Physically-Feasible Reactive Synthesis for Terrain-Adaptive Locomotion

Journal Title
XX(X):1–26

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Ziyi Zhou[1], Qian Meng[2], Hadas Kress-Gazit[2], and Ye Zhao[1]

Abstract

We present an integrated planning framework for quadrupedal locomotion over discrete, unforeseen terrain configurations revealed at runtime. 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 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, runtime symbolic repair, kinematic-feasibility re-targeting, and a delay-aware coordination scheme enable 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/.

Keywords

Legged Robots, Optimization and Optimal Control, Formal Methods in Robotics and Automation

1 Introduction

Terrain-adaptive locomotion is essential for enhancing the mobility of legged robots and moving beyond blind walking strategies (Di Carlo et al. 2018; Kim et al. 2019; Zhou et al. 2022b). Many existing methods achieve terrain adaptability by adjusting footholds around a nominal trajectory in real time (Fankhauser et al. 2018; Grandia et al. 2023). To further boost traversability, some approaches have explored variable gait patterns. In (Park et al. 2015; Kim et al. 2020; Gilroy et al. 2021), 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 (Norby and Johnson 2020), or rely on pre-trained classifiers to assess feasibility (Chignoli et al. 2022). 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 agile, variable-gait locomotion (Zhao et al. 2022; Shamsah et al. 2023) that fully account for the robot's physical limitations.

Mixed-integer programming (MIP)-based methods (Valenzuela 2016; Shirai et al. 2022) 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 (Deits and Tedrake 2014). When dynamics and constraints are suitably relaxed, the resulting convex approximation (MICP) provides a global certificate of feasibility upon convergence (Dai et al. 2019), 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 (Ding et al. 2020; Fey et al. 2024), constraining timing (Ponton et al. 2016; Risbourg et al. 2022; Acosta and Posa 2023), or limiting the number of footsteps (Aceituno-Cabezas et al. 2017). 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 (Fankhauser et al. 2018). 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 (Bloem et al. 2012), which enables us to systematically

[1]Georgia Institute of Technology, Atlanta, GA, USA
[2]Cornell University, Ithaca, NY, USA

Corresponding author:
Ye Zhao, Georgia Institute of Technology, Atlanta, GA 30332, USA.
Email: ye.zhao@me.gatech.edu

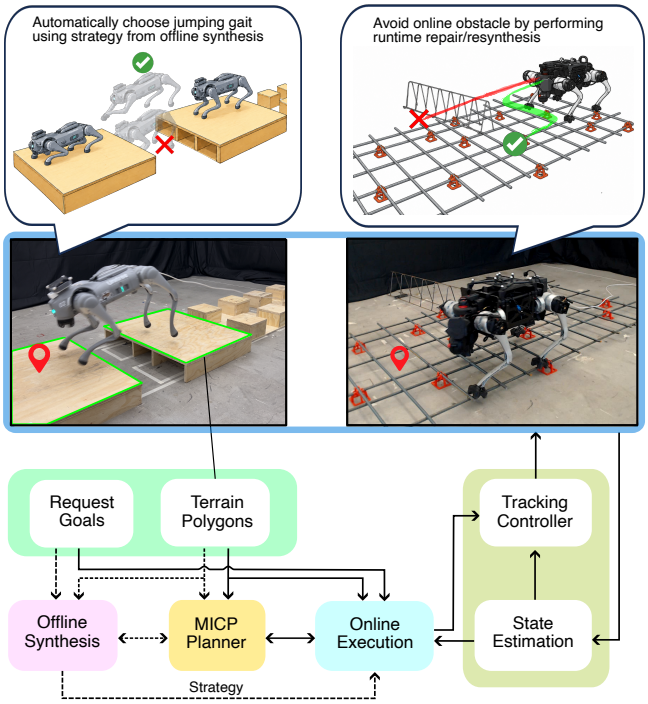


Figure 1. System architecture overview. The solid lines indicate online communication, while dashed lines represent offline processes. The request goals in the offline phase are user-defined, while those in the online phase are provided by a global planner based on the global goal.

construct a correct-by-construction strategy that guides the robot through local decisions toward achieving its intermediate goals. For readers unfamiliar with formal methods, LTL is a logic for expressing temporal properties of system behaviors using operators such as "always," "eventually," and "until" (Piterman et al. 2006; Bloem et al. 2012); reactive synthesis automatically constructs a controller (named as a *strategy automaton* in this paper) that satisfies an LTL specification against any admissible environment behavior, providing correct-by-construction guarantees at the symbolic level.

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 unforeseen terrain configurations revealed at runtime. 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. The framework is two-stage: 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. 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.

The two-stage architecture exposes two scalability challenges that we address with two complementary mechanisms. First, the full task specification has a state space that grows exponentially with the number of variables encoding the surrounding terrains, so direct synthesis is computationally impractical (Bloem et al. 2012); to address this, we propose a high-level manager that fixes the local terrain and goal information and only retains the skills needed for the current local environment, yielding an 80.9–90.1% reduction in the symbolic variable count in our experiments. Second, gait-free MICP—used to certify dynamic feasibility—is itself computationally expensive; we therefore use a symbolic repair process that suggests only the missing transitions whose feasibility must be checked, yielding a 71.6–97.6% reduction in costly gait-free MICP solves relative to exhaustive enumeration. During online execution, the gait-fixed MICP is re-solved against the actual perceived terrain to generate a new nominal trajectory, and runtime symbolic repair handles any unforeseen terrain that the offline phase did not cover.

Closing the loop between this synthesis-MICP framework and a real robot poses two practical challenges that we address through two additional online-execution mechanisms. (i) Online MICP solve times are inherently variable, ranging from hundreds of milliseconds to several seconds, and can exceed the trajectory horizon used by the tracking controller. We propose a *delay-aware coordination* scheme that asynchronously appends or stalls the tracking trajectory based on a formal two-case analysis of the per-transition timing budget; without it, the closed loop between the discrete strategy automaton and the continuous-time tracker is ill-defined under bounded solve time. (ii) The online MICP tends to become infeasible when the offline-checked target pose mismatches the actual terrain. We add a lightweight, zero-horizon *kinematic-feasibility re-targeting* step that pre-checks and adjusts the desired pose at each transition, decoupling offline pose specification from online terrain geometry; the same online MICP is also augmented with additional collision-avoidance constraints for swing-foot clearance during a transition. Together with the offline scalability machinery, these mechanisms make the framework end-to-end deployable, which we validate on two robot hardware platforms and benchmark against both a pure-MIP planner and a heuristic foothold planner.

The proposed framework remains flexible and modular—it can be driven by commands from a higher-level planner (Wermelinger et al. 2016; Fernbach et al. 2017; Norby and Johnson 2020; Chignoli et al. 2022; Liao et al. 2023; Feng et al. 2023) or directly from a user, and it easily accommodates new behaviors by expanding the symbolic transition set.

The main contributions of this work are summarized as follows:

- **Integrated reactive-synthesis and MICP framework for terrain-adaptive locomotion.** Symbolic-level guidance limits the planning horizon and the

number of terrain features any single MICP must consider, bridging high-level symbolic reasoning with low-level physical feasibility.

- **Scalable offline synthesis via a high-level manager and symbolic repair with dynamic feasibility checks.** The high-level manager reduces the symbolic state space by 80.9–90.1%, and symbolic repair, equipped with dynamic feasibility checks via gait-free MICP, generates only the necessary skills—reducing costly gait-free MICP solves by 71.6–97.6% versus exhaustive enumeration.
- **Online execution module that bridges offline synthesis and real-world deployment.** The module resolves a gait-fixed MICP against the actual perceived terrain at each symbolic transition, with delay-aware coordination between the strategy automaton and the tracking controller, online kinematic-feasibility re-targeting of the desired pose, and additional collision-avoidance constraints in the online MICP. Runtime symbolic repair handles previously unseen scenarios that were not considered offline.
- **Comprehensive experimental validation.** We report simulation across four terrain scenarios with full per-module timing, comparisons against a pure MIP planner and a heuristic foothold planner, hardware deployment of the integrated stack on two robot platforms (Unitree Go2 and SkyMul Chotu), and a feasibility study of yaw orientation as a generalization case study.

A conference version of this work is presented in (Zhou et al. 2025). Beyond the conference paper, the current article contributes the delay-aware coordination and online re-targeting mechanisms motivated above, full hardware validation of the integrated framework on two robot platforms, systematic benchmarking against a pure-MIP planner and a heuristic foothold planner, a preliminary yaw-extension feasibility study, and substantially expanded preliminaries, related-work survey, and discussion of limitations.

2 Related Work

2.1 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 (Kolter et al. 2008; Kalakrishnan et al. 2010; Winkler et al. 2014; Mastalli et al. 2015; Fankhauser et al. 2018; Griffin et al. 2019) or optimization-based approaches (Deits and Tedrake 2014; Tonneau et al. 2020; Risbourg et al. 2022; Ren et al. 2025). Although non-fixed gait/contact plans can be generated for very rough terrains (Hauser et al. 2005; Bretl 2006; Tonneau et al. 2018), 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 (Jenelten et al. 2020; Kim et al. 2020; Agrawal et al. 2022) or learning approaches (Magana et al. 2019; Villarreal et al. 2020; Yu et al. 2021; Gangapurwala et al. 2022; Wu et al. 2025; Kamohara et al. 2025). In (Grandia et al. 2021; Jenelten et al. 2022; Grandia et al. 2023), 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 (Muenprasitivej et al. 2024; Shamsah et al. 2025; Muenprasitivej et al. 2025).

However, the above dynamic locomotion works consider a fixed gait pattern that usually prohibits achieving versatile behaviors, such as jumping. In (Park et al. 2015; Kim et al. 2020; Gilroy et al. 2021), jumping motions are generated offline and triggered through heuristics. In (Fernbach et al. 2017; Norby and Johnson 2020; Chignoli et al. 2022), 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 (Norby and Johnson 2020) or decided by a pre-trained feasibility classifier (Chignoli et al. 2022). 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 (Da et al. 2021; Yang et al. 2022; Xie et al. 2022; Margolis et al. 2022; Yu et al. 2024) and model-based methods (Boussema et al. 2019; Wang et al. 2023; Sun et al. 2023).

A complementary line of work uses end-to-end reinforcement learning (RL) to train perceptive locomotion policies that directly map terrain observations to joint commands, achieving impressive robustness on stairs, gaps, and unstructured outdoor terrain (Miki et al. 2022a; Agarwal et al. 2023; Zhuang et al. 2023; Hoeller et al. 2023; Cheng et al. 2024; Luo et al. 2024). Compared with these data-driven approaches, our model-based, formal-methods framework offers explicit symbolic-level realizability guarantees and physical-feasibility certificates from MICP, at the cost of weaker generalization and the need for a terrain abstraction; we view the RL approaches and ours as complementary rather than competing.

2.2 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 (Posa et al.

2014) or smoothed (Mordatch et al. 2012; Drnach and Zhao 2021) complementarity constraints, which accurately model the physical contact interaction behavior but remains computationally inefficient due to the non-smoothness and high nonlinearity. Promising online contact-implicit MPC results are reported in (Aydinoglu and Posa 2022; Le Cleac'h et al. 2024; Drnach et al. 2022) 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 (Valenzuela 2016; Shirai et al. 2022; Jiang et al. 2023b). 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 (Dai et al. 2019), 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 (Ponton et al. 2016; Ding et al. 2020; Acosta and Posa 2023) assume the contact sequence and timing are chosen *a priori*, and only optimize the contact plane selection. Aceituno-Cabezas et al. (Aceituno-Cabezas et al. 2017) 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.

2.3 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 (Garrett et al. 2021; Zhao et al. 2024). For more complex physical contact resoning, Toussaint et al. propose Logic Geometric Programming (LGP) (Toussaint 2015) and embed the high-level logic representation into the low-level motion planner, demonstrating contact-rich tool-use behaviors (Toussaint et al. 2018) after defining abundant action primitives. Building on this, a broader range of motions has been showcased, including versatile manipulation (Migimatsu and Bohg 2020; Zhao et al. 2021) and loco-manipulation tasks (Sleiman et al. 2023). In a similar vein, works such as (Ding et al. 2021; Shirai et al. 2021; Jelavic et al. 2021; Amatucci et al. 2022; Chen et al. 2021; Zhang et al. 2023; Zhu et al. 2023; Asselmeier et al. 2024) 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 (Kress-Gazit et al. 2018). 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 (Kulgod et al. 2020; Zhou et al. 2022a; Jiang et al. 2023a) or locomotion keyframes for reduced-order models (Gu et al. 2022; Shamsah et al. 2023; Zhao et al. 2022). Reactive synthesis (Pnueli and Rosner 1989), widely applied to mobile robots (Kress-Gazit et al. 2009), has been employed to enable prompt decision-making in response to more complex environments (Shamsah et al. 2023) or external perturbations (Gu et al. 2022). Notably, (Zhao et al. 2022) 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.

2.4 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 (DeCastro et al. 2014; Fainekos et al. 2006) 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 (Bhatia et al. 2010), switched (Liu et al. 2013), and hybrid systems (Maly et al. 2013) 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 (Pacheck et al. 2020; Pacheck and Kress-Gazit 2023) 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. (Meng and Kress-Gazit 2024) extend this type of approach to online symbolic repair. Inspired by these works, Zhou et al. (Zhou et al. 2025) 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 (Zhou et al. 2025) through seamless integration with a low-level tracking control module, enabling systematic benchmarking, and demonstrating real-world hardware deployment.

3 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.

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.*

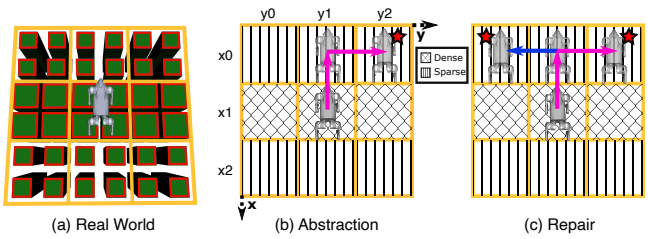


Figure 2. Terrain abstraction and skill definition. (a) Top-down view of the real-world terrain before abstraction. The red polygons denote the segmented terrain polygons. (b) Abstraction of the terrain and robot's skills (pink) moving from one location to another. (c) Repair process to find a new skill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

3.1 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: $\mathcal{X} := \{x_0, \ldots, x_{n-1}\}$ and $\mathcal{Y} := \{y_0, \ldots, y_{m-1}\}$.

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 $\mathcal{I}_{\text{robot}} := \{\pi_x \mid x \in \mathcal{X}\} \cup \{\pi_y \mid y \in \mathcal{Y}\}$ to represent the robot's location. Request inputs are given by $\mathcal{I}_{\text{req}} := \{\pi_x^{\text{req}} \mid x \in \mathcal{X}\} \cup \{\pi_y^{\text{req}} \mid y \in \mathcal{Y}\}$ to represent the desired goal cell. The terrain inputs are defined as $\mathcal{I}_{\text{terrain}} := \{n_x^y \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$, 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 $[z] := \{0, \ldots, z-1\}$. 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 (Ehlers and Raman 2016). The robot state abstraction can also be extended to incorporate orientation, such as the yaw angle of the floating base (see Sec. 8.8).

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 $\text{G} : \Sigma_{\mathcal{I}} \to 2^{\mathcal{Z}}$ 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 $\text{G}(\sigma_{\text{robot}}) := \{z \in \mathcal{Z} \mid \exists \pi_x, \pi_y \in \mathcal{I}_{\text{robot}}.\sigma_{\text{robot}}(\pi_x) \wedge \sigma_{\text{robot}}(\pi_y) \wedge \text{robot\_pose}(z) \in \text{cell}(x, y)\}$. Similarly, for request states $\sigma_{\text{req}} \in \Sigma_{\text{req}}$, we set $\text{G}(\sigma_{\text{req}}) := \{z \in \mathcal{Z} \mid \exists \pi_x^{\text{req}}, \pi_y^{\text{req}} \in \mathcal{I}_{\text{req}}.\sigma_{\text{req}}(\pi_x^{\text{req}}) \wedge \sigma_{\text{req}}(\pi_y^{\text{req}}) \wedge \text{request\_goal}(z) \in \text{cell}(x, y)\}$. For terrain states $\sigma_{\text{terrain}} \in \Sigma_{\text{terrain}}$, the grounding $\text{G}(\sigma_{\text{terrain}})$ 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 $\text{G}(\sigma_{\mathcal{I}}) := \text{G}(\sigma_{\text{robot}}) \cap \text{G}(\sigma_{\text{req}}) \cap \text{G}(\sigma_{\text{terrain}})$. Finally, we introduce an inverse grounding function $\text{G}^{-1} : \mathcal{Z} \to \Sigma_{\mathcal{I}}$ that maps each physical state $z \in \mathcal{Z}$ back to its corresponding symbolic input state.

In Example 1, the inputs are defined as $\mathcal{I}_{\text{robot}} := \{\pi_{x_0}, \pi_{x_1}, \pi_{x_2}, \pi_{y_0}, \pi_{y_1}, \pi_{y_2}\}$, $\mathcal{I}_{\text{req}} := \{\pi^{\text{req}} \mid \pi \in \mathcal{I}_{\text{robot}}\}$, and $\mathcal{I}_{\text{terrain}} := \{n_{x_i}^{y_j} \mid i, j \in 0, 1, 2\}$, 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. 5.1).

3.2 Specifications

We adopt the Generalized Reactivity (1) (GR(1)) fragment of linear temporal logic (LTL) (Bloem et al. 2012) to encode task specifications. An LTL formula φ follows the grammar $\varphi := \pi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \Box\varphi \mid \Diamond\varphi$, where $\pi \in AP$ denotes an atomic proposition, $\neg$ and $\wedge$ are Boolean operators, and $\bigcirc$ ("next"), $\Box$ ("always"), and $\Diamond$ ("eventually") are temporal operators. For a comprehensive introduction to LTL, we refer the reader to (Baier and Katoen 2008).

A GR(1) specification is written in the form $\varphi = \varphi_e \rightarrow \varphi_s$, where $\varphi_e = \varphi_e^i \wedge \varphi_e^t \wedge \varphi_e^g$ encodes the assumptions about the (possibly adversarial) environment, and $\varphi_s = \varphi_s^i \wedge \varphi_s^t \wedge \varphi_s^g$ represents the guarantees that define the system's behavior. For $\alpha \in \{e, s\}$, the components $\varphi_\alpha^i, \varphi_\alpha^t, \varphi_\alpha^g$ describe the initial conditions, safety requirements, and liveness objectives, respectively. We refer below and throughout the rest of this article to *symbolic repair*, the process of augmenting the symbolic transition set with new, dynamically feasible skills whenever the current specification becomes unrealizable. The full procedure is formalized in Sec. 5.3; for the purpose of the upcoming definitions, it suffices to treat symbolic repair as a black box that proposes candidate symbolic transitions whose physical feasibility is then verified by MICP. In the context of symbolic repair (Sec. 5.3), we further partition the safety constraints $(\varphi_e^t, \varphi_s^t)$ into two parts: skill-related constraints $(\varphi_e^{t,\text{skill}}, \varphi_s^{t,\text{skill}})$ and hard constraints $(\varphi_e^{t,\text{hard}}, \varphi_s^{t,\text{hard}})$. The skill constraints encode the preconditions and postconditions of skills (see Sec. 5.2) 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 (Zhou et al. 2025), 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 $\pi \in S^{\texttt{False}}$.

4 Problem Statement and Approach Overview

4.1 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.

4.2 Approach Overview

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. 5), online execution (Sec. 6), and low-level tracking control (Sec. 7). 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 evolving environments. Additionally, the system re-targets the desired pose during each transition to reflect current terrain constraints, and coordinates the strategy automaton with the tracking controller to handle MICP solve-time variability without compromising execution continuity.

5 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 $\Sigma_{\text{req}}^{\text{poss}} \subseteq \Sigma_{\text{req}}$ from the local request goals and mapping the predefined terrain polygons into possible terrain states $\Sigma_{\text{terrain}}^{\text{poss}} \subseteq \Sigma_{\text{terrain}}$. Afterward, we evaluate the feasibility of each candidate skill using a gait-fixed MICP formulated with the provided locomotion gaits (Sec. 5.1), and record all feasible transitions as symbolic skills in the specification (Sec. 5.2). 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. 5.3).

5.1 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

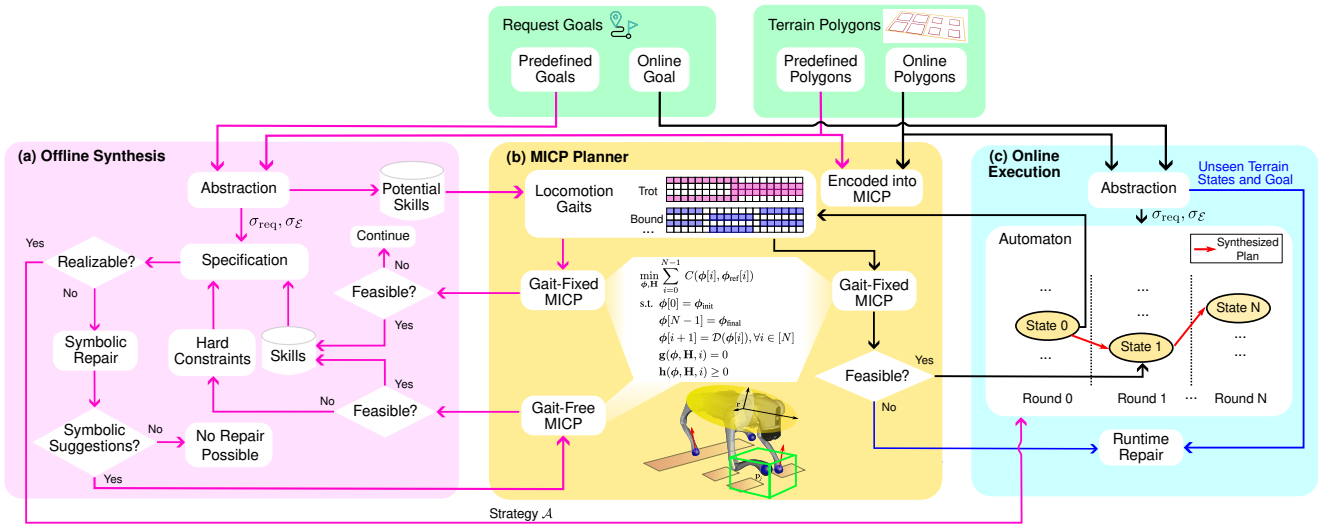


Figure 3. System overview. During offline synthesis (pink arrows), an initial set of locomotion gaits is provided, and symbolic skills are iteratively generated by solving the MICP. When the task specifications are unrealizable, a symbolic repair is triggered to seek missing skills. During the online 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.

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 $\Sigma_{\text{req}}^{\text{poss}}$ and terrain states $\Sigma_{\text{terrain}}^{\text{poss}}$, 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. 8.8.

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 problem considered in this work can formulated as:

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

$$\text{s.t. } \boldsymbol{\phi}[0] = \boldsymbol{\phi}_{\text{init}} \tag{1a}$$

$$\boldsymbol{\phi}[N-1] = \boldsymbol{\phi}_{\text{final}} \tag{1b}$$

$$\boldsymbol{\phi}[i+1] = \mathcal{D}(\boldsymbol{\phi}[i]), \forall i \in [N] \tag{1c}$$

$$\mathbf{g}(\boldsymbol{\phi}, \mathbf{H}, i) = 0 \tag{1d}$$

$$\mathbf{h}(\boldsymbol{\phi}, \mathbf{H}, i) \geq 0 \tag{1e}$$

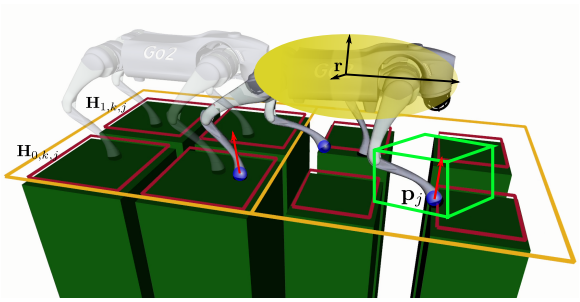


Figure 4. Demonstration of decision variables and polygons when solving MICP for skill a_0 in Example 1 using a trotting locomotion gait.

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 function $\mathcal{D}(\cdot)$ encodes the discrete-time system dynamics that propagate the continuous state $\boldsymbol{\phi}[i]$ to $\boldsymbol{\phi}[i+1]$; the specific dynamics used in this work are detailed in Eq. (3). 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:

5.1.1 Continuous Decision Variables The continuous decision variables ϕ 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:

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

5.1.2 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.

We note that the rotational dynamics adopted in Eq. (3) are intentionally simplified: the rotational equation of motion is replaced by a double integrator on the Euler angles, which keeps the constraint convex but *neglects the true angular-momentum dynamics.* This makes the formulation tractable for online deployment, but it limits applicability to highly dynamic motions such as flips or aggressive yaw maneuvers at high speeds. Future work will explore convex relaxations of the full angular-momentum dynamics to expand the range of feasible motions while maintaining online tractability.

5.1.3 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 1.

5.1.4 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.

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

5.1.5 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.

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

where μ denotes the friction coefficient.

5.1.6 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.

5.1.7 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.

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

Table 1. 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)

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.

5.2 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 $\varphi_{\text{s}}^{\text{t}}$. Each skill is associated with a user-defined precondition set $\Sigma_o^{\text{pre}} \subseteq 2^{\mathcal{I}_{\text{robot}} \cup \mathcal{I}_{\text{terrain}}}$ and postcondition set $\Sigma_o^{\text{post}} \subseteq 2^{\mathcal{I}_{\text{robot}}}$, expressed over the robot and terrain inputs introduced in Sec. 3.1. The encoding rules below map these sets into the GR(1) components $\varphi_{\text{e}}^{\text{t,skill}}$ and $\varphi_{\text{s}}^{\text{t,skill}}$ deterministically.

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:

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

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:

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

The disjunction $\bigvee_{\sigma \in \Sigma_o^{\text{pre}}}$ in the equation above is taken over precondition pairs in Σ_o^{pre}: a skill becomes eligible to execute when the conjunction of input assignments specified by at least one full pair $(\sigma_{\text{robot}}, \sigma_{\text{terrain}}) \in \Sigma_o^{\text{pre}}$ holds, rather than when only some individual atomic propositions within a pair hold. 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. 5.3). 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: $\square\big(\bigwedge_{o \in \mathcal{O}} \neg o \rightarrow \bigwedge_{\pi \in \mathcal{I}_{\text{robot}}} (\pi \leftrightarrow \bigcirc \pi)\big)$.

5.3 High-level Manager and Symbolic Repair

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 (Ehlers and Raman 2016). 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 $\varphi' := \varphi[\sigma_{\text{terrain}}^{\mathcal{B}} \cup \sigma_{\text{req}}, \mathcal{I}_{\text{terrain}}^{\mathcal{B}} \cup \mathcal{I}_{\text{req}} \setminus \sigma_{\text{terrain}}^{\mathcal{B}} \cup \sigma_{\text{req}}]$ (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. 5.1) failed to produce sufficient skills to reach the request under the given terrain. To address this, we rely on gait-free MICP (Sec. 5.4), 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 (Pacheck and Kress-Gazit 2023) 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.

5.4 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:

5.4.1 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).

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

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

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

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

$$\mathbf{H}_{r,m,j} \in \{0,1\} \quad (24)$$

5.4.2 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.

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

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

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

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

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

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.

6 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. 5). 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. 6.1). This reference trajectory along with the corresponding contact information will be further tracked by a tracking controller in real time (Sec. 7). 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. 6.2). 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.

6.1 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 terrain information from a terrain segmentation module that produces labeled convex polygons. 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. Note that, perception is intentionally treated as out of scope in this paper. We assume an upstream segmentation pipeline (e.g., elevation-mapping with plane segmentation (Miki et al. 2022b; Fankhauser et al. 2018)) provides labeled convex polygons; in our hardware experiments we use motion-capture-aided manual labeling of polygons, which is sufficient to validate the planning and control framework, although it does not represent a fully autonomous perception stack. Obtaining convex collision-free regions in unstructured environments is non-trivial, and we make no claim that our framework includes a solution to this perception problem. Robust perception integration is left for future work. In addition, the following modifications are highlighted when attempting to transition between two symbolic states in the automaton.

6.1.1 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.

6.1.2 Collision Avoidance and Swing Foot Constraints Our framework handles obstacles at *two distinct levels*, and

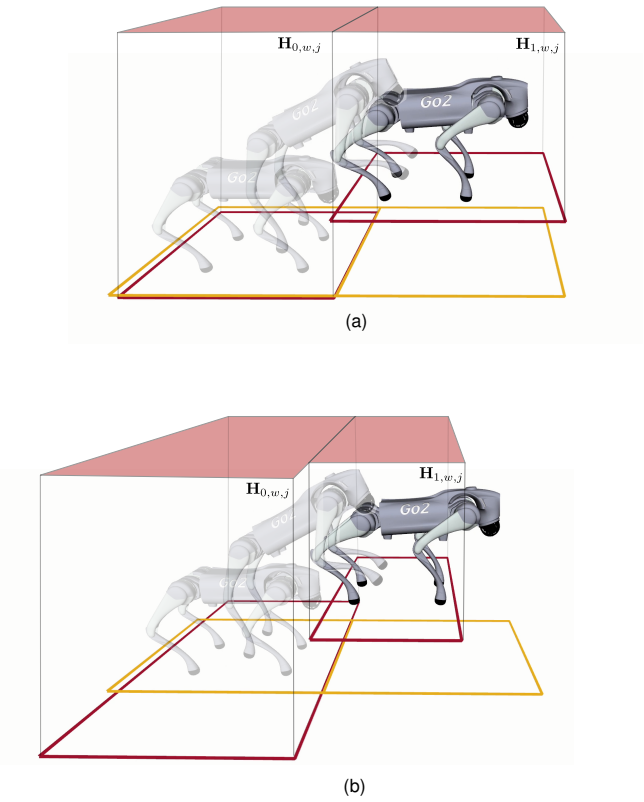


Figure 5. Demonstrations of (a) offline MICP for a skill transitioning from a flat terrain to a high terrain with predefined polygons; (b) online MICP for the same skill but with online terrain polygons.

we will explicitly elaborate which level is responsible in which scenario.

- *Symbolic-level obstacle avoidance.* When an obstacle is large enough that an entire abstracted cell becomes unsteppable, it is classified as the *obstacle* terrain class in the local symbolic abstraction (Sec. 3.1). The symbolic specification rules out transitions into such cells, and if an obstacle appears at runtime—as in the rebar-with-obstacle hardware experiment, where the obstacle is detected via motion capture and added to the local abstraction online—runtime symbolic re-synthesis (Sec. 6.2) finds a detour around it. No collision-region variables need to be added to the online MICP in this case.

- *MICP-level collision-free region selection (this subsection).* For smaller obstacles that the swing foot must clear within a single symbolic transition, we encode collision avoidance directly in the online MICP. Because the resulting trajectory is sent directly to the tracking controller, the quality of end-effector (EE) trajectories at this layer is critical for tracking performance. We introduce binary variables $\mathbf{H}_{s,w,j}$ that constrain the swing-foot trajectory of foot j to lie inside one of S pre-labeled collision-free convex polytopes at each time step. These free-space polytopes come from the same labeled-polygon segmentation pipeline as the steppable terrain, with a separate label set distinguishing free-space polytopes from steppable terrain polygons. In simulation, this mechanism is exercised in the *Unstructured–8* case with collision avoidance enabled (see Table 4), where the binary-variable count rises to 256.

We do not perform full-body, mesh-level collision checking against arbitrary geometry; the body footprint is implicitly handled by steppable-region selection keeping each foot inside a labeled terrain polygon. The formulation here trades fidelity for online tractability.

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. We emphasize that the collision-free region selection in Eq. (29) and the steppable-region selection in Eq. (6) handle different geometric primitives: the steppable-region constraint forces *stance-foot* placement onto a labeled steppable terrain polygon at the first stance time step, while the collision-free region constraint here keeps the *swing-foot* trajectory inside a free-space polytope across all time steps within a swing phase. The two are complementary and not redundant.

To enable larger swing foot clearance from the terrain, we also add constraints to force the swing foot height above a user-defined threshold h_{swing}, given the fixed contact timing. This constraint targets cleanly clearing the terrain during a transition—for example, when jumping onto a higher platform. While such clearance could in principle be enforced through the collision-free region selection alone, doing so would require a fine time-step discretization of the binary variables and would tend to yield trajectories that only marginally clear the terrain. The height threshold h_{swing} instead provides a direct, continuously enforced clearance margin that is straightforward to tune.

Note that the collision-avoidance and swing-foot constraints are also incorporated in the offline MICP as part of the feasibility-checking rules for the relevant terrain transitions. We highlight them here because their impact on online tracking performance is substantially more critical.

6.2 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 $\varphi_{\text{s}}^{\text{t}}$ in the specification φ (see Sec. 5.2) to $\varphi_{\text{s}}^{\text{t}} \wedge \square\varphi_{\text{disallow}}$, 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. 5.3.

7 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. 7.2).

7.1 Tracking Control Module

7.1.1 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 (Dai et al. 2014; Chignoli et al. 2021; Sleiman et al. 2021), 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 (Farshidian et al.), with a time horizon of one second and 100 knot points.

7.1.2 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) (Kuindersma et al. 2014) at 500 Hz. To improve the performance of agile motions such as leaping and jumping, centroidal momentum tracking (Wensing and Orin 2013) 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.

7.1.3 State Estimation The state estimator employs a linear Kalman filter similar to that in (Bledt et al. 2018), 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.

7.2 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:

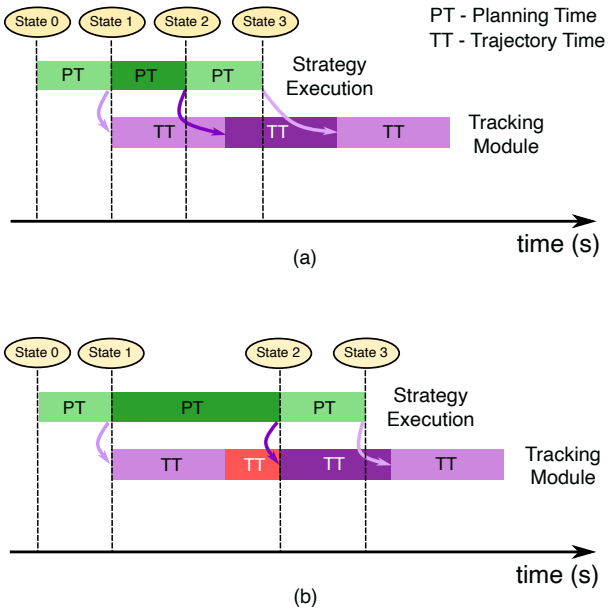


Figure 6. Coordination between strategy execution and tracking control module. (a) The MICP solving time (denoted by PT) for transitioning from State 1 to State 2 is shorter than the time horizon of the previous reference trajectory (denoted by TT), allowing the seamless appending for the new trajectory. (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.

7.2.1 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.

7.2.2 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.7.

7.2.3 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

Table 2. 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)

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)).

8 Simulation

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

8.1 Implementation Details

All MICP problems—including the offline gait-free MICP, the online gait-fixed MICP, the kinematic-feasibility re-targeting MICP, and the pure-MIP benchmark—are solved using Gurobi 9 (Gurobi Optimization, LLC 2025). The NMPC tracker is implemented with the OCS2 library (Farshidian et al.). All offline-synthesis, simulation, and online-planning experiments are run on a desktop workstation equipped with an Intel Core i9-13900K (24 cores) and 64 GB of RAM. Reported solve times reflect this desktop-class CPU and should be interpreted as such; onboard, low-power deployment is left as future work.

8.2 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. In our planner, we model both cases as point contact. The parameters for each robot are defined in Table 2 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 $\mathbf{I}$.

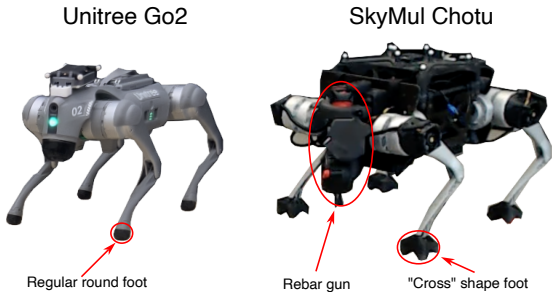


Figure 7. Hardware platforms used for experiments: Unitree Go2 and SkyMul Chotu (equipped with a rebar gun and "cross"-shaped feet).

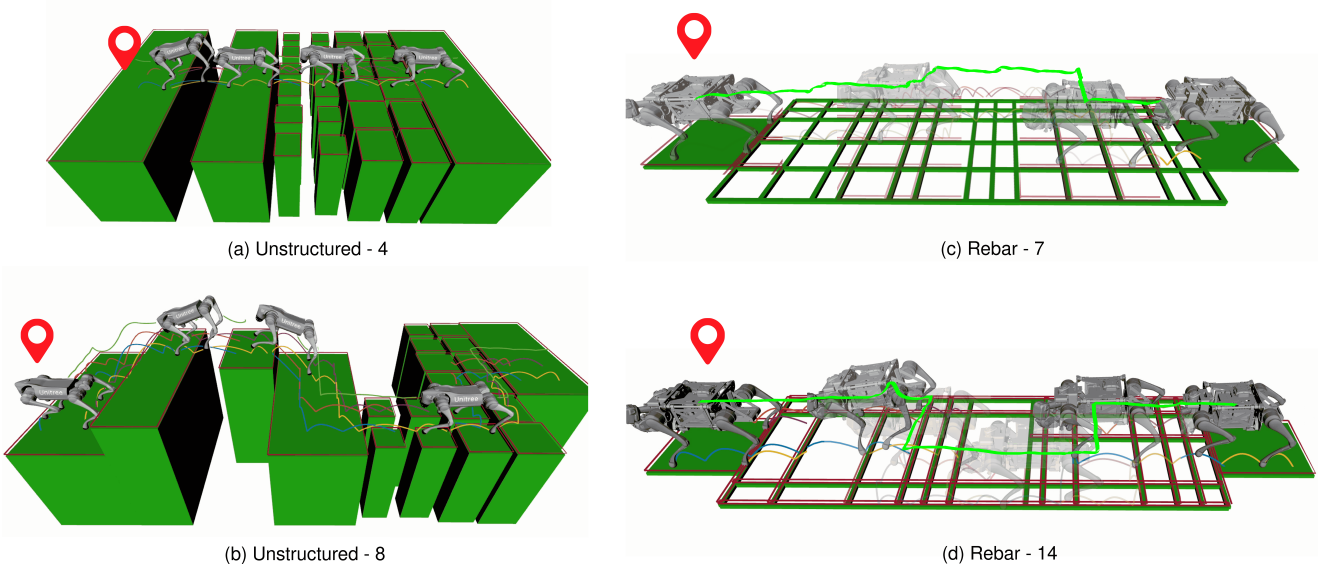
8.3 Simulation Environment Setup

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.

8.3.1 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.

8.3.2 Rebar Terrain Inspired by efforts to automate labor-intensive rebar tying tasks on construction sites (Asselmeier et al. 2024), we 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



(a) Unstructured - 4 (c) Rebar - 7

(b) Unstructured - 8 (d) Rebar - 14

Figure 8. Snapshots from online execution across the four simulation scenarios: (a) unstructured terrain with four terrain types, (b) unstructured terrain with the full eight-type abstraction, (c) rebar terrain with the 7-type abstraction, and (d) rebar terrain with the 14-type abstraction. Green trajectories in (c) and (d) are detours produced by runtime re-synthesis after online MICP solving failures.

Table 3. Offline Synthesis and Repair Time

Scenario	Grid Size	Terrain and Request State Pairs (Success/Total)	Number of Skills (Original/New/Total)	Symbolic Repair Time (s)	Feasibility Check Time (s) (Gait-Fixed/Gait-Free)
Unstructured - 4	3×3	169/169	18/13/64	8.07	18.97/449.94
	5×5	129/158	19/9/64	77.16	20.26/241.42
Unstructured - 8	3×3	250/264	19/20/256	14.90	22.92/382.63
	5×5	179/246	20/16/256	93.21	18.59/196.88
Rebar - 7	3×3	196/196	18/11/196	7.44	15.63/417.55
	5×5	196/196	18/10/196	21.63	14.83/374.89
Rebar - 14	3×3	237/237	20/29/784	10.34	21.92/701.38
	5×5	196/196	20/18/784	24.31	23.06/453.94

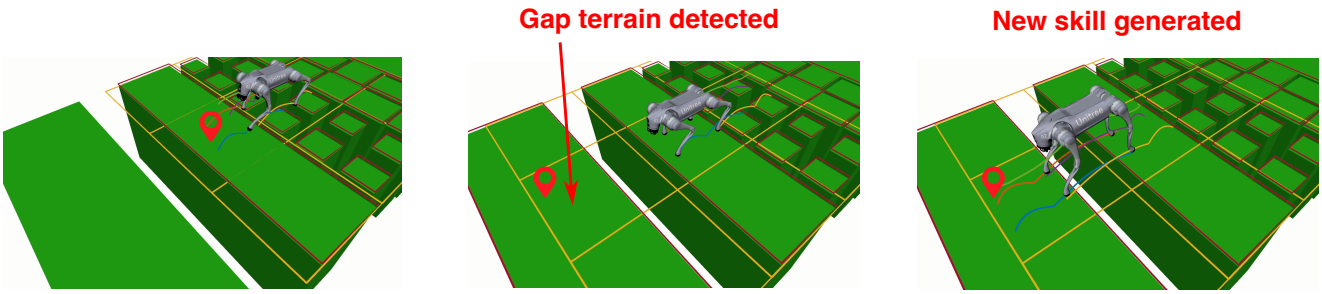
rebar sparsity randomly between 0.15 and 0.35 m, omitting *extreme sparse* regions. In contrast, the 14-type configuration in Fig. 8(d) includes sparsity ranging from 0.15 to 0.6 m, encompassing more complex and difficult combinations including *extreme sparse* types.

8.4 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 $\Sigma_{\text{terrain}}^{\text{poss}}$—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 $\Sigma_{\text{req}}^{\text{poss}}$ 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.

Table 3 reports (i) the number of terrain and request states pairs $(\sigma_{\text{terrain}}, \sigma_{\text{req}}) \in \Sigma_{\text{terrain}}^{\text{poss}} \times \Sigma_{\text{req}}^{\text{poss}}$, including both successful and total ones, (ii) the number of skills, including the original, the newly discovered, and the total possible ones, and (iii) offline repair and feasibility checking times, including both from gait-fixed and gait-free MICP. We first observe that the repair process successfully resolves 93.4% of the terrain and request state pairs. The remaining cases are deemed unrepairable due to either unreachable request states caused by obstacles or inherent physical infeasibility. As highlighted in red, the number of newly discovered skills is reduced by 71.6–97.6% compared to exhaustively checking all possible skills—each of which requires solving a computationally expensive gait-free MICP. On average, generating a new locomotion gait via gait-free MICP takes 27.23 s, with a maximum of 145.66 s. This demonstrates the effectiveness of offline repair in substantially reducing the number of costly gait-free MICP solves.

To investigate the scalability of offline repair, we perform repair across various sizes of the terrain and request states. As shown in Fig. 10, the repair runtime grows linearly in the size of the terrain and request states for both 3×3 and 5×5 grids. While fewer terrain and request states handled offline reduce checked time, they may lead to more frequent unforeseen states during online execution. Moreover, we note that the slope of 5×5 cases in Fig. 10 are steeper than those of 3×3 as a result of the increased state space. On average, repair takes 478.1% more time for 5×5 grids than 3×3 for one terrain and request states pair. In practice, the perception range and the robot size limit the grid size, so we do not test beyond 5×5 grids, though the user can adjust the resolution.



(a) Before a gap (b) Encountering a gap (c) After online repair

Figure 9. Runtime symbolic repair triggered by an unforeseen *gap* state in the *Unstructured-4* / 3×3 scenario, in which gap states are deliberately excluded offline. Repair proposes the missing transition and gait-free MICP synthesizes a leaping skill, taking 12.36 s total (0.18 s symbolic repair + 12.18 s gait-free MICP).

Table 4. 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

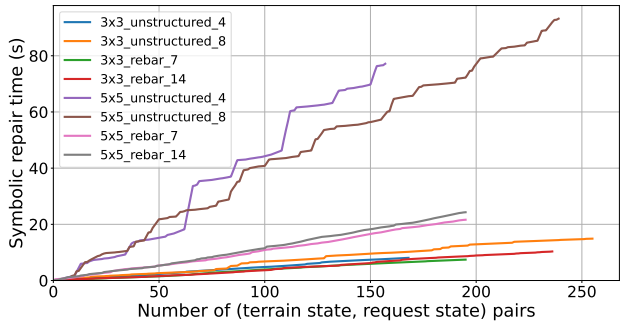


Figure 10. Offline symbolic-repair time versus the number of terrain–request state pairs for the 3×3 and 5×5 abstractions. Repair time grows approximately linearly in both cases; per state pair, the 5×5 grid takes 478.1% more time than the 3×3 grid.

8.5 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 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 4 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).

8.6 Runtime Repair Case Study

As a case study to demonstrate the framework's capability to handle unforeseen terrains and online failures, we highlight the following run-time scenarios that trigger runtime repair or re-synthesis.

8.6.1 Unforeseen Terrain States With limited prior knowledge of the global terrain map, the terrain states provided during offline synthesis may be insufficient when encountering new terrain configurations. Fig. 9 illustrates this in an *Unstructured - 4 types* scenario with a 3×3 grid size, where terrain states involving *gap* terrain are deliberately excluded from the offline phase. When the robot reaches a new terrain state that contains a *gap*, runtime repair is triggered for the current terrain and request states. A new skill with a leaping gait is generated to successfully cross the gap. The runtime repair takes 12.36 s in total, of which the symbolic-repair stage accounts for 0.18 s and the gait-free MICP for the new leaping gait accounts for 12.18 s—confirming that the dominant cost lies in the dynamic-feasibility check, not the symbolic search.

8.6.2 Solving Failure Another common runtime failure arises from MICP solving failures. Due to discrepancies

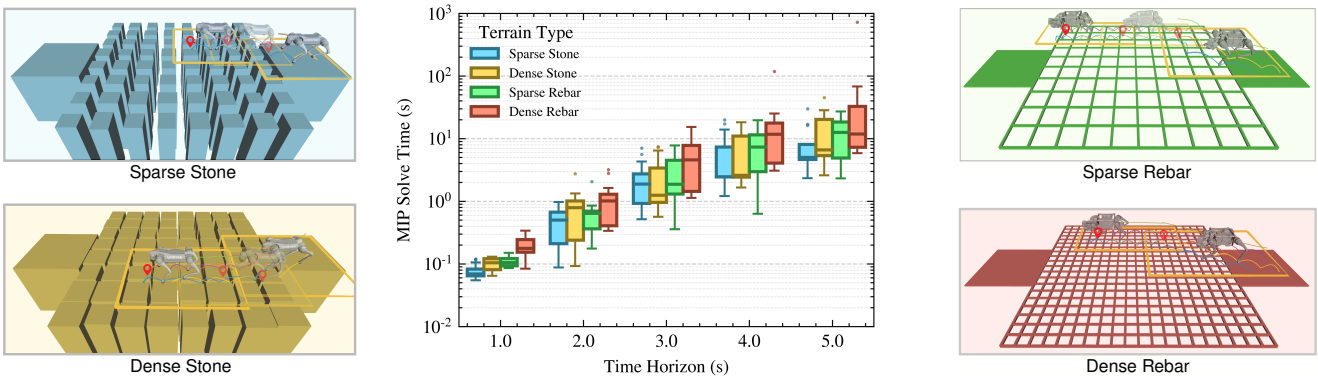


Figure 11. MIP solve time across four terrains—sparse stone (blue), dense stone (yellow), sparse rebar (green), dense rebar (red)—and horizons of 1–5 s, as box plots over 10 seeded waypoints per scenario. In each terrain subfigure, selected traversal snapshots at planning horizons of 1 s (blue), 2 s (yellow), 4 s (green), and 5 s (red) are overlaid for visualization.

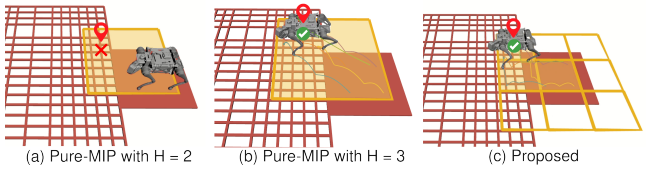


Figure 12. Representative dense rebar trial after randomly removing terrains for benchmarking different baselines: (a) Pure-MIP with $H{=}2$ s; (b) Pure-MIP with $H{=}3$ s; (c) Proposed approach with 3×3 grid.

between the predefined polygons used during offline synthesis and the actual online terrains—such as variations in shape and size—a skill deemed feasible offline may become infeasible when executed online, even if classified under the same symbolic transition. The detours shown in Fig. 8(c) and (d) illustrate the re-synthesis process triggered by solving failures in the 7-type and 14-type rebar cases. Solving failures occur more frequently in the rebar scenarios than in the unstructured ones, due to the higher uncertainty in the shapes and sizes of online rebar polygons. Re-synthesis takes 0.11 s and 0.07 s respectively for the two cases. Neither case triggers full runtime repair, because the existing skill set suffices to navigate to the goal under the failed transition.

8.7 Benchmark: Pure-MIP Planner

To evaluate how the symbolic-continuous planner separation 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. The pure-MIP approach can also be treated as an ablation study for the symbolic planning. 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. 11 with specific colors.

To make the comparison precise: the pure-MIP baseline uses the *same* convex relaxations of the dynamics (Eq. (3)), the *same* linearized friction-cone and fixed-Jacobian kinematic constraints, and the *same* cost weights as our online gait-fixed MICP. The only differences are (i) the planner solves a single long-horizon gait-fixed MICP without symbolic-level decomposition, and (ii) to evaluate performance across different planning horizons, the local planning region is also expanded proportionally to the horizon to encompass all relevant terrain polygons, rather than being limited to the cells adjacent to a symbolic transition. The same Gurobi solver and settings are used in both cases.

The pure-MIP planner uses an *adaptive* local region whose maximum size scales with the planning horizon H (in seconds): along each axis the region spans up to $\mathrm{gs} + (\mathrm{gs}/2) \cdot H$, where gs is the symbolic-cell grid spacing, with a floor of 1.5 gs corresponding to the $H{=}1$ s setting. So the $H{=}2$ s column reads as a like-for-like comparison for an equivalent two-cell transition footprint, while each additional second of horizon adds half a cell of reach. As shown in Fig. 11, the region further shrinks when the goal lies within the maximum range, avoiding redundant incorporation of terrain polygons in each MIP solve. In contrast, our proposed planner always plans against a fixed $3{\times}3$ symbolic grid centered on the robot, decoupling the per-MIP problem size from the total traversal distance. We omit the gait-free MICP from the benchmark because its solve time becomes prohibitively long once $H > 2$ s; the locomotion gait is repeated cyclically on a 1 s trotting pattern, and the planner receives a global waypoint and incrementally computes local waypoints as in the proposed framework.

Fig. 11 also reports the per-MIP solve time as box plots aggregated over 10 randomly sampled waypoints per scenario. Within the same terrain, the solve time grows almost exponentially with the horizon, and the dense variants consistently take longer than their sparse counterparts because more terrain polygons enter the MIP. The $H{=}1$ s setting is too myopic to commit to scenarios that require multi-step planning, while $H \geq 4$ s pushes the median solve into multiple seconds even for the easier scenarios. The $H{=}2$ s and $H{=}3$ s horizons sit on the knee of this trade-off curve, balancing optimality and per-step compute, and we therefore adopt them as the two reference pure-MIP configurations for the end-to-end benchmark below.

To assess closed-loop behavior beyond a per-MIP solve time, we run an end-to-end comparison on the dense rebar scenario, where the pure-MIP planner is most stressed. For each method, we dispatch 10 seeded random waypoints;

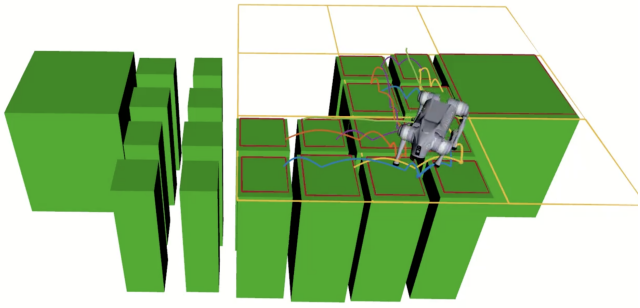


Figure 13. Demonstration of the robot executing multiple turning behaviors to continuously progress toward the global waypoint on terrain arranged in a zig-zag pattern.

between trials the robot is reset back to the launch pose, and we randomly remove a fraction of the rebar polygons as shown in Fig. 12 to add randomness and make the scenario more challenging. Identical random seeds across methods ensure all baselines see the byte-for-byte same waypoints and terrain, so any difference in outcome reflects the planner alone. Fig. 12 also shows one representative instance on which the pure-MIP planner with short horizons (i.e., 1 or 2 s) stalls—the planning region is too limited for the solver to discover a feasible routing path, whereas the proposed planner routes through a sequence of cells.

Table 5 reports the success rate, mean traversal time of successful trials, and the mean per-module active computation per trial—broken down into MIP transition solving, the pre-flight MIP feasibility check (FC), reactive synthesis (strategy query plus strategy reload), and runtime repair (online specification regeneration). In this scenario, the proposed planner attains 90% success rate with $18.5\,\mathrm{s}$ mean traversal time, matching the $H{=}3\,\mathrm{s}$ pure-MIP success rate (90%) at roughly three quarters of the traversal time and at less than half the MIP transition cost ($5.26\,\mathrm{s}$ vs. $11.54\,\mathrm{s}$ per trial). The shorter $H{=}2\,\mathrm{s}$ pure-MIP setting reaches only 60%, indicating that its horizon is too myopic to recover from the gaps. The two ablation rows confirm the value of the online additions introduced in this work: disabling the FC drops the success rate substantially in both planners ($60\% \rightarrow 20\%$ for pure-MIP at $H{=}2\,\mathrm{s}$ and $90\% \rightarrow 60\%$ for the proposed planner); disabling the delay-aware coordination (DAC) preserves the 90% success rate of the proposed planner but inflates the mean traversal time from $18.5\,\mathrm{s}$ to $23.4\,\mathrm{s}$, because each symbolic transition now must wait for the previous trajectory to be fully tracked before the next MIP solve can start.

8.8 Preliminary Study: Yaw Command Extension

This subsection demonstrates that the proposed framework can be extended to non-zero yaw waypoints in principle, but we report the result as a *preliminary feasibility study* only. We do not claim a full demonstration of generalizability of any angular movements. Specifically: (i) the symbolic abstraction in Sec. 3.1 grows combinatorially when yaw is added as a discretized input, and we have not characterized how the manager and symbolic-repair scaling numbers reported in Sec. 8 change with yaw discretization; (ii) the angular-momentum coupling neglected by the convex-dynamics simplification in Eq. (3) becomes more relevant once yaw is treated as a planned variable, especially for fast turns; (iii) we report simulation results only—no hardware deployment of the yaw extension.

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 demonstrate the framework's modularity, we provide a preliminary extension to the existing abstraction of the robot state to allow yaw 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. 5.2, 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. 13, 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.

9 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.

Table 5. End-to-end benchmark on the dense rebar scenario (10 seeded waypoints, identical seed and terrain across methods, with randomly removed rebars per trial to add randomness). Traversal time and per-module times are mean values computed over successful trials only. Reactive synthesis combines the per-step strategy query and the per-spec-switch strategy reload. Dashes denote modules that do not exist in the corresponding method or are disabled by the ablation.

Method	Success rate	Traversal time (s)	MIP trans. solving (s)	MIP feas. check (s)	Reactive synth. (query + load, s)	Runtime repair (s)
Pure MIP (H=2 s)	60% (6/10)	14.76	3.04	0.21	–	–
Pure MIP (H=3 s)	90% (9/10)	24.73	11.54	0.19	–	–
Pure MIP (H=2 s), no FC	20% (2/10)	9.00	1.38	–	–	–
Pure MIP (H=3 s), no FC	40% (4/10)	14.23	5.17	–	–	–
Proposed	90% (9/10)	18.48	5.26	0.19	0.29	0.03
Proposed, no FC	60% (6/10)	20.08	5.85	–	0.33	0.05
Proposed, no DAC	90% (9/10)	23.43	5.11	0.27	0.28	0.04

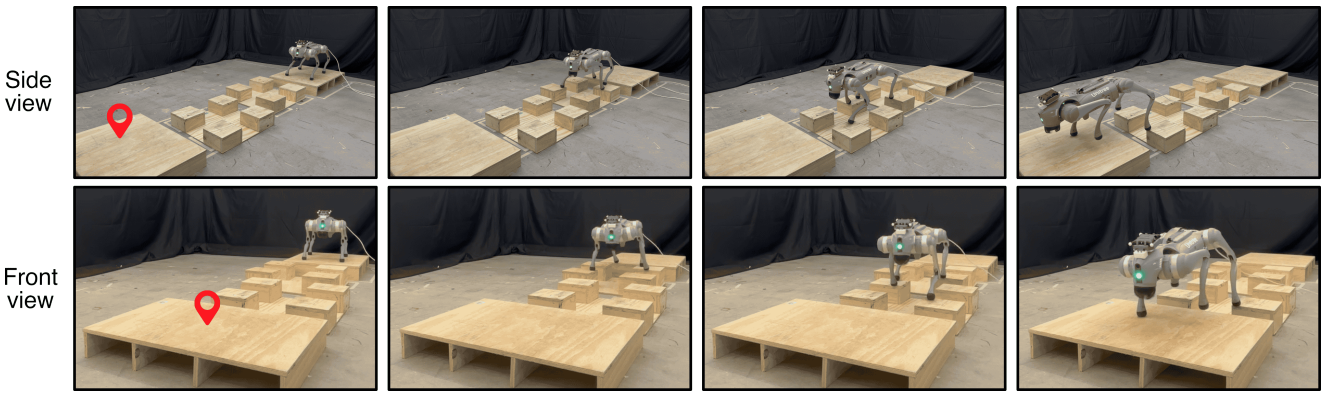


Figure 14. Hardware experiment on the first unstructured-terrain scenario: sparse stepping stones (0.18–0.23 m apart) interleaved with flat regions. Side and front views highlight the lateral misalignment of the stones; the online MICP produces a sideways trajectory and a 3 s trotting gait is used throughout.

Second, we illustrate the framework's assured safety through obstacle avoidance and selection of dynamically feasible gaits, benchmarked against a heuristic-based planner.

9.1 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 × 4-foot rebars and six 1/2-inch × 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.

9.2 Unstructured Terrain

As shown in Fig. 14, 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 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. 15, 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. 16. 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.

9.3 Rebar Terrain

For the rebar terrain scenarios shown in Fig. 17, 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

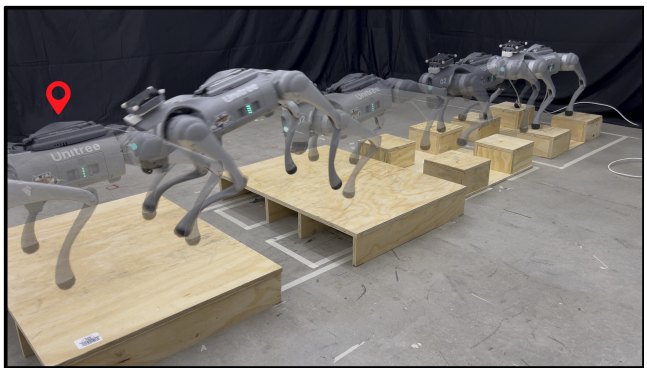


Figure 15. Hardware experiment demonstration for the second unstructured terrain scenario with sparse stepping stones, flat, and gap terrains.

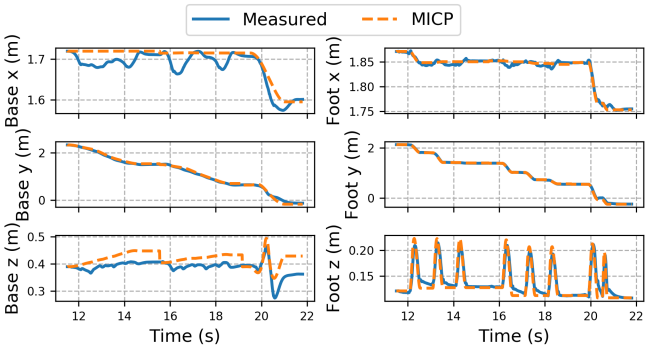


Figure 16. Tracking performance of the base and front-left foot positions: comparison between measured states and MICP-generated reference trajectories.

waypoint. In the second case (bottom row), an obstacle is added to the environment but only detected during the online execution phase. In our hardware setup, the obstacle's location is not estimated by an autonomous perception stack but rather provided through motion-capture-aided manual annotation: when the obstacle enters the robot's local field of view, its position is added to the online terrain set and the affected cell is classified as the *obstacle* terrain class at the symbolic level—i.e., obstacle avoidance in this scenario is handled entirely at the symbolic level (Sec. 3.1), not through the MICP-level collision-region variables $\mathbf{H}_{s,w,j}$ in Eq. (29). This setup is explicit about what is and is not validated—the planning and runtime-repair pipeline is exercised end-to-end, while the terrain-segmentation problem itself is not. 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. We note that the gait timings adopted on hardware—3 s trotting, 1.5 s leaping, 2.25 s static walking—are conservative compared to the most agile motions reported in the legged-locomotion literature; however, this locomotion gait setup still verifies the framework's gait-switching, online MICP, and runtime-repair mechanisms, which is the validation target of this article. Aggressive, high-speed gaits would require revisiting the convex-dynamics simplification (Eq. (3)) discussed in Sec. 6.

Table 6. Online Gait-Fixed MICP Solving Time for Hardware

Scenario	Polygons	Min (s)	Mean (s)	Std (s)	Max (s)
Unstructured - Scene 1	6	0.81	4.04	2.50	7.86
Unstructured - Scene 2	5	0.16	0.53	0.25	0.96
Rebar - Scene 1	8	3.88	6.54	2.20	9.26
Rebar - Scene 2	7.5	3.60	9.13	5.50	19.23
Rebar - Scene 3	6.33	2.45	7.37	4.41	14.7

9.4 Benchmark: Heuristics-Based Planner

We create a third rebar scenario, shown in Fig. 18, 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 (Fankhauser et al. 2018; Kim et al. 2020), 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).

We chose this nearest-feasible-foothold heuristic because it isolates the value of feasibility-aware, longer-horizon planning over a single-step heuristic, and because such heuristics remain widely used in deployed perceptive-locomotion stacks due to its computational efficiency. Comparisons against more sophisticated baselines are discussed as future work in Sec. 10.

Without re-performing the offline synthesis, our framework directly uses the results from Sec. 9.3. 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.

9.5 Computational Time

We report the hardware computational time in Table 6. 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 (Marcucci et al. 2024; Lin 2024) remains an important direction for future work.

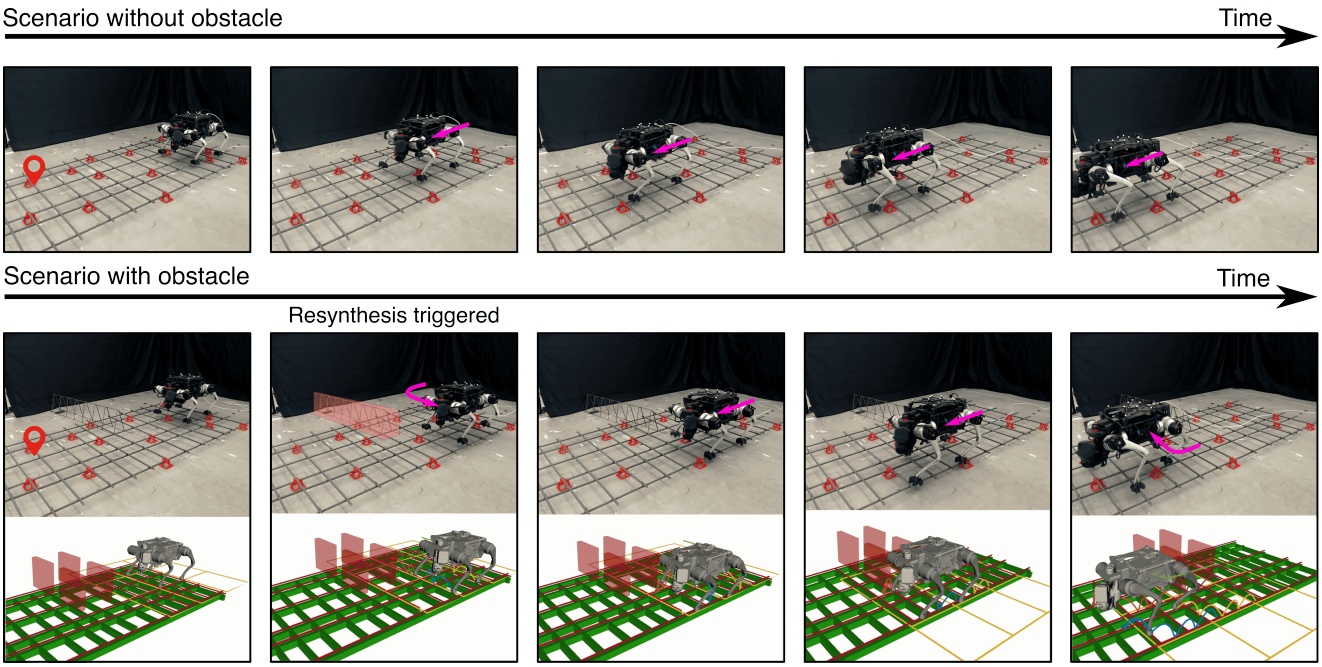


Figure 17. Execution results in rebar terrain without (top row) and with obstacle (bottom row). In the obstacle case, runtime resynthesis is triggered to generate a detour strategy and ensure successful traversal.

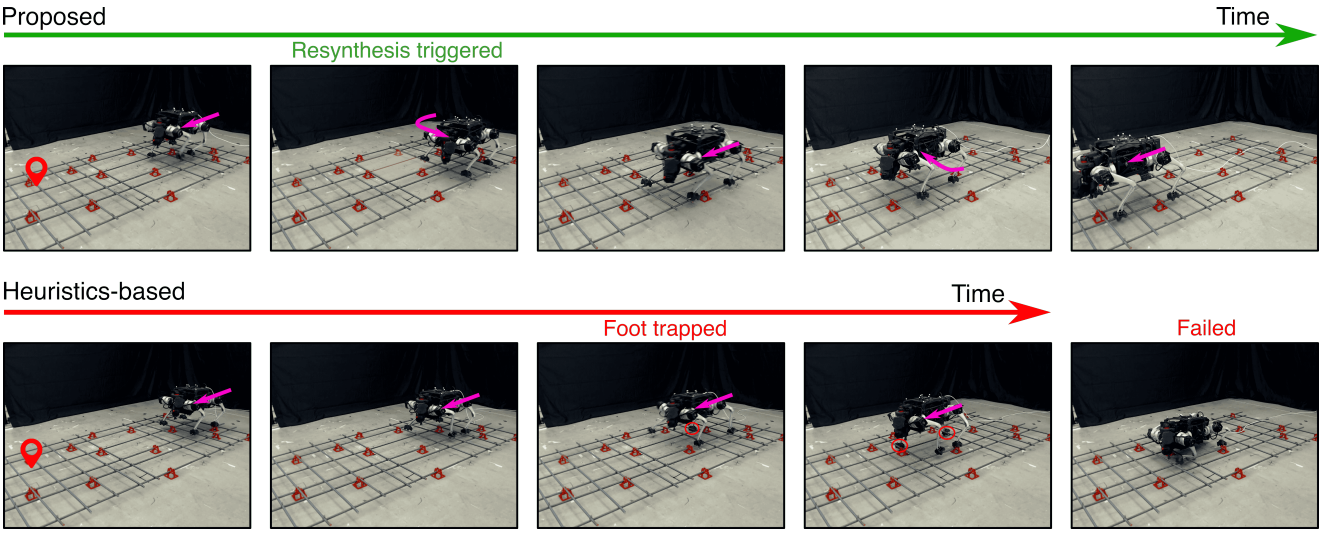


Figure 18. Comparison between the proposed method and a heuristics-based planner in an extreme sparse rebar scenario. The proposed method performs resynthesis and succeeds, while the heuristics-based planner fails due to foot trapping.

10 Discussion

10.1 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. 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.

10.2 Optimality

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

10.3 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 (Miki et al. 2022b). 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. In this work we deliberately treated the perception layer as out of scope and used motion-capture-aided manual labeling on hardware (Sec. 9); a fully autonomous deployment will need to address how perception uncertainty—segmentation noise, polygon misalignment, and per-frame perception delay—propagates through the MICP feasibility certificates and the symbolic-level realizability guarantees. In the current framework, runtime symbolic repair already provides a partial fallback: when perceived terrain causes the symbolic specification to become unrealizable, repair triggers and synthesizes new skills. Quantifying the resilience of this fallback under realistic perception noise is an important next step.

10.4 Reactivity and Update Frequencies

Across the experiments reported in this article, the symbolic strategy automaton evolves at the timescale of one symbolic transition, which corresponds to one online MICP solve—typically hundreds of milliseconds to a few seconds (Tables 4 and 6). Two complementary directions exist for tightening this loop in future work.

First, more frequent MICP updates—e.g., re-solving the online gait-fixed MICP at every NMPC tick rather than once per symbolic transition—are easily adaptable in the proposed framework: the delay-aware coordination scheme already supports asynchronous MICP solves of arbitrary duration, so any speed-up of the underlying mixed-integer solver, or any per-transition simplification, translates directly into a higher MICP update rate. The current limit is therefore the MICP solve time itself rather than any architectural constraint.

Second, more frequent symbolic-level planning is also achievable but requires careful design of the symbolic abstraction. Higher-frequency symbolic decisions would generally call for a finer abstraction—more atomic propositions per cell, finer terrain or robot-state granularity, or a larger skill repertoire. Reactive synthesis, however, scales exponentially in the number of variables in the specification (Bloem et al. 2012), so any finer abstraction must be balanced against synthesis cost. The high-level manager and symbolic repair adopted in this article already mitigate this scalability barrier; pushing toward truly real-time symbolic planning is an interesting future direction at the abstraction level rather than a limit of the framework itself.

10.5 Stronger Baseline Comparisons as Future Work

The experimental validation in this article compares against a pure-MIP planner and a nearest-feasible-foothold heuristic. Two stronger comparison points that we did not implement here remain valuable future targets: (i) optimization-based planners such as TOWR-style phase-based end-effector parameterization (Winkler et al. 2018), which jointly optimize gait timing and footholds in continuous time; and (ii) end-to-end perceptive reinforcement-learning controllers, in particular parkour-style policies such as (Zhuang et al. 2023; Hoeller et al. 2023; Cheng et al. 2024; Luo et al. 2024). A fair comparison against these two methods requires either equipping them with an additional discrete terrain selection mechanism or a comparable high-level navigation layer, which we did not pursue in the present validation.

10.6 Extension to Bipedal and Other Platforms

This article focuses on quadrupedal locomotion. The framework's symbolic-level machinery—terrain abstraction, GR(1) specifications, symbolic manager, symbolic repair, runtime repair—is platform-agnostic and applicable directly to other systems such as bipedal robots. The MICP layer requires more careful adaptation: angular-momentum modulation becomes critical even at moderate speeds—a regime where the convex-dynamics simplification adopted here (Eq. (3)) becomes a stronger limitation. Generalizing to humanoids would also require revisiting the kinematic-feasibility re-targeting MICP to account for arm-swing and balance constraints. We see this as a future direction.

11 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.

Acknowledgements

We thank Eohan George, Suphakorn Lertruchtkul, and Jane Ivanova for their hardware support on Chotu and contributions to rebar scenario design, and Pengyuan Shu for constructing the real-world terrains. The authors would also like to thank Hemanth Surya for the early exploration of the terrain segmentation.

Author Contributions

All authors contributed to the study. Ziyi Zhou: Conceptualization, Methodology, Formal analysis, Investigation, Software, Project administration, Writing - original draft. Qian Meng: Methodology, Formal analysis, Investigation Software, Writing - original draft. Hadas Kress-Gazit: Methodology, Supervision, Writing – review and editing. Ye Zhao: Funding acquisition, Resources, Methodology, Supervision, Writing – review and editing. All authors reviewed and approved the final version of the manuscript.

Ethical Considerations

Not applicable.

Consent to Participate

Not applicable.

Consent for Publication

Not applicable.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported through a sponsored research grant from SkyMul Inc. and a Georgia Tech IRIM/IPaT Aware Home Seed Grant. This work was also supported in part by the National Science Foundation (NSF) under Grant #CMMI-2144309, and Grant #FRR-2328254, Office of Naval Research (ONR) under Grant N000142312223, and in part by the United States Department of Agriculture (USDA) under Grant 2023-67021-41397.

References

Aceituno-Cabezas B, Mastalli C, Dai H, Focchi M, Radulescu A, Caldwell DG, Cappelletto J, Grieco JC, Fernández-López G and Semini C (2017) Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters* 3(3): 2531–2538.

Acosta B and Posa M (2023) Bipedal walking on constrained footholds with mpc footstep control. In: *2023 IEEE-RAS 22nd International Conference on Humanoid Robots*. IEEE, pp. 1–8.

Agarwal A, Kumar A, Malik J and Pathak D (2023) Legged locomotion in challenging terrains using egocentric vision. In: *Conference on robot learning*. PMLR, pp. 403–415.

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

Amatucci L, Kim JH, Hwangbo J and Park HW (2022) Monte carlo tree search gait planner for non-gaited legged system control. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4701–4707.

Asselmeier M, Ivanova J, Zhou Z, Vela PA and Zhao Y (2024) Hierarchical experience-informed navigation for multi-modal quadrupedal rebar grid traversal. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8065–8072.

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

Baier C and Katoen JP (2008) *Principles of model checking*. MIT press.

Bhatia A, Kavraki LE and Vardi MY (2010) Sampling-based motion planning with temporal goals. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2689–2696.

Bledt G, Powell MJ, Katz B, Di Carlo J, Wensing PM and Kim S (2018) 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, pp. 2245–2252.

Bloem R, Jobstmann B, Piterman N, Pnueli A and Saar Y (2012) Synthesis of reactive (1) designs. *Journal of Computer and System Sciences* 78(3): 911–938.

Boussema C, Powell MJ, Bledt G, Ijspeert AJ, Wensing PM and Kim S (2019) Online gait transitions and disturbance recovery for legged robots via the feasible impulse set. *IEEE Robotics and automation letters* 4(2): 1611–1618.

Bretl T (2006) Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *The International Journal of Robotics Research* 25(4): 317–342.

Chen C, Culbertson P, Lepert M, Schwager M and Bohg J (2021) 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, pp. 8262–8268.

Cheng X, Shi K, Agarwal A and Pathak D (2024) Extreme parkour with legged robots. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 11443–11450.

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

Chignoli M, Morozov S and Kim S (2022) Rapid and reliable quadruped motion planning with omnidirectional jumping. In: *International Conference on Robotics and Automation*. IEEE, pp. 6621–6627.

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

Dai H, Izatt G and Tedrake R (2019) Global inverse kinematics via mixed-integer convex optimization. *The International Journal of Robotics Research* 38(12-13): 1420–1441.

Dai H, Valenzuela A and Tedrake R (2014) Whole-body motion planning with centroidal dynamics and full kinematics. In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 295–302.

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

Deits R and Tedrake R (2014) Footstep planning on uneven terrain with mixed-integer convex optimization. In: *2014 IEEE-RAS international conference on humanoid robots*. IEEE, pp. 279–286.

Di Carlo J, Wensing PM, Katz B, Bledt G and Kim S (2018) 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, pp. 1–9.

Ding Y, Li C and Park HW (2020) 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, pp. 3998–4005.

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

Drnach L, Zhang JZ and Zhao Y (2022) Mediating between contact feasibility and robustness of trajectory optimization through chance complementarity constraints. *Frontiers in Robotics and AI* 8: 785925.

Drnach L and Zhao Y (2021) Robust trajectory optimization over uncertain terrain with stochastic complementarity. *IEEE Robotics and Automation Letters* 6(2): 1168–1175.

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

Fainekos GE, Loizou SG and Pappas GJ (2006) Translating temporal logic to controller specifications. In: *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, pp. 899–904.

Fankhauser P, Bjelonic M, Bellicoso CD, Miki T and Hutter M (2018) Robust rough-terrain locomotion with a quadrupedal robot. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 5761–5768.

Farshidian F et al. (????) OCS2: An open source library for optimal control of switched systems. [Online]. Available: https://github.com/leggedrobotics/ocs2.

Feng S, Zhou Z, Smith JS, Asselmeier M, Zhao Y and Vela PA (2023) Gpf-bg: A hierarchical vision-based planning framework for safe quadrupedal navigation. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1968–1975.

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

Fey N, Frei RJ and Wensing PM (2024) 3d hopping in discontinuous terrain using impulse planning with mixed-integer strategies. *IEEE Robotics and Automation Letters* .

Gangapurwala S, Geisert M, Orsolino R, Fallon M and Havoutis I (2022) Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *IEEE Transactions on Robotics* 38(5): 2908–2927.

Garrett CR, Chitnis R, Holladay R, Kim B, Silver T, Kaelbling LP and Lozano-Pérez T (2021) Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems* 4(1): 265–293.

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

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

Grandia R, Taylor AJ, Ames AD and Hutter M (2021) 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, pp. 8352–8358.

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

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

Gurobi Optimization, LLC (2025) Gurobi Optimizer Reference Manual. https://www.gurobi.com.

Hauser K, Bretl T and Latombe JC (2005) Non-gaited humanoid locomotion planning. In: *5th IEEE-RAS International Conference on Humanoid Robots, 2005.* IEEE, pp. 7–12.

Hoeller D, Rudin N, Sako D and Hutter M (2023) Anymal parkour: Learning agile navigation for quadrupedal robots. *arXiv preprint arXiv:2306.14874* .

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

Jenelten F, Grandia R, Farshidian F and Hutter M (2022) Tamols: Terrain-aware motion optimization for legged systems. *IEEE Transactions on Robotics* 38(6): 3395–3413.

Jenelten F, Miki T, Vijayan AE, Bjelonic M and Hutter M (2020) Perceptive locomotion in rough terrain–online foothold optimization. *IEEE Robotics and Automation Letters* 5(4): 5370–5376.

Jiang J, Coogan S and Zhao Y (2023a) Abstraction-based planning for uncertainty-aware legged navigation. *IEEE Open Journal of Control Systems* .

Jiang X, Chi W, Zheng Y, Zhang S, Ling Y, Xu J and Zhang Z (2023b) Locomotion generation for quadruped robots on challenging terrains via quadratic programming. *Autonomous Robots* 47(1): 51–76.

Kalakrishnan M, Buchli J, Pastor P, Mistry M and Schaal S (2010) Fast, robust quadruped locomotion over challenging terrain. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2665–2670.

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

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

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

Kolter JZ, Rodgers MP and Ng AY (2008) A control architecture for quadruped locomotion over rough terrain. In: *2008 IEEE International Conference on Robotics and Automation*. IEEE, pp. 811–818.

Kress-Gazit H, Fainekos GE and Pappas GJ (2009) Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics* 25(6): 1370–1381.

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

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

Kulgod S, Chen W, Huang J, Zhao Y and Atanasov N (2020) Temporal logic guided locomotion planning and control in cluttered environments. In: *American Control Conference*. IEEE, pp. 5425–5432.

Le Cleac'h S, Howell TA, Yang S, Lee CY, Zhang J, Bishop A, Schwager M and Manchester Z (2024) Fast contact-implicit model predictive control. *IEEE Transactions on Robotics* .

Liao Q, Li Z, Thirugnanam A, Zeng J and Sreenath K (2023) 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, pp. 2723–2730.

Lin X (2024) Accelerate hybrid model predictive control using generalized benders decomposition. *arXiv preprint arXiv:2406.00780* .

Liu J, Ozay N, Topcu U and Murray RM (2013) Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Transactions on Automatic Control* 58(7): 1771–1785.

Luo S, Li S, Yu R, Wang Z, Wu J and Zhu Q (2024) Pie: Parkour with implicit-explicit learning framework for legged robots. *IEEE Robotics and Automation Letters* 9(11): 9986–9993.

Magana OAV, Barasuol V, Camurri M, Franceschi L, Focchi M, Pontil M, Caldwell DG and Semini C (2019) Fast and continuous foothold adaptation for dynamic locomotion through cnns. *IEEE Robotics and Automation Letters* 4(2): 2140–2147.

Maly MR, Lahijanian M, Kavraki LE, Kress-Gazit H and Vardi MY (2013) Iterative temporal motion planning for hybrid systems in partially unknown environments. In: *Proceedings of the 16th international conference on Hybrid systems: computation and control*. pp. 353–362.

Marcucci T, Umenberger J, Parrilo P and Tedrake R (2024) Shortest paths in graphs of convex sets. *SIAM Journal on Optimization* 34(1): 507–532.

Margolis GB, Chen T, Paigwar K, Fu X, Kim D, Kim Sb and Agrawal P (2022) Learning to jump from pixels. In: *Proceedings of the 5th Conference on Robot Learning*. PMLR, pp. 1025–1034.

Mastalli C, Havoutis I, Winkler AW, Caldwell DG and Semini C (2015) On-line and on-board planning and perception for quadrupedal locomotion. In: *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, pp. 1–7.

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

Migimatsu T and Bohg J (2020) Object-centric task and motion planning in dynamic environments. *IEEE Robotics and Automation Letters* 5(2): 844–851.

Miki T, Lee J, Hwangbo J, Wellhausen L, Koltun V and Hutter M (2022a) Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science robotics* 7(62): eabk2822.

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

Mordatch I, Todorov E and Popović Z (2012) Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (ToG)* 31(4): 1–8.

Muenprasitivej K, Jiang J, Shamsah A, Coogan S and Zhao Y (2024) 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, pp. 11264–11271.

Muenprasitivej K, Zhao Y and Chou G (2025) Probabilistically-safe bipedal navigation over uncertain terrain via conformal prediction and contraction analysis.

Norby J and Johnson AM (2020) Fast global motion planning for dynamic legged robots. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 3829–3836.

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

Pacheck A, Moarref S and Kress-Gazit H (2020) Finding missing skills for high-level behaviors. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 10335–10341.

Park HW, Wensing PM and Kim S (2015) Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. In: *Robotics: Science and Systems*.

Piterman N, Pnueli A and Sa'ar Y (2006) Synthesis of reactive (1) designs. In: *Verification, Model Checking, and Abstract Interpretation: 7th International Conference, VMCAI 2006, Charleston, SC, USA, January 8-10, 2006. Proceedings 7*. Springer, pp. 364–380.

Pnueli A and Rosner R (1989) On the synthesis of a reactive module. In: *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. pp. 179–190.

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

Posa M, Cantu C and Tedrake R (2014) A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research* 33(1): 69–81.

Ren J, Lin X, Mineyev R, Feigh KM, Coogan S and Zhao Y (2025) Accelerating signal-temporal-logic-based task and motion planning of bipedal navigation using benders decomposition. *arXiv preprint arXiv:2508.13407* .

Risbourg F, Corbères T, Léziart PA, Flayols T, Mansard N and Tonneau S (2022) Real-time footstep planning and control of the solo quadruped robot in 3d environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 12950–12956.

Shamsah A, Gu Z, Warnke J, Hutchinson S and Zhao Y (2023) Integrated task and motion planning for safe legged navigation

in partially observable environments. *IEEE Transactions on Robotics* .

Shamsah A, Jiang J, Yoon Z, Coogan S and Zhao Y (2025) 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, pp. 12352–12358.

Shirai Y, Lin X, Mehta A and Hong D (2021) 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, pp. 7533–7539.

Shirai Y, Lin X, Schperberg A, Tanaka Y, Kato H, Vichathorn V and Hong D (2022) 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, pp. 13563–13570.

Sleiman JP, Farshidian F and Hutter M (2023) Versatile multicontact planning and control for legged loco-manipulation. *Science Robotics* 8(81): eadg5014.

Sleiman JP, Farshidian F, Minniti MV and Hutter M (2021) A unified mpc framework for whole-body dynamic locomotion and manipulation. *IEEE Robotics and Automation Letters* 6(3): 4688–4695.

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

Tonneau S, Del Prete A, Pettré J, Park C, Manocha D and Mansard N (2018) An efficient acyclic contact planner for multiped robots. *IEEE Transactions on Robotics* 34(3): 586–601.

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

Toussaint M (2015) Logic-geometric programming: An optimization-based approach to combined task and motion planning. In: *IJCAI*. pp. 1930–1936.

Toussaint MA, Allen KR, Smith KA and Tenenbaum JB (2018) Differentiable physics and stable modes for tool-use and manipulation planning .

Valenzuela AK (2016) *Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain*. PhD Thesis, Massachusetts Institute of Technology.

Villarreal O, Barasuol V, Wensing PM, Caldwell DG and Semini C (2020) Mpc-based controller with terrain insight for dynamic legged locomotion. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2436–2442.

Wang K, Hu ZJ, Tisnikar P, Helander O, Chappell D and Kormushev P (2023) When and where to step: Terrain-aware real-time footstep location and timing optimization for bipedal robots. *arXiv preprint arXiv:2302.07345* .

Wensing PM and Orin DE (2013) Generation of dynamic humanoid behaviors through task-space control with conic optimization. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3103–3109.

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

Winkler A, Havoutis I, Bazeille S, Ortiz J, Focchi M, Dillmann R, Caldwell D and Semini C (2014) 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, pp. 6476–6482.

Winkler AW, Bellicoso CD, Hutter M and Buchli J (2018) Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters* 3(3): 1560–1567.

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

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

Yang Y, Zhang T, Coumans E, Tan J and Boots B (2022) Fast and efficient locomotion via learned gait transitions. In: *Conference on Robot Learning*. PMLR, pp. 773–783.

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

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

Zhang M, Jha DK, Raghunathan AU and Hauser K (2023) Simultaneous trajectory optimization and contact selection for multi-modal manipulation planning. *arXiv preprint arXiv:2306.06465* .

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

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

Zhao Z, Zhou Z, Park M and Zhao Y (2021) Sydebo: Symbolic-decision-embedded bilevel optimization for long-horizon manipulation in dynamic environments. *IEEE Access* 9: 128817–128826.

Zhou Z, Lee DJ, Yoshinaga Y, Balakirsky S, Guo D and Zhao Y (2022a) Reactive task allocation and planning for quadrupedal and wheeled robot teaming. In: *IEEE International Conference on Automation Science and Engineering*. IEEE, pp. 2110–2117.

Zhou Z, Meng Q, Kress-Gazit H and Zhao Y (2025) 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.

Zhou Z, Wingo B, Boyd N, Hutchinson S and Zhao Y (2022b) Momentum-aware trajectory optimization and control for agile quadrupedal locomotion. *IEEE Robotics and Automation Letters* 7(3): 7755–7762.

Zhu H, Meduri A and Righetti L (2023) Efficient object manipulation planning with monte carlo tree search. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 10628–10635.

Zhuang Z, Fu Z, Wang J, Atkeson C, Schwertfeger S, Finn C and Zhao H (2023) Robot parkour learning. *arXiv preprint arXiv:2309.05665* .