Learning High-Value Footstep Placements for Quadruped Robots

Jeremiah M. Coholich, Zsolt Kira

***Abstract*—Learning policies for quadruped locomotion from scratch with reinforcement learning is challenging and motivates the need for behavioral priors. In this paper, we demonstrate that the combination of two such priors, gait trajectory generators and foot placement selection, are effective means to train robust policies. We specifically aim to learn a locomotion policy over a terrain consisting of stepping stones, where footstep placements are limited. To do this, we aim to add a behavioral prior for choosing footstep placements proposing a method to choose footstep targets which are optimal according to the value function of a policy trained to hit random footstep targets. We implement this method in simulation on flat ground and a difficult stepping stones terrain with some success and hypothesize about directions of future work which could improve our approach.**

***Index Terms*—Legged locomotion, reinforcement learning, rough terrain, quadruped robot**

I. Introduction and Related Work

In recent years, there has been an explosion of interest in using reinforcement learning (RL) for the planing and control of quadruped robots. It is possible to learn locomotion policies from scratch in an end-to-end manner [1]–[4]; however, this is typically challenging and requires extensive reward function engineering, hyperparameter tuning, or environment engineering. While RL promises to be a general framework for robots to autonomously acquire a wide variety of skills, legged locomotion poses a difficult learning and control problem due to underactuation, high-dimensional state spaces, and inverse-pendulum dynamics.

To avoid these issues and increase the success rate of learning locomotion policies, researchers began to incorporate various priors into RL algorithms. Most notably, [5] proposes a gait trajectory generator (TG) and limits the RL policy to learning residuals which are added to the output of the TG. This approach was subsequently adopted by many others [6]–[9] as it greatly improves the development time, sample efficiency, and success rate of learning locomotion policies. Other forms of prior knowledge are feasible as well. [10] acquires policies by imitating animals directly. [11] learns a policy that makes corrections to trajectories generated by an established physics-based planner, while [12], [13] constrain the learning process to respect physics-based feasibility criteria.

In this work, our ultimate goal is to learn polices which traverse footstep-placement-constrained terrain or "stepping stones". In this domain, legged robots clearly trump wheeled robots, since legged robots require only small, discrete contacts with terrain. Planning these contacts, or footstep placements, is therefore crucial in unlocking the full capability of legged robots. Our hypothesis is that biasing our policy architecture with focus on footstep placements will improve our ability to learn locomotion over stepping stones.

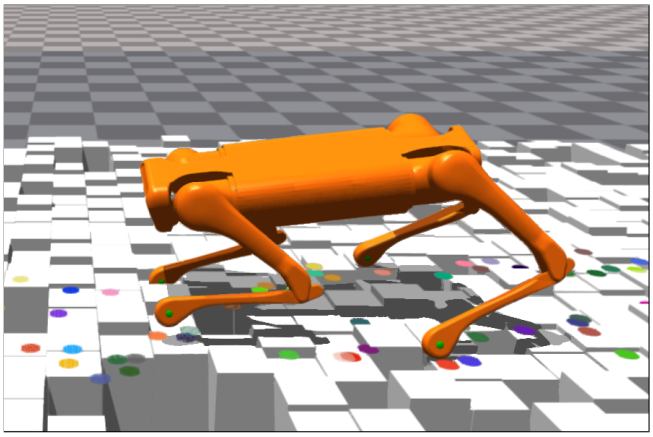


Fig. 1: The Aliengo robot in the training environment with randomly-generated footstep targets. The observation consists of robot proprioception and exteroception. The robot is rewarded for hitting footstep targets.

Our method involves a two-layer hierarchy, where footstep target locations are passed from the high level policy (HLP) to the low level policy (LLP). The core contribution is the development of a novel method for planning footstep locations which leverages the value function that is obtained for free during the training of the LLP. First, we train the LLP to control an Aliengo quadruped in simulation to hit a sequence of randomly-generated footstep targets over the stepping stones. Crucially, the relative location of future footstep targets is included in the LLP observation, which is the input to the policy and value functions. During runtime (after the LLP is trained), the HLP chooses footsteps according to an objective that combines value (according to the LLP value function) and a directional objective. The HLP therefore does not require RL training. The LLP value function encodes information about the surrounding terrain and current state of the robot, meaning that our method does not rely on heuristics or hand-derived constraints for footstep locations.

The works most similar to ours are [14] and [13]. In [14], the authors train bipeds in simulation to walk on increasingly

This work was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate (NDSEG) Fellowship Program.

difficult stepping stone sequences. In our work, we solve the problem of planning foot placements over terrain with many stepping stones, instead of traversing a fixed sequence. In [13], the authors propose a two-layer hierarchy where footstep targets are communicated between the high-level and low-level. However, they train their high-level with RL to satisfy a linear-program to check for the feasibility of each robot stance (in lieu of simulating physics), which is distinct from our value-function-based approach.

II. Low-Level Policy Training

The low-level policy (LLP) is trained on the task of controlling the quadruped to hit randomly-generated footstep targets.

A. RL Preliminaries

We formulate the task of hitting footstep targets as a Markov decision process, which is a tuple $(S, A, p, r, \rho_0, \gamma)$. S is the set of environment states, A is the set of policy actions, $p : S \times A \rightarrow S$ is the transition function of the environment, $r : S \times A \times S \rightarrow \mathbb{R}$ is the reward function, ρ_0 is the distribution of initial states, and γ is a discount factor. Our goal is to find an optimal policy $\pi^* : S \rightarrow A$ that maximizes the discounted sum of future rewards $J(\pi)$ over time horizon H.

$$J(\pi) = \mathbb{E}_{\{s_i, a_i\}_0^H \sim \pi, \rho_0} \left[\sum_{t=0}^{H} \gamma^t r\left(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}\right) \right] \quad (1)$$

$$\pi^* = \arg\max_{\pi} J(\pi) \quad (2)$$

We use proximal policy optimization [15] with generalized advantage estimation (GAE) [16] to solve for π^*, with $\lambda = 0.95$ and $\gamma = 0.99$. Additionally, we train a value network to predict the value of a state given the current policy. The value network is trained with the mean-squared error loss, where the target values are calculated from samples.

$$V_\pi\left(\mathbf{s_0}\right) := \mathbb{E}_{\tau \sim \pi} \left[r\left(\mathbf{s_0}, \mathbf{a}_0, \mathbf{s}_1\right) + \sum_{t=1}^{H} \gamma^t r\left(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}\right) \right] \quad (3)$$

The policy and value networks are parameterized as separate multilayer perceptrons with two hidden layers of 512 and 256 neurons each.

B. Training Environment

We train the LLP to hit randomly-placed footstep targets over a bed of procedurally-generated stepping stones. The training terrain contains varying difficulties of stepping stones as characterized by stepping stone density and stepping stone height variation.

As our action space, we use the Policies Modulating Trajectory Generators (PMTG) architecture [5] with the foot trajectories given in [6].

We generate sequences of footstep targets corresponding to a trotting gait, where the robot is tasked with hitting targets for two feet at a time, alternating between the front left and rear right feet and the front right and rear left feet. In our notation, each foot is associated index [1, 4, 2, 3], in the order the feet were listed. The pair of feet that the robot has active targets for at a given time is denoted as $\mathcal{N} \in \{\{1, 4\}, \{2, 3\}\}$. Each environment contains a sequence of footstep targets with a random heading and step length. The robot observation consists of proprioception, the relative position of the next footstep targets, and a heightmap of 284 points around the robot's feet.

We train the agent with a reward function that is dominated by a term to encourage the agent to hit footstep targets, given in equation 4. A target is considered hit if the foot makes contact with at least 5 N of force in a 7.5 cm radius around the target while the trajectory generator is in the contact phase for that foot. $\mathbf{d}$ is the distance in the x-y plane from the foot center to the target center. If a foot hits closer to the center of the target, the reward will increase by up to 50%. Equation 4 is heavily inspired by [14]. If the robot hits both active footstep targets at once, the reward for each foot is added, the total is tripled, and the environment advances to the next pair of targets.

$$1 + 0.5 * \left(1 - \frac{\mathbf{d}}{0.075\, m}\right) \quad (4)$$

The full reward function contains nine terms and is given in Appendix B. Further details about the training environment including terrain generation, footstep target generation, observations, and termination conditions are included in Appendix D

C. Training Results

We train policies in simulation using NVIDIA Isaac Gym [17]. We sample 60 steps from 3,333 environments for a total of 199,980 samples per policy update. Each policy is trained for 5k iterations or $\sim 10^9$ total samples. Figure 2 shows the RL training curve for our proposed method with and without the trajectory generator. Similar to prior works, the trajectory generator is crucial to getting our method to work repeatably.

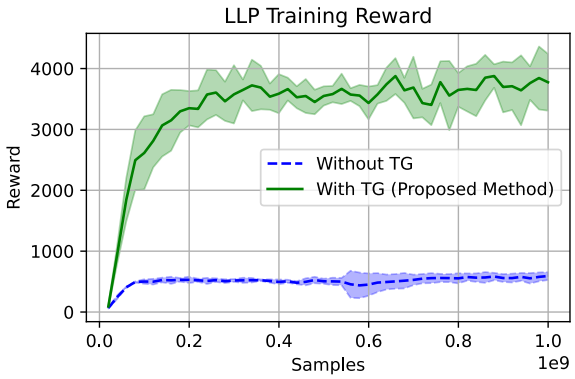


Fig. 2: Training reward obtained by the low-level policy with and without the trajectory generator. Each curve is the average of 5 random seeds, with the shaded regions indicating the standard deviation.

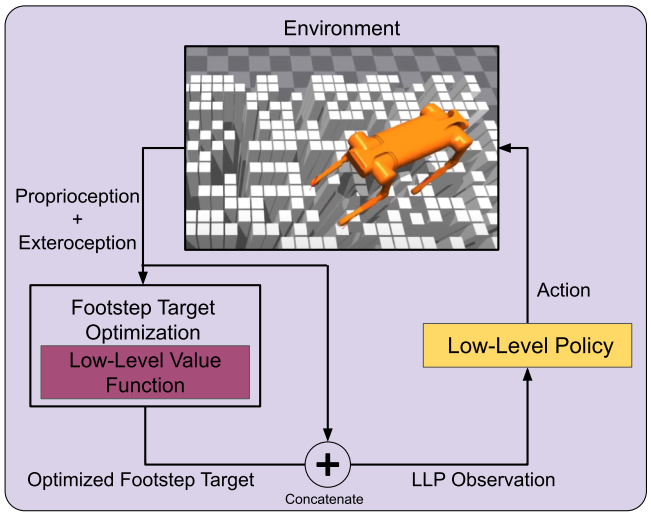


Fig. 3: Policy architecture incorporating a high-level footstep planner which makes use of the low-level policy's value function for selecting high-value footstep targets.

III. High Level Optimization

During runtime, the high-level policy (HLP) chooses footstep targets for the LLP. The high level policy has access to the current observation $\mathcal{O}_t$, the value network $V(\cdot)$ of the LLP, plus the robot yaw θ_{yaw}. The value network is a function of the relative location of footstep targets $\mathbf{d}$, since $\mathbf{d}$ is included in $\mathcal{O}_t$. The 4-dimensional action space of the HLP includes the x and y location of the next footstep targets, as defined in equation XX.

$$\mathbf{d}_{\text{next}} = \{d_{a,x}, d_{a,y}, d_{b,x}, d_{b,y}\} := \{d_{i,j} : i \in \mathcal{N},\ j \in \{x, y\}\}$$

First, let the HLP simply choose the highest-value footstep targets for the LLP. This is formulated as the optimization problem in Equation 5.

$$\mathbf{d}^*_{\text{next}} = \arg\max_{\mathbf{d}_{\text{next}}} V(\mathcal{O}) \tag{5}$$

Recall that the value function maps observations to expected discounted future rewards. Therefore, $\mathbf{d}^*_{next}$ are the footstep targets that yield the highest long-term reward for the LLP.

In order to encourage locomotion in a particular direction, we can add a secondary objective to equation 5 to yield equation 7. This directional objective $\mathbf{H}$ (equation 6) is parameterized by a heading angle α and a weight k. We use the robot's yaw to convert the relative footstep target positions ($\mathbf{d}_{next}$) from the robot's frame to the global frame. $R_z(\theta_{yaw})$ is the 3D rotation matrix about the vertical axis.

$$\mathbf{H} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \end{bmatrix} R_z(\theta_{yaw}) \begin{bmatrix} d_{a,x} & d_{b,x} \\ d_{a,y} & d_{a,y} \\ 0 & 0 \end{bmatrix} \tag{6}$$

$$\mathbf{d}^*_{\text{next}} = \arg\max_{\mathbf{d}_{\text{next}}} V(\mathbf{s}_0) + k\mathbf{H} \tag{7}$$

The hyperparameter k controls the tradeoff between picking targets that maximize the expected success of the LLP with picking targets that advance the robot's movement in direction α.

There are multiple options for solving equation 7, including stochastic gradient descent since the value function is differentiable. However, we instead choose to discretize the 4-dimensional space of $\mathbf{d}_{next}$ into a box $[-0.15, 0.15]_4$ with 17 points per axis and use grid search. There are several reasons for this. First of all, our action space is small enough in dimensionality for this simple method to be feasible. Additionally, the action space is tolerant of coarse discretizations. If the footstep targets are suboptimal due to a coarse discretization, the LLP will simply choose to place its feet slightly off-target; the HLP output targets are not a hard constraint. Finally, there is a benefit in performance from grid search. We can batch the set of test observation and evaluate them all in single forward pass through the value network using GPU-acceleration.

IV. Results and Experiments

A. Stepping In-place

In this experiment, the HLP chooses footstep targets with equation 5 (i.e. without any directional objective). We record the reward, episode length, and other metrics obtained by the LLP given these optimized footstep targets. In table I we compare these metrics to those obtained by an HLP outputting random and fixed footstep targets. Random footstep targets predictably lead to poor performance, but solving equation 5 curiously does not yield footstep targets that are better than fixed targets. More details can be found in Appendix C.

Possible explanations for this include:

- **Gap between training and testing environments**: During training, the footstep targets are fixed in the global frame. During testing, the footstep targets change at every timestep in response to the HLP output.
- **Optimization is too greedy**. The value function outputs an expected discounted sum of future rewards. We train with a discount factor of 0.99, which is perhaps too low and makes the policy short-sighted. Increasing the discount factor or incorporating other forms of look-ahead may improve performance.
- **Value function output is not smooth**: Neural networks are complex and highly non-linear functions which may produce unpredictable and unsmooth outputs. Perhaps

filtering the outputs of the LLP value function or running the optimization at a lower frequency will improve results.

B. Locomotion on challenging terrain

We evaluate our proposed method and end-to-end trained policy on a series of stepping-stone terrains of varying difficulty. We vary the stepping stone density and height variation. Figure 5 reports the success rates of each approach on each terrain, where success is defined as traveling 10 meters forward in under 10,000 timesteps without termination. The end-to-end approach outperforms our proposed method. Results for a wider range of environments and metrics can be found in Appendix A.

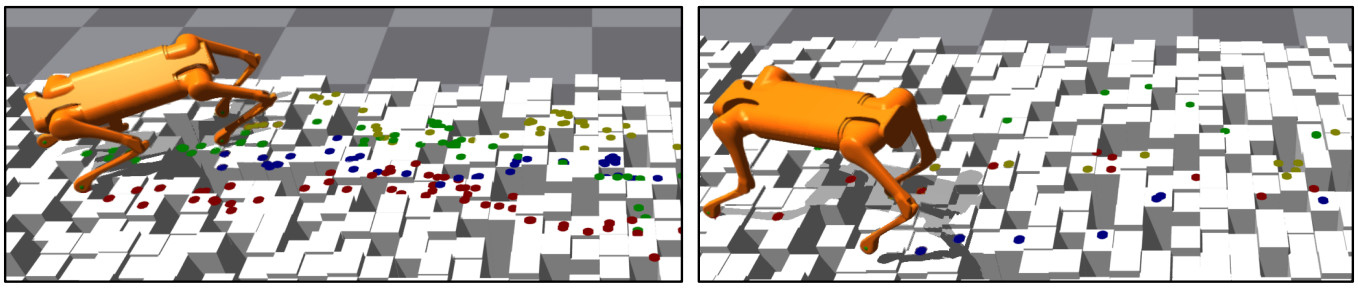
In figure 4, we show the quadruped traversing the stepping stone environment and plot the locations where each foot makes contact with the terrain. The end-to-end policy learns to take longer steps compared to our method.

V. Conclusion

We propose a strategy for incorporating an architectural bias into a learned policy for quadruped locomotion, with a priority on learning what good footstep placements are. Specifically, we choose footstep targets which are optimal according to the the value function of a policy that is trained to hit random footstep targets. We implement this method in simulation on flat ground and on difficult stepping stones terrain. Thus far, we have not obtained evidence that our proposed method outperforms an end-to-end policy. The root of the issue is likely related to the inability of the HLP to outperform fixed footstep targets (as explained in section IV-A). Changing our approach to decrease the short-sightedness of our HLP, smooth the HLP outputs, and eliminate the gap between training and testing environments will likely help. Additional future work includes swapping the terrain heightmap observation for egocentric vision and running real-world experiments.

TABLE I: Metrics for the stepping-in-place task for different high-level policies. The random agent selects a footstep target within the search space (a 0.3 meter box around the previous footstep target) at every timestep with a uniformly random probability. The fixed agent always gives the same footstep targets corresponding to a neutral standing position. The optimized agent gives the highest value footstep targets according to equation 5. The metrics are averages across 20 rollouts for each of 5 policies trained with a different random seed (100 rollouts total). The environment is set to timeout after 5,000 timesteps.

Footstep Target Selection Method	Reward per Timestep	Reward per Footstep	Reward per Episode	Episode Length (timesteps)	Footstep Targets Hit	Timesteps per Footstep
Random	1.21	**52.11**	668.	556.	12.95	43.62
Fixed	**3.93**	39.29	**19,643.**	**5,000.**	**499.96**	**10.00**
Optimized	3.51	41.39	6,905.	1,878.	170.63	12.00



(a) (b)

Fig. 4: Quadruped crossing the stepping stones terrain with density of 75% and 0.1 meter height variation using (a) our proposed approach and (b) a end-to-end learned policy. The previous foot contact locations with terrain are plotted. (●: Front Left Foot, ●: Rear Left Foot, ●: Front Right Foot, ●: Rear Right Foot)

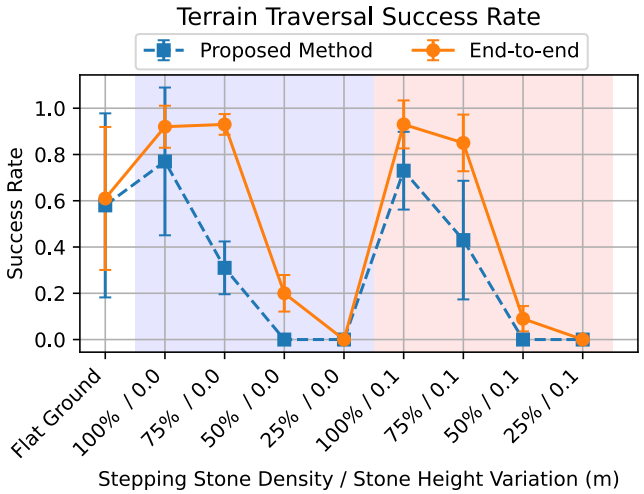


Fig. 5: A comparison of the proposed value-function-based approach with an end-to-end learned policy, averaged across 100 episodes (5 policies, 20 episodes each). Error bars give the standard deviation between policies. Background colors delineate between environments with different stepping stone height variation.

REFERENCES

[1] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," *arXiv preprint arXiv:1812.11103*, 2018.

[2] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," *arXiv preprint arXiv:2002.08550*, 2020.

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

[4] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, "Learning fast adaptation with meta strategy optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2950–2957, 2020.

[5] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," in *Conference on Robot Learning*. PMLR, 2018, pp. 916–926.

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

[7] 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.

[8] A. Escontrela, G. Yu, P. Xu, A. Iscen, and J. Tan, "Zero-shot terrain generalization for visual locomotion policies," *arXiv preprint arXiv:2011.05513*, 2020.

[9] A. Iscen, G. Yu, A. Escontrela, D. Jain, J. Tan, and K. Caluwaerts, "Learning agile locomotion skills with a mentor," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2019–2025.

[10] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[11] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Real-time trajectory adaptation for quadrupedal locomotion using deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5973–5979.

[12] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3642–3649, 2020.

[13] 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*, vol. 5, no. 2, pp. 3699–3706, 2020.

[14] Z. Xie, H. Y. Ling, N. H. Kim, and M. van de Panne, "Allsteps: Curriculum-driven learning of stepping stone skills," in *Computer Graphics Forum*, vol. 39, no. 8. Wiley Online Library, 2020, pp. 213–224.

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

[16] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[17] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

APPENDIX A
EVALUATION OF POLICIES ACROSS TERRAIN DIFFICULTY

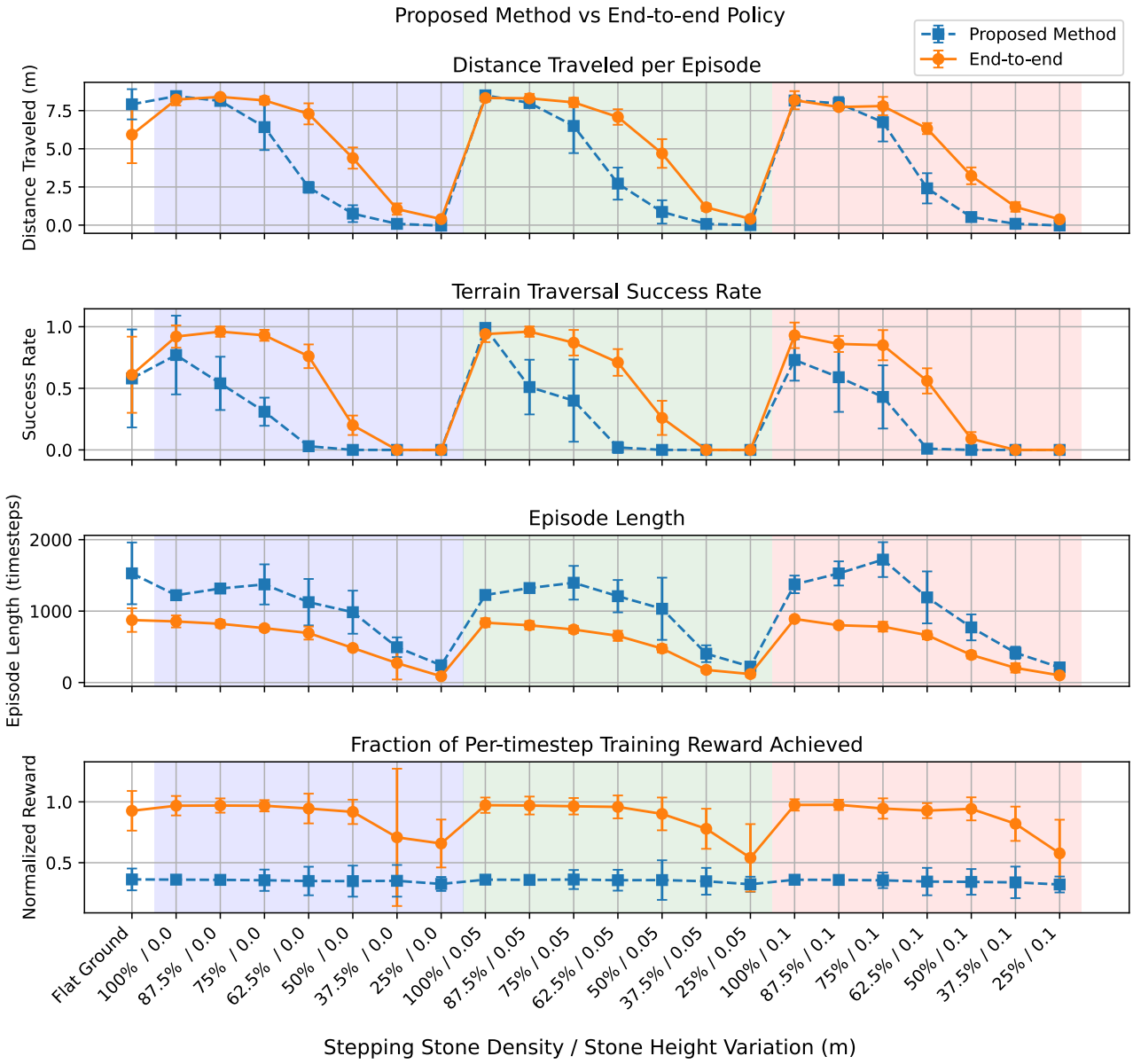


Fig. 6: A comparison of the proposed value-function-based approach with an end-to-end learned policy. Each data point is an average across 100 episodes (5 policies, 20 episodes each). The error bars show the standard deviation between the averages of the 5 policies. The background colors group environments with the same stepping stone height variation.

APPENDIX B
REWARD FUNCTION

The reward function for the LLP is the sum of the following reward terms.

A. Hit Footstep Target Reward

A simplified version of this reward term is given in equation 4 in Section II-B. $\mathbf{d}_{i,t}$ is the distance in the xy plane from the foot center to the target center. If a foot hits the center of the target, the reward increases 50%. $h_{i,t} \in \{0, 1\}$ indicates whether or not foot i has hit its footstep target at time t. $\mathbf{d}_{hit}$ is the xy distance threshold for hitting a footstep targets (set to 7.5 cm). The factor $\left(2.0 \prod_{i \in \mathcal{N}} h_{i,t} + 1\right)$ triples the per-foot rewards if both footstep targets are achieved on the same timestep.

$$2.0 \left(2.0 \prod_{i \in \mathcal{N}} h_{i,t} + 1\right) \sum_{i \in \mathcal{N}} h_{i,t} \left(1 + 0.5 * \left(1 - \frac{\mathbf{d}_{i,t}}{\mathbf{d}_{hit}}\right)\right)$$

B. Torque Penalty

This term penalizes the sum of squared joint torques.

$$-1 \times 10^{-6} \left\|\tau\right\|_2^2$$

C. Smoothness Reward

More realistic robot motions are obtained by penalizing the norm of second-order finite differences of the actions.

$$-0.03125 \left\|\mathbf{a_t} - 2\mathbf{a_{t-1}} + \mathbf{a_{t-2}}\right\|_2$$

D. Foot Slip Penalty

This term penalizes xy translation greater than 2 cm of feet that are in contact. $c_{i,t}$ gives the vertical contact force for foot i at time t in Newtons. $x_{i,t} \in \mathbb{R}^2$ is the global position in meters on the x-y plane for foot i at time t.

$$-0.0625 \left|\left\{i : \left\|x_{i,t} - x_{i,t-1}\right\|_2 > 0.02,\ c_{i,t} > 0,\ c_{i,t-1} > 0\right\}\right|$$

E. Velocity Towards Target

In order to provide denser rewards that encourage hitting footstep targets, we reward foot velocity towards targets.

$$-0.25 \sum_{i \in \mathcal{N}} \dot{\mathbf{d}}_{i,t}$$

F. Foot Stay Reward

Only two feet have active footstep targets at any time. In order to prevent the robot from immediately moving its feet off of footstep targets after they are hit, we reward the agent for keeping its feet on previous targets.

$$0.25 \sum_{i \in \{1,2,3,4\} \setminus \mathcal{N}} h_{i,t} \left(1 + 0.5 * \left(1 - \frac{\mathbf{d}_{i,t}}{0.075}\right)\right)$$

G. Collision Penalty

$\mathbf{g}$ is the number of robot linkages in that have collided with other linkages or terrain, excluding foot collisions with terrain.

$$-0.125\mathbf{g}$$

H. Trajectory Generator Swing Phase Reward

This term rewards the trajectory generator for entering mid-swing-phase, weighted by the frequency of the trajectory generator (f_{PMTG}). $\mathbb{1}\{\cdot\}$ is the indicator function. Empirically, we observe that this term is essential for preventing the learning of a degenerate policy that remains still and collects maximum rewards for foot stay, foot slip, smoothness, torque, and orientation.

$$\mathbb{1}_{[0.25\pi, 0.75\pi] \cup [1.25\pi, 1.75\pi]}\{\phi_t\}\ f_{\text{PMTG}}$$

I. Orientation Penalty

This term penalizes the roll and pitch of the robot.

$$-0.1\ (|\theta_{roll}| + |\theta_{pitch}|)$$

APPENDIX C
IN-PLACE FOOTSTEP OPTIMIZATION

Figure 7 shows the average trajectories taken by policies trained with different random seeds when the HLP was tasked with solving Equation 5. For each random seeds, the robot's position drifts in a different direction with a different amount of rotation. As explained in section IV-A, the robot obtains lower reward on average when following the optimized footstep targets versus the fixed targets in the first plot.

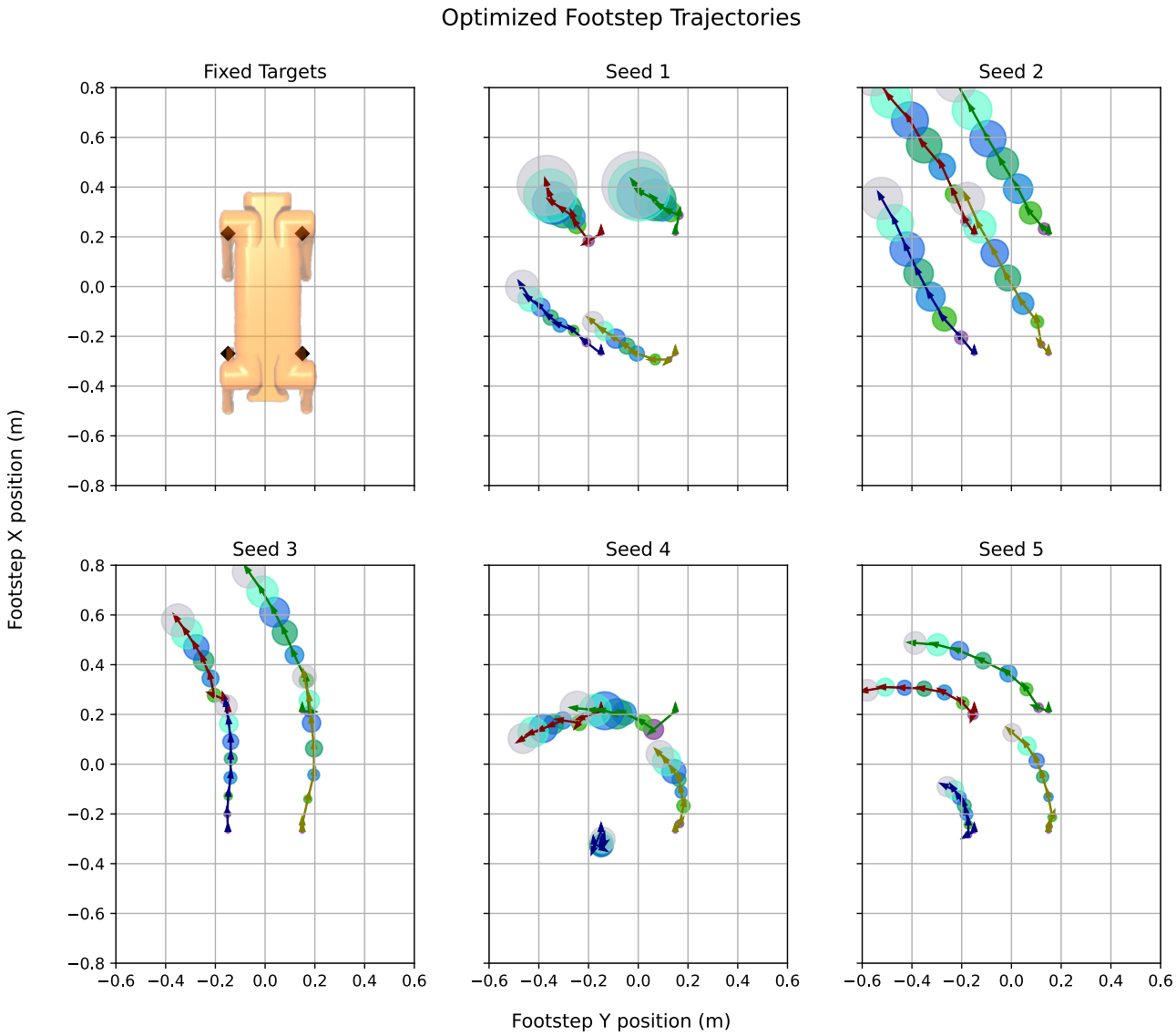


Fig. 7: Plots of data obtained from stepping-in-place experiments from Section IV-A. The plot in the top-left corner shows the fixed footstep targets which correspond to a neutral robot standing position. Other plots show the trajectory of footstep targets for five policies trained with different random seeds averaged over 20 rollouts. Each circle represents an average footstep target location, with its size indicating variance.

APPENDIX D
TRAINING ENVIRONMENT

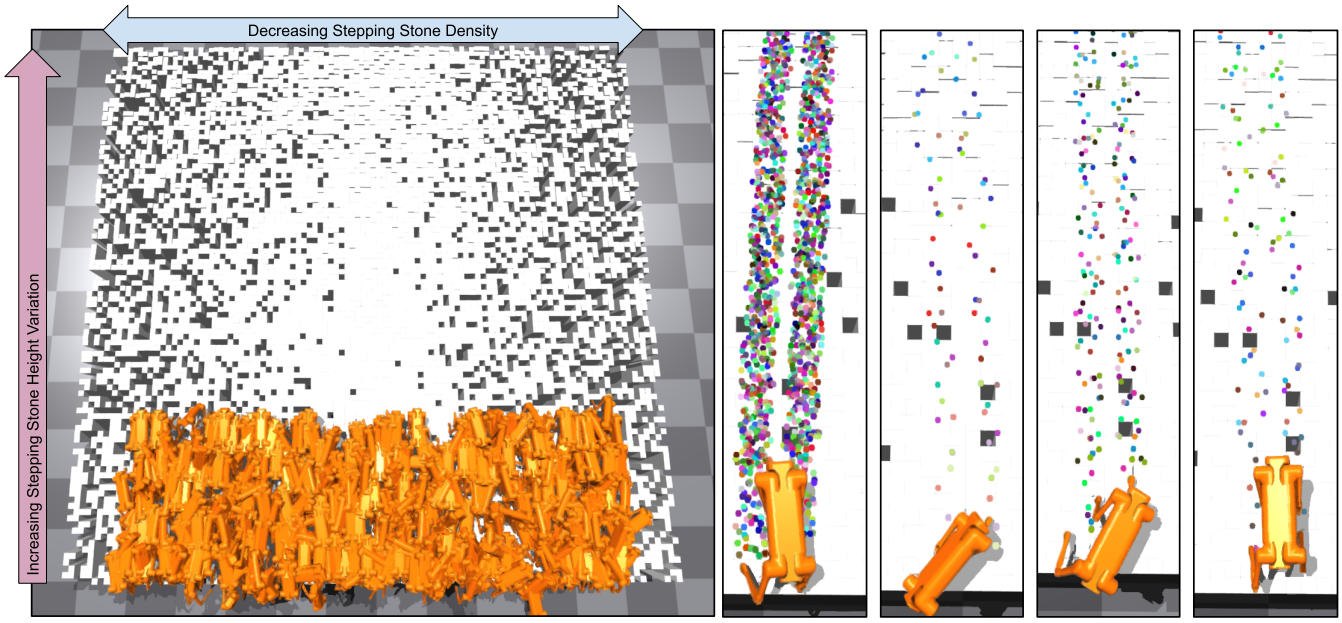


Fig. 8: **Left**: The training environment at 10% scale. The actual training environment is 10x as wide with 10 as many robots simulated in parallel. **Right**: Samples of randomly generated footstep target sequences for training. Paired targets are plotted in the same color.

1) Footstep Target Generation: We generate sequences of footstep targets corresponding to a trotting gait, where the robot is tasked with hitting targets for two feet at a time, alternating between the front left and rear right feet and the front right and rear left feet. In our notation, each foot is associated index [1, 4, 2, 3], in the order the feet were listed. The pair of feet that the robot has active targets for at a given time is denoted as $\mathcal{N} \in \{\{1, 4\}, \{2, 3\}\}$. We have found empirically that the trotting gait is the most suitable for implementation on the Aliengo robot in terms of robustness and speed.

A target is considered hit if the foot makes contact with at least 5 N of force in a 7.5 cm radius around the target while the trajectory generator is in the contact phase for that foot. If the robot hits both active footstep targets at once, the environment advances to the next pair of targets. The agent receives a reward for hitting footstep targets, plus a bonus for hitting both at once and another bonus for hitting closer to the center of the target, inspired by [14].

Each environment contains a sequence of footstep targets parameterized by a random step length sampled from $U(0, 0.1)$ m and a random heading sampled from $U(-8^{\circ}, 8^{\circ})$. Additionally, all targets are independently shifted by $U(-0.1, 0.1)$ m in the x and y directions.

2) Observation Space: The policy observation is a vector $\mathcal{O}_t = \{\mathbf{x},\ \dot{\mathbf{x}},\ \tau,\ \mathbf{O},\ \mathbf{c},\ \mathbf{d},\ \mathbf{f},\ \cos\phi,\ \sin\phi,\ \mathbf{a_{t-1}},\ \mathbf{a_{t-2}},\ \mathcal{F}\} \in \mathbb{R}_{372}$ where $\mathbf{x}$ represents the foot positions in the hip-frame, $\dot{\mathbf{x}}$ is the foot velocities, τ is the joint torques, $\mathbf{O}$ is the IMU data consisting of $\{\theta_{roll}, \theta_{pitch}, \dot{\theta}_{roll}, \dot{\theta}_{pitch}\}$, $\mathbf{c} \in \{0, 1\}_4$ is a vector giving the contact state of each foot, $\mathbf{d} = \{d_{1,x}, d_{1,y}, d_{2,x}, d_{2,y}, d_{3,x}, d_{3,y}, d_{4,x}, d_{4,y}\}$ gives the x and y distances from each foot to the corresponding to the next (for $i \in \mathcal{N}$) or previous ($i \notin \mathcal{N}$) footstep targets, $\mathbf{f} \in \{0, 1\}_4$ is a multi-hot encoding of $\mathcal{N}$, ϕ is the phase of the trajectory generator, $\mathbf{a_{t-1}}$ and $\mathbf{a_{t-2}}$ give the previous two actions taken by the policy, and $\mathcal{F}$ is a scan of points around each foot.

The foot scan $\mathcal{F}$ gives the difference in height between each foot and 71 points on the surrounding terrain. These points are equally spaced around seven rings with radii [0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.2], where each ring contains 10 points. There is one additional point directly under each foot. $\mathcal{F}$ is clipped to [-1.0, 1.0] m.

As in [12], the observation is corrupted with Gaussian noise with zero mean and standard deviation given by the vector $\{[0.001]_{12}, [0.02]_{12}, [0.1]_{12}, [0.05, 0.05, 0.1, 0.1], [0]_4, [0]_8, [0]_4, [0.05]_{284}, [0]_{15}, [0]_{15}\}$ to simulate sensor noise and inaccuracy.

3) Termination Conditions: A training episode will terminate if any of the following conditions are met:

- $|\theta_{roll}| > 90^{\circ}$ or $|\theta_{pitch}| > 90^{\circ}$
- The number of timesteps exceeds 3600

- It has been 300 timesteps (5 seconds) since the robot last hit a pair of footstep targets.
- The robot is within 0.5 meters of the end of the stepping stone terrain
- The robot has hit all of the footstep targets

4) Action Space: We use the Policies Modulating Trajectory Generators (PMTG) architecture [5] with the foot trajectories given in [6]. Our 15-dimensional action space consists of trajectory generator frequency, step length, standing height, and 12 residuals corresponding to the 3D position of each foot. The trajectory generator outputs foot positions in the hip-centered frame (as defined in [6]). These are converted into joint positions with analytical inverse kinematics and tracked with PD control.

5) Procedural Terrain Generation: We generate a bed of non-overlapping stepping stones to be shared between all environments, with varying the height variation and density as shown in Fig 8. All robots are spawned along the y-axis and tasked with traveling in the $+x$ direction. Each stepping stone surface is a 0.1 m square spaced 0.001 m from neighboring stones. The heights of the stepping stones are drawn randomly from a height of 2 meters plus or minus half of the stepping stone height variation.