Learning a Vision-Based Footstep Planner for Hierarchical Walking Control

Minku Kim, Brian Acosta, Pratik Chaudhari and Michael Posa

***Abstract*— Bipedal robots demonstrate potential in navigating challenging terrains through dynamic ground contact. However, current frameworks often depend solely on proprioception or use manually designed visual pipelines, which are fragile in real-world settings and complicate real-time footstep planning in unstructured environments. To address this problem, we present a vision-based hierarchical control framework that integrates a reinforcement learning high-level footstep planner, which generates footstep commands based on a local elevation map, with a low-level Operational Space Controller that tracks the generated trajectories. We utilize the Angular Momentum Linear Inverted Pendulum model to construct a low-dimensional state representation to capture an informative encoding of the dynamics while reducing complexity. We evaluate our method across different terrain conditions using the underactuated bipedal robot Cassie and investigate the capabilities and challenges of our approach through simulation and hardware experiments.**

I. Introduction

Bipedal robots hold immense potential for traversing unstructured terrains, making them invaluable for applications such as search and rescue and disaster response [1]. Humans demonstrate remarkable adaptive locomotion through a combination of proprioceptive feedback and visual perception. When walking, we simultaneously evaluate our surroundings for safe and stable footholds while planning our next step. This seamless integration of sensory information and control is essential for effective navigation in outdoor environments.

To replicate this behavior, vision-based locomotion controllers generally follow a modular framework composed of perception, planning, and control. The perception module processes visual data to build a spatial map of the local environment, which then informs a high-level planner to determine foot placements or motion trajectories. Finally, a low-level controller translates these plans into actuator commands. Although this pipeline has demonstrated success in structured settings, it often relies on handcrafted visual features and model-based strategies that are sensitive to noise and typically limited to simplified terrain assumptions such as piecewise-planar surfaces [2], [3], [4], [5].

An important component in achieving tractable planning for bipedal locomotion is the use of reduced-order models (ROMs), which are simplified representations that capture the key dynamics of complex robotic systems. These models offer interpretable and computationally efficient formulations, making them useful for real-time motion planning and control. Although ROMs have traditionally been employed in model-based control frameworks [6], [7], recent research has demonstrated their usefulness within reinforcement learning frameworks as well [8], [9], [10].

The authors are with the General Robotics, Automation, Sensing and Perception (GRASP) Laboratory, University of Pennsylvania, Philadelphia, PA, 19104, USA {minkukim, bjacosta, pratikac, posa}@seas.upenn.edu

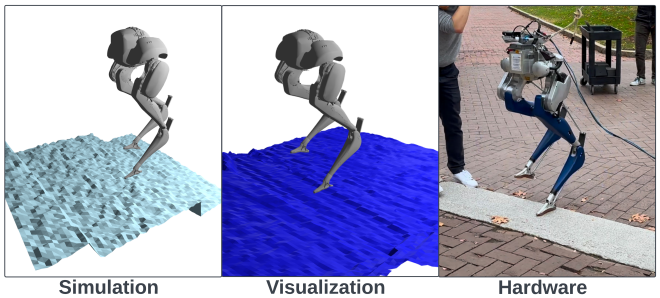

Fig. 1. The system incorporates local terrain information through an elevation map, enabling vision-based footstep planning via a reinforcement learning (RL) policy trained in simulation. The approach is validated on hardware. Left: Cassie training in simulation. Center: Visualization of the real-world elevation map. Right: Hardware evaluation.

Recent developments in reinforcement learning (RL) have significantly advanced the integration of visual perception and locomotion control by enabling both end-to-end control policies [11], [12] and hybrid control architectures [13], [14] that operate directly on raw or minimally processed visual inputs. These approaches reduce the reliance on manually engineered features, thereby increasing flexibility and generalization. However, this introduces challenges in transferring policies from simulation to real-world hardware.

Inspired by recent works, we propose a vision-based hierarchical control framework that integrates an RL-based footstep planner with a low-level operational space controller (OSC). The RL policy utilizes visual and low-dimensional state representation inputs to generate 3D footstep placements in real-time, while the low-level OSC tracks the spline trajectories derived from these footsteps.

The main contributions of this paper are:

- A vision-based hierarchical controller that uses a single depth camera and a reduced-order model to enable efficient and interpretable 3D footstep planning via reinforcement learning.
- Hardware validation of the full pipeline on both structured and unstructured terrains, with benchmarking against a model predictive control (MPC) baseline in simulation.
- An analysis of the ALIP model as a policy input and hierarchical control structure, showing limitations on complex terrain and impact on sim-to-real transfer.

II. Related Work

A. Optimal Control for Bipedal Locomotion

Model-based control for bipedal locomotion is often structured as an optimal control problem, where the multi-body dynamics are embedded as constraints in the control formulation [15]. However, due to the high dimensionality of full-order dynamics in bipedal robots, this formulation becomes computationally impractical for real-time optimization. Consequently, simplified reduced-order models, such as the Linear Inverted Pendulum (LIP) model [16] are used for online control along with its variations, including SLIP [17], ALIP [7], and H-LIP [18]. Controllers that use reduced-order dynamics with Model Predictive Control (MPC) for footstep planning have been successfully deployed in real-world environments [19]. However, there still remains limited investigation into robust vision-based control strategies for bipedal robots in uneven terrain, where previous methods have lacked robustness against noisy visual inputs such as poor lighting, making the pipeline unreliable [2], [3], [20].

B. Reinforcement Learning for Bipedal Locomotion

In response to the limitations of traditional control methods, recent research has actively explored deep reinforcement learning (DRL) as an alternative. Leveraging advancements in computation and physics simulations, DRL has enabled robust walking controllers capable of navigating diverse terrains in simulation and real-world environments [21], [22] and learning unified frameworks that can perform various dynamic gaits such as walking, running, and jumping while maintaining robustness against perturbations [23]. Recent work has also shown that hierarchical control architectures that combine model-based methods with DRL can enhance generalizability, interpretability, and sample efficiency by decomposing the locomotion problem into separate layers of objectives [9], [13], [14], [24]. Learning approaches demonstrate better performance in visual-locomotion integration compared to optimal control methods for both quadrupeds and bipeds, primarily due to their capacity to establish direct mappings between visual inputs and motor commands or footstep planning [11], [12], [11], [25].

C. Sim-To-Real Transfer

Despite the promising capabilities of DRL in developing locomotion controllers, several challenges emerge when deploying these controllers on physical robots. The primary limitation is due to modeling errors between the simulated environment and the real-world environment, making direct transfer of policies from simulation to hardware difficult. Domain randomization [26] has emerged as a prevalent approach to bridge this reality gap. Instead of carefully tuning model parameters to match real-world conditions, domain randomization involves extensively randomizing the simulated environment. By exposing the policy to a range of model distributions, the policy learns the distribution shift between the real-world and simulated environment [22], [27].

III. Background

A. Angular Momentum Linear Inverted Pendulum model

The classical Linear Inverted Pendulum (LIP) model [16] simplifies legged robots to a point mass at the center of mass (CoM) supported by a massless leg. The Angular Momentum Linear Inverted Pendulum (ALIP) model [7] is a reparameterization of the traditional LIP model, where the linear velocity of the CoM is replaced by the angular momentum about the contact point as the velocity variable. In the idealized case, the ALIP and LIP are mathematically equivalent. However, for real robotic systems with distributed mass and articulated limbs, the ALIP model is less sensitive to model error introduced by the movement of these limbs. In this work, we use a mass-normalized ALIP model, where the angular momentum is divided by the robot's mass to give the state components similar magnitudes:

$$\dot{x} = \begin{bmatrix} \dot{x}_{\text{com}} \\ \dot{y}_{\text{com}} \\ \dot{l}_x \\ \dot{l}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{H} \\ 0 & 0 & -\frac{1}{H} & 0 \\ 0 & -g & 0 & 0 \\ g & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{\text{com}} \\ y_{\text{com}} \\ l_x \\ l_y \end{bmatrix}$$

where x_{CoM} and y_{CoM} represent the horizontal positions of the CoM in the x and y directions, and l_x and l_y are the mass-normalized horizontal components of the angular momentum about the contact point in the x and y directions, and H denotes the height of the CoM.

B. Reinforcement Learning

In a standard reinforcement learning (RL) task, an agent engages in a sequential decision-making process, interacting with an environment by observing the current state, selecting actions based on a policy, and receiving rewards that guide future decisions. The problem is modeled using a Markov Decision Process (MDP) defined by a tuple of (S, A, P, R, γ), where S represents a set of states s, A denotes a set of actions a. $P(s'|s,a)$ describes the probability of transitioning from the current state s to the next state s' when an action a is taken. $R(s,a)$ gives an immediate scalar reward for each transition made from a state and action pair. The goal of RL is to learn an optimal policy π_θ^* that maximizes the expected value of the cumulative rewards and is formalized as:

$$\max_\theta J(\theta; s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0\right] \tag{1}$$

IV. Learning and Control Architecture

A. Perception Module

We use an Intel RealSense D455 to provide point-cloud updates to a robot-centric elevation mapping framework [28]. The D455 is mounted to the pelvis and pointed downward in front of the robot. We pre-process the point clouds to mask out the robot's legs before inputting the point clouds to the elevation map. The map is updated at 30 Hz based on the most recent point-cloud measurements. The robot state is estimated by a contact-aided invariant EKF [29], which uses simulated sensor measurements (encoder and IMU values) during training, and real sensor measurements

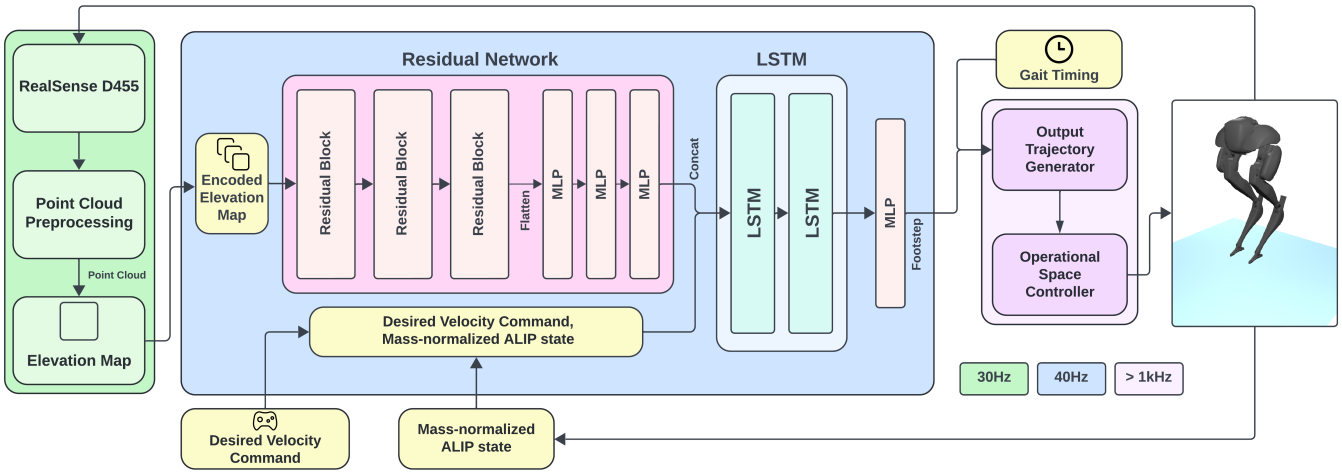

Fig. 2. Overview of the system diagram. The perception module (green) generates elevation maps at 30Hz from a RealSense D455. The high-level footstep policy (blue) outputs footstep actions at 40Hz. These actions are sent to the low-level controller (purple) for joint control.

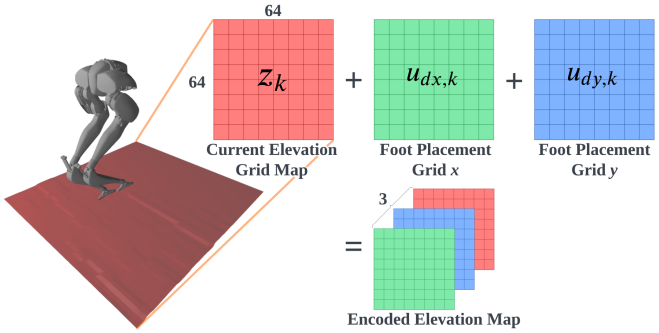

Fig. 3. The elevation map is cropped to a 64×64 grid and concatenated with XY footstep location grids. The central position of these grids corresponds to the desired footstep placement, which is calculated using an ALIP trajectory based on the velocity command and a constant stance width of 20 cm.

when deployed on hardware. This state estimate is also used to compute the ALIP state observation. The map is cropped into a 64×64 grid with a resolution of 0.025 meters per cell, which provides sufficient terrain detail and aligns with the range of the camera. Unknown heights are filled via nearest-value interpolation, and a median filter smooths out the values. The elevation map is tiled with a grid of potential footstep locations, centered with the desired footstep position (u_{dx}, u_{dy}) computed from a periodic ALIP trajectory to provide structured spatial information (Fig. 3).

B. Reinforcement Learning for Footstep Planning

The RL footstep planner uses the elevation map, ALIP states, and user-provided velocity commands to determine the footstep location local to the stance frame. The RL footstep planner is trained in simulation and then transferred to the physical robot. The policy is first pre-trained with ALIP trajectories from a blind Linear Quadratic Regulator (LQR) controller before being fine-tuned with model-free RL. The training process randomizes the terrain, velocity command, and model parameters using a curriculum learning approach to facilitate stable training and sim-to-real transfer.

C. Task-Space Objectives for Whole-Body QP

To realize the output of the learned footstep planner on the robot, we use an inverse-dynamics operational-space control QP [30]. The gains can be seen in Table I, where the weight and gain matrices are diagonal and are represented as vectors. The swing foot trajectory is represented as a single-segment polynomial spline that ends at the target footstep location. This spline is replanned for every new footstep target with a quadratic program (QP) that ensures continuity of the desired position, velocity, and acceleration, as well as minimizing swing foot acceleration and distance of the spline's midpoint from a target midpoint, which ensures ground clearance [31]. The center of mass height is controlled to a virtual plane that defines the ALIP model for the current and upcoming stance foot. The swing foot angle is also controlled to be parallel to this plane. To fully control Cassie's remaining degrees of freedom, we control the pelvis pitch and roll to zero, the swing leg yaw angle to zero, and the pelvis yaw rate to match a turning rate commanded by an operator. The commanded pelvis yaw rate is zero during training.

TABLE I

FEEDBACK GAINS FOR THE OSC CONTROLLER

OSC Objective	W	K_p	K_d
Toe joint angle	1	1500	10
Hip yaw angle	2	100	4
Pelvis [x, y]	[2, 4]	[200, 200]	[10, 10]
Pelvis heading [yaw]	0.02	0	10
CoM [z]	10	80	5
Foot [x, y, z]	[4, 4, 2]	[400, 400, 400]	[20, 20, 25]

V. Learning a Vision-Based Footstep Planner

A. Policy Design

The policy π_θ takes as inputs the desired velocity v_{des} provided by the operator, the mass-normalized ALIP state s_{ALIP}, and an encoded elevation map $M_{elevation}$ and outputs the foot placement coordinates in the local stance frame $[p_x, p_y, p_z]$. The use of task space actions improves sample efficiency and enables broader exploration to discover solutions compared to joint space actions, as shown in [32].

The policy architecture consists of two main components (Fig. 2): a residual network and a Long Short-Term Memory (LSTM) network. The residual network processes the encoded elevation map $M_{elevation}$ to produce a latent representation, which is combined with the current observation vector

TABLE II

REWARD AND PENALTY TERMS

Terms	Name	Value
Reward	Forward Velocity (r_{v_x})	$0.5 \times e^{-2\|v_{x,des} - v_x\|}$
	Lateral Velocity (r_{v_y})	$0.25 \times e^{-2\|v_{y,des} - v_y\|}$
	Height (r_z)	$0.3125 \times e^{-4\|a_{z,GT} - a_z\|}$
	Pelvis Stability (r_ϕ)	$0.1875 \times e^{-2\|\omega_z\|}$
	Action Smoothness (r_{a_t})	$0.125 \times e^{-3\|a_t - a_{t-1}\|}$
	Action Regulation (r_{reg})	$0.125 \times e^{-2\|a_{x,y,des} - a_{x,y}\|}$
Penalty	Tracking Penalty (p_{track})	$e^{3.5(err - 0.05)} - 1$
	Torque Penalty (p_τ)	$-0.000007 \times \sum_i \tau_i^2$
	Edge Penalty (p_{edge})	$1.5 \times \mathbf{1}_{\{step\ on\ edge\}}$
	Collision Penalty ($p_{collision}$)	$1.5 \times \mathbf{1}_{\{collision\ with\ terrain\}}$

$[v_{des}, s_{ALIP}]$. This serves as input to the LSTM network, which generates the actions. Prior to sending the actions out, the actions are concatenated with gait timing information to ensure temporal consistency with the gait cycle. The actor and critic networks do not share any layers.

B. Reward Function

We denote v as the linear velocity, a as the footstep action, ω as the angular velocity, and τ as the torque. The reward function is the sum of the reward terms and penalty terms shown in Table II. The total reward is constrained to be non-negative; If the cumulative penalty exceeds the cumulative reward, the total reward is set to zero. This design choice helps prevent the destabilizing effects of large penalties and promotes more stable policy learning.

$$r = \min\left(\mathbf{w}_r^\top \begin{bmatrix} r_{v_x} \\ r_{v_y} \\ r_z \\ r_\phi \\ r_{a_t} \\ r_{reg} \end{bmatrix} - \mathbf{w}_p^\top \begin{bmatrix} p_{track} \\ p_\tau \\ p_{edge} \\ p_{collision} \end{bmatrix}, 0\right) \tag{2}$$

The reward terms r_{v_x} and r_{v_y} encourage the policy to track the desired velocities, while r_ϕ promotes pelvis stability by minimizing yaw angular velocity. The r_z term aligns footstep heights with terrain elevation, and r_{a_t} encourages action continuity within the same stance. Finally, r_{reg} regulates actions to remain close to optimal footsteps determined by the LQR equation. The penalty term p_{track} penalizes deviations in swing foot trajectory tracking, p_τ penalizes torque usage to discourage the policy from generating excessively large torque commands, p_{edge} discourages stepping near edges detected by a Sobel filter [33], and $p_{collision}$ penalizes front foot collisions with the terrain.

Excluding edge penalties led to foot placements near stair edges, resulting in frequent slips and elevation map artifacts caused by the drift correction. Without the collision penalties, Cassie developed a toe-probing strategy before stepping up. While this strategy is viable in simulation, this behavior results in hardware failures.

C. Policy Training

The Proximal Policy Optimization (PPO) algorithm [34] is used to learn the footstep controller in a Drake simulation environment [35]. To enhance the accuracy of the value estimation and speed up training, we employ an asymmetric actor-critic in which only the critic has access to the privileged information. Unlike the actor, the critic additionally receives the ground truth height map, joint positions, and pelvis pose. We include a symmetric mirror loss [36] to the original PPO objective function to discourage asymmetric footsteps and keep a healthy symmetric gait:

$$L_{mirror}(\theta) = w \sum_{i=0}^{N} \|\pi_\theta(s_i) - \Psi_{act}\left(\pi_\theta\left(\Psi_{obs}(s_i)\right)\right)\|^2 \tag{3}$$

where π_θ is the policy, s_i represents the ith observation, $\Psi_{act}(\cdot)$ and $\Psi_{obs}(\cdot)$ mirrors the actions and observations, and N is the number of rollout samples. The coefficient w defines the importance of symmetry, and $w = 2$ is used for training.

We initialize the agent in a randomized pose and swing phase. The commanded velocities are sampled from $v_x \in [-0.8, 0.8]$, $v_y \in [-0.4, 0.4]$ for flat terrain and $v_x \in [0, 0.8]$, $v_y \in [-0.4, 0.4]$ for non-flat terrain. Backward walking is restricted to only flat terrains.

The policy is trained across six terrain categories (Fig. 4): (1) flat terrain; (2) flat terrain with randomly distributed obstacles of varying dimensions; (3) flat terrain with randomly distributed blocks of varying sizes; (4) stairs with varying width and height; (5) stairs with slopes; (6) slopes with varying angles. Episodes are conducted for 400 timesteps, equivalent to 10 seconds in simulation time, across random terrains and modeling parameters. Early termination occurs under three conditions: (1) foot-to-pelvis distance is less than 20 cm, indicating a fall; (2) the magnitude of swing foot tracking error exceeds 50% or (3) self-collision.

TABLE III

TERRAIN CATEGORIES

Terrain	Parameters	Range
Flat	×	×
Flat w/ Obstacle	Obstacle Dimension XYZ [m]	[0.2, 0.5]
	No. of Obstacle	[30, 40]
Block	Block Dimension XY [m]	[0.5, 1.0]
	Block Dimension Z [m]	[0.05, 0.15]
	No. of Block	[10, 20]
Stair	Stair Height [m]	[0.075, 0.17]
	Stair Width [m]	[0.5, 1.5]
Stair w/ Slope	Stair Height [m]	[0.07, 0.14]
	Stair Width [m]	[0.8, 1.6]
	Slope Angle [rad]	[0.03, 0.07]
Slope	Slope Angle [rad]	[0.1, 0.34]

VI. SIM-TO-REAL TRANSFER

A. Domain Randomization

To address modeling and measurement uncertainties between the simulated and real-world environments, we introduce randomization across dynamics parameters and observable states. Empirically, we found that randomizing all of the parameters in Table IV is necessary for sim-to-real. All randomization follows a uniform distribution sampled within a predefined range shown in Table IV. We introduce noise into the mass-normalized ALIP states to tackle discrepancies between simulation and hardware due to the differences in dynamics during ground contact. In addition, we perturb the elevation map through a combination of local and global displacements. At the beginning of each episode and at every

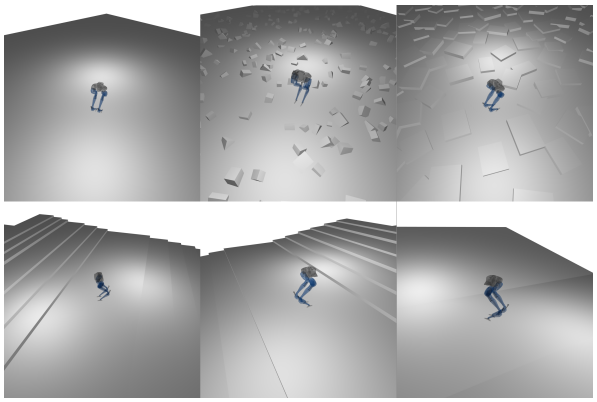

Fig. 4. Terrain types used for training. Top row (left to right): flat, flat with obstacles, and block terrains. Bottom row (left to right): stairs, stairs with slopes, and sloped terrains.

TABLE IV
DOMAIN RANDOMIZATION PARAMETERS

	Parameters	Range
Dynamics Model	PD Gains	$[0.5, 1.5] \times$ Default
	Joint Damping	$[0.5, 2.5] \times$ Default
	Link Mass	$[0.6, 1.4] \times$ Default
	ALIP state	$[-0.03, 0.03]$
	Friction Coefficient	$[0.3, 1.1]$
Elevation Map	Shift XY per episode	$[-0.03, 0.03]$m
	Shift XY per timestep	$[-0.02, 0.02]$m
	Shift Z per episode	$[-0.02, 0.02]$m
	Shift Z per timestep	$[-0.01, 0.01]$m
	Uniform Noise	$[-0.02, 0.02]$m
	Point Cloud Bias XYZ	$[-0.03, 0.03]$m
Communication	Delay	$[0, 0.025]$s
Perturbations	Force XY on pelvis	$[-20, 20]$N
	Force Z on pelvis	$[-10, 10]$N

timestep, the entire elevation map is shifted along the x, y, z axes. Uniform noise is further applied to the elevation map to regularize the encoder. To emulate imperfections in real-world sensor calibration, we also introduce systematic biases in the point cloud data along the x, y, z directions.

B. Curriculum Learning

We adopt a curriculum that adjusts the environment and domain randomization parameters for stable training. In the initial phase, training is conducted on flat terrain, staircases with a maximum height of 10 cm, and slopes. During this phase, domain randomization is excluded, as premature introduction of these elements causes policy divergence and cheating of the agent. After convergence of the initial stage, we include all terrain types and apply domain randomization. We constrain the probability of the flat terrain at 10% to avoid catastrophic forgetting of learned negative velocity commands since the flat terrain is the only terrain type that associates with negative x direction velocity.

C. Elevation Map Drift Correction

Due to the lack of direct global position sensing on the perception module, we observe consistent drift in the vertical (z) direction of the state estimator. This drift is primarily caused by impacts during foot contact, which perturb the floating base estimate over time, and as a result, the elevation map underestimates the terrain height. To address this problem,

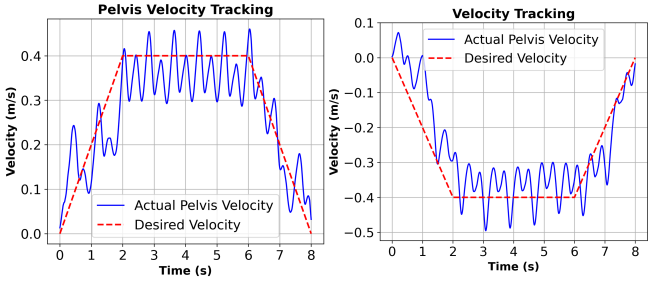

Fig. 5. Velocity tracking performance in simulation using a predefined velocity profile for motion in the x direction on flat terrain.

a drift correction strategy based on the known position of the stance foot is used [5]. Before each update of the map, we compute the vertical offset between the current stance foot position and the corresponding elevation in the map. This difference is applied as a correction offset and maintains alignment between the map and the ground surface.

VII. SIMULATION EXPERIMENTS

A. Simulation Setup

We trained two policies for analysis using identical network architecture, where the *ALIP* policy refers to a model using an observation space that includes the elevation map, desired velocity commands, and the mass-normalized ALIP state (Fig. 2). The *Joint* policy shares this observation space, with the addition of joint positions. Both policies are trained using all the methods discussed in this paper. In simulation, we compare *ALIP*, *Joint*, and a vision-based ALIP MPC footstep planner from [5]; denoted as *MPC*.

The controllers are evaluated using two primary metrics: *velocity tracking accuracy* and *success rates* across varying terrain types and friction coefficients. Stair terrains are generated using the parameters listed in Table III, with a 17-degree incline used for the slope terrain and friction coefficients of 0.4, 0.8, and 1.1 as specified in Table V. For each experimental configuration, 200 episodes are collected with random target velocities. All evaluations are performed without any noise. An episode is considered successful if the robot does not fall for 15 seconds.

B. Simulation Evaluation

1) Controller Comparison: As shown in Table V, both the RL policies consistently outperform *MPC* in velocity tracking across all terrain types and achieve higher success rates for ascending and descending stair and slope terrains. Furthermore, the *Joint* policy demonstrates better performance in velocity tracking compared to the *ALIP* policy across all of the evaluated terrains with similar success rates.

2) Collision Evaluation: Hard collisions with staircases accounted for a large proportion of failures, contributing to the lower performance of *MPC* on stair terrains. To further investigate, we assess the footstep planners based on footstep collisions with forces exceeding 1000 N. We quantify the frequency of such collisions and identify instances where these collisions directly lead to failure. A collision is classified as failing if the robot falls within two seconds of the impact.

TABLE V

PERFORMANCE COMPARISON OF RL POLICY ARCHITECTURES VS. MPC

Metric	Terrain Type	ALIP RL			Joint RL			MPC [5]		
		μ=0.4	μ=0.8	μ=1.1	μ=0.4	μ=0.8	μ=1.1	μ=0.4	μ=0.8	μ=1.1
Mean Squared Error of Velocity Tracking (m^2/s^2)	Stair (ascend)	0.0388	0.0514	0.0505	**0.0203**	**0.0209**	**0.0236**	0.0834	0.0726	0.0725
	Stair (descend)	0.0299	0.0351	0.0398	**0.0228**	**0.0247**	**0.0260**	0.0709	0.0665	0.0708
	Slope (ascend)	0.0186	0.0153	0.0211	**0.0046**	**0.0064**	**0.0072**	0.0258	0.0213	0.0227
	Slope (descend)	0.0424	0.0323	0.0346	**0.0129**	**0.0104**	**0.0099**	0.1166	0.0708	0.0654
Success Rate (%)	Stair (ascend)	90.5	88	81.5	**95.5**	**88.5**	**91.5**	56	72.5	68.5
	Stair (descend)	**92**	**94.5**	**95**	91.5	91.5	91	63.5	78	83
	Slope (ascend)	**100**	**100**	96	**100**	**100**	**100**	**100**	**100**	**100**
	Slope (descend)	82.5	**100**	**100**	**99.5**	99	99	5	80.5	96.5

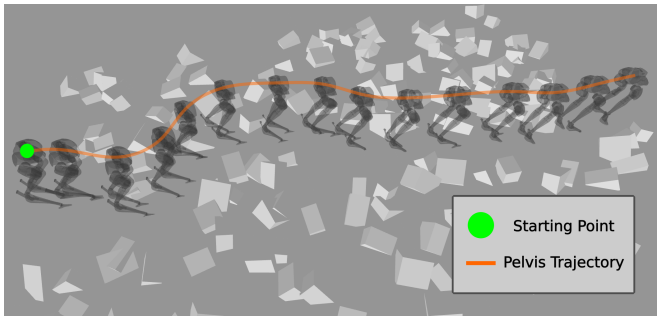

Fig. 6. Given a forward velocity command of 0.4m/s and no lateral velocity input, the policy successfully avoids obstacles while moving forward.

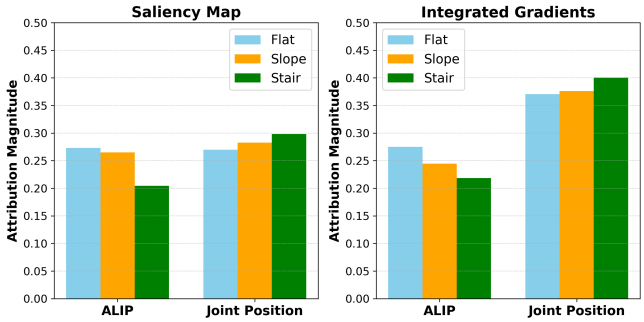

Fig. 7. Feature attribution visualizations for flat, slope, and stair terrains on the *Joint* policy. Left: Saliency maps, Right: Integrated gradients.

Evaluation results indicate that the *MPC* baseline exhibits a significantly higher frequency of hard collisions per episode compared to the RL-based planners. *MPC* achieves an average of 0.4833 hard collisions per episode, while *ALIP* and *Joint* achieve lower rates of 0.1467 and 0.11, respectively.

3) Recovery: In addition to reducing the frequency of hard collisions, the RL-based policies demonstrate improved recovery capabilities following hard collision events. The failure rate following a collision for *MPC* is 52.68%, whereas *ALIP* and *Joint* show considerably lower failure rates of 38.88% and 27.71%, respectively. Notably, recovery behaviors observed in simulation also transfer to hardware, where the RL-based policies show the ability to maintain balance and continue locomotion following disturbances.

4) Obstacle Avoidance: We demonstrate that incorporating obstacles directly into the simulation environment without introducing explicit safety or obstacle avoidance reward/penalty terms is sufficient for the policy to learn obstacle avoidance behaviors. Even in the absence of lateral velocity commands, the policy can navigate around obstacles while maintaining forward motion (Fig. 6). However, the policy may get stuck between obstacles, resulting in an in-place stepping behavior.

VIII. HARDWARE EXPERIMENTS

A. Hardware Setup

The state estimator and operational space controller are executed on Cassie's onboard NUC computer, and we transmit the joint torque commands to Cassie's target PC via UDP. The RL footstep planner and perception module operate on a ThinkPad p15 Laptop, equipped with an 8-core, 2.3 GHz Intel 1180H processor and 24 GB of RAM. The laptop is carried by one of the safety bar carriers and is networked with the NUC over Ethernet for LCM [37] communication.

B. Hardware Evaluation

The policy transferred to hardware despite a bent right bar and a difference in pelvis mass, with the custom battery used for hardware testing being approximately 1 kg lighter than the nominal URDF model. We demonstrate our walking controller across three scenarios: indoor flat terrain, outdoor flat-like terrain, and outdoor stairs with non-uniform, irregular slopes, all with perception in the loop. Trials are shown in Fig. 8. The controller performs successfully on flat-like terrains in both indoor and outdoor environments. However, shows poor performance when ascending stairs, where *ALIP* underperforms relative to *Joint*, achieving a maximum of two steps before failure, compared to four steps with *Joint*.

IX. ANALYSIS

A. ALIP as a Policy Input

We evaluate the comparative performance of the *ALIP* and *Joint* policies in stair-climbing tasks using the *mean stairs-to-failure* metric. In this evaluation, episodes are extended to 50 seconds, and episodes are terminated if the robot falls, has hard collisions with the stairs, or deviates significantly in the lateral (y) direction. A total of 200 episodes were collected for each policy. *Joint* averaged 10.024 steps before failure, outperforming *ALIP*, which averaged 6.534 steps. This is consistent with hardware observations and reflects the limitations of the ALIP model in handling height variations.

To assess the importance of ALIP states in policy decision-making, we employ two attribution methods: saliency maps [38] and integrated gradients [39]. As shown in Fig. 7, the influence of joint position inputs increases, while the contribution of ALIP state decreases with increasing terrain complexity in the *Joint* policy. This suggests the need for more expressive inputs in challenging discontinuous environments such as stair climbing. Our observation aligns with

Fig. 8. Sequence of images of Cassie walking on outdoor terrains. Top Row: Walking on outdoor flat terrain. Bottom Row: Descending (left three) and ascending (right three) outdoor stair terrain.

TABLE VI

L2 RATIO OF OSC OUTPUT TRACKING ↓

Trajectory	RL Real	RL Sim	Sim2real Error (RL)	MPC Real	MPC Sim	Sim2real Error (MPC)
Swing Foot x	0.2448	0.1698	0.075	0.1960	0.1279	0.0681
Swing Foot $\dot{x}$	0.7482	0.4966	0.2516	0.4921	0.4874	0.0047
Swing Foot y	0.0688	0.0529	0.0159	0.0607	0.0415	0.0192
Swing Foot $\dot{y}$	1.3454	0.1747	1.1707	1.0785	0.2417	0.8368
Swing Foot z	0.1440	0.1274	0.0166	0.1117	0.1169	0.0052
Swing Foot $\dot{z}$	0.3265	0.3084	0.0181	0.3036	0.2288	0.0748
CoM z	0.0179	0.0119	0.0060	0.0084	0.0094	0.0010
CoM $\dot{z}$	3.9889	3.0995	0.8894	2.2148	3.0187	0.8039

the findings of [9], which demonstrate successful policy performance on sloped terrains but do not report evaluations on discontinuous or stair-like environments, which are likely to be less successful due to the limitations of the ALIP reduced-order model.

B. Hierarchical Control

We evaluate the tracking performance of the operational space controller (OSC) using reference trajectories from the *MPC* and *Joint* footstep planners, averaged over four stair ascent trials in both simulation and hardware (Table. VI). To quantify performance, we report the L2 ratio, defined as:

$$\text{L2 Ratio} = \frac{\sqrt{\sum_t \|x_{\text{err}}(t)\|^2}}{\sqrt{\sum_t \|x_{\text{ref}}(t)\|^2}} \tag{4}$$

where $x_{err}(t)$ is the tracking error and $x_{ref}(t)$ is the reference trajectory at time t. The scale-invariant metric allows comparison across trials with different trajectory magnitudes. Ideally, if the OSC successfully feedback linearized the output, the L2 ratio would remain invariant to the reference signal. However, Table VI shows this is not the case. While both planners yield relatively low and comparable L2 ratios in simulation, the RL planner consistently results in higher L2 ratios across all output components in hardware.

This indicates that in the real-world, while the MPC planner produces trajectories that remain within the effective operating range of the OSC, the RL policy outputs footstep positions that drive the OSC into domains where the performance degrades. This effect is seen in hardware, where the sim-to-real gap further exacerbates the degradation in OSC performance, as reflected by the elevated L2 ratios.

X. CONCLUSION AND FUTURE WORKS

We present a learned hierarchical control framework for vision-based bipedal locomotion. The RL footstep planner utilizes a reduced-order ALIP model and an elevation map derived from a single depth camera. By simplifying both the observation and action spaces, we reduce the overall problem complexity. We validated the framework through simulation and hardware experiments, and interpreted the collected data to identify limitations of reduced-order models and hierarchical control within the context of hierarchical reinforcement learning frameworks.

While the use of the ALIP model and a hierarchical architecture offers benefits in terms of model simplicity and training efficiency, we show their limitations in handling complex tasks and real-world deployment. The ALIP-based footstep planner underperforms in complex environments due to its limited representational capacity. Additionally, although hierarchical controllers are known to improve training efficiency, interpretability, and modularity [24], our results indicate they complicate sim-to-real transfer. Transferring policies across layers increases the overall complexity of the transfer process, and the hierarchical structure imposes communication dependencies between layers, which require precise modeling and can hinder policy generalization.

Future work will explore more expressive reduced-order models for handling discontinuous terrain and focus on improving the alignment between the learned high-level policy and the low-level controller to enhance sim-to-real transfer.

REFERENCES

[1] T. Yoshiike, M. Kuroda, R. Ujino, H. Kaneko, H. Higuchi, S. Iwasaki, Y. Kanemoto, M. Asatani, and T. Koshiishi, "Development of experimental legged robot for inspection and disaster response in plants," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4869–4876.

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

[3] M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, "Continuous humanoid locomotion over uneven terrain using stereo fusion," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 881–888.

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

[5] B. Acosta and M. Posa, "Perceptive mixed-integer footstep control for underactuated bipedal walking on rough terrain," *IEEE Transactions on Robotics*, vol. 41, pp. 4518–4537, 2025.

[6] X. Xiong and A. D. Ames, "Coupling reduced order models via feedback control for 3d underactuated bipedal robotic walking," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–9.

[7] Y. Gong and J. W. Grizzle, "Zero dynamics, pendulum models, and angular momentum in feedback control of bipedal locomotion," *Journal of Dynamic Systems, Measurement, and Control*, vol. 144, no. 12, p. 121006, 2022.

[8] 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.

[9] G. A. Castillo, B. Weng, S. Yang, W. Zhang, and A. Hereid, "Template model inspired task space learning for robust bipedal locomotion," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 8582–8589.

[10] Y.-M. Chen, H. Bui, and M. Posa, "Reinforcement learning for reduced-order models of legged robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5801–5807.

[11] 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.

[12] H. Duan, B. Pandit, M. S. Gadde, B. Van Marum, J. Dao, C. Kim, and A. Fern, "Learning vision-based bipedal locomotion for challenging terrain," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 56–62.

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

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

[15] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, 2023.

[16] 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. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1. IEEE, 2001, pp. 239–246.

[17] J. Rummel, Y. Blum, H. M. Maus, C. Rode, and A. Seyfarth, "Stable and robust walking with compliant legs," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 5250–5255.

[18] X. Xiong and A. Ames, "3-d underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2405–2425, 2022.

[19] G. Gibson, O. Dosunmu-Ogunbi, Y. Gong, and J. Grizzle, "Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6724–6731.

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

[21] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. Van de Panne, "Feedback control for cassie with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1241–1246.

[22] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," *arXiv preprint arXiv:2105.08328*, 2021.

[23] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *arXiv preprint arXiv:2401.16889*, 2024.

[24] L. Bao, J. Humphreys, T. Peng, and C. Zhou, "Deep reinforcement learning for bipedal locomotion: A brief survey," *arXiv preprint arXiv:2404.17070*, 2024.

[25] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on robot learning*. PMLR, 2023, pp. 403–415.

[26] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[27] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2811–2817.

[28] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026, 2018.

[29] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended kalman filtering for robot state estimation," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 402–430, 2020. [Online]. Available: https://doi.org/10.1177/0278364919894385

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

[31] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti, "Walking control based on step timing adaptation," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 629–643, 2020.

[32] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst, "Learning task space actions for bipedal locomotion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1276–1282.

[33] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of solid-state circuits*, vol. 23, no. 2, pp. 358–367, 1988.

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

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

[36] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.

[37] A. S. Huang, E. Olson, and D. C. Moore, "Lcm: Lightweight communications and marshalling," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4057–4062.

[38] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[39] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 3319–3328.