2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

RL-augmented Adaptive Model Predictive Control for Bipedal Locomotion over Challenging Terrain

Junnosuke Kamohara[1], Feiyang Wu[1], Chinmayee Wamorkar[1], Seth Hutchinson[2], Ye Zhao[1]

Abstract— **Model predictive control (MPC) has demonstrated effectiveness for humanoid bipedal locomotion; however, its applicability in challenging environments, such as rough and slippery terrain, is limited by the difficulty of modeling terrain interactions. In contrast, reinforcement learning (RL) has achieved notable success in training robust locomotion policies over diverse terrain, yet it lacks guarantees of constraint satisfaction and often requires substantial reward shaping. Recent efforts in combining MPC and RL have shown promise of taking the best of both worlds, but they are primarily restricted to flat terrain or quadrupedal robots.**

In this work, we propose an RL-augmented MPC framework tailored for bipedal locomotion over rough and slippery terrain. Our method parametrizes three key components of single-rigid-body-dynamics-based MPC: system dynamics, swing leg controller, and gait frequency. We validate our approach through bipedal robot simulations in NVIDIA IsaacLab across various terrains, including stairs, stepping stones, and low-friction surfaces. Experimental results demonstrate that our RL-augmented MPC framework produces significantly more adaptive and robust behaviors compared to baseline MPC and RL. Project page: `https://rl-augmented-mpc.github.io/rlaugmentedmpc/`

I. Introduction

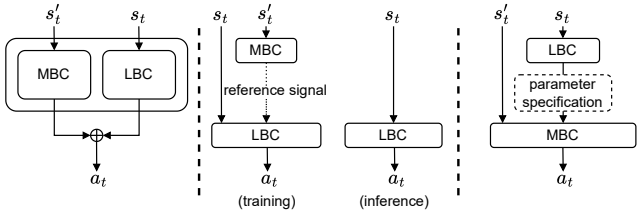
Legged locomotion conventionally employs model-based controllers (MBCs), particularly Model Predictive Control (MPC), due to their optimization-based constraint satisfaction [1], [2]. While whole-body dynamics models [3] are more accurate, researchers use simplified models [4]–[6] for computational efficiency and consequently suffer from model mismatch due to simplification of the dynamics. As a result, simplified models exhibit poorer tracking accuracy and instability, particularly during contact [7]. Additionally, MPC with simplified dynamics usually requires predefined contact sequence and swing leg trajectory, which limits its adaptivity to diverse terrains. Overall, the deterministic but inaccurate dynamic model and manual constraint design of MPC restrict its robustness and versatility, limiting its applicability to diverse terrains in the real world.

In contrast, learning-based controls (LBC), exemplified by Reinforcement Learning (RL) methods, have gained wide attention for their robustness and agility [8]–[12]. By training policies parameterized by neural networks, RL policies can achieve zero-shot transfer from simulation to reality. However, training robust policies requires substantial environmental interactions and extensive reward shaping. Furthermore, RL policies lack explicit constraint satisfaction because of the absence of explicit constraints.

Motivated by the unique advantages of both sides, recent years have witnessed a surge of methods combining model-based and learning-based approaches, leveraging the safety offered by MPC's explicit constraints as well as powerful reactive behaviors offered by RL [13], [14]. In legged robotics, there are two main threads of combination. The first thread uses MPC within a policy. Recent works either adopt a *hierarchical architecture*, where RL parametrizes MPC's components, including system dynamics, center of mass reference trajectory, and gait frequency [7], [15]–[17]; or follows a *parallel architecture*, where RL policies refine MPC outputs by adding corrective actions such as footholds and joint commands [18]–[20]. Another thread uses MPC as an expert policy, training the policy through behavior cloning or RL with imitation loss to increase sample efficiency and motion accuracy [12], [21]–[23]. Each of these designs carries trade-offs: MPC as an expert improves training efficiency by imitating MPC motions, yet it incurs significant computational overhead during training due to repeated optimization solves, making training in parallelized RL environments particularly challenging [23]. While parallel architectures offer flexibility by directly augmenting MPC outputs, they raise safety concerns since the RL policy bypasses feasibility constraints from optimization. Hierarchical architectures, in contrast, preserve the optimization structure and computational complexity, as the policy is evaluated before solving the optimization problem. This ensures the feasibility and constraint satisfaction within the optimization framework.

Despite these advances, most combined approaches for bipedal locomotion remain limited to flat terrain, as prior works primarily emphasize improving tracking accuracy rather than adaptability [18], [20], leaving integration of MPC and RL for rough-terrain-adaptive locomotion unexplored. In this work, we aim to enhance the adaptability of humanoid locomotion, enabling responsive and robust behaviors in the face of terrain disturbances. We leverage a hierarchical method that augments MPC via RL by incorporating rich whole-body information into the simplified system model, adjusting the gait frequency to modulate step length, and modifying the swing foot trajectory to improve robustness against challenging terrain. We focus on addressing the limitations of MPC with simplified dynamics: model mismatch, predefined swing leg curve, and static gait frequency. The RL policy learns residual dynamics through whole-body dynamics simulation, as well as swing leg curve

[1] Georgia Institute of Technology, GA 30332, USA. {jkamohara3, feiyangwu, cwamorkar3, yezhao}@gatech.edu
[2] Northeastern University, 360 Huntington Ave, Boston, MA 02115, USA. s.hutchinson@northeastern.edu



(a) Parallel arch. (b) Imitation arch. (c) Hierarchical arch.

Fig. 1: Overview of existing MPC and RL combined approaches. MBC and LBC take state s'_t and observation s_t as inputs. The final control output is a_t.

parameters, including apex height and control points, and dynamic gait frequency within one locomotion cycle.

These learned adaptations enable reactive behaviors, including recovery from foot entrapment and severe slippage. We implement our method on bipedal locomotion tasks with the HECTOR [24] in NVIDIA IsaacLab, a state-of-the-art GPU-accelerated simulator [25]. Our framework significantly improves robustness against disturbances on diverse terrains, including slippery surfaces, stairs, and stepping stones. Additionally, we conduct ablation studies on the three residual modules to analyze the contribution of each component.

II. Related work

Recent works have explored how to combine MPC and RL controllers [13], [14]. Existing literature in legged locomotion follows three types of architectures: parallel, imitation, and hierarchical architecture, as shown in Fig. 1.

Parallel architecture: With a parallel architecture, MPC's output is refined by a learning-based module to improve the performance. Learning-based controllers are usually trained to output the residual action, which is then added to the MPC output to form the final control. A2C-MPC [26] outputs forward acceleration and angular velocity in the steering angle to improve tracking in off-road navigation problems. Authors in [19], [20] augment joint commands in an attempt to achieve agile locomotion on rough terrain. In bipedal locomotion, authors in [18] refine suboptimal footholds from MPC by training a residual foot placement planner using RL. While these approaches are intuitive and can directly override control inputs, they may fail to produce stable walking motions due to the lack of explicit constraint satisfaction.

Imitation architecture: This approach employs MPC to guide the RL policy during training [12], [22], [23], leveraging trajectory optimization solutions as additional supervision signals. Model-based reference generation can help accelerate training, but it also has significant drawbacks. Training with offline-generated trajectory limits the policy's generalizability, as the trained policy only imitates the generated trajectories [12], [22]. Employing MPC as an online trajectory generator can yield a more generalizable policy, but the MPC in this setup is often overly simplified or restricted due to heavy computational overhead [23].

Hierarchical architecture: With a hierarchical architecture, RL specifies MPC's parameters, and MPC produces the final control inputs. This line of research often leverages residual estimation [7], [15], [27] and learning-based planners [16], [17]. For residual estimation, the authors in [15], [19] estimate the uncertainty sets of system dynamics via RL, achieving locomotion on diverse terrain. Another approach [7], [27] employs supervised learning to estimate the residual dynamics from offline proprioceptive data. For learning-based planners, CAjun [17] trains an RL policy that outputs Center of Mass (CoM) velocity commands, gait frequency, and refined footholds to achieve jumping. GLiDE [16] estimates reference CoM acceleration, which is tracked by a quadratic programming-based force controller. Nonetheless, for bipedal robots, hierarchical training with RL and MPC remains an area of investigation. Inspired by this, our proposed RL-augmented MPC for humanoid robots augments the system dynamics through RL training, while simultaneously enabling adaptive swing leg trajectory generation and gait frequency modulation to further enhance versatility and robustness.

Bipedal locomotion over challenging terrain: Recent works have demonstrated successful bipedal locomotion over diverse rough terrains [11], [12], [22], [28], [29]. However, RL-based policies raise safety concerns and demand significant reward shaping efforts. MPC and RL combined approaches aim to address these issues, but they are largely limited to quadrupedal locomotion, and the evaluated terrains are restricted to a short staircase [19], stepping stones [16], and gravel and grass [15]. Still, it remains to be seen if these works can succeed on more general rough terrains like stairs with a larger number of steps or terrains with random elevation patterns.

III. Preliminaries

The following section briefly introduces the single rigid body dynamics (SRBD) MPC controller [5], [6].

A. Single rigid body dynamics MPC

The SRBD can be formulated as follows:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{\mathbf{p}} \\ \mathbf{R}_b \boldsymbol{\omega} \\ \sum_i \mathbf{F}_i / m + \mathbf{g} \\ I_{\mathbf{w}}^{-1} \sum_i (\mathbf{r}_i \times \mathbf{F}_i + \mathbf{M}_i), \end{bmatrix} \tag{1}$$

where $\mathbf{x} = [\mathbf{p}, \boldsymbol{\Theta}, \dot{\mathbf{p}}, \boldsymbol{\omega}] \in \mathbb{R}^{12}$ is the state, and $\mathbf{u} = [\mathbf{F}_1, \mathbf{F}_2, \mathbf{M}_1, \mathbf{M}_2] \in \mathbb{R}^{12}$ is the control input. The state $\mathbf{x}$ consists of the center of mass (CoM) position $\mathbf{p} \in \mathbb{R}^3$, orientation $\boldsymbol{\Theta} \in \mathbb{R}^3$, linear velocity $\dot{\mathbf{p}} \in R^3$, and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$. The control input $\mathbf{u}$ represents the ground reaction force $\mathbf{F} \in \mathbb{R}^3$ and the reaction wrench $\mathbf{M} \in \mathbb{R}^3$, where $i \in \{0, 1\}$ is the index of leg in contact. Additionally, $\mathbf{g} \in \mathbb{R}^3$ is the gravity vector, m is the lumped robot mass, and $I_{\mathbf{w}} \in \mathbb{R}^{3\times3}$ is the robot's centroidal inertia w.r.t the global coordinate, $\mathbf{r}_i \in \mathbb{R}^3$ is the vector from the CoM to the i^{th} contact point, and $\mathbf{R}_b \in \mathbb{R}^{3\times3}$ is the world-to-base coordinate transformation matrix. The nonlinear terms $\mathbf{R}_b\boldsymbol{\omega}$ and $\mathbf{r}_i \times \mathbf{F}_i$ are linearized at the current state to obtain linear dynamics. We refer to this linearized state dynamics as the nominal dynamics $\widetilde{\mathbf{f}}(\mathbf{x}, \mathbf{u}) : \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, with

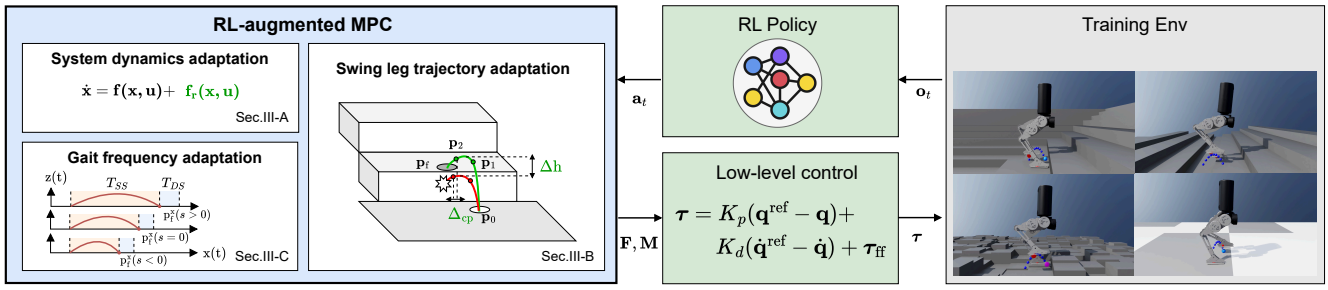


Fig. 2: System architecture of our proposed method. The RL policy augments three key modules in the single rigid body MPC. The low level controller then tracks the contact force $\mathbf{F}$ and moment $\mathbf{M}$ computed by MPC to generate joint torque commands. We leverage the whole-body dynamics of the robot to train the policy across diverse terrains, as illustrated on the right side of the figure.

an augmented state $\mathbf{x} = [\mathbf{p}, \boldsymbol{\Theta}, \dot{\mathbf{p}}, \boldsymbol{\omega}, 1] \in \mathbb{R}^{13}$ and state-space matrices $\mathbf{A} \in \mathbb{R}^{13\times13}$ and $\mathbf{B} \in \mathbb{R}^{13\times12}$. We include the gravity term as an additional state variable to cast the dynamics into state-space form [5], [6], and we continue to denote the augmented state as $\mathbf{x}$, by slight abuse of notation.

Using the linearized dynamics, the convex MPC controller [6] solves a constrained optimization problem that computes the contact wrench to track the reference trajectory:

$$\min_{\boldsymbol{\mathcal{X}}} \sum_{k=0}^{N-1} (\mathbf{x}_{k+1} - \mathbf{x}_{k+1}^{\text{ref}})^\top \mathbf{Q}_k (\mathbf{x}_{k+1} - \mathbf{x}_{k+1}^{\text{ref}}) + \mathbf{u}_k^\top \mathbf{R}_k \mathbf{u}_k$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k \tag{2a}$$

$$(1 - \mathrm{c}_{k,i})\, \mathbf{u}_{k,i} = \mathbf{0} \tag{2b}$$

$$|\mathrm{F}^{\mathrm{x}}{}_{k,i}|, |\mathrm{F}^{\mathrm{y}}{}_{k,i}| \le \mu |\mathrm{F}^{\mathrm{z}}{}_{k,i}| \tag{2c}$$

$$0 \le \mathrm{F}^{\mathrm{z}}{}_{k,i} \le \mathrm{F}_{\max} \tag{2d}$$

$$-l_t \mathbf{S}_{\mathrm{z}} \mathbf{R}_{\mathrm{f},i}^\top \mathbf{F}_{k,i} \le \mathbf{S}_{\mathrm{y}} \mathbf{R}_{\mathrm{f},i}^\top \mathbf{M}_{k,i} \le l_h \mathbf{S}_{\mathrm{z}} \mathbf{R}_{\mathrm{f},i}^\top \mathbf{F}_{k,i} \tag{2e}$$

$$\forall k = 0, \ldots, N-1, \forall i = 0, 1$$

where $\boldsymbol{\mathcal{X}}$ is the full set of decision variables composed of the state $\mathbf{x}_k$ and control input $\mathbf{u}_k$ at all timesteps $k = 0, \ldots, N$. The cost function includes the trajectory tracking error and input cost, where $\mathbf{x}_k^{\text{ref}} \in \mathbb{R}^3$ is the state reference, and $\mathbf{Q}_k \in \mathbb{R}^{13\times13}$ and $\mathbf{R}_k \in \mathbb{R}^{12\times12}$ are diagonal positive semi-definite cost-weight matrices. The equality constraints include the discrete-time dynamics (2a), where $\mathbf{A}_k$ and $\mathbf{B}_k$ represent the discrete-time state-space matrices from Euler integration, and the contact constraint (2b), where $\mathrm{c}_{k,i}$ and $\mathbf{u}_{k,i}$ are the contact state (0 or 1) and the contact wrench of the i^{th} foot at timestep k, respectively. The inequality constraints include the friction pyramid (2c), force saturation (2d), and line contact constraints (2e). Here, μ represents the friction coefficient; $\mathbf{F}_{k,i}$ and $\mathbf{M}_{k,i}$ represent the ground reaction force and moment of the i^{th} foot at timestep k; $\mathrm{F}_{\max}$ represents the maximum vertical force; l_t and l_h denote the distances from the ankle joint to the toe and heel; $\mathbf{S}_{\mathrm{y}} \in \mathbb{R}^{1\times3}$ and $\mathbf{S}_{\mathrm{z}} \in \mathbb{R}^{1\times3}$ are row vectors that select the y and z elements, respectively; and $\mathbf{R}_{\mathrm{f},i}$ is the rotation matrix of the i^{th} foot w.r.t the world frame.

B. Low level control

Low-level control follows existing literature [5], [6], where the stance foot torque is computed from the contact Jacobian mapping, and the swing foot torque is calculated using a proportional–derivative (PD) controller whose target setpoints are generated via analytical inverse kinematics (IK). The analytical IK computes joint position commands corresponding to the desired foot position $\mathbf{p}_{\text{foot}}^{\text{ref}}$, which is derived from a heuristic swing leg trajectory whose end point is specified by the foot placement planner. For the foot placement planner, we use Raibert heuristics [30] to update the target footholds $\mathbf{p}_{\text{foothold}} \in \mathbb{R}^3$ at each control timestep:

$$\mathbf{p}_{\text{foothold}} = \mathbf{p}_{\text{hip}}^{\text{ref}} + \tfrac{1}{2}\dot{\mathbf{p}}\delta T + k_d(\dot{\mathbf{p}} - \dot{\mathbf{p}}^{\text{ref}}), \tag{3}$$

where $\mathbf{p}_{\text{hip}}^{\text{ref}} \in \mathbb{R}^3$ represents the hip position reference; δT denotes the remaining step duration in a single walking step; $\dot{\mathbf{p}}^{\text{ref}} \in \mathbb{R}^3$ is the reference CoM velocity in the world frame; and k_d is a feedback gain (set to 0.05 in this work). Finally, the reference swing foot trajectory $\mathbf{p}_{\text{foot}}^{\text{ref}} \in \mathbb{R}^3$ is computed using a cubic Bézier curve $\mathcal{B}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_{\text{foothold}}, \phi_t)$, where $\mathbf{p}_0$ is the stance foot position right before contact switch, and $\mathbf{p}_1$ and $\mathbf{p}_2$ are the control points given by $\mathbf{p}_1 = \mathbf{p}_0 + \frac{1}{3}(\mathbf{p}_{\text{foothold}} - \mathbf{p}_0)$, $\mathbf{p}_2 = \mathbf{p}_0 + \frac{2}{3}(\mathbf{p}_{\text{foothold}} - \mathbf{p}_0)$, and ϕ_t is a swing phase variable that is mapped to a predefined swing state (0 or 1). At each control step, ϕ_t is updated by dt/T_{s}, where dt is the control timestep and T_{s} is the swing phase duration.

IV. Methodology

In this section, we introduce our RL-augmented MPC, where the RL policy augments the system dynamics, swing leg trajectory, and step duration. The following subsections describe how the RL policy augments each component.

A. System dynamics adaptation

Single rigid-body dynamics (SRBD) represents a reduced-order model with several simplifications: the leg masses are ignored, and the robot's centroidal inertia does not change drastically [5]. In addition, the robot's center of mass experiences various disturbances, including foot–terrain interactions and inertial forces from the swing legs [31]. While MPC with whole-body dynamics offers a potential solution, this approach incurs significant computational costs and implementation complexity. Therefore, we maintain the use of simplified dynamics, but augment this model using RL. Our baseline MPC is solved at 100 Hz; consequently, we retrieve residual dynamics from RL at the same frequency.

Following existing literature [7], [15], [19], we define residual dynamics $\mathbf{f}_{\mathrm{r}}(\mathbf{x}, \mathbf{u})$ as offset terms to the linear and

angular accelerations that account for unmodeled foot inertia and ground impact effects. We express these offset terms in an affine control system [32], a standard model formulation, as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \mathbf{f}_{\text{r}}(\mathbf{x}, \mathbf{u}), \mathbf{f}_{\text{r}} = \begin{bmatrix} \mathbf{0_{6\times 1}} \\ \hat{\mathbf{f}}_{\mathbf{x}}(\mathbf{x}) + \hat{\mathbf{f}}_{\mathbf{u}}(\mathbf{x})\mathbf{u} \end{bmatrix} \tag{4}$$

where $\hat{\mathbf{f}}_{\mathbf{x}}(\mathbf{x}) \in \mathbb{R}^6$ represents the residual state term, and $\hat{\mathbf{f}}_{\mathbf{u}}(\mathbf{x}) \in \mathbb{R}^{6\times 12}$ denotes the residual actuation matrix term. The first six rows of $\mathbf{f}_{\text{r}}(\mathbf{x}, \mathbf{u})$ are zeros, since the residual dynamics only model perturbations at the acceleration level.

Furthermore, we simplify the $\hat{\mathbf{f}}_{\mathbf{u}}(\mathbf{x})$ matrix by assuming that moments do not affect linear acceleration and forces do not affect angular acceleration. This simplification reduces the original actuation matrix $\hat{\mathbf{f}}_{\mathbf{u}}(\mathbf{x}) \in \mathbb{R}^{6\times 12}$ to two low-dimensional matrices $\hat{\mathbf{f}}_{\mathbf{u}}^{(1)}, \hat{\mathbf{f}}_{\mathbf{u}}^{(2)} \in \mathbb{R}^{3\times 3}$. Then the linearized augmented dynamics are further simplified as

$$\dot{\mathbf{x}} = \widetilde{\mathbf{f}}(\mathbf{x}, \mathbf{u}) + \begin{bmatrix} \mathbf{0_{6\times 1}} \\ \hat{\mathbf{f}}_{\mathbf{x}}^{(1)} + \hat{\mathbf{f}}_{\mathbf{u}}^{(1)} \sum_i \mathbf{F}_i \\ \hat{\mathbf{f}}_{\mathbf{x}}^{(2)} + \hat{\mathbf{f}}_{\mathbf{u}}^{(2)} \sum_i \mathbf{M}_i \end{bmatrix} \tag{5}$$

where $\widetilde{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ denotes the linearized single rigid body dynamics from the previous section, $\hat{\mathbf{f}}_{\mathbf{x}}^{(1)}, \hat{\mathbf{f}}_{\mathbf{x}}^{(2)} \in \mathbb{R}^3$ are residual accelerations, and $\hat{\mathbf{f}}_{\mathbf{u}}^{(1)}, \hat{\mathbf{f}}_{\mathbf{u}}^{(2)} \in \mathbb{R}^{3\times 3}$ are simplified residual actuation matrices. We further simplify the matrices $\hat{\mathbf{f}}_{\mathbf{u}}^{(1)}, \hat{\mathbf{f}}_{\mathbf{u}}^{(2)}$ as $\text{diag}(\hat{\text{f}}_{\text{u11}}^{(1)}, \hat{\text{f}}_{\text{u22}}^{(1)}, \hat{\text{f}}_{\text{u33}}^{(1)}) \in \mathbb{R}^{3\times 3}$ and $\text{diag}(\hat{\text{f}}_{\text{u11}}^{(2)}, \hat{\text{f}}_{\text{u22}}^{(2)}, \hat{\text{f}}_{\text{u33}}^{(2)}) \in \mathbb{R}^{3\times 3}$ by ignoring off-diagonal elements to reduce the number of variables from 9 to 3 for each matrix. Consequently, the residual dynamics parameters to be learned are $[\hat{\mathbf{f}}_{\mathbf{x}}^{(1)}, \hat{\mathbf{f}}_{\mathbf{x}}^{(2)}, \hat{\text{f}}_{\text{u11}}^{(1)}, \hat{\text{f}}_{\text{u22}}^{(1)}, \hat{\text{f}}_{\text{u33}}^{(1)}, \hat{\text{f}}_{\text{u11}}^{(2)}, \hat{\text{f}}_{\text{u22}}^{(2)}, \hat{\text{f}}_{\text{u33}}^{(2)}] \in \mathbb{R}^{12}$.

B. Swing leg trajectory adaptation

Designing collision-free swing leg trajectories is critical, especially for locomotion on non-flat terrain. To this end, we design an adaptive swing leg trajectory by adjusting its apex height and control points, as illustrated in Fig. 2. Specifically, we modify the cubic Bézier curve $\mathcal{B}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_{\text{foothold}}, \phi_t)$ using the adaptive control points $\mathbf{p}_1, \mathbf{p}_2$:

$$\begin{aligned} \mathbf{p}_1 &= \mathbf{p}_0 + (\tfrac{1}{3} + \Delta_{\text{cp}})(\mathbf{p}_{\text{foothold}} - \mathbf{p}_0) \\ \mathbf{p}_2 &= \mathbf{p}_0 + (\tfrac{2}{3} + \Delta_{\text{cp}})(\mathbf{p}_{\text{foothold}} - \mathbf{p}_0) \\ \text{p}_1^{\text{z}} &= \text{p}_2^{\text{z}} = \tfrac{1}{6}(8\text{h} - \text{p}_0^{\text{z}} - \text{p}_{\text{foothold}}^{\text{z}}) \\ \text{h} &= \text{p}_0^{\text{z}} + \widetilde{\text{h}} + \Delta\text{h} \end{aligned} \tag{6}$$

where $\mathbf{p}_0$ is the stance foot position right before contact switch, $\mathbf{p}_{\text{foothold}}$ is the planned foothold, h is the apex height of reference foot trajectory, $\widetilde{\text{h}}$ is the nominal swing foot height. We introduce two residual parameters: the residual apex height Δh and the control point coefficient Δ_{cp}, which modify the shape of the swing leg trajectory. These two parameters adjust the foot's reference height and the position of the apex, respectively, as illustrated in Fig. 2. A positive Δ_{cp} shifts the trajectory forward, whereas a negative value shifts it backward. This formulation enables flexible manipulation of the trajectory while maintaining a small search space, compared to approaches that learn residuals for the entire joint PD setpoints [19].

TABLE I: List of action space.

Action terms	Scaling parameters	Unit
Res. lin. acc. $\hat{\mathbf{f}}_{\mathbf{x}}^{(1)}$	$(2.0, 2.0, 4.0)$	(m/s^2)
Res. ang. acc. $\hat{\mathbf{f}}_{\mathbf{x}}^{(2)}$	$(1.0, 1.0, 1.0)$	(rad/s^2)
Res. act. mat. $\hat{\mathbf{f}}_{\mathbf{u}}^{(1)}$	$(\frac{0.2}{13.856}, \frac{0.2}{13.856}, \frac{0.2}{13.856})$	$(/\text{kg})$
Res. act. mat. $\hat{\mathbf{f}}_{\mathbf{u}}^{(2)}$	$(\frac{0.2}{0.5413}, \frac{0.2}{0.52}, \frac{0.2}{0.0691})$	$(/\text{kg}\cdot\text{m}^2)$
Sampling time coef. s	0.3	-
Swing foot Δh	0.05 (slippery), 0.15 (rough)	(m)
Control point Δ_{cp}	0.33 (slippery), 0.66 (rough)	-

C. Gait frequency adaptation

Our controller uses a periodic gait to pre-generate the contact sequence for both legs. We achieve variable walking gaits by parameterizing the swing phase duration as $T_s = H dt_{\text{mpc}}$, where H represents MPC's horizon length and dt_{mpc} denotes the discretization timestep (or sampling time). Therefore, the step duration can be adjusted by manipulating the MPC sampling time. To this end, we formulate dt_{mpc} as follows:

$$dt_{\text{mpc}} = \widetilde{dt}_{\text{mpc}}(1 + s) \tag{7}$$

where $\widetilde{dt}_{\text{mpc}}$ is the nominal MPC sampling time, and s is the MPC sampling time coefficient. The double-support and single-support durations are computed as $T_{DS} = 2dt_{\text{mpc}}$ and $T_{SS} = 8dt_{\text{mpc}}$, respectively. Thus, the residual double-support and single-support durations are $\Delta T_{DS} = 2s\widetilde{dt}_{\text{mpc}}$ and $\Delta T_{SS} = 8s\widetilde{dt}_{\text{mpc}}$, respectively. The MPC then uses the updated double-support and single-support durations to generate the contact sequence at each RL step. We assume these durations remain constant during the MPC prediction horizon, since the horizon is short ($0.2 \sim 0.3$ s) and the MPC solution is updated at a high frequency (100 Hz).

D. RL formulation

The objective of the RL policy is to estimate the residual action parameters discussed above. We treat the problem of finding these residual parameters as a Markov Decision Process (MDP). At each timestep t, the agent receives the observation $\mathbf{o}_t$, performs an action $\mathbf{a}_t$ according to its current policy $\pi(\cdot|\mathbf{o}_t)$, and obtains a scalar reward R_t. The goal of the agent is to maximize the discounted sum of rewards $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t]$, where $\gamma \in [0, 1)$ is the discount factor.

We apply model-free deep reinforcement learning (DRL) methods to train a neural network that maximizes the return and finds the optimal residuals for the MPC. The predicted residual parameters $\mathbf{a}_t \sim \pi(\cdot|\mathbf{o}_t)$ augment the MPC controller to enable adaptive locomotion. We leverage the robotics simulation platform NVIDIA IsaacLab [25] with its state-of-the-art physics engine for whole-body dynamics simulation.

Action space: The action space $\mathcal{A}$ consists of dynamics residuals $[\hat{\mathbf{f}}_{\mathbf{x}}^{(1)}, \hat{\mathbf{f}}_{\mathbf{x}}^{(2)}, \hat{\text{f}}_{\text{u11}}^{(1)}, \hat{\text{f}}_{\text{u22}}^{(1)}, \hat{\text{f}}_{\text{u33}}^{(1)}, \hat{\text{f}}_{\text{u11}}^{(2)}, \hat{\text{f}}_{\text{u22}}^{(2)}, \hat{\text{f}}_{\text{u33}}^{(2)}] \in \mathbb{R}^{12}$, swing foot controller residuals $[\Delta\text{h}, \Delta_{\text{cp}}] \in \mathbb{R}^2$, and the MPC sampling time coefficient $s \in \mathbb{R}$. The actions from the policy are scaled to match the range of each residual term using the scaling parameters listed in Table I.

TABLE II: List of observation space.

Observation	Dim	Observation	Dim
Projected gravity	3	Base linear vel.	3
Base angular vel.	3	Velocity commands	3
Joint pos.	10	Joint vel.	10
Previous action	9	Swing phase	2
Planned footholds	4	Foot pos.	6
Reference foot pos.	6		

TABLE III: List of reward functions.

Reward	Expression	Weight
Track lin. vel. XY	$\exp\left(-\|\dot{\mathbf{p}} - \dot{\mathbf{p}}^{\text{ref}}\|^2/\sigma^2\right)$	1.0
Track ang. vel. Z	$\exp\left(-\|\boldsymbol{\omega} - \boldsymbol{\omega}^{\text{ref}}\|^2/\sigma^2\right)$	0.5
Track height	$\exp\left(-(\mathrm{p}^{\mathrm{z}} - \mathrm{p}^{\mathrm{ref,z}})^2/\sigma^2\right)$	0.1
Lin. vel. Z	$\dot{\mathrm{p}}_{\mathrm{z}}$	-1e-2
Ang. vel. XY	${\omega_{\mathrm{x}}}^2 + {\omega_{\mathrm{y}}}^2$	-1e-4
Joint vel.	$\|\dot{\mathbf{q}}_{\text{joint}}\|^2$	-2.5e-4
Action smoothness	$\|\mathbf{a_t} - \mathbf{a_{t-1}}\|^2$	-0.015
Feet slide	$\sum_i \mathrm{c}_i \|\dot{\mathbf{p}}_{\text{foot},i}\|^2$	–0.01
Knee collision	1_{knee}	-5.0
Leg-base angle	θ_{leg}^2	-1.0
Step width	d^2	-0.2

Observation space: The observation space consists of proprioceptive sensory data from the robot and foot states from the MPC controller. Proprioceptive information includes base linear velocity $\dot{\mathbf{p}}$ and angular velocity $\boldsymbol{\omega}$, joint positions $\mathbf{p}_{\text{joint}}$, and joint velocities $\dot{\mathbf{p}}_{\text{joint}}$. Additionally, we include the command velocity and the computed projected gravity $\hat{\mathbf{g}}$ based on IMU data. Foot states consist of the current foot positions $\mathbf{p}_{\text{foot}}$, reference foot positions $\mathbf{p}_{\text{foot}}^{\text{ref}}$, planned footholds $\mathbf{p}_{\text{foothold}}$, and the swing phase ϕ. Finally, the actions from the previous timestep $\mathbf{a}_{t-1}$ are also incorporated into the observation, which has been shown to produce smooth actions [9].

Reward functions: Our reward design is summarized in Table III. We leverage existing reward functions implemented in IsaacLab for bipedal locomotion tasks, which include velocity command tracking, center of mass height tracking, and others. Reward functions specific to our problem include penalties on the base-to-stance-foot angle θ_{foot} and the step width d.

Implementation detail: In this work, we employ Soft Actor-Critic [33], an off-policy model-free RL algorithm. An off-policy algorithm is suitable for our approach because it is more sample efficient [33], which is crucial since environment rollouts with MPC are computationally expensive. Both the actor and critic networks are implemented as separate three-layer MLPs with hidden sizes of (512, 256, 128) with the ELU activation function. The policy is parameterized as a Gaussian distribution during training, with the mean and standard deviation predicted by a single MLP.

Training: The RL policy is trained on an NVIDIA RTX 4090 using IsaacLab [25]. Training is performed using the SAC algorithm implemented in `rl_games` [34], a GPU-accelerated RL library. The hyperparameters used during training are provided in the open-source code. During both training and inference, MPC and RL run at 100 Hz, while low-level control and physics simulation run at 400 Hz. The parameters of MPC are listed in Table IV.

Given this setup, the policy often converged to suboptimal behavior, outputting saturated actions most of the time. To mitigate this poor exploration, we introduce a squared action loss implemented in `rl_games`. With this loss, our policy does not saturate.

TABLE IV: List of MPC parameters.

Name	Value
Horizon length H	10
Sampling time $\widetilde{dt}_{\text{mpc}}$	0.025 s
Swing height $\widetilde{\text{h}}$	0.1 m
Toe length l_t	0.07 m
Heel length l_h	0.04 m
Max. force $F_{\max}$	500 N
State weight $\mathbf{Q}$	$\text{diag}(150, 150, 250, 100, 100, 250, 1, 1, 1, 10, 10, 1, 1)$
Control weight $\mathbf{R}$	$\text{diag}(10^{-5}, \ldots, 10^{-4}, \ldots, 10^{-4})$

V. Evaluation

A. Terrain setups

1) Pyramid stairs: This type of terrain contains a set of ascending steps arranged in all four directions. Each step has a width of 25 cm and a height of 10 cm.
2) Random stairs: This type of terrain contains a set of steps with a random ascending/descending pattern. The width and height of each step are the same as the pyramid stairs.
3) Stepping stones: This terrain is constructed by extruding or intruding a flat grid cell by a random height. Each grid has a width of 25 cm, and the height is uniformly sampled from $\mathcal{U}(-h_{\max}, h_{\max})$ with $h_{\max} = 7$ cm.
4) Slippery surface: This type of terrain contains a set of row-indexed sub-terrains, where the friction range decreases monotonically with the row index. In each sub-terrain, the friction coefficient is randomly assigned either a high value ($\mu = 0.5$) or a low value sampled from the predefined range. As the row index increases, the minimum friction value decreases to 0.05.

During training, a terrain curriculum is used, adapting the robot's initial position to a more difficult region, depending on whether walking succeeds at the current region. For example, on slippery terrain, if the robot can walk in the current region, it is moved to a lower-friction region.

B. Evaluation metrics

We use success rate (SR), velocity/orientation tracking accuracy $e^v, e^{\theta_x}, e^{\theta_y}$ as metrics with a thorough analysis of the action outputs. The success rate is measured by running 100 episodes and counting the number of trials that succeed in reaching the end of a terrain. An episode is counted a failure if the robot violates height constraints ($p_z \geq 0.4$), orientation constraints ($\theta_x, \theta_y \in (-\frac{\pi}{6}, \frac{\pi}{6})$), or fails to reach the end of the terrain within 20 s. Tracking accuracy is measured as the absolute error of the base velocity, which evaluates the policy's command tracking capability, with episodes run for 6 s. Finally, we analyze the action outputs to assess the contribution of the RL policy. A successful policy should output near-zero actions under nominal conditions, such as on flat terrain with a normal frictional coefficient,

TABLE V: Comparison between the baseline MPC and our approach. The table shows success rate (SR), forward velocity tracking error (e^{v}), and roll/pitch errors ($e^{\theta_x}, e^{\theta_y}$) collected from 100 episodes on rough terrain and slippery terrain with varying levels of difficulty. The highest success rate and the lowest tracking errors are highlighted in bold.

Env	height (cm) / μ (-)	$\text{SR}_{\text{MPC}}\uparrow$ (%)	$\text{SR}_{\text{ours}}\uparrow$ (%)	$e^{v}_{\text{MPC}}\downarrow$ (m/s)	$e^{v}_{\text{ours}}\downarrow$ (m/s)	$e^{\theta_x}_{\text{MPC}}\downarrow$ (deg)	$e^{\theta_x}_{\text{ours}}\downarrow$ (deg)	$e^{\theta_y}_{\text{MPC}}\downarrow$ (deg)	$e^{\theta_y}_{\text{ours}}\downarrow$ (deg)
pyramid	8	100 ± 0	100 ± 0	0.12 ± 0.09	0.12 ± 0.09	$\mathbf{0.55 \pm 0.38}$	0.55 ± 0.40	1.71 ± 0.88	$\mathbf{1.6 \pm 0.94}$
stairs	9	48 ± 5	$\mathbf{96 \pm 1.96}$	0.15 ± 0.12	$\mathbf{0.14 \pm 0.1}$	0.68 ± 1.03	$\mathbf{0.59 \pm 0.46}$	1.96 ± 1.58	$\mathbf{1.68 \pm 1.1}$
(height)	10	1 ± 0.99	$\mathbf{86 \pm 3.47}$	0.22 ± 0.17	$\mathbf{0.16 \pm 0.13}$	0.97 ± 1.87	$\mathbf{0.63 \pm 0.58}$	2.6 ± 3.03	$\mathbf{1.7 \pm 1.16}$
random	8	99 ± 0.99	99 ± 0.99	0.1 ± 0.09	0.1 ± 0.09	0.57 ± 0.39	$\mathbf{0.56 \pm 0.4}$	1.39 ± 0.84	$\mathbf{1.32 \pm 0.88}$
stairs	9	69 ± 4.62	$\mathbf{98 \pm 1.4}$	0.12 ± 0.11	$\mathbf{0.12 \pm 0.1}$	0.66 ± 0.89	$\mathbf{0.58 \pm 0.43}$	1.59 ± 1.51	$\mathbf{1.37 \pm 0.95}$
(height)	10	15 ± 3.57	$\mathbf{85 \pm 3.57}$	0.19 ± 0.15	$\mathbf{0.12 \pm 0.11}$	0.96 ± 1.94	$\mathbf{0.61 \pm 0.48}$	2.08 ± 2.58	$\mathbf{1.47 \pm 1.18}$
stepping	$\mathcal{U}(-5, 5)$	100 ± 0	100 ± 0	0.09 ± 0.08	0.09 ± 0.08	0.56 ± 0.39	$\mathbf{0.54 \pm 0.34}$	1.24 ± 0.64	$\mathbf{1.18 \pm 0.6}$
stones	$\mathcal{U}(-6, 6)$	86 ± 3.47	$\mathbf{96 \pm 1.96}$	0.1 ± 0.09	0.1 ± 0.09	0.6 ± 0.59	$\mathbf{0.57 \pm 0.49}$	1.35 ± 1.14	$\mathbf{1.25 \pm 0.87}$
(height)	$\mathcal{U}(-7, 7)$	68 ± 4.66	$\mathbf{89 \pm 3.13}$	0.12 ± 0.11	$\mathbf{0.11 \pm 0.09}$	0.67 ± 1.0	$\mathbf{0.61 \pm 0.68}$	1.58 ± 1.82	$\mathbf{1.28 \pm 0.95}$
slippery	0.2	100 ± 0	100 ± 0	0.09 ± 0.08	$\mathbf{0.09 \pm 0.07}$	$\mathbf{0.59 \pm 0.29}$	0.73 ± 0.34	$\mathbf{1.11 \pm 0.43}$	1.4 ± 0.38
surface	0.15	81 ± 3.92	$\mathbf{100 \pm 0}$	0.1 ± 0.08	$\mathbf{0.1 \pm 0.07}$	0.79 ± 0.94	$\mathbf{0.65 \pm 0.34}$	$\mathbf{1.26 \pm 0.76}$	1.41 ± 0.4
(μ)	0.1	19 ± 3.92	$\mathbf{100 \pm 0}$	0.16 ± 0.12	$\mathbf{0.11 \pm 0.07}$	1.56 ± 2.97	$\mathbf{0.61 \pm 0.37}$	1.55 ± 1.7	$\mathbf{1.42 \pm 0.42}$
	0.05	5 ± 2.18	$\mathbf{74 \pm 4.39}$	0.22 ± 0.16	$\mathbf{0.13 \pm 0.09}$	1.95 ± 3.6	$\mathbf{0.78 \pm 1.29}$	1.56 ± 1.84	$\mathbf{1.48 \pm 0.71}$

and produce large corrective actions on slippery surfaces and rough terrain.

VI. Results

In this section, we evaluate the performance of the proposed algorithm for robust walking on different types of terrain. The robot's state is initialized with the values listed in Table VI at the beginning of each episode.

TABLE VI: Initialization parameters. $U(a)$ denotes the uniform distribution $\mathcal{U}(-a, a)$.

	Name	Value	Unit
Training	Pos. $\mathbf{p}$	$(U(2.0), U(2.0), 0.55)$	(m)
	Ori. $\boldsymbol{\Theta}$	$(0, 0, U(\pi))$	(rad)
	Cmd. $\dot{\mathrm{p}}^{\mathrm{ref,x}}$	0.5	(m/s)
Inference	Pos. $\mathbf{p}$	$(U(0.3), U(0.3), 0.55)$	(m)
pyramid stairs	Ori. $\boldsymbol{\Theta}$	$(0, 0, \pm\frac{n\pi}{2})$	(rad)
random stairs	Ori. $\boldsymbol{\Theta}$	$(0, 0, \pm n\pi)$	(rad)
stepping stones	Ori. $\boldsymbol{\Theta}$	$(0, 0, U(\pi))$	(rad)
slippery surface	Ori. $\boldsymbol{\Theta}$	$(0, 0, U(\pi))$	(rad)
	Cmd. $\dot{\mathrm{p}}^{\mathrm{ref,x}}$	0.5	(m/s)

A. Per-terrain evaluation

We evaluate performance across four terrain types, varying step height for rough terrains and minimum friction for slippery terrain. Table V summarizes benchmark results under various terrain types. Across all terrain types, our method achieves substantially higher success rates and higher tracking accuracy compared to the MPC baseline.

Pyramid stairs: As the step height increases, the performance gap between the baseline controller and our approach becomes more pronounced. The baseline controller rapidly deteriorates, achieving only a 1% success rate on 10 cm stairs, indicating its limited robustness to challenging terrain. In contrast, our method sustains a success rate exceeding 85%, highlighting its ability to reliably handle significant terrain variations. In addition to robustness, our method maintains better tracking performance across different levels of terrain difficulty. The variation in tracking error remains within 0.04 m/s between mild and harsh terrain, while the baseline shows much larger deviations, up to 0.1 m/s, further underscoring the advantage of our method.

Random stairs: Overall, the success rate is slightly lower than on pyramid stairs due to the random descending–ascending transitions, where overspeed during descent increases the risk of tripping on the subsequent step. Even at the highest difficulty, our method sustains a success rate above 80%, whereas the baseline falls below 20%. Tracking error remains small across stair heights for our approach, in contrast to the baseline.

Stepping stones: Our method consistently outperforms the baseline in success rate. Tracking errors are comparable between the two methods across all stone spacings, reflecting the relatively lower difficulty of stepping stones.

Slippery surface: As the friction coefficient μ decreases, the baseline drops to a 5% at $\mu = 0.05$, whereas our method maintains above 70%. Tracking accuracy shows a similar trend: our method limits error to 0.13 m/s on the most slippery surface, while the baseline exceeds 0.2 m/s. Moreover, our policy stabilizes torso roll angle more effectively, which is critical for resisting lateral slip.

B. Role of RL Residuals

We analyze the policy's corrective actions to illustrate how RL complements MPC on pyramid stairs and slippery surfaces, as presented in Fig. 3. Since the policy does not include perception in its observation, it reacts to the terrain only after the foot makes contact or experiences slippage.

On stairs, pronounced spikes appear in $\Delta\mathrm{h}$, Δ_{cp}, and s at foot contact instants, indicated by the dotted vertical lines. When stepping onto stairs, $\Delta\mathrm{h}$ increases by up to 10 cm, raising the swing foot reference height to reduce the risk of tripping. Simultaneously, Δ_{cp} decreases to approximately -0.2, shifting the swing trajectory backward. This adaptation is critical for versatile and failure-free locomotion, as a symmetric trajectory ($\Delta_{\mathrm{cp}} = 0$) often causes foot collisions when the prior foothold is near a stair edge. Residuals in linear and angular accelerations also increase, particularly in the vertical direction, thereby increasing the normal force at contact. In contrast, during steady walking on flat terrain ($t = 0.6$–1.1 s), residuals remain small and periodic, underscoring their adaptive, event-driven activation.

On slippery terrain, both the amplitude and frequency of

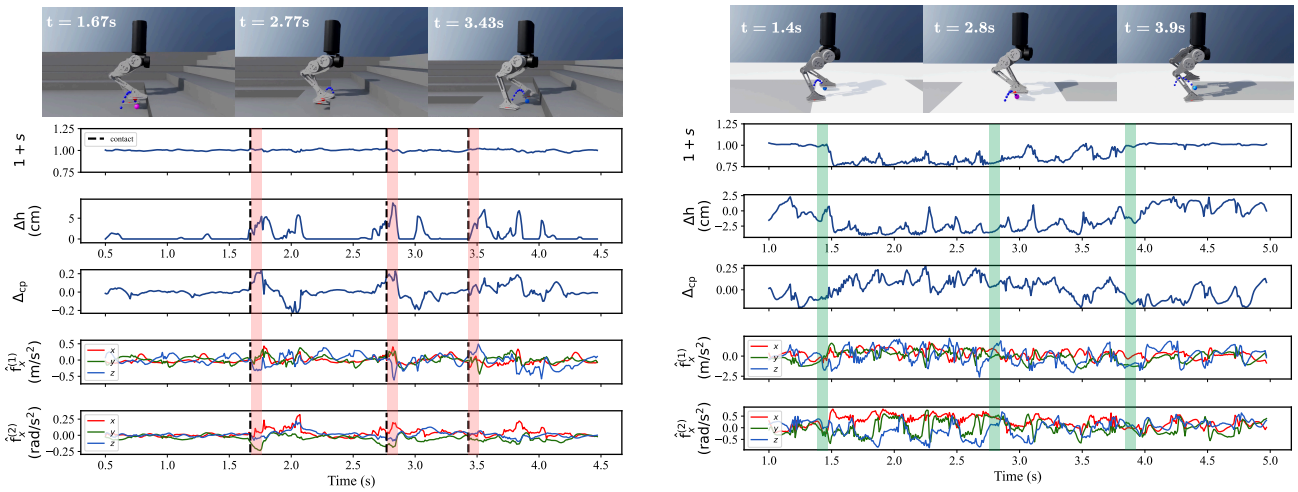


Fig. 3: The figure illustrates actions taken by the RL policy and time-lapse images of the robot at times marked by red (rough terrain instances) and green (slippery terrain instances) shaded regions. The action plots illustrate the outputs of each term on the pyramid stairs (left) and slippery surface (right). Dotted vertical lines in the left figure indicates the instance of toe contact.

the residuals increase in response to reduced friction. At $t = 1.4$ s, the ratio of MPC sampling time to its nominal value decreases from 1.0 to 0.75, leading to shorter step durations. These shorter steps reduce stride length and mitigate lateral slip. Meanwhile, Δh decreases to about -2.5 cm, lowering the swing foot to enable faster step-down motion. Linear and angular acceleration residuals adapt strongly, with pitch and yaw accelerations exhibiting the largest oscillations.

C. Ablation study

Our proposed method integrates multiple adaptation modules into the nominal MPC controller. To access the contribution of each module, we conduct an ablation study against eight variants, each representing a different combination of the adaptation modules. Specifically, *res-all* corresponds to our proposed method with all three modules, *vanilla-MPC* to the baseline MPC, and *vanilla-RL* to a joint space RL policy trained with rewards implemented in IsaacLab for bipedal locomotion tasks. Following the main evaluation metrics, we use success rate and velocity/orientation tracking as metrics, measured by running 100 episodes on 10 cm pyramid stairs. As summarized in Table VII, our method consistently outperforms the baseline MPC and RL controllers in terms of success rate and tracking accuracy.

The ablation study reveals specific weakness in the baselines. For example, the configurations without swing foot trajectory adaptation (*dyn-gait*, *dyn*, *gait*) exhibit catastrophic failure, underscoring the critical role of swing leg adaptation on rough terrain. Furthermore, *swing-gait* and *dyn-swing* achieve lower success rates and larger tracking errors than *swing* alone, highlighting that all three modules are essential for achieving the best performance.

D. Batched MPC on GPU

Lastly, we present preliminary results of a batched MPC controller scalable to thousands of parallel environments, accelerating policy training by over 20 times compared to a CPU baseline. We implement a sparse quadratic programming (QP) solver based on the primal-dual interior point method in CusADi [35] to enable training with a batch size of 4096. Fig. 4 presents benchmark results on computation time across different batch sizes. We compare our solver against `qpOASES` [36], a CPU-based QP solver used in the baseline MPC, and `qpth` [37], a state-of-the-art batched QP solver implemented in PyTorch. As the batch size increases, our solver maintains significantly lower computation times, while both `qpOASES` and `qpth` exhibit substantial increases. Furthermore, it achieves locomotion performance comparable to the CPU baseline, with the policy attaining a 85% success rate on 10 cm pyramid stairs. Given these performance gains, we are migrating our case studies to this GPU-accelerated implementation and will report updated results in future work.

TABLE VII: Results of the ablation study on 10 cm pyramid stairs. The highest success rate and lowest tracking errors are bolded.

Name	SR↑ (%)	e^v↓ (m/s)	e^{θ_x}↓ (deg)	e^{θ_y}↓ (deg)
res-all	$\mathbf{86 \pm 3.47}$	0.16 ± 0.13	$\mathbf{0.63 \pm 0.58}$	$\mathbf{1.7 \pm 1.16}$
swing	74 ± 1.0	0.17 ± 0.14	0.67 ± 0.54	1.72 ± 1.13
swing-gait	67 ± 4.7	$\mathbf{0.15 \pm 0.12}$	0.69 ± 0.6	1.91 ± 1.36
dyn-swing	51 ± 5.0	0.18 ± 0.15	0.69 ± 0.66	1.85 ± 1.48
dyn-gait	2 ± 1.4	0.21 ± 0.17	0.98 ± 2.08	2.53 ± 2.82
dyn	1 ± 0.99	0.23 ± 0.18	1.02 ± 2.08	2.69 ± 3.16
gait	1 ± 0.99	0.21 ± 0.17	0.9 ± 1.64	2.65 ± 3.01
vanilla-MPC	1 ± 0.99	0.22 ± 0.17	0.97 ± 1.87	2.60 ± 3.03
vanilla-RL	15 ± 3.57	0.18 ± 0.15	3.27 ± 2.00	5.85 ± 2.29

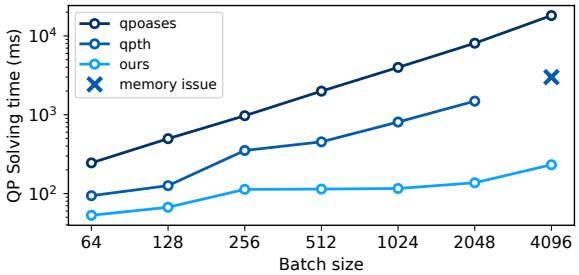


Fig. 4: Computational performance of `qpOASES`, `qpth`, and our method across different batch sizes in log scale.

VII. Conclusion

In this work, we presented an RL-augmented MPC for robust bipedal locomotion over diverse terrains. Extensive simulation results demonstrate that our method achieves significantly higher success rates and more accurate tracking compared to the baseline. Future research involves extensive hardware experiments and extending the current approach to perceptive locomotion.

References

[1] J. Köhler, E. Andina, R. Soloperto, M. A. Müller, and F. Allgöwer, "Linear robust adaptive model predictive control: Computational complexity and conservatism," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 1383–1388.

[2] C. Feller and C. Ebenbauer, "Relaxed logarithmic barrier function based model predictive control of linear systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1223–1238, 2016.

[3] C. Khazoom, S. Hong, M. Chignoli, E. Stanger-Jones, and S. Kim, "Tailoring solution accuracy for fast whole-body model predictive control of legged robots," *IEEE Robotics and Automation Letters*, 2024.

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

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

[6] J. Li and Q. Nguyen, "Force-and-moment-based model predictive control for achieving highly dynamic locomotion on bipedal robots," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 1024–1030.

[7] M.-G. Kim, D. Kang, H. Kim, and H.-W. Park, "A modular residual learning framework to enhance model-based approach for robust locomotion," *IEEE Robotics and Automation Letters*, no. 99, pp. 1–8, 2025.

[8] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[9] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.

[10] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science robotics*, vol. 7, no. 62, p. eabk2822, 2022.

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

[12] H. Jung, Z. Gu, Y. Zhao, H.-W. Park, and S. Ha, "Ppf: Pre-training and preservative fine-tuning of humanoid locomotion via model-assumption-based regularization," *IEEE Robotics and Automation Letters*, 2025.

[13] R. Reiter, J. Hoffmann, D. Reinhardt, F. Messerer, K. BaumgÃĪrtner, S. Sawant, J. Boedecker, M. Diehl, and S. Gros, "Synthesis of model predictive control and reinforcement learning: Survey and classification," *arXiv preprint arXiv:2502.02133*, 2025.

[14] Z. Gu, J. Li, W. Shen, W. Yu, Z. Xie, S. McCrory, X. Cheng, A. Shamsah, R. Griffin, C. K. Liu *et al.*, "Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning," *arXiv preprint arXiv:2501.02116*, 2025.

[15] A. Pandala, R. T. Fawcett, U. Rosolia, A. D. Ames, and K. A. Hamed, "Robust predictive control for quadrupedal locomotion: Learning to close the gap between reduced-and full-order models," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6622–6629, 2022.

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

[17] Y. Yang, G. Shi, X. Meng, W. Yu, T. Zhang, J. Tan, and B. Boots, "Cajun: Continuous adaptive jumping using a learned centroidal controller," in *Conference on Robot Learning*. PMLR, 2023, pp. 2791–2806.

[18] S. H. Bang, C. A. Jové, and L. Sentis, "Rl-augmented mpc framework for agile and robust bipedal footstep locomotion planning and control," in *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2024, pp. 607–614.

[19] Y. Chen and Q. Nguyen, "Learning agile locomotion and adaptive behaviors via rl-augmented mpc," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 436–11 442.

[20] J. Cheng, D. Kang, G. Fadini, G. Shi, and S. Coros, "Rambo: Rl-augmented model-based optimal control for whole-body loco-manipulation," *arXiv preprint arXiv:2504.06662*, 2025.

[21] H. J. Lee, S. Hong, and S. Kim, "Integrating model-based footstep planning with model-free reinforcement learning for dynamic legged locomotion," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 11 248–11 255.

[22] F. Liu, Z. Gu, Y. Cai, Z. Zhou, H. Jung, J. Jang, S. Zhao, S. Ha, Y. Chen, D. Xu *et al.*, "Opt2skill: Imitating dynamically-feasible whole-body trajectories for versatile humanoid loco-manipulation," *arXiv preprint arXiv:2409.20514*, 2024.

[23] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "Dtc: Deep tracking control," *Science Robotics*, vol. 9, no. 86, p. eadh5401, 2024.

[24] J. Li, J. Ma, O. Kolt, M. Shah, and Q. Nguyen, "Dynamic loco-manipulation on hector: Humanoid for enhanced control and open-source research," *arXiv preprint arXiv:2312.11868*, 2023.

[25] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[26] P. Gupta, J. M. Smereka, and Y. Jia, "Reinforcement learning compensated model predictive control for off-road driving on unknown deformable terrain," *arXiv preprint arXiv:2408.09253*, 2024.

[27] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," in *2019 international conference on robotics and automation (icra)*. IEEE, 2019, pp. 9784–9790.

[28] Q. Liao, B. Zhang, X. Huang, X. Huang, Z. Li, and K. Sreenath, "Berkeley humanoid: A research platform for learning-based control," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 2897–2904.

[29] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[30] M. H. Raibert, H. B. Brown Jr, M. Chepponis, J. Koechling, and J. K. Hodgins, "Dynamically stable legged locomotion," Tech. Rep., 1989.

[31] Z. Gu, Y. Zhao, Y. Chen, R. Guo, J. K. Leestma, G. S. Sawicki, and Y. Zhao, "Robust-locomotion-by-logic: Perturbation-resilient bipedal locomotion via signal temporal logic guided model predictive control," *IEEE Transactions on Robotics*, 2025.

[32] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[33] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. Pmlr, 2018, pp. 1861–1870.

[34] D. Makoviichuk and V. Makoviychuk, "rl-games: A high-performance framework for reinforcement learning," https://github.com/Denys88/rl_games, May 2021.

[35] S. H. Jeon, S. Hong, H. J. Lee, C. Khazoom, and S. Kim, "Cusadi: A gpu parallelization framework for symbolic expressions and optimal control," *IEEE Robotics and Automation Letters*, 2024.

[36] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[37] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," *arXiv preprint arXiv:1703.00443*, 2017.