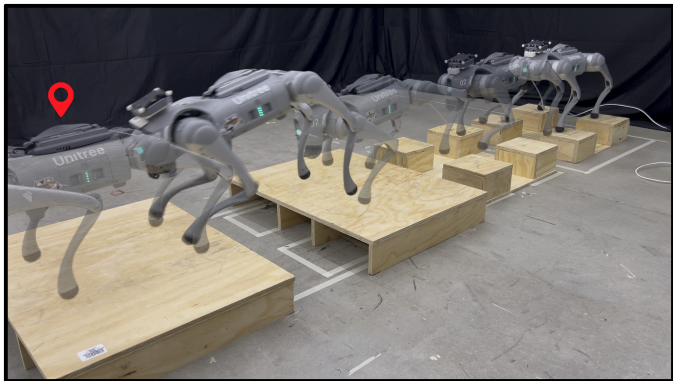


Fig. 12. Hardware experiment demonstration for the second unstructured terrain scenario with sparse stepping stones, flat, and gap terrains.

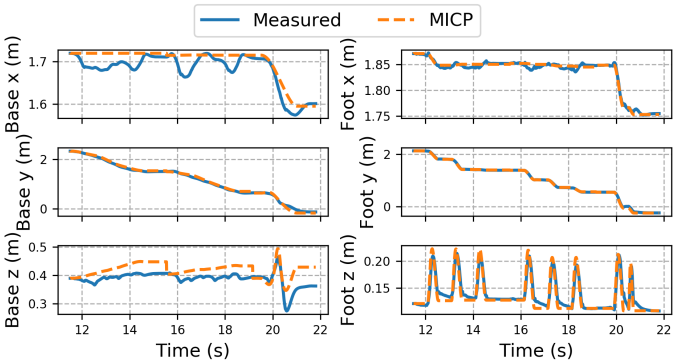


Fig. 13. Tracking performance of the base and front-left foot positions: comparison between measured states and MICP-generated reference trajectories.

views are provided to highlight the misalignment of stepping stones along the robot walking direction. The MICP planner successfully generates a reference trajectory involving non-trivial sideways movement, enabling the robot to safely land on the stepping stones. All transitions utilize a 3 s trotting gait.

In the second scenario, shown in Fig. 12, we further include a gap terrain segment. The gap, approximately 30 cm wide, necessitates the use of a leaping gait. Since the gap terrain was incorporated during offline synthesis, our framework can directly select an optimized 1.5 s leaping gait generated by gait-free MICP upon encountering this gap terrain. The task-space tracking performance between the measured state and the reference trajectory generated by the online MICP is shown in Fig. 13. This includes the floating base position and the end-effector (EE) position for the front-left foot. The EE tracking maintains errors within 0.01 m in the x and y axes, ensuring precise foot placement. Meanwhile, the NMPC effectively adjusts the base position to compensate for model simplifications at the MICP level, enabling successful execution of both trotting and leaping motions.

C. Rebar Terrain

For the rebar terrain scenarios shown in Fig. 14, we test two cases: one without any obstacles and one with an obstacle introduced during execution. The rebar spacing varies between 0.1 m and 0.3 m. Offline synthesis is conducted without prior knowledge of obstacles. In the first case (top row), the robot Chotu successfully performs the entire maneuver without encountering any runtime failures, consistently moving forward until reaching the goal waypoint. In the second case (bottom row), an obstacle is added to the environment but only detected during the online execution phase. Upon encountering a new terrain/request pair caused by the obstacle, the planner promptly triggers a runtime resynthesis. Leveraging previously generated skills, it computes a new strategy within 0.05 s to detour around the obstacle and continue toward the goal. All transitions in both scenarios use a 2.25 s static walking gait, where only one foot is lifted at a time to ensure safety.

D. Benchmark: Heuristics-Based Planner

We create a third rebar scenario, shown in Fig. 15, by removing two rebars to further increase the maximum spacing to 0.48 m. This highlights the safety advantage of our planner compared to a heuristics-based footstep planner. The heuristic baseline follows a similar strategy to [4], [7], adapted for the rebar scenario by selecting the nearest rebar polygon and the closest feasible foothold to a nominal target. The resulting footstep plan and linearly interpolated base trajectory are sent to the same tracking control module. For fairness, all parameters are tuned to achieve successful traversal at a speed of 0.1 m/s (as shown in the supplemental video).

Without re-performing the offline synthesis, our framework directly uses the results from Sec. X-C. It classifies the extreme sparse region as an obstacle and performs runtime resynthesis, generating a detour to safely reach the goal. In contrast, we task the heuristics-based planner with the same goal at 0.2 m/s to match our planner's effective gait speed (i.e., 0.45 m per 2.25 s transition). Without reasoning about the compatibility between terrain geometry and gait feasibility, the heuristic planner suffers from poor EE tracking after a few steps, leading to foot slippage and eventual failure as the robot is entangled by the sparse rebar layout.

E. Computational Time

We report the hardware computational time in Table IV. Except for the second scenario in the unstructured terrain, we observe a notable increase in solve time compared to simulation. This is mainly due to two factors: (1) In the first unstructured terrain scenario, the two sets of stepping stones are laterally misaligned from the robot's viewpoint, requiring non-trivial deviation from the nominal foot location and sideways movement; (2) In the rebar scenarios, the real-world rebar polygons are not perfectly aligned with the x or y-axis as in simulation, introducing additional numerical complexity for branch-and-bound solvers. Further acceleration of MIP solving through tighter convex relaxations [106], [107] remains an important direction for future work.

XI. Discussion

A. Generalization to More Types of Terrain

The current framework relies on manual terrain abstraction and classification, which requires expert knowledge to define terrain types and assign them to discrete symbolic categories.

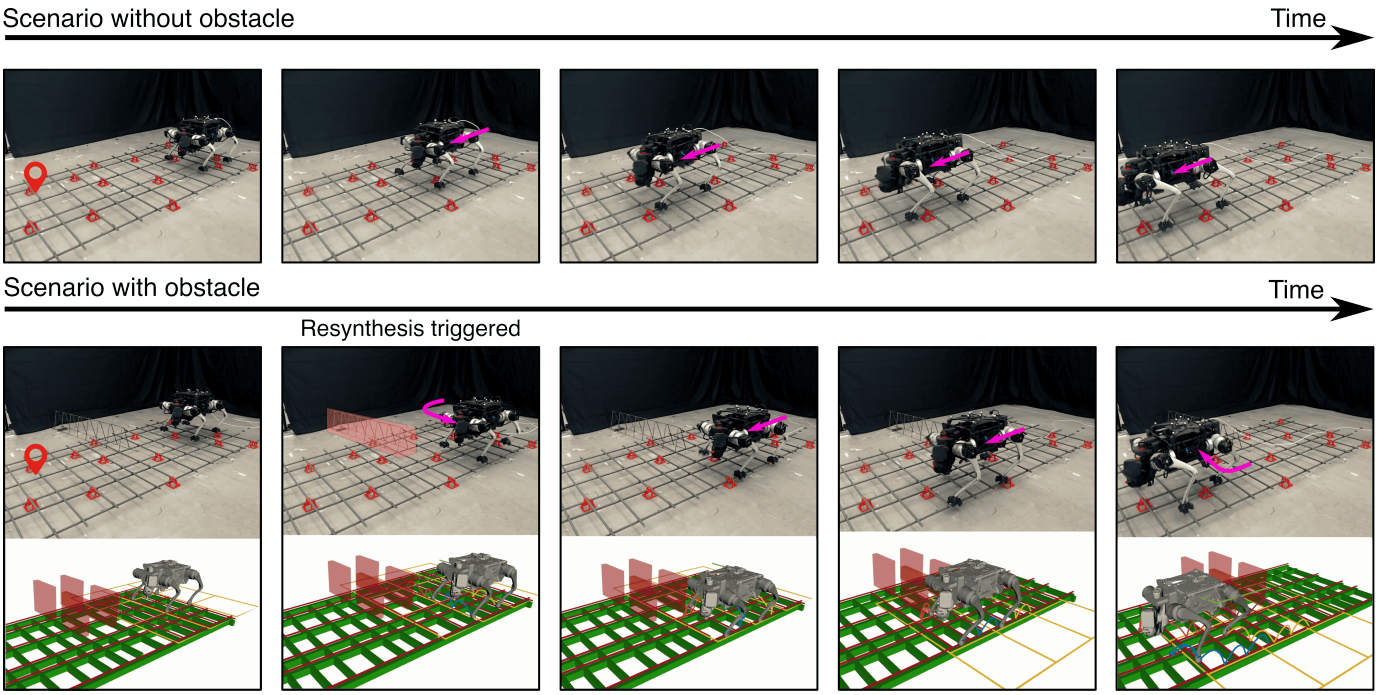


Fig. 14. Execution results in rebar terrain without (top row) and with obstacle (bottom row). In the obstacle case, runtime resynthesis is triggered to generate a detour strategy and ensure successful traversal.

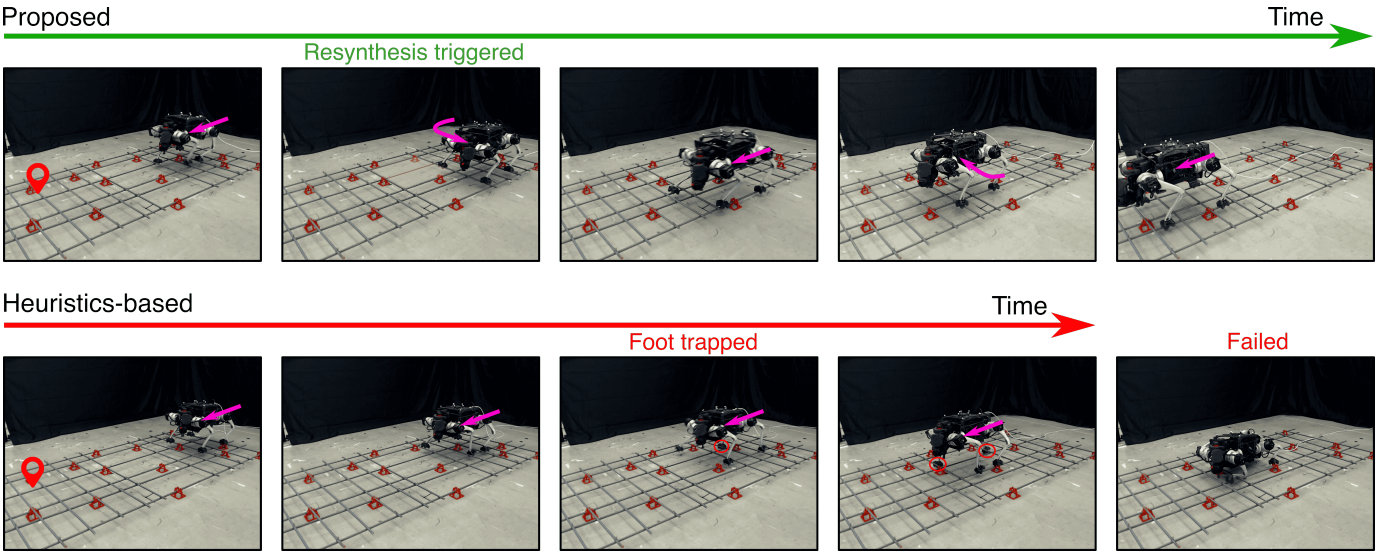


Fig. 15. Comparison between the proposed method and a heuristics-based planner in an extreme sparse rebar scenario. The proposed method performs resynthesis and succeeds, while the heuristics-based planner fails due to foot trapping.

TABLE IV
ONLINE GAIT-FIXED MICP SOLVING TIME FOR HARDWARE

Scenario	Polygons	Min (s)	Mean (s)	Max (s)
Unstructured - Scene 1	6	0.81	4.04	7.86
Unstructured - Scene 2	5	0.16	0.53	0.96
Rebar - Scene 1	8	3.88	6.54	9.26
Rebar - Scene 2	7.5	3.60	9.13	19.23
Rebar - Scene 3	6.33	2.45	7.37	14.7

This process typically involves analyzing physical features such as terrain shape, height variation, and support area to determine whether a terrain segment should be labeled as flat, sparse, dense, high, gap, etc. While effective, this expert-driven approach may limit scalability and generalization to novel environments or terrain conditions. Future extensions could incorporate data-driven terrain classification and generation methods, such as supervised learning or clustering based on 3D point cloud statistics to automate the terrain abstraction process and reduce reliance on human heuristics.

B. *Optimality*

The synthesis process focuses on guaranteeing feasibility and correctness of transitions but does not account for the rel-

ative cost or quality of different feasible gaits. When multiple locomotion gaits (e.g., trotting, bounding, leaping) are viable for a given symbolic transition, the current method does not evaluate their optimality in terms of energy efficiency, robustness, execution time, or other performance metrics. Instead, it selects the first gait that satisfies the physical feasibility conditions via MICP. Integrating cost-aware synthesis would enable the framework to prioritize higher-quality behaviors and make more informed decisions under trade-offs.

C. Perception Module

The proposed framework is modular and can be directly integrated with a wide range of perception pipelines that provide terrain geometry in the form of segmented polygonal regions [108]. Existing perception modules that perform terrain segmentation using LiDAR, RGB-D sensors, or stereo vision can be adapted to supply the required input to the MICP planner and symbolic abstraction layers.

XII. Conclusion

In this work, we presented an integrated planning framework for terrain-adaptive locomotion that combines reactive synthesis with MICP, enabling safe and reactive responses in dynamically changing environments. To address unrealizable specifications arising from limited motion primitives, we introduced a symbolic repair approach that incorporates dynamic feasibility checks, automatically identifying missing transitions necessary for navigating adversarial terrains. In the online execution phase, we tackled the disparity between offline synthesis and real-world conditions by using an online MICP solver and a runtime repair process based on real-world terrain data, including unforeseen terrain conditions. Our approach not only enhances motion feasibility and safety but also provides a scalable solution for legged robots maneuvering in complex and safety-critical environments.

References

[1] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1–9.

[2] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[3] Z. Zhou, B. Wingo, N. Boyd, S. Hutchinson, and Y. Zhao, "Momentum-aware trajectory optimization and control for agile quadrupedal locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7755–7762, 2022.

[4] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5761–5768.

[5] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," *IEEE Transactions on Robotics*, 2023.

[6] H.-W. Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *Robotics: Science and Systems*, 2015.

[7] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 2464–2470.

[8] S. Gilroy, D. Lau, L. Yang, E. Izaguirre, K. Biermayer, A. Xiao, M. Sun, A. Agrawal, J. Zeng, Z. Li *et al.*, "Autonomous navigation for quadrupedal robots with optimized jumping through constrained obstacles," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 2132–2139.

[9] J. Norby and A. M. Johnson, "Fast global motion planning for dynamic legged robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2020, pp. 3829–3836.

[10] M. Chignoli, S. Morozov, and S. Kim, "Rapid and reliable quadruped motion planning with omnidirectional jumping," in *International Conference on Robotics and Automation*. IEEE, 2022, pp. 6621–6627.

[11] Y. Zhao, Y. Li, L. Sentis, U. Topcu, and J. Liu, "Reactive task and motion planning for robust whole-body dynamic locomotion in constrained environments," *The International Journal of Robotics Research*, p. 02783649221077714, 2022.

[12] A. Shamsah, Z. Gu, J. Warnke, S. Hutchinson, and Y. Zhao, "Integrated task and motion planning for safe legged navigation in partially observable environments," *IEEE Transactions on Robotics*, 2023.

[13] A. K. Valenzuela, "Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain," Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[14] Y. Shirai, X. Lin, A. Schperberg, Y. Tanaka, H. Kato, V. Vichathorn, and D. Hong, "Simultaneous contact-rich grasping and locomotion via distributed optimization enabling free-climbing for multi-limbed robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 563–13 570.

[15] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS international conference on humanoid robots*. IEEE, 2014, pp. 279–286.

[16] H. Dai, G. Izatt, and R. Tedrake, "Global inverse kinematics via mixed-integer convex optimization," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1420–1441, 2019.

[17] Y. Ding, C. Li, and H.-W. Park, "Kinodynamic motion planning for multi-legged robot jumping via mixed-integer convex program," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3998–4005.

[18] N. Fey, R. J. Frei, and P. M. Wensing, "3d hopping in discontinuous terrain using impulse planning with mixed-integer strategies," *IEEE Robotics and Automation Letters*, 2024.

[19] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 842–849.

[20] F. Risbourg, T. Corbères, P.-A. Léziart, T. Flayols, N. Mansard, and S. Tonneau, "Real-time footstep planning and control of the solo quadruped robot in 3d environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 12 950–12 956.

[21] B. Acosta and M. Posa, "Bipedal walking on constrained footholds with mpc footstep control," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots*. IEEE, 2023, pp. 1–8.

[22] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2017.

[23] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911–938, 2012.

[24] Z. Zhou, Q. Meng, H. Kress-Gazit, and Y. Zhao, "Physically-feasible reactive synthesis for terrain-adaptive locomotion via trajectory optimization and symbolic repair," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025.

[25] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1184–1189.

[26] P. Fernbach, S. Tonneau, A. Del Prete, and M. Taïx, "A kinodynamic steering-method for legged multi-contact locomotion," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3701–3707.

[27] Q. Liao, Z. Li, A. Thirugnanam, J. Zeng, and K. Sreenath, "Walking in narrow spaces: Safety-critical locomotion control for quadrupedal robots with duality-based optimization," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2723–2730.

[28] S. Feng, Z. Zhou, J. S. Smith, M. Asselmeier, Y. Zhao, and P. A. Vela, "Gpf-bg: A hierarchical vision-based planning framework for safe

quadrupedal navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1968–1975.

[29] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 811–818.

[30] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2665–2670.

[31] A. Winkler, I. Havoutis, S. Bazeille, J. Ortiz, M. Focchi, R. Dillmann, D. Caldwell, and C. Semini, "Path planning with force-based foothold adaptation and virtual model control for torque controlled quadruped robots," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6476–6482.

[32] C. Mastalli, I. Havoutis, A. W. Winkler, D. G. Caldwell, and C. Semini, "On-line and on-board planning and perception for quadrupedal locomotion," in *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, 2015, pp. 1–7.

[33] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain," in *2019 IEEE-RAS 19th international conference on humanoid robots (humanoids)*. IEEE, 2019, pp. 9–16.

[34] S. Tonneau, D. Song, P. Fernbach, N. Mansard, M. Taïx, and A. Del Prete, "Sl1m: Sparse l1-norm minimization for contact planning on uneven terrain," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6604–6610.

[35] J. Ren, X. Lin, R. Mineyev, K. M. Feigh, S. Coogan, and Y. Zhao, "Accelerating signal-temporal-logic-based task and motion planning of bipedal navigation using benders decomposition," *arXiv preprint arXiv:2508.13407*, 2025.

[36] K. Hauser, T. Bretl, and J.-C. Latombe, "Non-gaited humanoid locomotion planning," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.* IEEE, 2005, pp. 7–12.

[37] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 317–342, 2006.

[38] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

[39] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain–online foothold optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5370–5376, 2020.

[40] A. Agrawal, S. Chen, A. Rai, and K. Sreenath, "Vision-aided dynamic quadrupedal locomotion on discrete terrain using motion libraries," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4708–4714.

[41] O. A. V. Magana, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini, "Fast and continuous foothold adaptation for dynamic locomotion through cnns," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2140–2147, 2019.

[42] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2436–2442.

[43] W. Yu, D. Jain, A. Escontrela, A. Iscen, P. Xu, E. Coumans, S. Ha, J. Tan, and T. Zhang, "Visual-locomotion: Learning to walk on complex terrains with vision," in *5th Annual Conference on Robot Learning*, 2021.

[44] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2908–2927, 2022.

[45] F. Wu, X. Nal, J. Jang, W. Zhu, Z. Gu, A. Wu, and Y. Zhao, "Learn to teach: Sample-efficient privileged learning for humanoid locomotion over real-world uneven terrain," *IEEE Robotics and Automation Letters*, 2025.

[46] J. Kamohara, F. Wu, C. Wamorkar, S. Hutchinson, and Y. Zhao, "Rl-augmented adaptive model predictive control for bipedal locomotion over challenging terrain," 2025.

[47] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8352–8358.

[48] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "Tamols: Terrain-aware motion optimization for legged systems," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3395–3413, 2022.

[49] K. Muenprasitivej, J. Jiang, A. Shamsah, S. Coogan, and Y. Zhao, "Bipedal safe navigation over uncertain rough terrain: Unifying terrain mapping and locomotion stability," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 11 264–11 271.

[50] A. Shamsah, J. Jiang, Z. Yoon, S. Coogan, and Y. Zhao, "Terrain-aware model predictive control of heterogeneous bipedal and aerial robot coordination for search and rescue tasks," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 12 352–12 358.

[51] K. Muenprasitivej, Y. Zhao, and G. Chou, "Probabilistically-safe bipedal navigation over uncertain terrain via conformal prediction and contraction analysis," 2025.

[52] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg, "Learning a contact-adaptive controller for robust, efficient legged locomotion," in *Conference on Robot Learning*. PMLR, 2021, pp. 883–894.

[53] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and efficient locomotion via learned gait transitions," in *Conference on Robot Learning*. PMLR, 2022, pp. 773–783.

[54] Z. Xie, X. Da, B. Babich, A. Garg, and M. v. de Panne, "Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2022, pp. 523–539.

[55] G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. b. Kim, and P. Agrawal, "Learning to jump from pixels," in *Proceedings of the 5th Conference on Robot Learning*. PMLR, 2022, pp. 1025–1034.

[56] S. Yu, N. Perera, D. Marew, and D. Kim, "Learning generic and dynamic locomotion of humanoids across discrete terrains," *arXiv preprint arXiv:2405.17227*, 2024.

[57] C. Boussema, M. J. Powell, G. Bledt, A. J. Ijspeert, P. M. Wensing, and S. Kim, "Online gait transitions and disturbance recovery for legged robots via the feasible impulse set," *IEEE Robotics and automation letters*, vol. 4, no. 2, pp. 1611–1618, 2019.

[58] K. Wang, Z. J. Hu, P. Tisnikar, O. Helander, D. Chappell, and P. Kormushev, "When and where to step: Terrain-aware real-time footstep location and timing optimization for bipedal robots," *arXiv preprint arXiv:2302.07345*, 2023.

[59] H. Sun, J. Yang, Y. Jia, and C. Wang, "Free gait generation of quadruped robots via impulse-based feasibility analysis," *IEEE/ASME Transactions on Mechatronics*, 2023.

[60] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[61] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (ToG)*, vol. 31, no. 4, pp. 1–8, 2012.

[62] 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.

[63] A. Aydinoglu and M. Posa, "Real-time multi-contact model predictive control via admm," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3414–3421.

[64] S. Le Cleac'h, T. A. Howell, S. Yang, C.-Y. Lee, J. Zhang, A. Bishop, M. Schwager, and Z. Manchester, "Fast contact-implicit model predictive control," *IEEE Transactions on Robotics*, 2024.

[65] L. Drnach, J. Z. Zhang, and Y. Zhao, "Mediating between contact feasibility and robustness of trajectory optimization through chance complementarity constraints," *Frontiers in Robotics and AI*, vol. 8, p. 785925, 2022.

[66] X. Jiang, W. Chi, Y. Zheng, S. Zhang, Y. Ling, J. Xu, and Z. Zhang, "Locomotion generation for quadruped robots on challenging terrains via quadratic programming," *Autonomous Robots*, vol. 47, no. 1, pp. 51–76, 2023.

[67] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, no. 1, pp. 265–293, 2021.

[68] Z. Zhao, S. Cheng, Y. Ding, Z. Zhou, S. Zhang, D. Xu, and Y. Zhao, "A survey of optimization-based task and motion planning: From classical to learning approaches," *IEEE/ASME Transactions on Mechatronics*, 2024.

[69] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning." in *IJCAI*, 2015, pp. 1930–1936.

[70] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," 2018.

[71] T. Migimatsu and J. Bohg, "Object-centric task and motion planning in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 844–851, 2020.

[72] Z. Zhao, Z. Zhou, M. Park, and Y. Zhao, "Sydebo: Symbolic-decision-embedded bilevel optimization for long-horizon manipulation in dynamic environments," *IEEE Access*, vol. 9, pp. 128 817–128 826, 2021.

[73] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Versatile multicontact planning and control for legged loco-manipulation," *Science Robotics*, vol. 8, no. 81, p. eadg5014, 2023.

[74] Y. Ding, M. Zhang, C. Li, H.-W. Park, and K. Hauser, "Hybrid sampling/optimization-based planning for agile jumping robots on challenging terrains," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2839–2845.

[75] Y. Shirai, X. Lin, A. Mehta, and D. Hong, "Lto: lazy trajectory optimization with graph-search planning for high dof robots in cluttered environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7533–7539.

[76] E. Jelavic, F. Farshidian, and M. Hutter, "Combined sampling and optimization based planning for legged-wheeled robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8366–8372.

[77] L. Amatucci, J.-H. Kim, J. Hwangbo, and H.-W. Park, "Monte carlo tree search gait planner for non-gaited legged system control," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4701–4707.

[78] C. Chen, P. Culbertson, M. Lepert, M. Schwager, and J. Bohg, "Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8262–8268.

[79] M. Zhang, D. K. Jha, A. U. Raghunathan, and K. Hauser, "Simultaneous trajectory optimization and contact selection for multi-modal manipulation planning," *arXiv preprint arXiv:2306.06465*, 2023.

[80] H. Zhu, A. Meduri, and L. Righetti, "Efficient object manipulation planning with monte carlo tree search," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10 628–10 635.

[81] M. Asselmeier, J. Ivanova, Z. Zhou, P. A. Vela, and Y. Zhao, "Hierarchical experience-informed navigation for multi-modal quadrupedal rebar grid traversal," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 8065–8072.

[82] H. Kress-Gazit, M. Lahijanian, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 211–236, 2018.

[83] 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, pp. 5425–5432.

[84] Z. Zhou, D. J. Lee, Y. Yoshinaga, S. Balakirsky, D. Guo, and Y. Zhao, "Reactive task allocation and planning for quadrupedal and wheeled robot teaming," in *IEEE International Conference on Automation Science and Engineering*. IEEE, 2022, pp. 2110–2117.

[85] J. Jiang, S. Coogan, and Y. Zhao, "Abstraction-based planning for uncertainty-aware legged navigation," *IEEE Open Journal of Control Systems*, 2023.

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

[87] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," in *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 1989, pp. 179–190.

[88] 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.

[89] J. A. DeCastro, R. Ehlers, M. Rungger, A. Balkan, P. Tabuada, and H. Kress-Gazit, "Dynamics-based reactive synthesis and automated revisions for high-level robot control," *arXiv preprint arXiv:1410.6375*, 2014.

[90] G. E. Fainekos, S. G. Loizou, and G. J. Pappas, "Translating temporal logic to controller specifications," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 899–904.

[91] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2689–2696.

[92] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1771–1785, 2013.

[93] 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 *Proceedings of the 16th international conference on Hybrid systems: computation and control*, 2013, pp. 353–362.

[94] A. Pacheck, S. Moarref, and H. Kress-Gazit, "Finding missing skills for high-level behaviors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 335–10 341.

[95] A. Pacheck and H. Kress-Gazit, "Physically feasible repair of reactive, linear temporal logic-based, high-level tasks," *IEEE Transactions on Robotics*, 2023.

[96] Q. Meng and H. Kress-Gazit, "Automated Robot Recovery from Assumption Violations of High-Level Specifications," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. IEEE, Aug. 2024, pp. 4154–4161.

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

[98] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[99] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[100] M. Chignoli, D. Kim, E. Stanger-Jones, and S. Kim, "The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors," in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2021, pp. 1–8.

[101] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified mpc framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.

[102] F. Farshidian *et al.*, "OCS2: An open source library for optimal control of switched systems," [Online]. Available: https://github.com/leggedrobotics/ocs2.

[103] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2589–2594.

[104] P. M. Wensing and D. E. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3103–3109.

[105] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2245–2252.

[106] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake, "Shortest paths in graphs of convex sets," *SIAM Journal on Optimization*, vol. 34, no. 1, pp. 507–532, 2024.

[107] X. Lin, "Accelerate hybrid model predictive control using generalized benders decomposition," *arXiv preprint arXiv:2406.00780*, 2024.

[108] T. Miki, L. Wellhausen, R. Grandia, F. Jenelten, T. Homberger, and M. Hutter, "Elevation mapping for locomotion and navigation using gpu," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2273–2280.