# Linear Policies are Sufficient to Realize Robust Bipedal Walking on Challenging Terrains

Lokesh Krishna, Guillermo A. Castillo, Utkarsh A. Mishra, Ayonga Hereid, *Member, IEEE*, and Shishir Kolathaya, *Member, IEEE*

***Abstract*—In this work, we demonstrate robust walking in the bipedal robot Digit on uneven terrains by just learning a single linear policy. In particular, we propose a new control pipeline, wherein the high-level trajectory modulator shapes the end-foot ellipsoidal trajectories, and the low-level gait controller regulates the torso and ankle orientation. The foot-trajectory modulator uses a linear policy and the regulator uses a linear PD control law. As opposed to neural network based policies, the proposed linear policy has only 13 learnable parameters, thereby not only guaranteeing sample efficient learning but also enabling simplicity and interpretability of the policy. This is achieved with no loss of performance on challenging terrains like slopes, stairs and outdoor landscapes. We first demonstrate robust walking in the custom simulation environment, MuJoCo, and then directly transfer to hardware with no modification of the control pipeline. We subject the biped to a series of pushes and terrain height changes, both indoors and outdoors, thereby validating the presented work.**

***Index Terms*—Humanoid and bipedal locomotion, reinforcement learning, machine learning for robot control, legged robots.**

## I. Introduction

CLASSICAL works like spring loaded inverted pendulum (SLIP) with Raibert's heuristic controller [1], Zero Moment Point (ZMP) [2], the linear inverted pendulum [3] and Hybrid Zero Dynamics (HZD) [4] are designed to achieve robust walking behaviors on rough terrains. Despite the benefits of these works, like interpretability, existence of formal guarantees, scaling them for more complex tasks is not straightforward, involving a series of optimizations and tuning. For example, [5] used supervised learning in conjunction with gait libraries to enable rough terrain walking. Other examples include re-optimizations and gain scheduling [6] to realize multiple periodic gaits depending on the terrain.

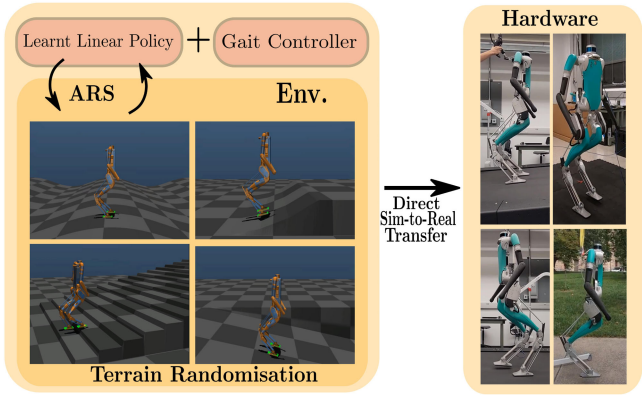


Fig. 1. Overview of the proposed learning framework.

Deep Reinforcement Learning (DRL) in robotics, and particularly in the context of legged locomotion [7]–[10] has shown appreciable progress in the development of state-of-the-art (SOTA) control frameworks. It has witnessed wide applications ranging from simulated physics-based animations [11], [12] to robust movements in real hardware [10], [13]. With the DRL framework, we let the bipeds learn to walk by themselves, thereby avoiding the complex tuning process. However, the existing frameworks for DRL need significant prior data to realize robust locomotion policies. In addition, they use significantly large networks with thousands or millions of parameters, which translate to additional computational costs. Furthermore, with the non-linearity of the neural networks, we lost the possibility of obtaining useful insights to leverage our understanding of the implicit learned behavior. This is in stark contrast to classical Raibert's controllers [1] that demonstrated robust locomotion behaviors and yet maintained simplicity and interpretability.

With a view toward simplicity of controller development, and based on the success of learning linear policies in simulation [14], [15], and in low-cost quadruped hardware [16], [17], we propose to extend this further and realize bipedal walking on challenging terrains in hardware. The proposed pipeline has two parts: a) The high-level foot-trajectory modulator and b) The low-level gait controller (see Fig. 1). The trajectory modulator shapes both the swing and stance leg trajectories, while the gait controller generates the trajectories to be tracked by the leg.

Manuscript received September 9, 2021; accepted December 27, 2021. Date of publication January 14, 2022; date of current version January 25, 2022. This letter was recommended for publication by Associate Editor C. Chevallereau and Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported in part by the Robert Bosch Centre of Cyber Physical Systems, in part by Pratiksha Trust, and in part by OSU under the M&MS Discovery Theme Initiative. *(Lokesh Krishna and Guillermo A. Castillo contributed equally to this work.) (Corresponding author: Lokesh Krishna; Guillermo A. Castillo.)*

Lokesh Krishna is with the Department of Electronics Engineering, Indian Institute of Technology (BHU) Varanasi, India (e-mail: lokesha1b2c3@gmail.com).

Guillermo A. Castillo and Ayonga Hereid are with the Department of Mechanical and Aerospace Engineering, Ohio State University, Columbus, OH 43210 USA (e-mail: g.castillom90@gmail.com; hereid.1@osu.edu).

Utkarsh A. Mishra and Shishir Kolathaya are with the Department of Computer Science and Automation and the Centre for Cyber-Physical Systems, Indian Institute of Science, Bengaluru, India (e-mail: umishra@me.iitr.ac.in; shishirk@iisc.ac.in).

Digital Object Identifier 10.1109/LRA.2022.3143227

Furthermore, the torso and ankle regulators enable correction for perturbations of the torso and terrain respectively. The trajectory modulator and the regulators are linear, thereby allowing us to learn/tune in a straightforward manner. The primary contributions of our letter are as follows:

*Linear abstraction of the control framework:* The main contribution lies in the extraction and exploitation of linear relations within the control of a highly non-linear system. We focus on learning and tuning the foot trajectory modulator and the torso-ankle regulator while keeping the nonlinearity fixed. Akin to Raibert's controller [1], this is motivated by the linear policy's ability to show seamless transfer to real hardware without requiring additional techniques like motor modeling [18], dynamics randomization [10] and others.

*Learning leveraged through heuristics:* The SOTA learning based frameworks for robot control, which often involves over-parameterized neural networks, leaves little to no space for integrating valuable physical insights/heuristics owing to their "black box" nature. These sophisticated learning algorithms often tend to exploit the practices like reference trajectory imitation (which aim at physics-driven policy optimization) and over fit to the simulation settings. To this end, we propose a flexible framework that utilizes the well-known priors of bipedal walking by incorporating user-designed heuristics.

Despite the advantages of linear parameterizations, the need for locomotion on more challenging terrains is indispensable. As opposed to existing works on Digit walking, our formulation is much simpler and yet yields versatile walking behaviours on complex terrains. We demonstrate robustness to external pushes, stair climbing and outdoor walking.

The letter is structured as follows: Section II will describe the robot model, notations and the associated hardware considerations. Section III presents the control framework, and Section IV provides the description of the training process. Finally, Section V showcases the simulation results, analysis, comparison with baseline and successful hardware experiments, followed by the conclusion in Section VI.

## II. Robot Model and Hardware Testbed

In this section, we describe the robot model on which the proposed framework is tested. We also introduce the mathematical notations used throughout the letter.

### A. Robot Model and Notations

**Digit** is a 30 degrees of freedom (DoF) 3D biped developed by Agility Robotics, USA (see Fig. 2). The total weight of the robot is 48 kg, from which 22 kg corresponds to the upper body, and 13 kg to each leg. Each arm has 4 DoF corresponding to the shoulder roll, pitch and yaw joints ($q_{sr}, q_{sp}, q_{sy}$) and the elbow joint ($q_e$).[1] Each leg consists of eight joints, including three actuated hip joints (hip roll, yaw, and pitch ($q_{hr}, q_{hp}, q_{hy}$)), one actuated knee joint ($q_k$), two actuated ankle joints (toe pitch and roll ($q_{tp}, q_{tr}$)), and three passive joints corresponding

[1]In this letter, we do not utilize the arm joints for balancing, hence, they will be controlled at fixed angles

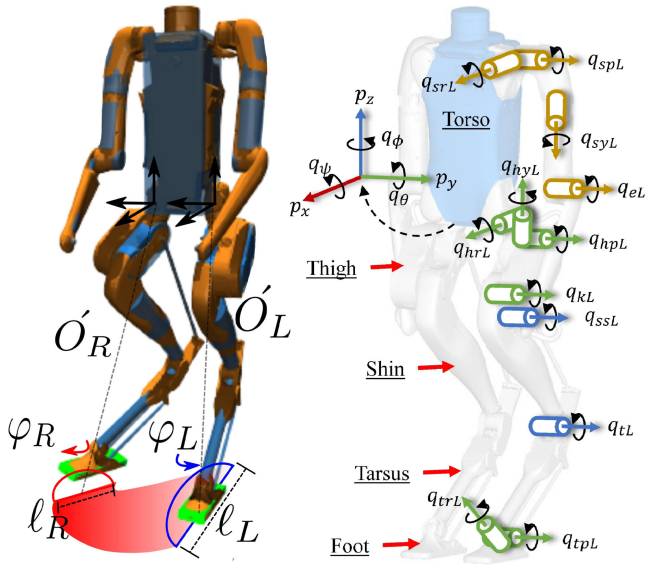


Fig. 2. Figure showing the Digit's kinematic-tree structure (right) and the action-space description (left) which portrays the transition from left to right leg. This is to highlight that the policy learns only the leg trajectory for the current swing leg irrespective of whichever of the two leg's is in the swing.

to shin-spring ($q_{ss}$), tarsus ($q_t$), and heel-spring joints ($q_{hs}$). To differentiate between left and right leg joints, we add the superscript $L$ and $R$ respectively to each of the joints. The position and orientation of the robot's base is denoted by:

$$\mathbf{q^b} = [p_x, p_y, p_z, \psi, \theta, \phi]^T, \tag{1}$$

where $p_x, p_y, p_z$ correspond to the base translation and $\psi, \theta, \phi$ correspond to the base orientation (roll, pitch and yaw angles) respectively. Therefore, the generalized coordinates of the robot are completely defined by:

$$\mathbf{q} = (\mathbf{q^b}, \mathbf{q^j}), \tag{2}$$

where $\mathbf{q^j}$ is defined by the robot joint angles:

$$\mathbf{q^j} = [q^L_{hr}, q^L_{hy}, q^L_{hp}, q^L_k, q^L_{ss}, q^L_t, q^L_{hs}, q^L_{tp}, q^L_{tr} q^L_{sr}, q^L_{sp}, q^L_{sy}, q^L_e,$$
$$q^R_{hr}, q^R_{hy}, q^R_{hp}, q^R_k, q^R_{ss}, q^R_t, q^R_{hs}, q^R_{tp}, q^R_{tr}, q^R_{sr}, q^R_{sp}, q^R_{sy}, q^R_e]^T.$$

In this letter, we denote $v_x, v_y, v_z$ as the torso velocity, $\dot{\psi}, \dot{\theta}, \dot{\phi}$ as the torso angular velocity about the roll, pitch and yaw axes, and the error as $e_\square = \square^d - \square$, where $\square^d$ is the desired value for that state.

*1) Forward Kinematics:* Given the generalized coordinates of the robot $\mathbf{q}$, forward kinematics (FK) can be used to compute the homogeneous transformation matrix $\mathbf{T} \in \mathbb{R}^{4\times4}$ of the robot's end-effector and center of mass (CoM). Several open-source packages can solve the FK by using the URDF model of the robot. We created a URDF of Digit from the XML model provided by the Agility Robotics, and used FROST [19] to obtain the symbolic expressions for $\mathbf{T}$.

For any homogeneous transformation:

$$\mathbf{T}_{\text{ac}} = \begin{bmatrix} \mathbf{R}_{\text{ac}} & \mathbf{p}_{\text{ac}} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}, \tag{3}$$

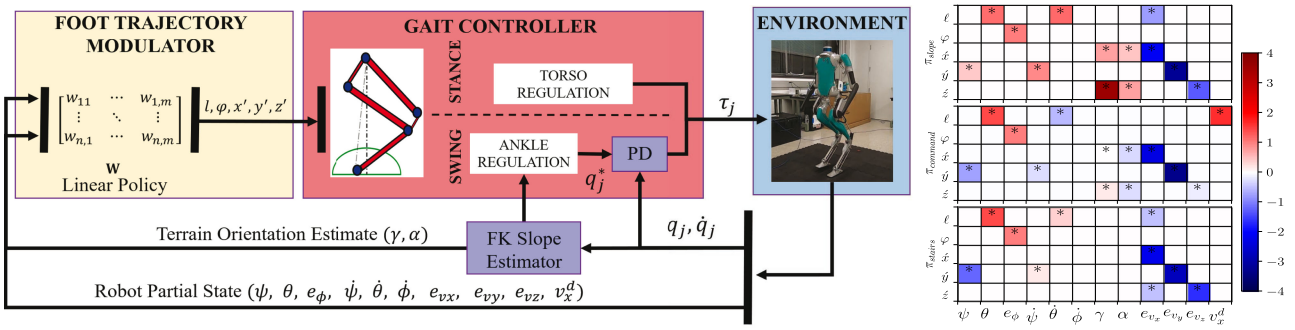


Fig. 3. Figure showing the control framework (left) and the policies (right) trained for walking 1) on arbitrary slopes, 2) on velocity command control, and 3) on stairs. In the policy matrices, the symbol '*' marks the terms that were allowed to be non-zero, to be optimised through ARS. The final values of the learnt policy is shown as the colour map.

where a and c denote any two frames of interest, $\mathbf{R}_{\text{ac}} \in \mathbb{R}^{3\times3}$ represent the rotation matrix, and $\mathbf{p}_{\text{ac}} \in \mathbb{R}^{3\times1}$ represents the relative position of the origin of frame c with respect to the origin of frame a. The orientation of the robot's feet with respect to the world is given through:

$$\mathbf{R}_{\text{wf}}^{L/R} = \mathbf{R}_{\text{wb}}^{L/R}\mathbf{R}_{\text{bf}}^{L/R}, \tag{4}$$

where w corresponds to the world fixed frame, f and b correspond to the robot's feet and base body frames, and $L/R$ determines left or right side.

By using the FK described above, we can use the orientation of the stance foot to estimate the support plane roll ($\gamma$) and pitch ($\alpha$) angles of the walking terrain by converting the rotation matrix $\mathbf{R}_{\text{wf}}^{L/R}$ to Euler angles. The rotation matrix $\mathbf{R}_{\text{wb}}^{L/R}$, is obtained from the IMU feedback of the torso orientation.

*2) Inverse Kinematics:* Following the work presented in [15], we only consider the foot position with respect to the robot base to solve the IK problem. In addition, given the particular closed-chains structure of Digit's leg, we keep the yaw hip angle constant and use simple trigonometric computations to transform the foot Cartesian position into virtual leg length, pitch angle, and roll angle. The virtual leg is the imaginary line that connects the hip of the robot with the leg ankle. In this decoupled system, the virtual leg length and pitch angle are determined by the position of the hip pitch and knee joint. With these values, we then solve the "reduced" IK subject to the constraints imposed by the leg kinematic structure. Finally, by solving the IK problem for a sufficiently diverse set of desired foot positions, we obtain a closed form solution using nonlinear regression.

*3) Contact Detection:* To switch between the left and right stance during the walking gait, we use the contact information of the feet with the ground. To detect the contact event, we estimate the ground reaction force of the stance foot by using the spring deflection and the contact Jacobian of the stance foot, as shown in [20].

## III. Methodology

In this section, we describe the control framework, and explain how the end-foot trajectories are modulated and tracked in real-time. A pictorial representation of our framework is shown in Fig. 3 (left).

### A. Overview of the Control Framework

As shown in Fig. 3 (left), we use a hierarchical structure with a high-level foot trajectory modulator and a low-level gait controller. The foot trajectory modulator comprises a linear policy that modulates a parameterized semi-elliptical foot trajectory. The trajectory thus generated is then fed into the gait controller, which uses a regulation based on the contact state of each of the legs. We chose to append an ankle regulation with the required joint targets obtained from the foot trajectory through inverse kinematics for the swing leg. If a leg is in the stance phase, we keep the ankle passive and activate the torso regulation to maintain the upper body attitude. Effectively, the policy is free to control the swing leg, whereas the stance leg is only used to stabilize the robot base. Since the stance ankle is passive and aligns the foot parallel to the support plane, we can accurately estimate the ground elevation using the forward kinematics. Ignoring the short-lived double support phase in every walking cycle, we only consider the single support phases and estimate the terrain for every walking step as discussed in Section II-A1. This estimate and the robot's torso states are used by the policy to modulate the properties of the swing leg trajectory. The proprioceptive robot state feedback is estimated by the inbuilt state estimator shipped along with Digit by Agility Robotics, with no additional customisations by us.

### B. Foot Trajectory Modulator

To modulate the foot trajectory, we propose to train a policy that uses only the relevant feedback deduced through physical insights from walking motion. For the given state space $\mathcal{S} \subset \mathbb{R}^n$ of dimension $n$, and action space $\mathcal{A} \subset \mathbb{R}^m$ of dimension $m$, we define our policy to be $\pi : \mathcal{S} \to \mathcal{A}$ as $\pi(s) := Ms$, where $M \in \mathbb{R}^{m\times n}$ is a matrix of learnable parameters. Our formulation drastically decreased the required control complexity, and hence a linear policy was sufficient to learn such a transformation. The observation and the action space choices are explained as follows.

*Observation Space:* In our prior work [15], we demonstrated the effectiveness of choosing a reduced observation space from all available robot states. However, to improve the terrain complexity that the policy could handle and develop command-controlled policies, we augmented the torso velocity error and

the desired heading velocity to the observation space in [15], including the torso orientation, torso angular velocity, and support plane elevation. For heading direction control, we choose to send the error in heading yaw in the place of the current torso yaw. Thus, the observation space is a 12 dimensional state vector defined as $s_t = \{\psi, \theta, e_\phi, \dot{\psi}, \dot{\theta}, \dot{\phi}, \gamma, \alpha, e_{v_x}, e_{v_y}, e_{v_z}, v_x^d\}$ (refer Section II-A for the notations).

*Action Space:* The semi-elliptical foot trajectory, is parameterized by the major axis (Step Length, $\ell$), the orientation of the ellipse along the Z-axis of the hip frame (hip yaw, $\varphi$) and translational shifts along X, Y and Z directions, $\acute{x}, \acute{y}, \acute{z}$ (together shown as $\acute{O}$), as seen in Fig. 2 (left). Here, step length and hip yaw describe the walking motion, whereas the shifts are heavily utilized to balance the robot actively. The semi-ellipse is then generated in the hip frame of reference as shown in [15]. To preserve the symmetry of the trajectories, we remove all the asymmetric conventions between the legs, outside the policy and apply a mirrored transformation according to the leg. This enables us to learn to predict a single set of parameters irrespective of the leg. Thus, the action space is a 5-dimensional vector such that, $a_t$=$\{\ell, \varphi, \acute{x}, \acute{y}, \acute{z}\}$.

### C. Gait Controller

The gait controller is responsible for keeping track of the gait parameters and tracking the generated foot trajectory. Based on the contact state of the leg (estimated as explained in Section II-A3), the gait controller augments the ankle regulation followed by joint level PD tracking for the swing leg and just the torso regulation for the stance leg, as in [13]. A phase-variable $\tau, \tau \in [0, 1)$ which is used to track the semi-elliptical trajectory gets reset once every walking step or upon a premature foot contact. Hence for an ideal walking cycle, the phase variable iterates from 0 to 1 twice. The gains of the joint level PD controllers are $K_p$ = 1000, 1000, 1000, 1500, 1000, 1000 and $K_d$ = 66.849, 26.1129, 38.05, 38.05, 15.55, 15.55 for the actuators at joints $q_{hr}$, $q_{hy}$, $q_{hp}$, $q_k$, $q_{tp}$,and $q_{tr}$ respectively.

*Ankle Regulation:* Since the generated foot trajectory does not include the two DoF at the ankle (actuated joints), we require an explicit regulation to control the foot orientation. For bipeds, control of the foot is crucial as the swing leg's angle of attack directly affects the torso orientation. The swing foot is kept parallel to the underlying terrain elevation to ensure proper landing on the ground. The desired position of the swing foot is determined from the kinematics of the robot's leg [13] as follows:

$$q_{tr}^d = q_{hr} + S_f(q_{tr}^{\text{off}} + \psi + \gamma) \tag{5}$$

$$q_{tp}^d = q_{hp} + S_f(q_{tp}^{\text{off}} - \theta - \alpha), \tag{6}$$

where $q_{tr}^d$ and $q_{tp}^d$ are the target angles and $q_{tr}^{\text{off}} = 0.366$ rad and $q_{tp}^{\text{off}} = 0.065$ rad are the joint angle offsets in accordance to the kinematic assembly for the ankle roll and pitch joints. The value of $S_f$ is defined as follows,

$$S_f = \begin{cases} -1 & \text{left leg in swing phase} \\ +1 & \text{right leg in swing phase} \end{cases}$$

*Torso Regulation:* The torso regulation is applied to ensure an upright torso, which is desired for a stable walking gait and, more importantly, to prevent the stance leg from sliding. The robot is assumed to have a rigid-body torso, and hence simple PD controllers defined below can be used for the the attitude control.

$$u_{hr} = P_{hr}(\psi_d - \psi) + D_{hr}(\dot{\psi_d} - \dot{\psi}) \tag{7}$$

$$u_{hp} = -S_f(P_{hp}(\theta_d - \theta) + D_{hp}(\dot{\theta_d} - \dot{\theta})), \tag{8}$$

where $u_{hr}, u_{hp}$ are the torques applied the the hip roll and pitch of the stance leg and $P_{hr}, D_{hr}, P_{hp}, D_{hp}$ are hand tuned gains. The desired targets for the torso attitude $(\psi_d, \theta_d, \dot{\psi_d}, \dot{\theta_d})$ are all set to zero for normal walking.

### D. Development of Heuristics

Being simple and interpretable, the linear policy allows us to manipulate the matrix elements based on physical insights. In [15] we developed a set of heuristic linear equations as a sub-optimal policy, hand-tuned the gains and deployed it as the initial policy for the training. This technique resulted in the policy converging to practical walking motions across different slopes. However, a lack of structure in the policy leads to the training algorithm making undesirable relations between the state and action variables. For example, the feedback of torso pitch ($\theta$) or yaw ($\phi$) is unnecessary for y-shift ($\acute{y}$), as it cannot control those DoF. The effect of these non-sparse terms in the matrix was insignificant in simulation but got amplified in our hardware trials, leading to the policy's failure. We hypothesize that the policy overfitting to the simulation dynamics through these non-zero stray terms affects the hardware performance due to the domain shift. To resolve this issue, we enforce a structure to the sparse matrix and only learn for the relevant terms. In this work, we intend to train policies for i) walking on arbitrary slopes, ii) walking on stairs, and iii) velocity controlled walking. Hence, we select certain terms common across all these matrices to ensure dynamic balance and several unique terms for each of them based on their task requirements.

*Stabilization Heuristics:* Irrespective of the task at hand, we require any policy to keep balance and walk forward. To this end, we define the following heuristic relations to stabilize the robot in each of the following planes.

*In the sagittal plane,*

- $\ell$ is to be used for correcting the disturbance in $\theta$,
- $\acute{x}$ is to be used for correcting the error in $v_x$, i.e. $e_{v_x}$,
- $\acute{z}$ is to be used for minimizing the torso oscillation along the z-direction, i.e. $e_{v_z}$

*In the transverse plane,*

- $\acute{y}$ is to be used for correcting the disturbance in $\psi$ and the error in $v_y$, i.e. $e_{v_y}$

*In the coronal plane,*

- $\varphi$ is to be used for correcting the error in heading direction, i.e. $e_\phi$

*Task-Specific Heuristics:* Apart from the stabilization heuristics, we add additional terms to the policy matrix based on the nature of the task for each of the following cases,

*Arbitrary Slope Policies:* In this case, there should be a dependency of the actions $\acute{x}$ and $\acute{z}$ with the support plane estimates $(\gamma, \alpha)$, to alter the foot placements in the sagittal plane based on the underlying terrain. Deducing a feasible target velocity for an arbitrary terrain is not straightforward, and we are also not keen on velocity tracking compared to stable walking on this challenging terrain. Hence, we relate the action $\ell$ with the state $e_{v_x}$, expecting the policy to converge to nominal walking step size in accordance with the objective (refer Section IV).

*Command Controlled Policies:* To learn a command controlled policy, we keep the same setup as for arbitrary slopes except for the step length ($\ell$) to be related with the commanded heading velocity ($v_x^d$) directly.

*Stair Policies:* The primary strategy to walk on stairs blindly are i) have a high swing height and ii) increase the z-shift upon accidental stubbing with a step. For the first strategy, we explicitly choose a higher foot clearance. To incorporate the second strategy, we enable the term connecting the state $e_{v_x}$ with the action $\acute{z}$. The intuition here is that when a foot collides with a step, a sudden change in the $v_x$ can be observed, and the feedback from $e_{v_x}$ can result in an increase in the $\acute{z}$.

These heuristics are shown visually in the Fig 3 (right), where the non-zero terms of the sparse matrices that the training algorithm can optimize for are marked with a '*'.

## IV. Policy Training

In this section, we discuss the training procedure used for learning the linear policy. Similar to [15], we use Augmented Random Search (ARS) [21], owing to the minimal number of hyper-parameters to tune, ease of use, and its effectiveness towards solving continuous-control problems. A point worth noting is that, instead of using the generic ARS setup, where the search space is in $\mathbb{R}^{m \times n}$, having enforced a heuristic structure to the policy matrix, we only search a sub-space of this parameter space.

### A. Reward Function:

Due to the ambiguity in finding a feasible target velocity for a given terrain type, we propose two different reward functions for training the i) Terrain Policies and ii) Command Controlled Policies. For terrain policies (slope and stair policies), we use a reward function defined as,

$$r = G_{w_1}(\psi) + G_{w_2}(\theta) + G_{w_3}(e_\phi) + G_{w_4}(e_{p_z}) + W\Delta_x \quad (9)$$

where, $e_{p_z}$ is the error in the robot's height with $p_z^d = 0.95\,m$ and $p_z$ being the current torso height, and $\Delta_x$ is the distance travelled along the heading direction in that time-step, weighted by $W$. The mapping $G : \mathbb{R} \rightarrow [0, 1]$ is the Gaussian kernel given by $G_{w_j}(x) = \exp(-w_j * x^2),\ w_j > 0$. The objective here is to walk as far as possible while ensuring the stability of the torso. For training the command controlled policies, we remove the $\Delta_x$ term and substitute it with a velocity tracking term as shown in (10). This is because, we require the policy to learn to react to changes in the velocity commands.

$$r = G_{w_1}(\psi) + G_{w_2}(\theta) + G_{w_3}(e_\phi) + G_{w_4}(e_{p_z}) + G_{w_5}(e_v) \quad (10)$$

where $e_v$ is the error in the heading velocity of the robot.

### B. Training Setup

As shown in Fig. 1, for terrain policies, we train on the variants of a given parameterized terrain type. A specific combination of terrain parameters is randomly chosen from a discrete set of that terrain's configurations at the beginning of an episode. The target heading velocity is kept to be a small positive value to prevent the policy from learning to walk in place (as $e_{v_x} \neq 0$). For the command-controlled policies, we only train on flat-ground and update the target velocity and desired yaw every three seconds. An episode is terminated when the robot topples, or if the robot's height decreases below a certain threshold or if the maximum episode length is reached. The ARS hyperparameters used for training are learning rate $(\beta) = 0.03$, noise $(\nu) = 0.04$ and episode length $= 15$ k simulation steps.

## V. Results

This section presents the simulation results, comparision study with baselines, behaviour analysis, and the hardware experiments conducted. For training our policies in simulation, we use a custom Gym environment with the MuJoCo physics engine. The hardware results presented below are with policies that showed direct sim-to-real transfer with no form of tuning or usage of explicit techniques like dynamics randomisation.

### A. Simulation Results

*1) Performance Analysis:* In simulation, we train three different policies for fulfilling three distinct tasks; namely, i) walking on arbitrary slopes ($\pi_{\text{slope}}$), ii) walking on stairs ($\pi_{\text{stair}}$) and ii) Omni-directional command controlled policies ($\pi_{\text{command}}$), as shown in Fig. 3 (left). For learning to walk on arbitrary slopes, we train by sampling a random terrain elevation chosen from $\{$-13°, -11°, -7°, 0°, 7°, 11°, 13°$\}$. With active feedback of the underlying terrain elevation, the policies learn to traverse inclines of up to 25° and declines of up to $-20°$ successfully. Apart from showing a direct extrapolation to uniform slopes of higher elevations, unseen during training, these policies also show zero-shot generalization to varying slopes and sinusoidal terrains (as seen from the attached video). This result shows that the policy learns a robust foot-placement strategy for the swing leg based on the current estimates of terrain obtained from the stance leg. Unlike generic vision-driven control techniques, this provides an elegant and efficient solution as it is unaffected by the feedback update-rate and does not require planning a future horizon at a very high dimensional space. In accordance with the well-shaped reward function, we observe that the oscillations in the torso are minimal and well-contained within the following ranges: $\psi \in (-1.5°, 1.5°), \theta \in (-4°, +4°), \phi \in (-5°, 5°)$, and $p_z \in [$terrain height - 0.02, terrain height + 0.02$]$m. Since

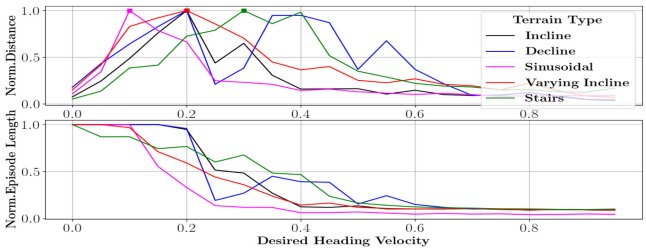


Fig. 4. Figure showing the terrain traversed Vs. the feasible heading velocity. The square markers mark the point of maximum distance travelled by the policy in each terrain.

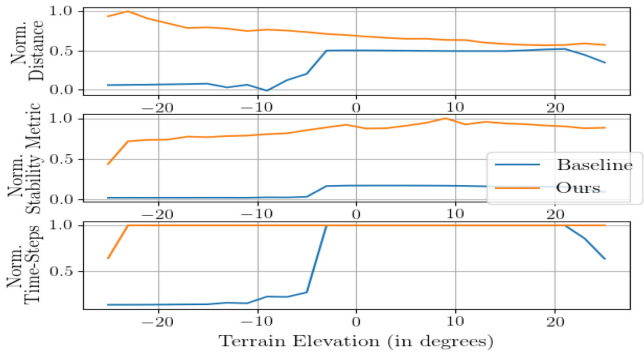


Fig. 5. Figure showing the performance metrics compared over slopes of different elevations between the baseline NN Policy and our proposed linear policy.

the terrain estimates are inadequate to provide any fruitful feedback about stairs, we intend to treat the steps as a terrain uncertainty. Following the strategy described in Section III-D, we train the $\pi_{\text{stair}}$ on staircases with parameters randomly sampled from $\{(0.3, 0.05), (0.4, 0.085), (0.5, 0.1)\}$, where the former value in each pair refers to the step length and the latter to the step height. Training only on these discrete staircases, the policy generalizes well within the configurations between these parameter limits. In simulation, we observe that the policy could traverse stairs of up to 15 cm heights. For learning a command controlled policy, $\pi_{\text{command}}$, we update the desired target velocity and desired heading yaw every three seconds from the beginning of each training episode. We limit the maximum change in velocity and yaw commands to be at $\pm 0.2$ m/s (longitudinally), $\pm 0.1$ m/s (laterally) and $\pm 2.5^{\circ}$, respectively. Such a training configuration exposes the policy to a wide range of direction commands and velocity transitions. These trained policies showed stable walking of velocities up to 0.5 m/s and quasi-static rotations of the torso yaw about the axis.

*2) Behavioural Study:* Owing to the linear relations and constrained structure of the policies, we can easily map a certain recovery behaviour directly to a parameter in the matrix. This allows training for various strategies by simply changing the matrix configuration. As seen in Fig. 3 (left), the policies learn parameters that tend to have different values even for the same state-action combination. It is worth noting that this subtle difference contrasts to classical hand-tuned heuristic gains, as the identical spatial elements neither need to have comparable values nor need to be of the same sign. Hence, the imposed heuristics are not restrictive, and the learning algorithm can develop emergent behaviours as per the task requirements. Another important design choice that we went with was not restricting the terrain policies ($\pi_{slope}, \pi_{stair}$) to track a certain desired velocity but converging upon a nominal velocity which was practical for the underlying terrain. To identify this nominal heading velocity to which that policy converged, we compare the distance travelled and the time steps before failure, across different terrains in Fig. 4. We observe that the slope policy tends to walk very slowly (0.1 m/s) on sinusoidal terrain whereas, reaching the maximum distance reliably at 0.2 m/s for incline, decline and varying inclines. At points 0.4 and 0.5 m/s, though the distance-travelled spikes-up for decline, the quality of motion is poor, and the robot falls soon enough, after taking some aggressive steps. The $\pi_{\text{stair}}$ is seen to walk the longest at 0.4 m/s. However, the times-steps before failing drop as the target velocity is increased. Thus, $\pi_{\text{stair}}$ could be operated within the range of 0 to 0.4 m/s.

TABLE I
COMPARISON OF THE MEAN AND STANDARD DEVIATION IN COT FOR DIFFERENT TERRAINS TYPES OBTAINED THROUGH MULTIPLE TRIALS ACROSS VARIED TERRAIN PARAMETERS

| Framework | Stats. | Ours | Baseline |
|---|---|---|---|
| Slope | $\mu$ | 0.1704 | 0.2594 |
| | $\sigma$ | 0.0339 | 0.0176 |
| Varying Slope | $\mu$ | 0.2158 | 0.3217 |
| | $\sigma$ | 0.0176 | 0.0197 |
| Sinusoidal | $\mu$ | 0.2539 | - |
| | $\sigma$ | 0.0118 | - |
| Stairs | $\mu$ | 0.3040 | - |
| | $\sigma$ | 0.0452 | - |

*3) Comparison With Baselines:* As the proposed methodology expresses the effectiveness of a simple linear behavioral policy to control Digit, we present comparison with a recent baseline on the robot with a compact non-linear neural network policy [13]. The key difference between the current and baseline algorithm, is planning in task space and joint space of the robot, respectively. Considering the same desired forward velocity, a comparison is conducted over performance metrics like the distance travelled, time-steps sustained before failure, and stability [2] of the torso. The stability metric for the torso is measured using a subset of the reward terms in (9) and (10), associated with the state of the torso orientation and height.

Fig. 5 illustrates the performance of the two policies in a wide range of terrain elevations.[3] An interesting observation was that the NN Framework failed to generalize towards declines and opted to take very small steps to counter-act the incline. On the other hand linear policies show consistent and superior performance throughout the evaluated elevation range of $-25^{\circ}$ to $25^{\circ}$. As an additional metric, we compare the Cost of Transport (CoT) between the policies of both of these frameworks based on the equations described in [10], across multiple variants of a given terrain. The results tabulated in Table I show that the CoT

[2]Here the word "stability" is used as an alias to represent the upright posture of the robot and not in the classical sense of the word.

[3]Due to the limitations of the baseline framework, it cannot be readily extended for stairs and fails to generalize for sinusoidal terrains.

Fig. 6. Tile plots of Digit recovering from a external push (top), blind walking on terraced surface (centre), and turning in place as per command (bottom) using the learned linear policies.

of our framework increases along with the terrain complexity and yet is consistently lower than the baseline.

### B. Hardware Results

We demonstrate that our proposed framework can be successfully transferred from simulation to hardware without additional tuning of the learned parameters. To evaluate the robustness of the learned policy, we extensively test our controller in a series of different experiments, including external disturbance rejection, and blind walking on challenging terrains such as slopes and stairs. These experiments are documented in the accompanying video submission.

*1) Direction Controlled Walking:* We used the learned policy to command the robot to walk in different directions (forward, backward, and lateral) and different heading angles. This enables our policy for safe navigation in real world scenarios. Fig. 6 (bottom) shows a tile plot of the robot turning to the left while walking in place. Furthermore, we empirically demonstrate the stability of the walking gait by analyzing the phase portrait of the joints. Fig. 7 shows that the phase portrait of the robot joints converge to a stable walking limit cycle.

*2) Walking on Slopes:* We tested the learned policy on slopes with varying inclinations, including upslopes of $5°, 7°, 9°$, and $11°$. In addition, the learned policy was tested in outdoor environments, where the same policy was able to successfully walk along paths that included transitions from flat ground to arbitrary slopes. Fig. 8 shows the velocity profile for this experiment and the slope estimation introduced in Section III-B. More of these results can be seen in the accompanying video submission.

*3) Blindly Walking on Stairs:* To evaluate the robustness of the learned policy to walk blindly trough stairs, we build a small testbed with stairs of different heights, including $4, 5, 8$ and $-4$ centimeters. The policy maintains a stable walking gait while moving forward and backward. Fig. 6 (centre) shows a tile plot of the experiment, while Fig. 9 shows the action given by the linear policy transformed to the desired trajectory for the robot's feet.

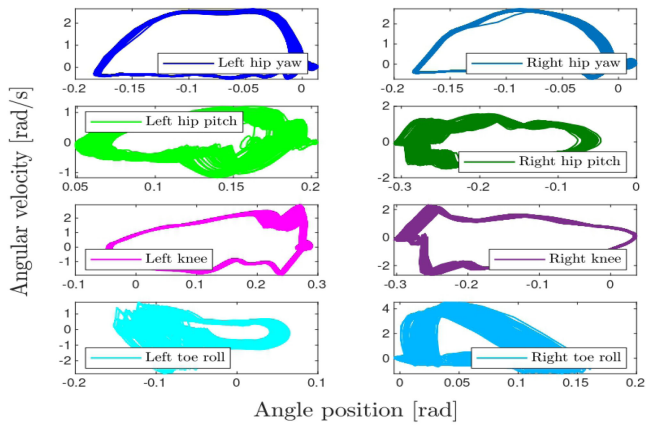


Fig. 7. Phase portrait of joints for learned policy. The phase portrait of the joints converge to a stable limit cycle, which empirically shows the stability of the walking gait.

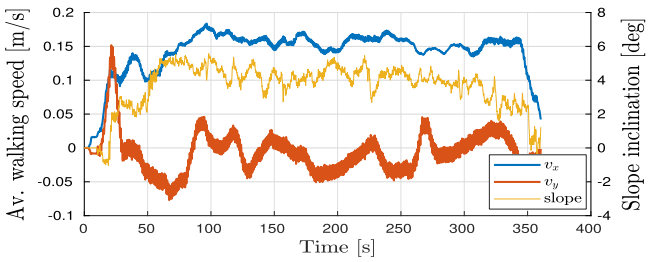


Fig. 8. Digit walking upslope in outdoors environment. The velocity profile demonstrate the robot keeps walking forward ($v_x$) while moving consistently upslope with almost no drift in the lateral direction ($v_y$). The walking gait is robust to the varying inclination of the terrain, which is estimated by the FK-based slope estimator.

*4) Disturbance Rejection:* Finally, we tested the robustness of the learned policy against external disturbances by pushing

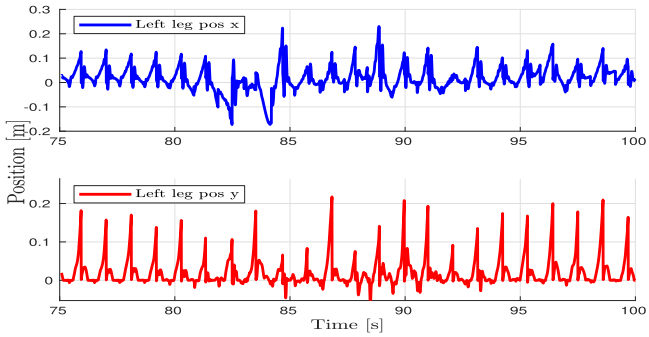


Fig. 9. The actions delivered by the learned policy for blind stair walking allows the robot to adapt its leg position with respect to the robot's base to keep a stable walking gait over stairs with different heights. The negative values in the x direction allows the robot to step back to recover from disturbances caused by its interaction with the stair.

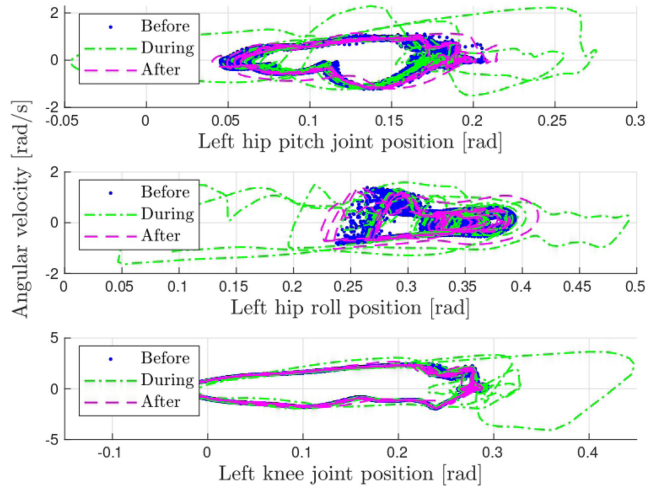


Fig. 10. The phase portraits of the joints are perturbed during the disturbance, but converge to the nominal walking limit cycles by the actions of the learned policy.

the robot in different directions while the robot is walking in place. To illustrate the policy performance, Fig. 10 presents the limit cycle of three of the robot's joints when the robot is pushed in the lateral direction. The policy recovers effectively from the push as the joint motion returns to a stable periodic orbit. This recovering behavior is also illustrated in the tile plot shown in Fig. 6 (top).

## VI. Conclusion

In this letter, we successfully demonstrated robust walking in the bipedal robot Digit in simulation and hardware with the help of linear policies. We show zero-shot generalization from training on constant inclines and declines to walking on varying inclines, sinusoidal terrains and stairs. Further, we extend the framework to direction controlled walking on flat surfaces. The proposed control formulation obtains linear relationships based on bipedal walking priors and several heuristics. The current approach, along with our previous contributions [14], [15], results in efficient synthesis of policies for legged robots (bipeds and quadrupeds) and simplifies the process of designing controllers for sim-to-real transfer for a wide variety of terrains. The video results accompanying this letter is shown here: https://youtu.be/IK9BSHnNQrk.

## References

[1] M. H. Raibert, "Legged robots," *Commun. ACM*, vol. 29, no. 6, pp. 499–514, 1986.
[2] R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3D biped," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2004, vol. 3, pp. 2849–2854.
[3] Y. Gong and J. Grizzle, "One-step ahead prediction of angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-inspired controller," in *Proc. Int. Conf. Robot. Automat.*, 2021, pp. 2832–2838.
[4] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Trans. Autom. Control*, vol. 48, no. 1, pp. 42–56, Jan. 2003.
[5] X. Da and J. Grizzle, "Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots," *Int. J. Robot. Res.*, vol. 38, no. 9, pp. 1063–1097, 2019.
[6] Z. Li *et al.*, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 2811–2817.
[7] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Proc. Conf. Robot Learn.*, 2020, pp. 317–329.
[8] J. Siekmann *et al.*, "Learning memory-based control for human-scale bipedal locomotion," in *Proc. Robot. Sci. Syst.*, 2020.
[9] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst, "Learning task space actions for bipedal locomotion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 1276–1282.
[10] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," in *Proc. Robotics: Sci. Syst.*, 2021.
[11] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, Jul. 2017.
[12] T. Kwon, Y. Lee, and M. Van De Panne, "Fast and flexible multilegged locomotion using learned centroidal dynamics," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 46–1, Jul. 2020.
[13] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Robust feedback motion policy design using reinforcement learning on a 3D digit bipedal robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5136–5143.
[14] K. Paigwar *et al.*, "Robust quadrupedal locomotion on sloped terrains: A linear policy approach," in *Proc. Conf. Robot Learn.*, 2020.
[15] L. Krishna, U. A. Mishra, G. A. Castillo, A. Hereid, and S. Kolathaya, "Learning linear policies for robust bipedal locomotion on terrains with varying slopes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5136–5141.
[16] M. Rahme, I. Abraham, M. L. Elwin, and T. D. Murphey, "Linear policies are sufficient to enable low-cost quadrupedal robots to traverse rough terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8446–8453.
[17] A. Iscen *et al.*, "Policies modulating trajectory generators," in *Proc. 2nd Conf. Robot Learn.*, 2018, pp. 916–926.
[18] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, 2019.
[19] A. Hereid and A. D. Ames, "FROST: Fast robot optimization and simulation toolkit," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 719–726.
[20] Y. Gong *et al.*, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *Proc. Amer. Control Conf.*, 2019, pp. 4559–4566.
[21] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1800–1809.