2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Integrating Model-Based Footstep Planning with Model-Free Reinforcement Learning for Dynamic Legged Locomotion

Ho Jae Lee[1], Seungwoo Hong[1], and Sangbae Kim[1]

***Abstract*— In this work, we introduce a control framework that combines model-based footstep planning with Reinforcement Learning (RL), leveraging desired footstep patterns derived from the Linear Inverted Pendulum (LIP) dynamics. Utilizing the LIP model, our method forward predicts robot states and determines the desired foot placement given the velocity commands. We then train an RL policy to track the foot placements without following the full reference motions derived from the LIP model. This partial guidance from the physics model allows the RL policy to integrate the predictive capabilities of the physics-informed dynamics and the adaptability characteristics of the RL controller without overfitting the policy to the template model. Our approach is validated on the MIT Humanoid, demonstrating that our policy can achieve stable yet dynamic locomotion for walking and turning. We further validate the adaptability and generalizability of our policy by extending the locomotion task to unseen, uneven terrain. During the hardware deployment, we have achieved forward walking speeds of up to 1.5 m/s on a treadmill and have successfully performed dynamic locomotion maneuvers such as 90-degree and 180-degree turns.**

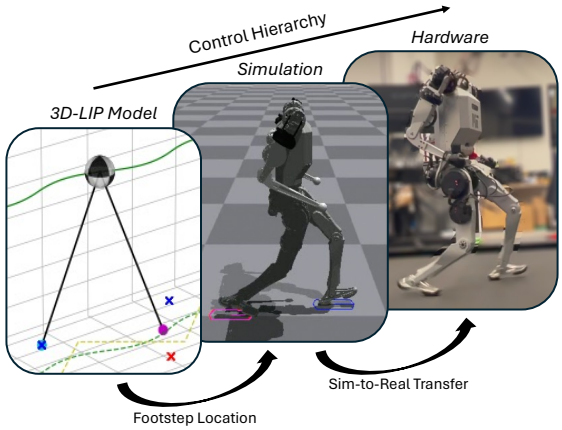


Fig. 1: Our control hierarchy that employs a 3D-LIPM to determine the desired footstep location for locomotion. We train an RL policy to track the given steps and deploy the policy on MIT Humanoid.

I. INTRODUCTION

Legged biological systems are capable of navigating through unstructured, complex, and discontinuous terrains, such as stepping stones. In the realm of legged robotics, researchers have long strived to enable legged robots to achieve mobility comparable to their natural counterparts, which would provide numerous practical real-world applications. However, designing controllers for legged robots is non-trivial because they have high degrees-of-freedom and their under-actuated floating base can only be indirectly controlled through external contact wrenches, making their equations of motion highly nonlinear and non-smooth.

Model-based control approaches, such as reactive and predictive control methods, have emerged as a highly effective strategy for solving these complex control challenges, showcasing remarkable performance in quadrupedal and bipedal robotics applications [1]–[9]. The major advantage of the model-based control approach lies in leveraging insights from physics models to predict robot behavior, thereby enhancing controller design. In particular, foot placement emerges as one of the main components in model-based control on uneven or discontinuous terrain, providing an interface to control the robot through contact forces. Numerous studies have successfully used simplified models, such as the linear inverted pendulum model (LIPM) [1]–[3] to calculate foot placements or utilized optimization-based approaches that leverage simplified models like spring-loaded inverted pendulum (SLIP) [6], single rigid body dynamics (SRBD) [10], centroidal dynamics [11] for simultaneous computation of foot placements and contact forces. However, these model-based control strategies that purely rely on simplified dynamics are inherently constrained by their simplifications and model mismatches, resulting in conservative locomotion that does not fully exploit the robot's capabilities.

Parallel to these developments, model-free Reinforcement Learning (RL) has emerged as a powerful tool for robotic control, demonstrating remarkable success in managing complex, dynamic environments [12]. The application of model-free RL to both quadrupedal and bipedal robots has showcased a great performance in the given task and robustness against external perturbations [13]–[16]. Nonetheless, the model-free RL lacks interpretability, making the process of reward shaping and hyperparameter tuning less straightforward and challenging. Furthermore, it has difficulty in generalizing learned policies to new tasks or environments without undergoing retraining. In response, numerous studies used heuristic-based references or model-based physics insights to inform and guide policy learning. Specifically, some of the studies have employed heuristic or sampling-based methods to generate footstep locations and track these references using RL policy [15], [17]. However, the absence of physical reasoning in footstep selection makes it challenging to accurately track target footstep locations while simultaneously maintaining balance without leading to instability or falls.

[1]All authors are with the Biomimetic Robotics Lab, Mechanical Engineering Department, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA (e-mail: {hjlee201, swhong, sangbae}@mit.edu).

This work was supported by NAVER Labs.

Other studies have used reduced-order models to guide the RL policy to follow the reference trajectories generated by these models [18]–[20]. However, directly tracking the reference body and joint trajectories [20] or imitating the offline motion library [18], [19] from simplified models causes the RL policy to become overly aligned with the model, restricting exploration during training. Consequently, the resulting policy may be excessively constrained by the simplified model, failing to fully utilize the potential of whole-body dynamics.

In this work, we aim to bridge the gap between these two paradigms, integrating the physics-driven insights of model-based approaches with the adaptive and robust characteristics of RL. Specifically, we propose a hierarchical control framework that employs physics-informed step placements, utilizing linear inverted pendulum (LIP) dynamics to generate target step patterns, while concurrently training an RL policy to ensure the robot adheres to these prescribed step placements. This partial guidance from the physics-based template model prevents the policy from being confined to the model and results in a stable control policy capable of dynamic locomotion tasks, such as fast walking and sharp turns. Furthermore, our approach exhibits the robustness and adaptability inherent to RL policies, extending its capability to navigate unseen, uneven terrains by dynamically adjusting desired steps during the swing phase. The effectiveness of our approach is demonstrated through simulations and hardware experiments on the MIT Humanoid robot [21], showcasing its potential in advancing robotic locomotion in complex environments (see video[1] and code[2]).

II. BACKGROUND

A single inverted pendulum that connects the supporting foot with its center of mass (CoM) via a massless telescopic leg is commonly used as a simplified model to represent bipedal locomotion [1]. By applying constraints to the inverted pendulum's motion, including a constant CoM velocity along the z-axis and a point-foot model without an actuated ankle joint, an analytical solution for the 3D-LIPM (Fig. 2a) [1] can be formulated, governed by a linearly independent equation of motion:

$$\ddot{\boldsymbol{r}} = \omega_0^2(\boldsymbol{r} - \boldsymbol{p}) \tag{1}$$

where $\boldsymbol{r} = (r_x, r_y)^T$ denotes the position of the CoM, $\omega_0 = \sqrt{g/z_0}$ indicates the natural frequency of the pendulum, and $\boldsymbol{p} = (p_x, p_y)^T$ represents the position of the foot. During the derivation of (1), it is assumed that the foot is in contact with the ground.

Integrating (1) yields the "orbital energy" [1], leading to the formulation of the Instantaneous Capture Point (ICP), which is a point on the ground that the system comes to a stop if it were to instantaneously place its foot there [22]:

$$\boldsymbol{\xi} = \boldsymbol{r} + \frac{\dot{\boldsymbol{r}}}{\omega_0} \tag{2}$$

[1]Supplementary Video Link
[2]Open-Source Code Link

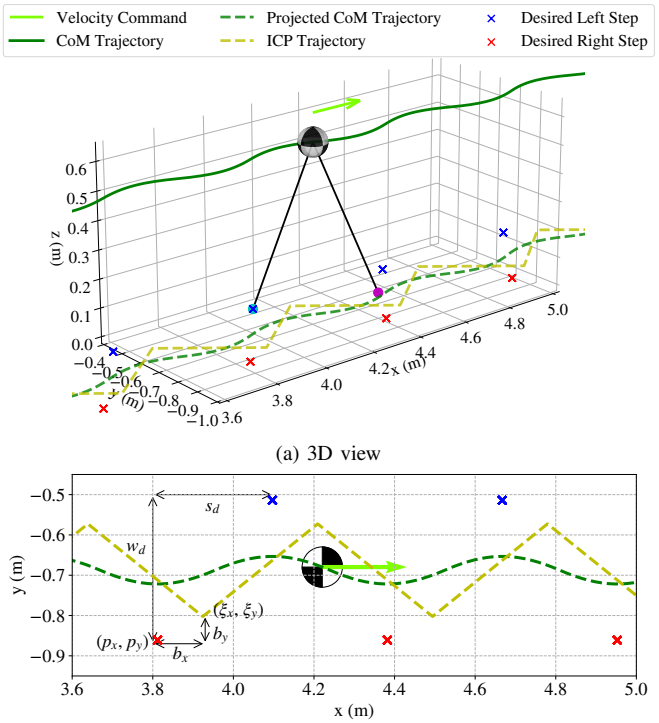


Fig. 2: Step pattern generation algorithms for 3D-LIPM from 3D (Figure 2a), 2D top-view (Figure 2b) perspective. Figure 2a depicts the LIPM with two legs. The LIP dynamics can predict the CoM trajectory (green lines, and green dashed lines). Our method calculates ICP trajectory (yellow lines) and adds offsets (b_x, b_y) to the final ICP (ξ_x^f, ξ_y^f) to determine desired step locations for tracking velocity commands. Figure 2b depicts the top view of the proposed method.

where $\boldsymbol{\xi} = (\xi_x, \xi_y)^T$ denotes the position of the ICP. By differentiating (2) with respect to time, and inserting (1) into that equation, we obtain the ICP dynamics:

$$\dot{\boldsymbol{\xi}} = \omega_0(\boldsymbol{\xi} - \boldsymbol{p}) \tag{3}$$

We can derive the solution of (3) as follows:

$$\boldsymbol{\xi}(t) = e^{\omega_0 t}\boldsymbol{\xi}(0) + (1 - e^{\omega_0 t})\boldsymbol{p} \tag{4}$$

These principles, described in (3) and (4), are pivotal for generating stable step patterns, as discussed in the next section.

III. STEP PATTERN GENERATION ALGORITHMS

In this section, we describe the process of generating a suitable step pattern for the 3D-LIPM to achieve a velocity tracking task [2], [3]. We incorporate these strategies into the RL problem in Sec. IV, ensuring the bipedal robot aligns its foot placement with calculated step locations.

Our main objective is to track the desired base velocity command. Fig. 2a outlines our step pattern generation algorithms, deriving the ICP trajectory and calculating the necessary offsets to determine the step locations for the left and right steps. Fig. 2b presents a top-view of our algorithms.

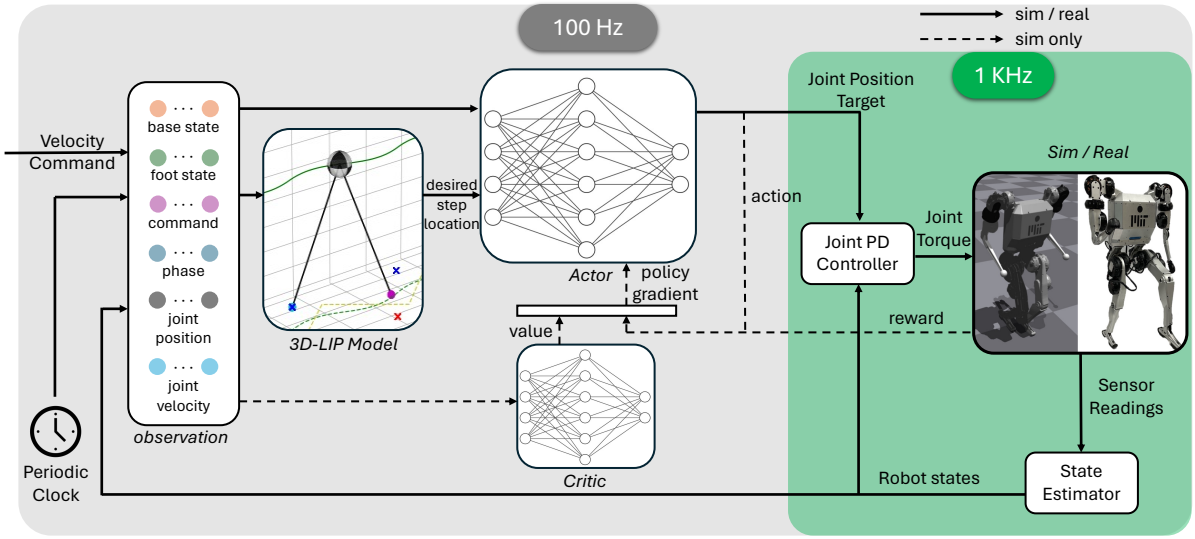


Fig. 3: Overall control diagram and training framework for both learning in simulation and deployment to hardware. Step pattern generation algorithms generate the desired step location by utilizing the robot's CoM position, velocity, and foot states. These algorithms and NN update at a frequency of 100 Hz where both the actor and critic are trained using the PPO algorithm. Once the policy (actor) outputs joint position targets, the joint PD controller is evaluated at 1 KHz, and the command torques are sent to the motor.

We assume the LIPM moves along the positive x-axis. First, we calculate the desired step length s_d based on the velocity command $\hat{\boldsymbol{v}} = (\hat{v}_x, \hat{v}_y)$:

$$s_d = |\hat{\boldsymbol{v}}| \cdot \delta T \tag{5}$$

Here, δT denotes the remaining step duration calculated by $\delta T = T_\mathrm{s} - t$, where T_s represents the user-defined step duration, and t indicates the elapsed time since the beginning of the step. At the start of each step, t is reset to 0 and progresses to T_s.

Similarly, we calculate the desired step width w_d based on the step width command $\hat{w}$:

$$w_d = |\hat{w}| \cdot \delta T / T_\mathrm{s} \tag{6}$$

Given the initial (i.e., at the beginning of the step) body state $\boldsymbol{r}^\mathrm{o} = (r_x^\mathrm{o}, r_y^\mathrm{o})$, $\dot{\boldsymbol{r}}^\mathrm{o} = (\dot{r}_x^\mathrm{o}, \dot{r}_y^\mathrm{o})$, we calculate the initial ICP $\boldsymbol{\xi}^\mathrm{o} = (\xi_x^\mathrm{o}, \xi_y^\mathrm{o})$. Then we predict the LIP's final (i.e., at the end of step) ICP $\boldsymbol{\xi}^\mathrm{f} = (\xi_x^\mathrm{f}, \xi_y^\mathrm{f})$ after δT using (4):

$$\begin{pmatrix} \xi_x^\mathrm{f} \\ \xi_y^\mathrm{f} \end{pmatrix} = \begin{pmatrix} \xi_x(\delta T) \\ \xi_y(\delta T) \end{pmatrix} = \begin{pmatrix} e^{\omega_0 \delta T} \cdot \xi_x^\mathrm{o} + (1 - e^{\omega_0 \delta T}) \cdot p_x \\ e^{\omega_0 \delta T} \cdot \xi_y^\mathrm{o} + (1 - e^{\omega_0 \delta T}) \cdot p_y \end{pmatrix} \tag{7}$$

Based on Fig. 2b, we observe that s_d and w_d can be readily expressed as follows:

$$\begin{pmatrix} s_d \\ w_d \end{pmatrix} = \begin{pmatrix} (\xi_x^\mathrm{f} - \xi_x^0) \\ (\xi_y^\mathrm{f} - \xi_y^0) + 2(\xi_y^0 - p_y) \end{pmatrix} \tag{8}$$

By inserting (8) into (7), we obtain the constant offset vector (b_x, b_y), which when added to the final ICP, guarantees the step pattern has the desired step length s_d and width w_d:

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} \xi_x^0 - p_x \\ \xi_y^0 - p_y \end{pmatrix} = \begin{pmatrix} \frac{\xi_x^\mathrm{f} - \xi_x^0}{e^{\omega_0 \delta T} - 1} \\ \frac{(\xi_y^\mathrm{f} - \xi_y^0) + 2(\xi_y^0 - p_y)}{e^{\omega_0 \delta T} + 1} \end{pmatrix} = \begin{pmatrix} \frac{s_d}{e^{\omega_0 \delta T} - 1} \\ \frac{w_d}{e^{\omega_0 \delta T} + 1} \end{pmatrix} \tag{9}$$

TABLE I: User-defined Variables for Step Pattern Generation Algorithms

Variable	Value
Velocity commands $\hat{\boldsymbol{v}}$	$U^2[-2.0,\ 2.0]$ m/s
Step width command $\hat{w}$	0.3 m
Step duration T_s	0.35 s
Base height $\hat{p}_{\mathrm{base},z}$	0.62 m
Base heading $\hat{\theta}_\mathrm{base}$	$tan^{-1}(\hat{v}_y/\hat{v}_x)$

Then we determine the desired step location $\hat{\boldsymbol{p}} = (\hat{p}_x, \hat{p}_y)^T$ by adding this constant offset vector to the final ICP [2]:

$$\begin{pmatrix} \hat{p}_x \\ \hat{p}_y \end{pmatrix} = \begin{pmatrix} \xi_x^\mathrm{f} - b_x \\ \xi_y^\mathrm{f} + (-1)^n b_y) \end{pmatrix} \tag{10}$$

where n indicates the step cycle (even n for the left step and odd n for the right step). In the case of turning in xy-plane, we modify (10) by rotating the constant offset vector:

$$\begin{pmatrix} \hat{p}_x \\ \hat{p}_y \end{pmatrix} = \begin{pmatrix} \xi_x^\mathrm{f} \\ \xi_y^\mathrm{f} \end{pmatrix} + \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \begin{pmatrix} -b_x \\ (-1)^n b_y \end{pmatrix} \tag{11}$$

where θ refers to the rotation angle around z-axis defined by $\theta = tan^{-1}(\hat{v}_y/\hat{v}_x)$. During the training, we calculate the desired step location only at the beginning of the step when $t = 0$ (i.e., $\delta T = T_\mathrm{s}$). The detailed values for each user-defined variable are given in Table I.

IV. RL PROBLEM FORMULATION

In this section, we describe our RL training framework to ensure the robot tracks the velocity commands and the step pattern generated by (11).

Fig. 3 shows the overview of our control framework. Our control policy is a fully connected neural network with 3 hidden layers, each layer with 256 nodes. The policy takes as input the robot states, step commands derived from our proposed step pattern algorithms, and user velocity commands,

TABLE II: PPO Hyperparameters

Parameter	Value
Horizon (H)	24
Adam learning rate	1×10^{-5}
Number of epochs	5
Number of mini-batches	4
Discount (γ)	0.99
Clipping parameter (ϵ)	0.2
Max gradient norm	1

and outputs the desired residual joint PD setpoints. We train our policy in the IsaacGym simulation engine using PPO [23] algorithms with parallelization of 4096 environments and input normalization. Detailed information on PPO hyperparameters can be found in Table II. Now, we introduce the state space, action space, and reward formulation for the RL problem.

A. State Space

The state space of our policy consists of the observed robot states, step commands, and user-defined velocity commands with a size of $\mathcal{S} \in \mathbb{R}^{51}$. In detail, $\mathcal{S}$ includes the base height, base linear velocity in the world frame, base angular velocity, projection of the gravity vector in the base frame, left and right foot location and heading in the base frame, left and right desired step location and heading in the base frame, velocity commands, phase clock in sine and cosine functions, joint position, joint velocity. The phase identifiers indicate the swing and stance phase of each foot through the contact scheduler. The base states are measured through the phase-based state estimator [24] that assumes the foot contact on the ground at the specified contact schedule.

B. Action Space

We define the action space $\mathcal{A} \in \mathbb{R}^{10}$ as the desired residual joint PD setpoints $\Delta\hat{\boldsymbol{q}}$, representing a deviation from the nominal joint position $\boldsymbol{q}^{\text{ref}}$ for hip yaw, hip abduction, hip pitch, knee and ankle joint respectively. The action from our policy is updated at a frequency of 100 Hz and fed into the joint PD controller. Then, the fixed-gain joint PD controller operates at 1 kHz. To be specific, the joint PD controller uses the following equation to convert the action into the desired torque command:

$$\hat{\boldsymbol{\tau}} = \mathbf{K_p}(\boldsymbol{q}^{\text{ref}} + \Delta\hat{\boldsymbol{q}} - \boldsymbol{q}) + \mathbf{K_d}(\mathbf{0} - \dot{\boldsymbol{q}}) \tag{12}$$

For the joint PD controller's gains, we have configured $\mathbf{K_p}$ to $diag(30, ..., 30)$, and $\mathbf{K_d}$ to $diag(1, ..., 1)$.

C. Rewards

We formulate the reward structure to ensure the robot tracks the desired step location while maintaining stability and adaptability. Since the desired step location is derived based on the LIPM, we incorporate specific rewards to satisfy the assumption of the LIPM. To retain the inherent flexibility and adaptability characteristic of RL policy, however, we do not impose explicit rewards to follow the LIPM's CoM trajectory.

The overall reward function is formulated as follows:

TABLE III: Regularization Rewards

Reward	Weight	Expression
Joint torques	1e-4	$-\lvert\boldsymbol{\tau}\rvert^2$
Joint torque limits	1e-2	$-\max(\lvert\boldsymbol{\tau}\rvert - 0.9\boldsymbol{\tau}_{max}, 0)$
Joint velocity	1e-3	$-\lvert\dot{\boldsymbol{q}}\rvert^2$
Joint limits	10	$-\text{clip}(\lvert\boldsymbol{q}\rvert - 0.9\boldsymbol{q}_{max}, 0, 1)$
Action smoothness 1	1e-3	$-\lvert(\boldsymbol{a}_t - \boldsymbol{a}_{t-1})/\Delta t\rvert^2$
Action smoothness 2	1e-4	$-\lvert(\boldsymbol{a}_t - 2\boldsymbol{a}_{t-1} + \boldsymbol{a}_{t-2})/\Delta t\rvert^2$
Hip joint regularization	1.25	$\exp(-(\boldsymbol{q}_{\text{hip},xz})^2/\sigma)$
Base roll-pitch velocity	1e-2	$-(\omega^2_{\text{base},x} + \omega^2_{\text{base},y})$
Base z-axis velocity	1e-1	$-\lvert v_{\text{base},z}\rvert^2$
Base tilting	1	$\exp(-(g^2_{\text{base},x} + g^2_{\text{base},y})/\sigma)$
Termination	100	$\begin{cases} -1, \text{ self-collision}, \\ -1, \lvert\boldsymbol{v}_{\text{base}}\rvert \geq 10 \text{ [m/s]}, \\ -1, \lvert\boldsymbol{\omega}_{\text{base}}\rvert \geq 5 \text{ [rad/s]}, \\ -1, g_{\text{base},x}, g_{\text{base},y} \geq 0.7, \\ -1, p_{\text{base},z} < 0.3 \text{ [m]}, \\ 0, \text{ otherwise}. \end{cases}$

$$r = r_{bh} + r_{bo} + r_{vt} + r_{cs} + r_{\text{Regularization}} \tag{13}$$

First, to address the LIPM's assumption of a constant height, we introduce a reward r_{bh} that encourages the robot to keep a constant base height $\hat{p}_{\text{base},z}$:

$$r_{bh} = \exp(-(\hat{p}_{\text{base},z} - p_{\text{base},z})^2/\sigma) \tag{14}$$

Given that the LIPM is represented solely by a point mass and lacks any orientation, it offers no direct control over the robot's orientation. Therefore, we assume that the robot's base should consistently orient towards the desired base heading $\hat{\theta}_{\text{base}}$ direction. To encapsulate this concept, the reward r_{bo} is designed:

$$r_{bo} = 2\exp(-|\hat{\theta}_{\text{base}} - \theta_{\text{base}}|/\sigma) \tag{15}$$

The desired step location calculated by step pattern generation algorithms in Sec. III results in the LIPM's passive dynamics naturally fulfilling the velocity command $\hat{\boldsymbol{v}}$. Given the robot's deviation from the LIPM, we implement the velocity tracking reward r_{vt} to ensure tracking of the velocity command $\hat{\boldsymbol{v}}$:

$$r_{vt} = 4\exp(-(\frac{\hat{\boldsymbol{v}} - \boldsymbol{v}_{\text{world}}}{1 + |\hat{\boldsymbol{v}}|})^2/\sigma) \tag{16}$$

Upon determining the desired step location, the robot must place its foot at this location for the specified step duration. The reward r_{cs} is crafted to incentivize the robot to conform to the contact schedule at the desired step location:

$$r_{cs} = 9(\mathbb{I}_{\text{r,contact}} - \mathbb{I}_{\text{l,contact}})\phi_{\text{contact}} \cdot \exp(-(||\hat{\boldsymbol{p}} - \boldsymbol{p}||_2)/\sigma) \tag{17}$$

Here, $\mathbb{I}_{\text{r,contact}}$ and $\mathbb{I}_{\text{l,contact}}$ are indicator functions for right and left foot ground contact, respectively. The contact schedule ϕ_{contact} is a continuous function that oscillates between -1 and 1 across each two-step duration $2T_{\text{s}}$:

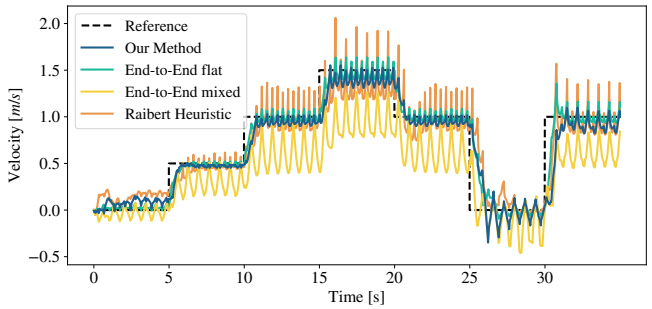


Fig. 4: Comparison of velocity tracking performance between our method, End-to-End policies trained on flat terrain versus mixed terrains (flat, rough, and gap), and Raibert heuristic policy. Commands were given in flat terrain. Our method exceeds the performance of the End-to-End approach trained on varied terrains and shows comparable results to the End-to-End policy trained exclusively on flat terrain.

$$\phi_{\text{contact}} = \frac{sin(2\pi\phi)}{\sqrt{sin^2(2\pi\phi) + 0.04}}\ , \quad \phi = \frac{t'}{2T_{\text{s}}} \tag{18}$$

where t' denotes the elapsed time from the start of the right-foot step, which is reset to 0 every two-step cycle, $2T_{\text{s}}$.

Furthermore, to mitigate any undesirable motions, a set of regularization rewards $r_{\text{Regularization}}$ is imposed to penalize excessive joint torque, velocities, unnecessary angular motion, policy termination due to falls, etc (see Table III). The reward shaping parameter σ for the exponential function is set to 0.25 during training.

V. EXPERIMENT RESULTS

We now present our simulation and hardware test results on MIT Humanoid to evaluate the effectiveness of our approach. The training process takes about three hours of wall clock time using a Nvidia GeForce 3090 GPU.

A. Simulation Results

1) Velocity tracking performance: Fig. 4 presents the velocity tracking performance of our method compared to: 1) End-to-End policy, which is trained to track the velocity commands without foot placement constraints; and 2) Raibert heuristic [25] policy, which replaces step pattern generation algorithms with Raibert heuristic. Our method and Raibert heuristic policy are trained exclusively on flat terrain. Additionally, we train two End-to-End policies: one on flat terrain; and the other on multiple terrains, including not only flat but also rough and gap-containing terrains. This plot depicts that our method exceeds the velocity tracking performance of the End-to-End policy trained across these diverse terrains. It also shows that our tracking accuracy is comparable to the End-to-End policy trained solely on flat terrain, which is recognized for its proficiency in single-task scenarios. The results validate that our method reliably tracks velocities up to 2.0 m/s. Furthermore, we observe that with the application of lateral velocity commands, the robot can execute dynamic maneuvers, including 90-degree and 180-degree turns.

2) Learning desired step duration: Fig. 5 shows the step duration T_{s} learned by the policy. Throughout the training phase, T_{s} was fixed at 0.35 seconds, indicating the ground contact duration for a single step. In our setting, a foot is considered to be in contact with the ground if either the toe or heel is touching the ground. This behavior is encouraged through a contact schedule reward (17). The plot confirms that the policy has successfully learned to maintain ground contact for 0.35 seconds for each leg, alternating between the left and right. This consistent step sequence is subsequently beneficial for employing a phase-based state estimator in hardware deployment.

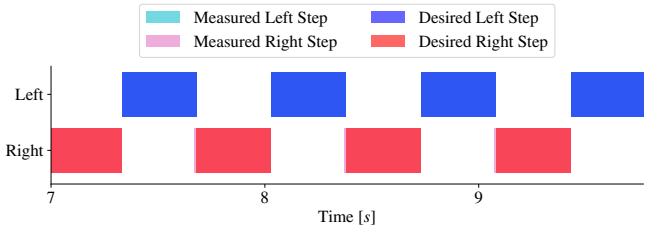


Fig. 5: Learned foot contact schedule. The step duration T_{s} that was set to 0.35 seconds was encouraged by a contact schedule reward. The error between the measured and desired foot contact schedule is less than 0.01 seconds.

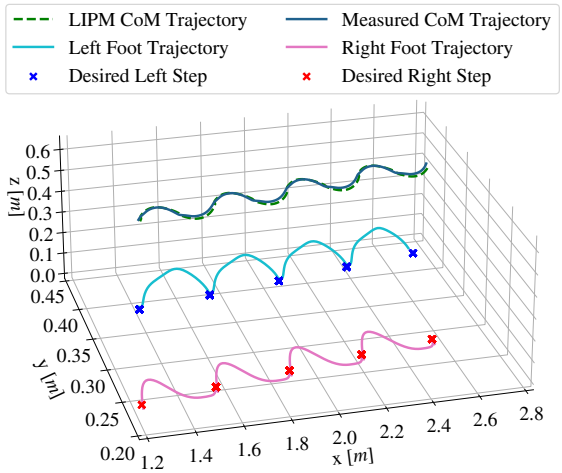
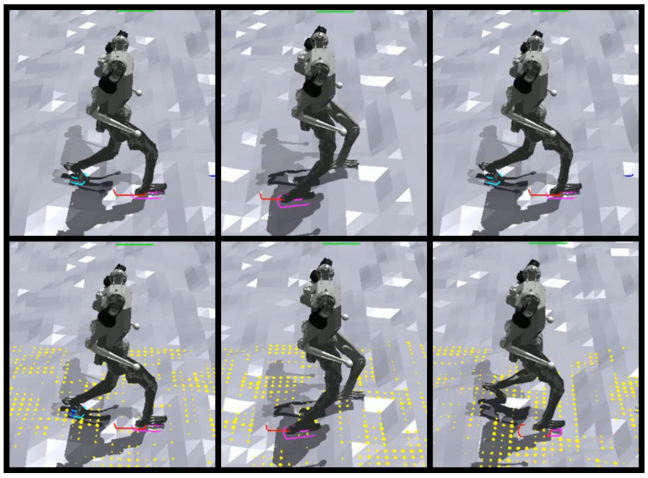


Fig. 6: Tracking desired step location and its resulting foot trajectory. Both left and right foot trajectories show a consistent and smooth path that culminates in an accurate touchdown at the target step locations. A notable observation is the robot's measured CoM trajectory exhibits a close correspondence to the LIPM CoM trajectory

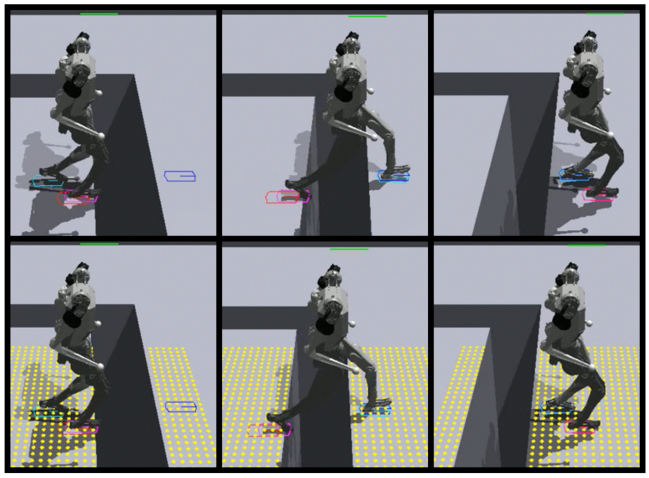
3) Tracking desired step location: Fig. 6 shows the robot's successful tracking of desired step location generated by step pattern generation algorithms. Both right and left foot trajectories form a smooth and regular trajectory ensuring accurate touch down on the target step location. Notably, the measured CoM trajectory of the robot closely aligns with the analytical LIPM CoM trajectory. This behavior is attributed to the implementation of the rewards that encourage the robot to satisfy the assumptions of LIPM.

4) Extension to rough terrain and gap terrain:

To evaluate the adaptability of our policy to unseen and uneven terrains, we conducted tests on both rough



(a) Rough terrain



(b) Gap terrain

Fig. 7: Adaptive locomotion on varied terrain. For rough terrain (Fig. 7a), the policy dynamically updates the desired step location by modifying δT in the step pattern generation algorithms to compensate for the irregularities in terrain that affect the robot's CoM height. On gap terrain (Fig. 7b), the policy adjusts step location to the nearest flat surface, demonstrating the robot's capability to navigate discontinuities in the surface and maintain a forward velocity $\hat{v}_x$.

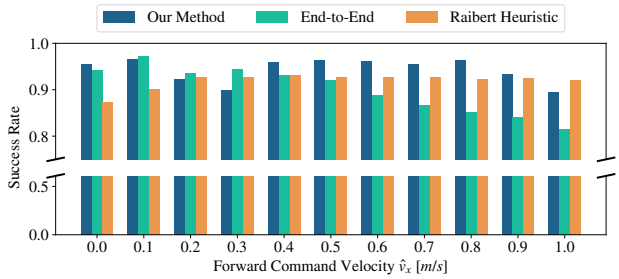


Fig. 8: Success rate for walking on the rough terrain. To navigate rough terrain, we modify the robot's desired step locations at each time step to account for the deviation from the LIPM dynamics caused by uneven terrain. These adjustments, along with step elevation refinements based on heightmap data, helped maintain the robot's balance while tracking velocity command. Our method proved more effective than an End-to-End and Raibert heuristic policy trained on flat terrain, as evidenced by on average a higher success rate in maintaining forward velocity without falling.

and gap terrain (Fig. 7). For the rough terrain, (Fig. 7a), we implemented dynamic adjustments of the desired step location by modifying δT in equations (5)-(7) every time step. This approach compensates for the inevitable deviations from the LIP dynamics due to the rough terrain's impact on the constancy of the robot's CoM height. Additionally, we refined the desired step elevation in accordance with the ground height data obtained from a heightmap. The efficacy of our method was quantified by comparing it to an End-to-End policy and Raibert heuristic policy, originally trained on flat terrain, using a success metric defined by the robot's ability to maintain a predetermined forward velocity command $\hat{v}_x$ for five seconds without falling. As depicted in Fig. 8, our policy showed on average a higher success rate. In gap terrain scenarios (Fig. 7b), if the desired step location falls into a gap, we adjust it to the closest flat ground using heightmap data. Through these deployments on both rough and gapped terrains, we have validated the robustness and adaptability of our policy: it can successfully modify the desired step location in response to real-time terrain alterations, thereby sustaining effective locomotion.

B. Hardware Results

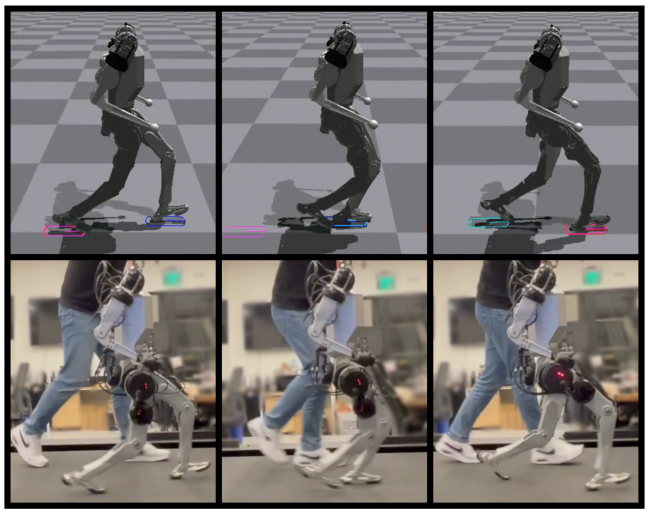
We successfully transferred the policies developed in simulation to robot hardware, showcasing the robust sim-to-real transfer capabilities of our policies (Fig. 9). The robot demonstrated the ability to maintain a consistent height and precisely track the desired step locations for the given step duration T_s. To compensate for state estimator noise, we dynamically modified the step locations at each timestep. The performance was evaluated through two specific locomotion tasks:

1) Forward walking: We evaluated the robot's ability to follow a forward velocity command on flat terrain using a treadmill, as shown in Fig. 9a, confirming its capacity to walk at speeds up to 1.5 m/s. Notably, the robot demonstrated a heel-to-toe motion closely resembling human walking. Despite the noise in the base linear velocity from the state estimator, the policy enabled stable walking while accurately tracking velocity commands, as shown in Fig. 10a.

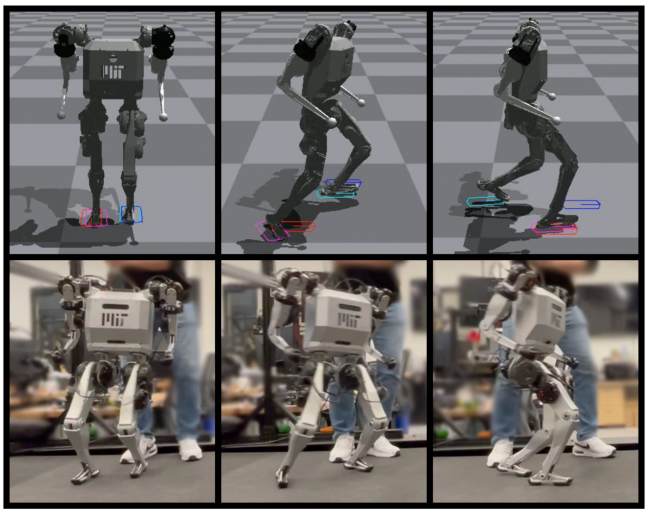
2) Dynamic turning: Dynamic locomotion tasks including 90-degree and 180-degree turns were evaluated, with the results showcased in the supplementary video. Due to spatial limitations of the testing area, only small lateral velocity commands could be issued, resulting in the robot's inability to track these commands precisely. However, the robot was still able to execute stable turns as demonstrated in Fig. 9b, and Fig. 10b.

VI. CONCLUSION AND FUTURE WORKS

In this work, we present an approach that combines LIPM with RL to learn the policy capable of accurately tracking

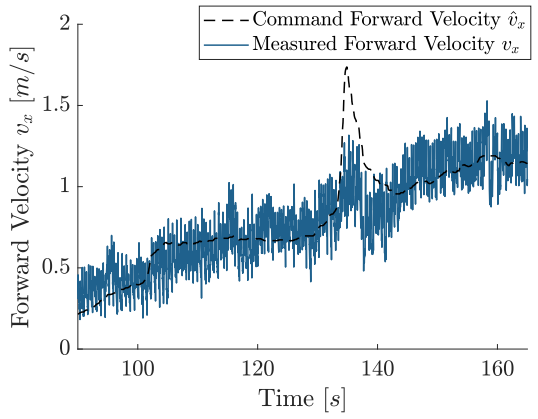


(a) Forward Walking

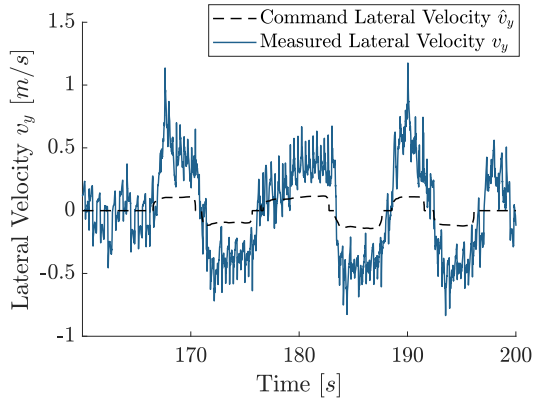


(b) 90° Turning

Fig. 9: Hardware experiment results for forward walking and 90-degree turning. We successfully achieved forward walking at speeds up to 1.5 m/s and executed 90-degree turns, both featuring a heel-to-toe motion during touchdown similar to human walking. These motions in the hardware precisely mirrored those observed in the simulation environment.



(a) Forward Walking



(b) Turning

Fig. 10: Velocity tracking plot for both forward walking and turning. Fig. 10a corresponds to Fig. 9a, and Fig. 10b corresponds to Fig. 9b. Despite the presence of noise in the base linear velocity readings from the state estimator, our policy is able to track the velocity command and execute the given locomotion tasks.

desired step locations determined by LIP dynamics. Specifically, our control framework forward predicts the robot states and determines the desired step location to track a given velocity command based on LIP dynamics. We demonstrated our approach on MIT Humanoid and confirmed that tracking these steps enables stable forward walking and dynamic turning. The learned policy further showcased flexibility and adaptability by adjusting desired steps during the swing phase proving its extendability to unseen and uneven terrains. We were able to deploy our policy on MIT Humanoid achieving a forward walking speed of 1.5 m/s and dynamic 90 and 180-degree turning.

In future work, our aim is twofold: 1) We plan to incorporate vision algorithms into our system to detect the height of the terrain. This will allow us to identify stable stepping locations, enhancing the robot's ability to navigate real-world uneven terrain. 2) We aim to refine our method of determining desired step locations by replacing the LIP dynamics with whole-body dynamics, employing a model predictive controller. This refinement is expected to further improve our control framework to predict better step locations across various locomotion tasks.

ACKNOWLEDGMENT

We thank the members of the Biomimetic Robotics Laboratory at MIT for insightful discussions and feedback on the paper. We especially thank Se Hwan Jeon, Elijah Stanger-Jones, and Charles Khazoom for their helpful support in setting up and conducting the hardware experiments.

REFERENCES

[1] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ International*

Conference on Intelligent Robots and Systems, IEEE, vol. 1, 2001, pp. 239–246.

[2] A. L. Hof, "The 'extrapolated center of mass' concept suggests a simple control of balance in walking," *Human movement science*, vol. 27, no. 1, pp. 112–125, 2008.

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

[4] C. Semini, V. Barasuol, J. Goldsmith, M. Frigerio, M. Focchi, Y. Gao, and D. G. Caldwell, "Design of the hydraulically actuated, torque-controlled quadruped robot hyq2max," *IEEE/Asme Transactions on Mechatronics*, vol. 22, no. 2, pp. 635–646, 2016.

[5] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[6] M. Rutschmann, B. Satzinger, M. Byl, and K. Byl, "Nonlinear model predictive control for rough-terrain robot hopping," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 1859–1864.

[7] 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.

[8] S. Hong, Y. Um, J. Park, and H.-W. Park, "Agile and versatile climbing on ferromagnetic surfaces with a quadrupedal robot," *Science Robotics*, vol. 7, no. 73, eadd1017, 2022.

[9] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous robots*, vol. 40, pp. 429–455, 2016.

[10] G. Bledt and S. Kim, "Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 6316–6323.

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

[12] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

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

[14] 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, eabk2822, 2022.

[15] H. Duan, A. Malik, M. S. Gadde, J. Dao, A. Fern, and J. Hurst, "Learning dynamic bipedal walking across stepping stones," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 6746–6752.

[16] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Conference on Robot Learning*, PMLR, 2020, pp. 317–329.

[17] H. Duan, A. Malik, J. Dao, A. Saxena, K. Green, J. Siekmann, A. Fern, and J. Hurst, "Sim-to-real learning of footstep-constrained bipedal dynamic walking," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 10 428–10 434.

[18] K. Green, Y. Godse, J. Dao, R. L. Hatton, A. Fern, and J. Hurst, "Learning spring mass locomotion: Guiding policies with a reduced-order model," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3926–3932, 2021.

[19] R. Batke, F. Yu, J. Dao, J. Hurst, R. L. Hatton, A. Fern, and K. Green, "Optimizing bipedal maneuvers of single rigid-body models for reinforcement learning," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, IEEE, 2022, pp. 714–721.

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

[21] A. SaLoutos, E. Stanger-Joncs, Y. Ding, M. Chignoli, and S. Kim, "Design and development of the mit humanoid: A dynamic and robust research platform," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, IEEE, 2023, pp. 1–8.

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

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] MIT Biomimetics, *Cheetah-Software: Software for the MIT Mini Cheetah*, `https://github.com/mit-biomimetics/Cheetah-Software`, Accessed: 2023-09-28, 2023.

[25] M. H. Raibert, H. B. Brown, M. Chepponis, E. Hastings, J. Koechling, K. N. Murphy, S. S. Murthy, and A. Stentz, *Dynamically stable legged locomotion*. Department of Computer Science, Carnegie-Mellon University, 1983.