Robust Quadrupedal Locomotion on Sloped Terrains: A Linear Policy Approach

Kartik Paigwar, Lokesh Krishna, Sashank Tirumala, Naman Khetan, Aditya Sagi, Ashish Joglekar, Shalabh Bhatnagar, Ashitava Ghosal, Bharadwaj Amrutur, and Shishir Kolathaya

Indian Institute of Science, Bangalore

Abstract: In this paper, with a view toward fast deployment of locomotion gaits in low-cost hardware, we use a linear policy for realizing end-foot trajectories in the quadruped robot, Stoch 2. In particular, the parameters of the end-foot trajectories are shaped via a linear feedback policy that takes the torso orientation and the terrain slope as inputs. The corresponding desired joint angles are obtained via an inverse kinematics solver, and tracked via a PID control law. Augmented Random Search, a model-free and a gradient-free learning algorithm, is used to train this linear policy. Simulation results show that the resulting walking is robust to terrain slope variations and external pushes. This methodology is not only computationally light-weight, but also uses minimal sensing and actuation capabilities in the robot, thereby justifying the approach.

Keywords: *Quadrupedal walking, Reinforcement Learning, Random Search*

1 Introduction

Over the last few years, there has been a surge in the development of low-cost and open-source quadruped robots [1], [2], [3]. Due to limitations in actuation/sensing capabilities, developing a control law that yields stable walking in these types of robots is cumbersome. Some of these robots do not possess BLDC motors for accurate torque control [3]. In addition, deployment of some of the advanced control laws like the model predictive control (MPC) [4], Deep Neural Networks (DNN) [5, 6] require expensive computational resources. Therefore, in this paper, we would like to answer the following question: What is the minimum possible control framework that can be deployed to realize stable locomotion behaviors in medium-size low-cost quadruped robots?

The domain of quadrupedal locomotion has reached maturity today with quite a few research labs/companies successfully commercializing their quadrupeds. A slew of techniques are in use—inverted pendulum model based controllers [7], zero-moment point based controllers [8], hybrid zero dynamics [9], model predictive controllers [4]—to name a few. However these techniques are not suitable for our goal because they require extensive domain expertise. The long term goal of our work is to create a quadruped controller akin to inexpensive flight controllers developed for remote piloting of autonomous aircrafts [10, 11]. The difficulty in developing such a controller is that it must be capable of adapting to different quadruped robots with minimum modifications.

With a view toward "automatically" developing control algorithms for a broad set of servo driven quadruped robots, we aim to learn a simple policy that outputs end-foot reference trajectories that can be tracked by the individual joints, resulting in stable walking. To this end, data driven approaches like Reinforcement Learning (RL) [12], which are capable of learning walking controllers by themselves seem very attractive. In our formulation we learn a linear policy i.e., a single matrix which can be implemented on microcontroller based hardware boards (like the STM32 series or Cortex M4 series) using RL and is capable of traversing slopes, rough terrain and rejecting disturbances.

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

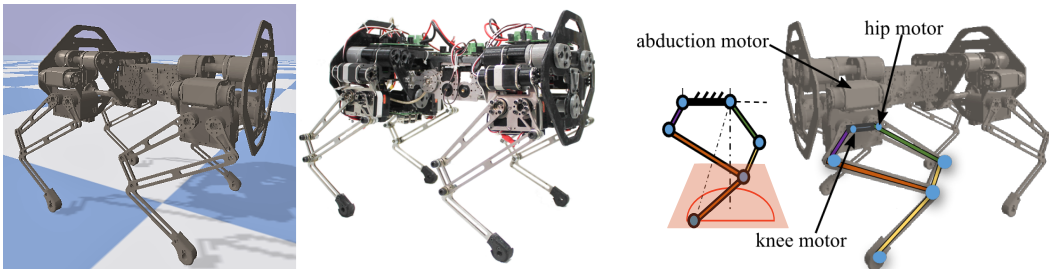

Figure 1: Simulated model and fabricated hardware of quadruped robot, Stoch 2 respectively (left two figures). The kinematic description of a leg (rightmost). The trajectory followed by the foot is a semi-ellipse which lies within a safe work space of the leg which is shown as the trapezoidal region.

1.1 Related work

Reinforcement learning (RL) for quadrupedal walking was first explored by [13], where policy gradient algorithms were used to learn optimal parameters for end-foot trajectories. The learned trajectories were then tracked in each foot in open-loop. A deep reinforcement learning framework (D-RL) was used in [6], [14], for training a deep neural network (DNN) based feedback policy for walking on flat terrains by first training the policy in simulation, and then bridging the sim-to-real gap with domain randomization and motor-modelling. However, these DNN based policies are computationally intensive and were unable to generalize to rough terrain walking. Similarly [15], [16], [17] trained a quadruped for a flat terrain, where the learning process was sped up significantly by parameterizing the control policy using central pattern generators, Bézier curves and rational Béziers respectively. In our paper, we also parameterize our control policy with elliptical curves to speed up training. [18] first demonstrated the capabilities of a linear policy for robotics control in simulation while [19] used the linear policy approach to create policies for flat terrain quadruped walking. We also use a similar linear policy paradigm for rough terrain quadruped walking. Walking on arbitrary terrains was formulated as a nonlinear optimization problem in [20]. [21] learnt the parameter initialization for the above nonlinear optimization problem, improving the accuracy and speed of the optimization and enabling the quadruped robot to navigate bumpy terrain. We used a similar initialization technique in our training process. A custom RL algorithm was used in [22], to enforce footstep constraints in order to train walking over stairs, slopes and ridges. Similarly [23] uses hierarchical apprenticeship learning and expert data to navigate a rocky terrain. However these policies require a complete terrain map of as an input to work. Our focus is mainly in trotting on slopes with proprioceptive feedback.

Outside of learning, a large number of techniques have been developed to navigate a rough terrain for a quadruped robot in a blind manner. The MIT Cheetah robot achieved this by using Model Predictive Control for the stance legs and Raibert's controller for the swing legs [24]. Similarly, [25] demonstrated a hierarchical blind whole body controller for the Anymal robot. However these techniques require expensive Series Elastic Actuators or Quasi Direct-Drive actuators in order to function. In contrast, our approach requires only hobby servo motors and mainly focuses on reinforcement learning for rough terrain locomotion.

1.2 Contributions and organization of the paper

In this paper we extend the work in [15], [16], [19] to include locomotion on rough terrain. In particular [16] used open-loop trajectories that were tracked in each leg to yield walking. Our approach includes a feedback control to dynamically shape these trajectories based on the body and plane orientation. Our controller is capable of rejecting disturbances and maintaining balance on slopes of various inclines. The support plane is estimated using forward kinematics and foot contacts similar to [24]. We also restrict our policy to being linear. This linearity requires low computation, thus enabling our policy to be run in real-time on an on-board embedded system. Augmented Random Search (ARS), a well-known algorithm for training linear policies [26], is used to learn our proposed policy. We validate our results on the indigenously developed quadruped robot, Stoch 2 in simulation.

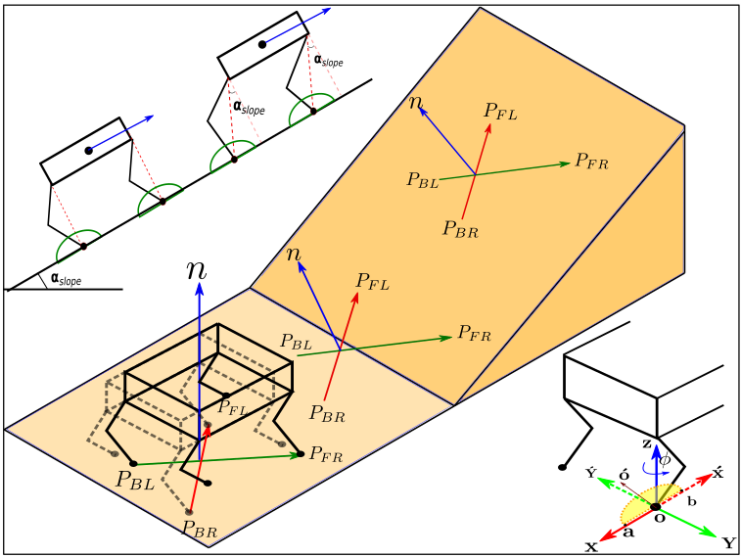

Figure 2: Figure showing the parameterization of the semi-elliptic trajectory tracked by the feet of the robot (bottom right), the foot position vectors used for the slope estimator (middle), and the two commonly used strategies for walking on slopes (top left).

The paper is structured as follows: Section 2 will provide a description of the robot and the associated control framework, Section 3 will describe the training process used. This will be followed by simulation results in Section 4.

2 Robot model and problem formulation

In this section, we will discuss our custom built quadruped robot Stoch 2. Specifically, we will provide details about the hardware, and the walking controller that we aim to learn with reinforcement learning.

2.1 Hardware description of Stoch 2

Stoch 2 is a second generation quarduped robot in the *Stoch* series designed and developed in the Indian Institute of Science [3]. It is a low cost hardware (costing less than $3000) which is built using hobby servo motors and 3D-printed parts. It has an inertial measurement unit (IMU) to detect the body pose, and integrated joint-encoders and motor current sensors which are built into the servo motors. The legs use a five-bar architecture and each leg contains three actuated joints for hip abduction, hip flexion/extension, and knee flexion/extension. The end foot point can follow any path that lies inside a trapezoidal region within the planar work-space of the five-bar leg (see Fig. 1). This planar workspace is extended in 3D with the abduction motors. The calculation of the leg work-space and its inverse kinematic details can be found in [3]. The URDF model used in the simulator was obtained from the SolidWorks assembly (see Fig. 1). Overall, the robot model consists of 6 floating degrees-of-freedom and 12 actuated degrees-of-freedom.

2.2 Walking control strategy for Stoch 2

As mentioned previously, we use trajectory based policies [16], i.e., we shape the semi-ellipses of every leg to realize walking in Stoch 2. This idea of shaping of the end-foot trajectories is mainly obtained from [27]. More details about the specific parameters used for shaping are provided in Fig. 2. The general idea is to use the orientation of the robot and the supporting plane (terrain), to obtain the trajectory parameters. We will first describe the mechanism to estimate the slope of the supporting plane, and then focus on walking controller used on arbitrary slopes.

Terrain slope estimation. Assuming a reasonably flat terrain, the surface on which the robot is walking can be modelled using a plane. An estimate of the terrain slope is required by the controller to enable a robot to dynamically adapt its posture based on the terrain. We use an Inertial Measurement Unit (IMU) and joint encoders to estimate the slope of the surface. A history of foot position data is used to estimate the slope of the terrain, as the foot lies on the slope during it's stance phase. The position $\mathbf{p}_i$ of the feet w.r.t. the robot frame of reference can be calculated using the joint encoder data from the motors using forward kinematics:

$$\mathbf{p}_i = \mathbf{f}(\alpha_{hip_i}, \alpha_{knee_i}, \alpha_{abd_i}), \tag{1}$$

where $\mathbf{f}$ is the forward kinematics function that maps the joint angles to the Cartesian coordinates of the foot, $\mathbf{p}_i$ is the position of the foot i, with $i \in \{FL, FR, BL, BR\}$, in the robots body frame of reference, and α_{hip_i} and α_{knee_i} are the angles of the hip and knee actuators (see Fig. 1) and α_{abd_i} is the angle of the abduction actuator for the leg i.

To convert p_i from the robot's reference frame to the world frame, we use the IMU Sensor data. In this case the world reference frame is assumed to be located at the center of mass of the robot with its z-axis pointing opposite to the direction of action of the gravity and with the x-axis pointing to the magnetic north. The positions of the robot feet, $\mathbf{p}_i$, can be expressed in the world reference frame, $\mathbf{P}_i$, by transforming the vectors to the new frame $\mathbf{P}_i = \mathbf{R}\mathbf{p}_i$, where $\mathbf{R}$ is the rotation matrix that defines the orientation of the robot body frame w.r.t. the world frame. The positions of the feet should be ideally captured at the exact moment of transition when all four legs are on the ground. Due to the inability to detect the time of the transition, data is captured from the stance leg pair just before the transition and the new stance leg pair just after the transition. The difference between the two data capture times are small and hence it can be reasonably approximated to have occurred at the same time. Normal to the plane (see Fig. 2) can be obtained by:

$$\mathbf{n} = (\mathbf{P}_{FR} - \mathbf{P}_{BL}) \times (\mathbf{P}_{FL} - \mathbf{P}_{BR}). \tag{2}$$

The roll and pitch orientation of the plane can be readily calculated from the normal vector.

Trajectory shaping on supporting planes. The strategy for walking on slopes introduces new challenges, and cannot be obtained directly from walking controllers used for horizontal planes. Depending on the slope, the semi-ellipses are shifted, reoriented and reshaped in real-time so that the walking is stable. The parameters for this mechanism are $\mathbf{O\acute{O}}$, ϕ, and $\mathbf{ab}$, which are shown in Fig. 2. This strategy is mainly motivated from the methods presented in [28]. In particular, two types of strategies were proposed in [28]: *telescopic strut* and *lever*.

In telescopic strut strategy, the feet are vertically below the hip, and in lever strategy, the legs are aligned with the normal to the supporting plane (see Fig. 2). In both of these strategies, the goal is to ensure that the torso is aligned with the supporting plane. Owing to our learning based approach, we shape our rewards accordingly, to ensure that the torso is aligned with the support plane. While we let our controller learn the right strategy by itself, we use the telescopic strut strategy to obtain the right seeds for training (see Section 3.3 for details). Accordingly, we model a linear policy that takes in the body and plane's orientations as the inputs and chooses the parameters of the semi-ellipse in real time, which are then tracked by the low level motor controllers.

2.3 Reinforcement learning framework

We formulate rough terrain locomotion as a reinforcement learning problem, with an observation space, action space and a reward function detailed below. Our policy π is a linear transformation that maps the observation vector S to the action vector A.

Observation space. Our observation space is constituted of the robot base orientation and terrain slope parameters, since these quantities are highly relevant observations for locomotion on slopes as explained in Section 2.2. The base orientation is measured directly by an inertial measurement unit (IMU) while the terrain slope parameters estimated by the equations provided in Section 2.2. The observation space forms a 11-dimensional state vector defined as $S_k = \{\Theta_k, \lambda_k, \gamma_k\}$, where $\Theta_k \in \mathbb{R}^9$ is a history of base orientations, λ_k and γ_k are the estimated roll and pitch of the slope respectively. The history of base orientations (roll, pitch and yaw) are provided for three time-steps at $k-2$, $k-1$ and k. It was observed that a history of body orientation was essential in training a robust locomotion policy (more analysis is presented in Section 4). The length of the history configuration was found empirically by analyzing the final performance of the policy.

Action space. The action space provides a set of transformations of the 2D semi-elliptic end-foot trajectories in a 3D work space of each leg. A semi-elliptic trajectory is chosen to allow for computationally simple implementation of the controller. The transformations consist of translation of the semi-ellipse along X, Y, and Z axes ($\mathbf{O\acute{O}}$ in Fig. 2), rotation about Z-axis (ϕ in Fig. 2) and variation of the length of the major axis ($\mathbf{ab}$ in Fig. 2) which is also the footstep length. All the transformations are defined in the local leg frame of reference (see Fig. 2). The action space is a 20-dimensional vector defining these five transformations for each of the four legs. Having obtained the end-foot trajectory after the corresponding transformations, the joint angles are obtained via an inverse kinematics solver. More details about the inverse kinematics for parallel five-bar linkages are provided in [3]. It is worth noting that the minor axis of the semi-ellipse is kept fixed as 0.06m since foot clearance can be varied by moving the trajectory higher when required. The detailed description of our semi-elliptical trajectory generator can be found in Appendix.

3 Training and Simulation

Having defined the model and the control methodology, we are now ready to discuss the policy and training algorithm used for Stoch 2.

3.1 Training algorithm

Augmented Random Search (ARS) [26] is a learning algorithm which is designed for finding linear deterministic policies, and is known to be on par with other model-free RL algorithms. We choose the policy to be $\pi(s) := Ms$, where $M \in \mathbb{R}^{20\times 11}$ is the matrix that maps the states to actions. Let the parameters of M be denoted by θ. In this case as we are not adding any constraints to the matrix M, the parameters θ of M are simply each element of M. Then the goal of ARS is to determine the θ, of the matrix M, that yields the best rewards which in turn leads to the best locomotion on non-flat terrain for Stoch 2.

ARS optimizes a parameterized policy by moving along the gradient of the function that maps the parameters of the policy to the expected reward. Since the function is stochastic in nature, different algorithms use different estimates of the gradient. ARS uses the method of finite differences as opposed to likelihood ratio methods used in other algorithms like PPO [29] or TRPO [30]. We use Version **V-1t** of ARS from [26]. We pick N i.i.d. directions $\{\delta_k\}_{\{k=1,\dots,N\}}$ from a normal distribution, where $N = 20 \times 11$ is the dimension of the policy parameters, i.e., $\theta \in \mathbb{R}^N$. With a scaling factor of $\nu > 0$, we perturb the policy parameters θ across each of these directions with the scaling ν. The policy is perturbed both along the positive and negative directions $\theta+\nu\delta_{(k)}$, $\theta-\nu\delta_{(k)}$. Executing this perturbed policy for one episode yields the return R for each direction. Therefore, for N directions, we collect $2N$ returns. We choose the best $N/2$ directions corresponding to the maximum returns. Let $\delta_{(k)}$, $k = 1, 2, \dots, N/2$ correspond to these best directions in decreasing order of returns. We update θ as follows:

$$\theta := \theta + \frac{\beta}{\frac{N}{2} * \sigma_R} \sum_{k=1}^{N/2} \left(R(\theta + \nu\delta_{(k)}) - R(\theta - \nu\delta_{(k)})\right) \delta_{(k)}, \tag{3}$$

where $\beta > 0$ is the step size, and σ_R is the standard deviation of the $N/2$ returns obtained. One iteration completes when θ gets updated in the matrix M. In one iteration we have $2N$ episodes i.e. 440 episodes and each episode has 400 steps.

3.2 Reward function

We design a reward function to encourage our robot to adapt to the underlying terrain and to move in the forward direction. We have

$$\begin{aligned} r = {} & G_{w_1}(r_{roll} - p_{roll}) + G_{w_2}(r_{pitch} - p_{pitch}) + G_{w_3}(r_{yaw} - d_{yaw}) + G_{w_4}(h - h_d) \\ & + W\Delta x - S_P, \end{aligned} \tag{4}$$

where $r_\square$ (with the square subscript being either $roll$, $pitch$ or yaw) is the orientation of the robot's base, $p_\square$ is the orientation of the support plane, d_{yaw} is the desired heading, h is the robot height, h_d is the desired height, Δx is the distance travelled, and S_P is the standing penalty. The mapping

$G : R \rightarrow [0, 1]$ is the Gaussian kernel function, and is given by $G_{w_j}(x) = \exp(-w_j * x^2)$, $w_j > 0$. Here $d_{yaw} = 0$ to maintain the robot heading constant along the X axis. Note that d_{yaw} can also be used as an input for steering the robot in the desired direction. We are also rewarding the robot for walking with the desired height so as to maintain sufficient clearance, and avoid multiple optimal solutions for different heights. We are normalizing Δx by dividing it by the maximum possible forward step length. W is its corresponding weight. With S_P, we are penalizing the robot for standing still, whenever the distance travelled is less than 2 cm in 50 time steps.

3.3 Policy training details

An open-source physics engine, PyBullet, was used to simulate and train our agent. Accurate measurements of link-lengths, moments of inertia and masses were stored in the URDF file. An asynchronous version of Augmented Random Search with 15 parallel agents was used to speed up the training on the robot. We trained our robot to walk on slopes of different inclination kept at varied orientations about the Z axis of the world frame. The inclinations are discretized at $0°, 5°, 7°, 9°, 11°$, and the orientations (yaw) in the range $0°$ to $90°$ degrees with $15°$ resolution, making 7 different orientations for each angle of inclination except for 0 degree, Thus giving us a grid of 29 combinations ($7 * 4 + 1$). If orientation is $0°$ then the incline is in uphill fashion with no roll and if orientation is $90°$, then the incline is in a side hill fashion with no pitch. After every policy update, we sample an incline degree and its orientation from their above mentioned ranges. Instead of starting with a random policy matrix, we used a guided initial policy to give a warm start to our ARS algorithm for policy exploration. We chose a few sets of hand-tuned fixed action values with reference to the telescopic strut strategy as discussed in Section 2.2, and simulated these actions to collect corresponding observations from the environment. Then, we did supervised learning to map the observations with the hand-tuned action values to obtain our guided initial policy. With our current action framework, it is quite intuitive to come up with such sets of action values that can perform fairly better than a random initial policy which significantly reduces the number of training iterations.

We are using the curriculum learning technique [31] for training the policy which makes the learning smooth, efficient, and effective. The curriculum is designed based on our intuition of tasks difficulty we begin with lower incline degrees and advances to steeper inclines. We introduce the curricula in two stages. In stage one we sampled $0°$, $5°$ and $7°$ degree slopes with all the described orientations. Similarly, in stage two we introduced $9°$ and $11°$ degree inclines after 30 training iterations. Also in this stage the sampling was favorable to the difficult tasks i.e steeper inclines. Similar approaches have been shown to be effective in [32], [33] and [34]. The hyper-parameters of ARS used in training are: Learning rate $(\beta) = 0.05$, noise $(\nu) = 0.04$ and episode length was of 400 steps.

Robustness. We incorporate robustness by using two schemes as similar to [5]: 1) domain randomization and 2) applying external forces on the robot base. In domain randomization, we randomize the surface friction, robot's mass distribution and the motor strength. The friction co-efficient was varied from 0.5 to 0.8, the additional masses of 0 to 200 grams were added to the front and back portions of the robot, and motor strength was varied from 5 to 8 Nm in addition to a randomly chosen inclined slope as discussed in Section 3.3. Similarly, we applied a random external force of $60 - 120$N to the base of our robot for a fixed interval of 10 steps (0.05 s) at the mid of episode.

Evaluation. We created a small evaluation dataset that spans over the entire training dataset. We evaluated our policy after every 3 training iterations and received the best total average reward of 1550 in only 72 iterations as shown in Fig. 3. Since we began our training using a guided initial policy, we achieved higher rewards in initial iterations itself as opposed to a random initial policy.

4 Results and discussion

With the proposed control framework, Stoch 2 was able to perform dynamic trotting on a variety of terrain slopes ($5°$, $7°$, $9°$, $11°$) kept at different orientations. Slopes $\geq 13°$ are not evaluated due to the kinematic limits of our robot. Measurements collected during a simulation experiment are shown in Fig. 4, where the robot starts trotting on flat terrain and then traverses a $9°$ slope by moving uphill and reaches an elevated flat terrain. The estimated pitch and roll angles of the terrain, the torso orientation, as well as the height of the robot along the normal to the terrain are shown in Fig. 4. The robot effectively adapts its pitch angle to match the terrain, while the tracking of the roll angle is

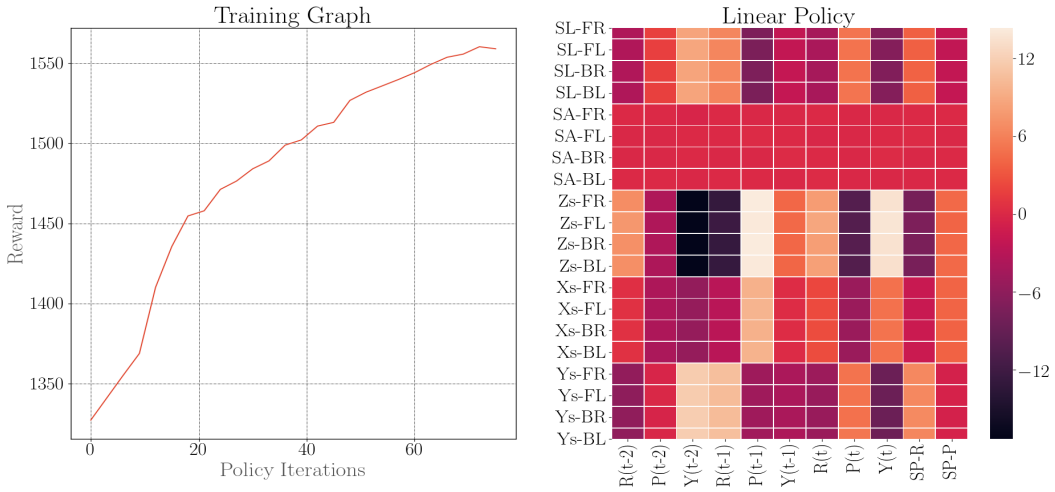

Figure 3: Left figure shows rewards evaluated after every three policy iterations. This policy is learned in 72 iterations. Right figure shows the heat-map of the learned matrix M. Here, labels SL, SA, Xs, Ys and Zs denotes the step length, steering angle, x shift, y shift and z shift respectively. $R(t), P(t), Z(t)$, denotes the Roll, Pitch, Yaw of the torso at timestep t. SP-R, SP-P corresponds to estimated support plane roll and pitch values respectively.

somewhat less accurate with a variation of ± 3 °. The yaw angle (heading direction) varies between ± 6 ° while climbing uphill but the robot is able to correct itself later.

Upon analysing the actions in Fig. 5 shows that the evolved policy shows similar characteristics as that of the *telescopic strut* strategy. In Fig. 5, the left figure shows the X shifts in the trajectory of the front left foot when the robot walks uphill. The X shift is increasing in a negative direction with the steepness in slopes. The figure on the right shows a similar pattern in Y shifts when the robot walks sidehill. The abduction joint limit was reached for a sideways slope of 11 degrees. This corresponds to the maximum allowed lateral shift (Y-shift) of 3.5 cm owing to the robot's kinematic limits. The linear policy (see Fig. 3) is capable of generalizing to negative roll and positive pitch environments as well. This is demonstrated in the supplementary video.

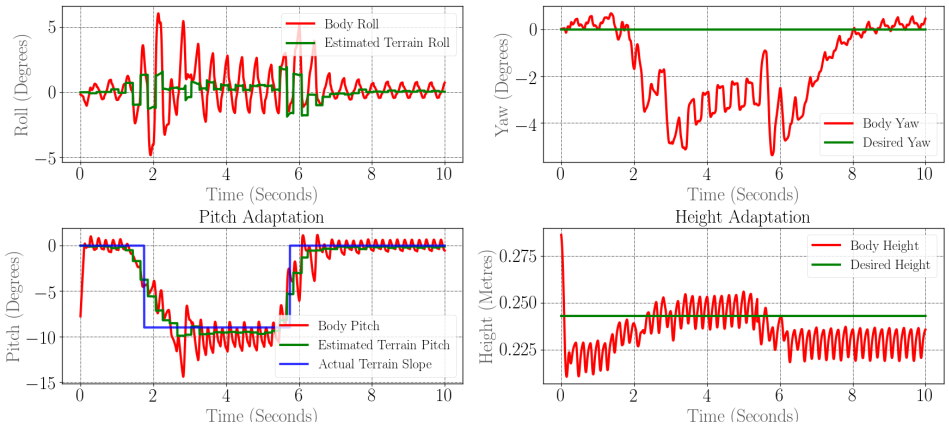

Figure 4: Figures showing how the policy adapts the center of mass orientation and height to that of the terrain in real time. The plane is at $-9°$ incline.

We demonstrate robustness to external disturbances by applying forces externally. The results of this are shown in Fig. 6. From time 0.7 to 0.9 second, an external force of 100N is applied onto the robot. This results in a change in the body orientation, which is then read by the linear policy to accordingly correct its direction of walking. As shown in Fig. 6, the step length and steering quickly adjust themselves within 0.5 s. In the video, we demonstrate robustness to external forces on arbitrary inclines. We also demonstrate robustness to changes in friction, masses and motor

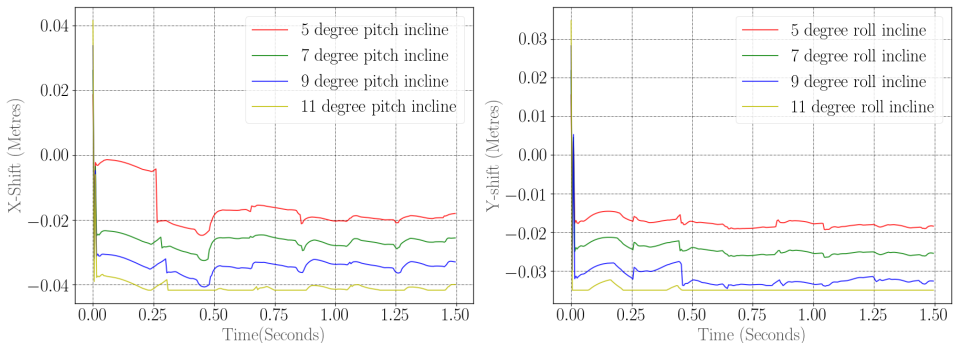

Figure 5: This figure shows how the response of the front left leg varies with different inclines. The left figure is for an up-down incline and shows the X-Shift parameter of the trajectory while the right figure is for a left-right incline and shows the Y-Shift parameter of the trajectory.

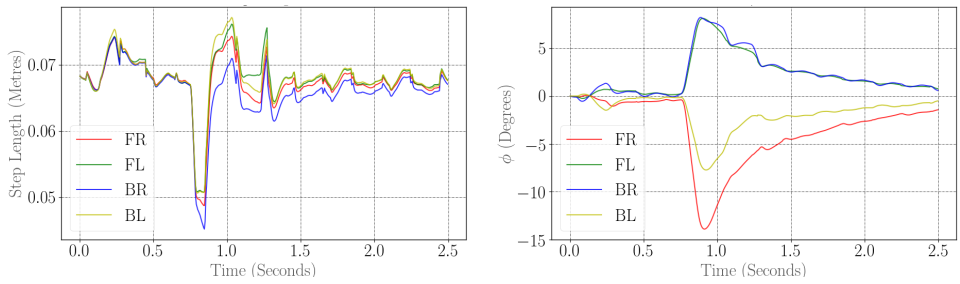

Figure 6: This figure shows how the legs adapt to external disturbances. The left figure indicates the step-length while the right figure indicates the steering angle. This is for a $-9°$ incline.

strengths on robot, thereby showing that domain randomization has made the linear policy more robust.

Our experiments shows that orientation history was essential in training a locomotion policy because it increases the representational capacity of a linear policy, allowing it to learn more complex behaviours as explained in [18]. Secondly, it enables the policy to detect foot slip and filter the sensor noise.

Other robots. We verified our approach on other robots with different physical properties as compared to Stoch2. Owing to the simplicity of our control framework, the mere scaling of the action space values was sufficient to match the kinematic limits of the new robot. The PD gains and the frequency of trajectory controller should also be updated accordingly. It is worth noting that no additional manual tuning is required to train a new policy. We validated our controller in robots namely, HyQ and Laikago that belong to the large and medium form factors respectively. The trained policies were found to generalize well on multiple inclinations ($0° - 15°$) and across varied orientations ($0° - 45°$) as demonstrated in the supplementary video.

Conclusion. We successfully demonstrated a single linear policy for quadrupedal locomotion on multiple types of sloped terrains, which is learnt via Augmented Random Search (ARS). By assuming stairs to be sloped terrains as done in [24], our policy is capable of traversing stairs as shown in the attached video. The linear policy takes the body and plane orientation as inputs and provides the trajectory parameters as outputs. This type of policy is easy to deploy on hardware, and, at the same time, do not require large computational resources, unlike the Deep Neural Network (DNN) based control algorithms existing in literature. Our approach can be used to quickly train linear policies for multiple robots, thus significantly simplifying the process of controller design and implementation for rough terrain walking on quadruped robots. Future work will involve testing the controller in hardware. The video submission accompanying this paper is shown here: youtu.be/KdQn1e3rI7o, and the code is released here: github.com/StochLab/SlopedTerrainLinearPolicy.

References

[1] N. Kau, A. Schultz, N. Ferrante, and P. Slade. Stanford doggo: An open-source, quasi-direct-drive quadruped. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6309–6315, May 2019. doi:10.1109/ICRA.2019.8794436.

[2] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, J. Fiene, A. Badri-Spröwitz, and L. Righetti. An open torque-controlled modular robot architecture for legged locomotion research, 2019.

[3] D. Dholakiya, S. Bhattacharya, A. Gunalan, A. Singla, S. Bhatnagar, B. Amrutur, A. Ghosal, and S. Kolathaya. Design, development and experimental realization of a quadrupedal research platform: Stoch. In *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, pages 229–234, April 2019. doi:10.1109/ICCAR.2019.8813480.

[4] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *ArXiv*, abs/1909.06586, 2019.

[5] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *CoRR*, abs/1804.10332, 2018. URL http://arxiv.org/abs/1804.10332.

[6] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. doi:10.1126/scirobotics.aau5872. URL https://robotics.sciencemag.org/content/4/26/eaau5872.

[7] M. H. Raibert. Legged robots. *Communications of the ACM*, 29(6):499–514, 1986.

[8] M. Vukobratović and B. Borovac. Zero-moment point—thirty five years of its life. *International journal of humanoid robotics*, 1(01):157–173, 2004.

[9] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56, Jan 2003. ISSN 0018-9286. doi:10.1109/TAC.2002.806653.

[10] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *2011 IEEE International Conference on Robotics and Automation*, pages 2992–2997, May 2011. doi:10.1109/ICRA.2011.5980229.

[11] L. Meier, D. Honegger, and M. Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240, May 2015. doi:10.1109/ICRA.2015.7140074.

[12] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[13] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, pages 2619–2624. IEEE.

[14] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *CoRR*, abs/1804.10332, 2018. URL http://arxiv.org/abs/1804.10332.

[15] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke. Policies modulating trajectory generators. In *Conference on Robot Learning*, pages 916–926, 2018.

[16] S. Kolathaya, A. Joglekar, S. Shetty, D. Dholakiya, Abhimanyu, A. Sagi, S. Bhattacharya, A. Singla, S. Bhatnagar, A. Ghosal, and B. Amrutur. Trajectory based deep policy search for quadrupedal walking. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–6, Oct 2019. doi:10.1109/RO-MAN46459.2019.8956369.

[17] S. Tirumala, K. Paigwar, A. Sagi, A. Joglekar, S. Bhatnagar, A. Ghoshal, B. Amrutur, and S. Kolathaya. Learning stable manoeuvres in quadruped robots from expert demonstrations. In *IEEE Robotics, Robot and Human Interaction (Ro-Man)*, 2020.

[18] A. Rajeswaran, K. Lowrey, E. Todorov, and S. Kakade. Towards generalization and simplicity in continuous control. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6553–6564, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[19] S. Tirumala, A. Sagi, K. Paigwar, A. Joglekar, S. Bhatnagar, A. Ghosal, B. Amrutur, and S. Kolathaya. Gait library synthesis for quadruped robots via augmented random search, 2019.

[20] A. W. Winkler, D. C. Bellicoso, M. Hutter, and J. Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters (RA-L)*, 3:1560–1567, July 2018. doi:10.1109/LRA.2018.2798285.

[21] O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon. Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations, 2020.

[22] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3699–3706, 2020.

[23] J. Z. Kolter, P. Abbeel, and A. Y. Ng. Hierarchical apprenticeship learning, with application to quadruped locomotion. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, page 769–776, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520.

[24] G. Bledt, M. Powell, B. Katz, J. Carlo, P. Wensing, and S. Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. 10 2018. doi:10.1109/IROS.2018.8593885.

[25] C. Dario Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter. Dynamic locomotion and whole-body control for quadrupedal robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3359–3365, 2017.

[26] H. Mania, A. Guy, and B. Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1800–1809, 2018.

[27] A. T. Sprowitz, A. Tuleu, M. Ajaoolleian, M. Vespignani, R. Moeckel, P. Eckert, M. D'Haene, J. Degrave, A. Nordmann, B. Schrauwen, et al. Oncilla robot: a versatile open-source quadruped research robot with compliant pantograph legs. *Frontiers in Robotics and AI*, 5: 67, 2018.

[28] C. Gehring, C. D. Bellicoso, S. Coros, M. Bloesch, P. Fankhauser, M. Hutter, and R. Siegwart. Dynamic trotting on slopes for quadrupedal robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5129–5135. IEEE, 2015.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`.

[30] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL `http://arxiv.org/abs/1502.05477`.

[31] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML'09*. ACM Press, 2009. doi:10.1145/1553374.1553380. URL `https://doi.org/10.1145/1553374.1553380`.

[32] W. Yu, G. Turk, and C. K. Liu. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics*, 37(4):1–12, Aug. 2018. doi:10.1145/3197517.3201397. URL `https://doi.org/10.1145/3197517.3201397`.

[33] N. H. K. Zhaoming Xie, Hung Yu Ling and M. van de Panne. Allsteps: Curriculum-driven learning of stepping stone skills. *ArXiv*, abs/2005.04323, 2020.

[34] S. Park, H. Ryu, S. Lee, S. Lee, and J. Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics*, 38(6):1–11, Nov. 2019. doi:10.1145/3355089.3356501. URL https://doi.org/10.1145/3355089.3356501.

5 Appendix

In a trot gait the diagonally opposite legs move in phase while the other two move out of phase. This means that the swing phase and stance phase of the legs last for the same duration. If we assume that the stance phase (the flat portion of the semi-elliptic trajectory) for any leg lasts for the first half of its cycle and the swing phase for the next half cycle, then taking the hip as the origin, we have the x, y, z position of the foot as:

$$\begin{aligned} &\text{stance}: x = l\cos\left(2\pi(1-\tau)\right), \quad y = 0,\ z = -0.243, && \tau \in [0, 0.5), \\ &\text{swing}: x = l\cos\left(2\pi(1-\tau)\right), \quad y = 0,\ z = -0.243 + 0.06\sin\left(2\pi(1-\tau)\right), && \tau \in [0.5, 1). \end{aligned}$$

where l is the step-length ($= \mathbf{ab}$ from Fig. 2), T is time period of one cycle, and $\tau = t/T$ is the time normalized w.r.t. the period T. The foot is required to move along a line which is 0.243m below the hip joint to maintain required height above the ground. This semi-ellipse which lies in the X-Z plane is transformed according to the action by:

$$x' = x_s + x\cos(\phi), \quad y' = y_s + x\sin(\phi), \quad z' = z_s + z, \tag{5}$$

where x', y' and z' are the coordinates of the foot in the leg frame of reference after the transformation, x_s, y_s and z_s are the shift (given by $\mathbf{O\acute{O}}$ in Fig. 2) in the three corresponding Cartesian directions as obtained from the controller, and ϕ is the yaw rotation of the semi-ellipse (see Fig. 2). x', y', z' are then represented in the common body frame by appropriate transformations.