BeeTLe: Blind Terrain-aware Learned Locomotion

Rogier Fransen, Richard Bowden and Simon Hadfield
Centre for Vision Speech and Signal Processing, University of Surrey, Guildford, Surrey, GU2 7XH, UK
{r.fransen, r.bowden, s.hadfield}@surrey.ac.uk

***Abstract*— One of the largest challenges in the deployment of legged robots in the real world is deriving effective general gaits. In this paper, we present BeeTLe, which is a framework that enables terrain aware locomotion without the need for dedicated terrain sensors. BeeTLe is realised as a multi-expert policy Reinforcement Learning (RL) algorithm. This enables multiple gaits, applicable to different surface types, to be stored and shared in a single policy. Sensor free terrain awareness is incorporated using a Recurrent Neural Network (RNN) to infer surface type purely from actuator positions over time. The RNN achieves an accuracy of 94% in terrain identification out of 8 possible options. We demonstrate that BeeTLe achieves a greater performance than the baselines across a series of challenges including: the traversal of a flat plane, a tilted plane, a sequence of tilted planes and geometry modelling a natural hilly terrain. This is despite not seeing the sequence of tilted planes and the natural hilly terrain during training.**

The code, policy and simulated environments are available at: `https://gitlab.surrey.ac.uk/rf00350/BeeTLe`

I. INTRODUCTION

The derivation of effective gaits for real world legged robots, remains a major challenge for the field. Different environments required different strategies and gaits need to be adaptable to on-the-fly changes in terrain. For example, a gait strategy used to successfully walk across a slippery laboratory floor, will not work when climbing a grassy incline. Many existing gaits are hand-crafted for a set of different terrain types that may be encountered. However, this causes problems when traversing terrain types not originally considered. This approach also implies an additional sensor load, as the robot must be able to identify the type of surface it is walking on.

In this paper, we propose BeeTLe as a solution to both of these challenges. Firstly, RL is used to train an ensemble of expert walking policies in an unsupervised manner. The terrain types to be navigated are not restrictive. Instead, when the robot encounters new terrain, an attention network appropriately combines the knowledge from the experts to derive a suitable new gait. This allows it to scale effectively to a much larger range of surface types, including those not seen during training as shown in Figure 1.

Secondly, we do not follow the traditional approach of deploying additional sensors (and associated data processing) to determine the characteristics of the surface being traversed. Instead, we propose a purely proprioceptive attention network. This observes the relationship between control inputs and the robot's internal state (i.e. the angular position of actuators) over time. Based on this, an RNN infers the

This work was supported by the Engineering & Physical Sciences Research Council under grant agreement EP/S035761/1

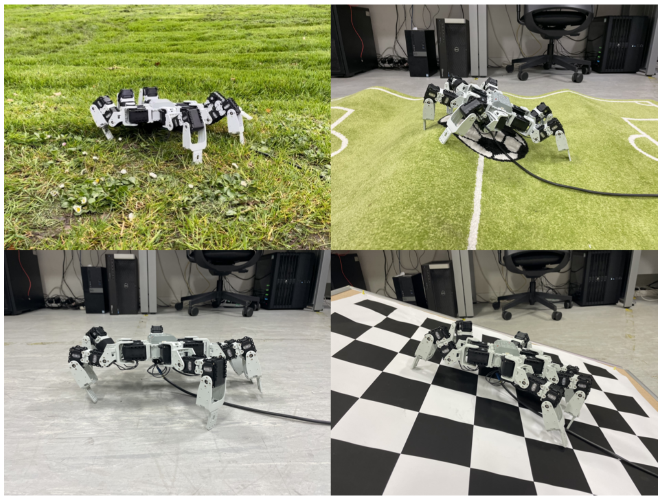


Fig. 1: Terrain aware hexapod robot using motor feedback to determine terrain type and adjust its walking policy accordingly

properties of the surface, and thus the suitability of the various experts within the ensemble.

Finally, we implement a system that translates actions from the policy trained in simulation, into control signals understood by the actuators of a real hexapod robot. The same system also translates back the angular positions read from the robot's actuators, into observations compatible with the learnt policy. This allows us to demonstrate the real-life deployment of our learned locomotion strategy.

To summarise the contributions of this paper are:

1) A general-purpose locomotion strategy based on blending an ensemble of expert policies on-the-fly.
2) A proprioceptive attention network, which infers terrain type and estimates expert suitability, based entirely on internal sensing of the robot's state over time.
3) A practical demonstration on commercially available robotic hardware, as well as a full code release including both the trained policy and the simulated environments, ensuring reproducibility.

II. LITERATURE REVIEW

A. *Learned Legged Locomotion*

Machine learning has been extensively applied to the problem of locomotion. Specifically, algorithms have been developed that allow legged robots to learn how to walk, run, or perform complex movements while adapting to their surroundings. Tan et al. [1] shows that a locomotion controller can be learned from scratch using RL with simple reward signals. More advanced controllers utilise state history to improve performance. Böhm et al. [2] use a gated recurrent unit feature extractor to encode a low-dimension representation of the state observation sequence before passing it to the RL training procedure. Similarly, Lee et al. [3] utilise a temporal convolutional network to produce a latent representation of the environment information using the history of observations. Furthermore, Lai et al. [4] propose a transformer model for locomotion control on various terrains.

To improve the generality of learned controllers, Christmann et al. [5] propose distributing the complexity of different gaits into dedicated locomotion policies, and transitioning between them depending on the latent state representations. Ubellacker and Ames [6] show that robustness in legged locomotion can be achieved by switching to and transitioning through suitable motion primitives. Yang et al. [7] propose a multi-expert learning architecture, which contains multiple expert neural networks and uses a gating neural network to fuse them dynamically into a new neural network. BeeTLe also consists of multiple expert networks, but utilises an RNN to produce the weights by which to blend the output of each expert.

B. *Terrain Awareness for Locomotion*

Common approaches for traversing varied terrains rely heavily on visual perception. Gangapurwala et al. [8] utilise proprioceptive and exteroceptive feedback to map sensory information and desired base velocity commands into footstep plans using an RL policy. Miki et al. [9] integrate exteroceptive and proprioceptive perception using an attention-based recurrent encoder to combine them into an integrated belief state. Acero et al. [10] also propose a learning framework that integrates exteroceptive and proprioceptive states, but additionally show that perceptual locomotion can be achieved using only sparse visual observations.

Many approaches seek to reduce the need for exteroceptive sensors by solely relying on proprioception [1], [7], [11]–[16]. Kumar et al. [11] use an environment encoder network, which initially uses privileged information about the environment from the simulator, to produce a latent space representation of the environment. Later, an adaptation module is trained using supervised learning to mimic the output of this environment encoder network using only the action and state space history as an input. Their state consists of joint positions, joint velocities, roll, pitch and binarized foot contact indicators. These are collected from the motor encoders, an IMU and foot sensors.

Nahrendra et al. [12], only use proprioceptive observations from joint encoders and an IMU. They use a variational autoencoder to implicitly imagine terrain properties from the history of observations. These terrain properties are in the form of a latent state, which is subsequently passed into a main policy network. Margolis and Agrawal [16] also only use proprioceptive observations, but use an accelerometer instead of an IMU, as well as joint encoders, to estimate the environment properties. In contrast to this, BeeTLe reduces the sensor load out further and solely uses the observations from joint encoders to predict the terrain properties.

III. Method

A. *Policy Ensemble*

Overview As outlined in Figure 2, the individual expert policies within BeeTLe's multi-expert ensemble operate in an actor-critic framework. Specifically, the ensemble contains n sets of independent actor and critic networks, each pair of which is referred to as a single expert. The RNN attention network produces a set of attention weights rating the relevance of the current state to the different experts. These attention weights are used to blend the outputs of the individual experts, producing the final output.

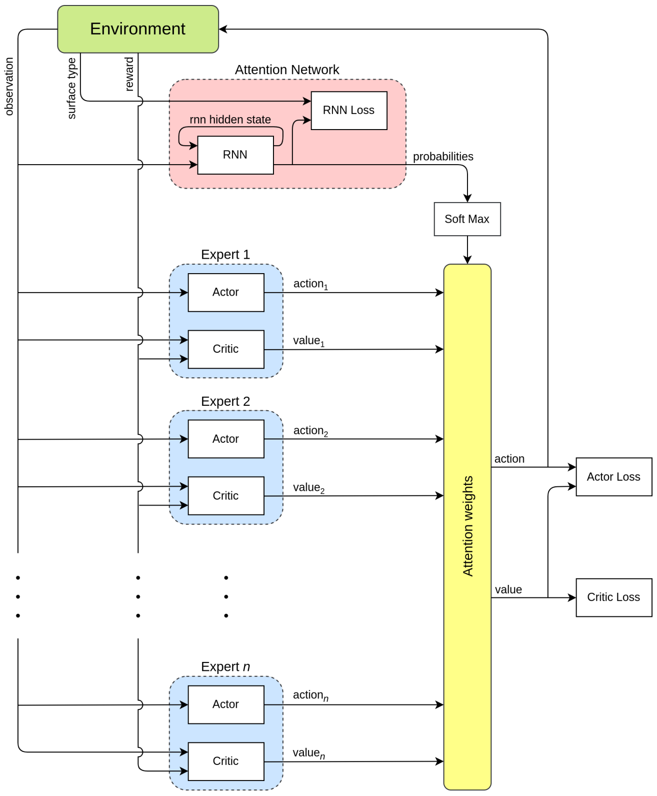


Fig. 2: Flow diagram showing BeeTLe's structure at training time

Expert Networks Within the ensemble, the ith expert network learns how to rate the value $V_i(s_t)$ of a particular environmental state s_t (the critic), and how to map observations to actions a_i (the actor policy $\pi_i(a_i|s_t)$), in a manner that results in walking for its observed training terrain type. The overall output of the ensemble is the average of the individual expert outputs, weighted according to the attention network scores ω_i

$$a = \sum_{i=1}^{n} a_i \omega_i(s_t), \qquad V(s_t) = \sum_{i=1}^{n} V_i(s_t)\omega_i(s_t). \tag{1}$$

We then define our RL losses using the standard Proximal Policy Optimization (PPO) formulation [17], which are the value loss L_V and the policy loss L_π, using these ensemble outputs as follows

$$L_V = \frac{1}{T}\sum_{t=1}^{T} \left(V\left(s_t\right) - \left(r_t + \gamma V\left(s_{t+1}\right)\right)\right)^2, \tag{2}$$

where T is the number of timesteps in the rollout buffer, r_t is the reward obtained at timestep t and γ is the discount factor that determines the importance of future rewards. As well as

$$L_\pi = -\frac{1}{T}\sum_{t=1}^{T}\min\Bigg(\frac{\pi(a|s_t)}{\pi_{\text{old}}(a|s_t)}A(s_t,a), \\ \text{clip}\left(\frac{\pi(a|s_t)}{\pi_{\text{old}}(a|s_t)}, 1-\epsilon, 1+\epsilon\right)A(s_t,a)\Bigg), \quad (3)$$

where $\pi(a|s_t)$ is the current policy's probability of taking action a in state s_t, $\pi_{\text{old}}(a|s_t)$ is the probability of taking action a according to the policy at the previous iteration, $A(s_t,a)$ is the advantage function (which is the difference between the Q function and the value function) and ϵ is a hyperparameter that determines the extent of policy clipping.

Thus each critic uses the reward returned by the environment to tell each actor how well it did. It is worth noting that on the backward pass, gradients are distributed to the individual experts according to their blending weights ω. As such, experts which have a greater impact on the result experience a more significant update to their model parameters.

Intuitively, if we imagine the attention network as an oracle function $\mathcal{O}$, which groups input states into non-overlapping clusters based on terrain categories, then the attention weights become a one-hot vector. In this case, a single expert is chosen to act alone at each step. This approach is easy to learn, and is used to initialise the training of BeeTLe. However, it is restrictive as only a limited number of terrain categories can be modelled, and there is no elegant way to handle new terrain types. It also leads to significant redundancy as there are certain skills that must be learned by every expert, as there is no sharing of information.

In contrast, our "soft-blending" approach enables information sharing between experts and a far more nuanced handling of diverse terrain types.

Attention Network The attention network is formed of an RNN and a cross entropy loss function. It produces the expert blending weights ω_t using the observation from the environment s_t, as well as its previous hidden state h_{t-1}

$$h_t(s_t) = b_h^2 + W_h^2 \max\left(0, b_h^1 + W_h^1\left(s_t \oplus h_{t-1}\right)\right), \quad (4)$$

$$\omega_t(s_t) = \sigma\left(b_\omega^2 + W_\omega^2 \max\left(0, b_\omega^1 + W_\omega^1\left(s_t \oplus h_{t-1}\right)\right)\right), \quad (5)$$

where b are the biases and W are the weights for the various linear network layers and σ is a softmax function, which the RNN output is passed through to form the final attention weights. Here the $\oplus$ symbols represent concatenations.

The gradients from the RL losses of equation 3 also impact the attention network. Specifically, the attention network will be encouraged to route particular training examples to the experts which perform best on the given terrain type. Similarly, the experts to whom the examples are routed will be further refined to operate more effectively on that class of example. Over the course of training, these two interactions lead to an emergent unsupervised clustering behaviour. Terrain types are naturally grouped into subsets that can effectively exploit the same walking policy, and the experts are tuned to specialise on these subsets.

During the earlier stages of training, an additional loss is introduced to simplify the training of the attention network. This helps to initialise the attention network and expert policies in a mutually effective arrangement. Here, the estimated blending weights are also passed to a cross entropy loss function, along with the "ground truth" from the oracle function $\mathcal{O}$ above

$$L(s_t) = -\sum_{i=1}^{n} g_i \log\left(\omega_i\left(s_t\right)\right), \quad (6)$$

where g_i is the ground truth label and ω_i is the output of the attention network.

B. Training

The ensemble of expert policies and attention network in BeeTLe are trained via a 3 step curriculum learning procedure. In practice, we found that training all components simultaneously from a random initialisation was ineffective, due to the heavy interdependence of the modules explained above.

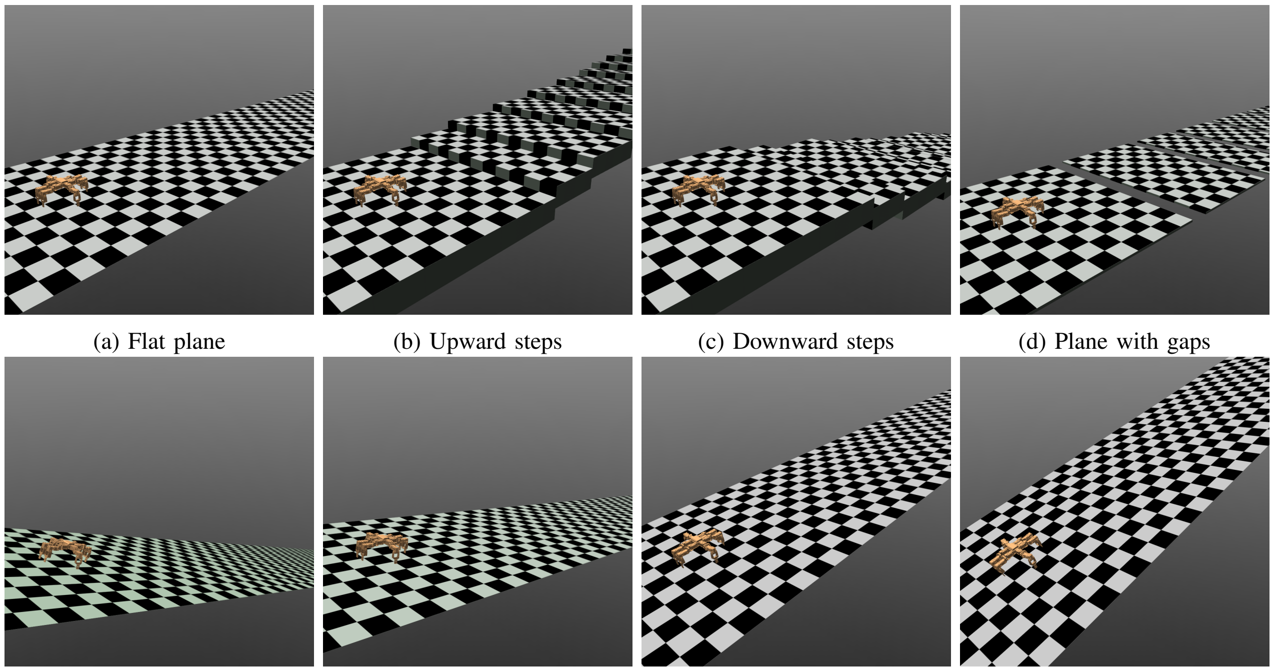
The curriculum learning procedure consists of first training the individual experts, via RL, on different pre-selected surface types using the oracle function, with the RNN frozen. Next, with the experts frozen, the RNN attention network is trained, via supervised learning, to identify those pre-selected surface types and produce an attention matrix that maps the correct expert to the output, roughly mirroring the oracle function. Finally, the cross-entropy loss is disabled and all components are jointly refined solely using the RL losses. This enables the grouping of states to be adapted away from the oracle function, enabling BeeTLe to discover more effective subsets of terrain type.

C. Simulation Environment

The simulator utilised during training is MuJoCo [18], using a custom environment modelled after a real hexapod robot, which can be seen in Figure 3. The hexapod consists of 18 actuators, which are position servos. The dynamics of the custom environment were set such that the simulated model behaved similarly to the real robot, given the same actions. The environment follows the Open AI Gym API [19], making it easy to integrate into other RL algorithms. It takes an action a_t and returns an observation s_t, as well as a reward r_t. The action is the set of 18 goal angular positions to be executed by the actuators. The observation is made up of the current angular positions of each of the actuators, followed by their angular velocities. The reward for the environment is defined as follows

$$r_t = \dot{y}_t - c_{ctrl}\sum(a_t - a_{t-1})^2, \quad (7)$$

where $\dot{y}_t$ is the y velocity at time t and c_{ctrl} is the control cost weight. The actions a_t and a_{t-1} are the joint positions at time t and $t-1$ respectively. In this equation, the $\dot{y}_t$ term encourages movement in the forward direction, while $c_{ctrl}\sum(a_t - a_{t-1})^2$ discourages large changes in positions between timesteps.


(a) Flat plane (b) Upward steps (c) Downward steps (d) Plane with gaps

(e) -20° tilted plane (f) -10° tilted plane (g) 10° tilted plane (h) 20° tilted plane

Fig. 3: MuJoCo training environments

IV. RESULTS

A. Experimental Setup

BeeTLe was implemented as a new PPO RL algorithm within the stablebaselines3 [20] library. The policy class has n sets of actors and critics, and an RNN attention network, which produces the attention weights by which to blend the values produced by each of these actor and critic networks. The robot model was created by modelling the real hexapod robot in a MuJoCo xml file. This xml file also defines the terrain. A new Gym environment class was written to manage the hexapod locomotion challenge.

BeeTLe was instantiated, with randomly initialised weights for each of the networks, and saved into a state dictionary. Next, 8 individual policies were trained on different terrains. These terrains include a flat plane, upward steps, downward steps, a flat plane with gaps and 4 tilted planes, which can all be seen in Figure 3. The distance between steps and gaps is randomised. The tilted planes were set to tilts of -20, -10, 10 and 20 degrees.

These policies formed the pre-trained experts for stage 1 of curriculum learning using the losses in equations 2 and 3, and the oracle function to produce ω. A dataset was then created from this partially pre-trained multi-expert policy. To create the dataset, every policy and terrain type combination was run for 200 timesteps, and the observations were recorded and associated with a label corresponding to the terrain type. As there are 8 different terrain types and 8 different experts, a total of 64 combinations were recorded. Each combination was repeated 1000 times for the training dataset, and 100 times for the test dataset. These datasets were saved into an FFCV [21] database.

These train and test datasets were then used to pre-train the attention RNN in a supervised manner for stage 2 of curriculum learning. The cross-entropy loss function of equation 6 was used to train this. As there are 64 types of sequences and 8 possible labels, an untrained model would be expected to have an average accuracy of 12.5%, when attempting to associate a sequence of observations to a terrain type. Our RNN achieved an accuracy of 94% after training. A learning rate scheduler was used to refine the model when the training loss would reach a plateau.

Finally, all of the networks within BeeTLe were trained at once, which is the last step of the curriculum learning procedure. In this step, the RL algorithm's optimiser has access to all of the network parameters within BeeTLe, including those of all the experts and those of the attention network. Thus the optimiser has the opportunity to redistribute knowledge in a more effective way.

B. Simulation Comparison

The performance of BeeTLe after training was compared against 4 baselines. The first of these was the manufacturer's default handcrafted gait. This is a set of 4 poses that the robot repeatedly cycles through at predefined time intervals. The second baseline was a simple neural network trained via imitation learning to imitate the behaviour of the manufacturer's gait. The third and fourth were a trained PPO [17] and a trained Soft Actor-Critic (SAC) [22] policy respectively.

To create the imitation learning baseline, a dataset of observations, and the corresponding actions given by the manufacturer's gait, was recorded and then used to train a neural network via supervised learning. The end result is a neural network which imitates the behaviour of the manufacturer's gait.

Testing was performed in various different scenarios, including on complex varying terrains that were not seen during training. Specifically, the testing modes were:

- Flat plane - A smooth horizontal plane (Figure 3a)
- Tilted plane - A plane tilted at 10 degrees (Figure 3g)
- Sequence of planes - A sequence of planes consisting of an uphill plane, followed by a flat plane, a downhill plane and then by another uphill plane (Figure 4a)
- Natural hilly terrain - A geometry modelling a natural hilly terrain, implemented via a height map (Figure 4b).

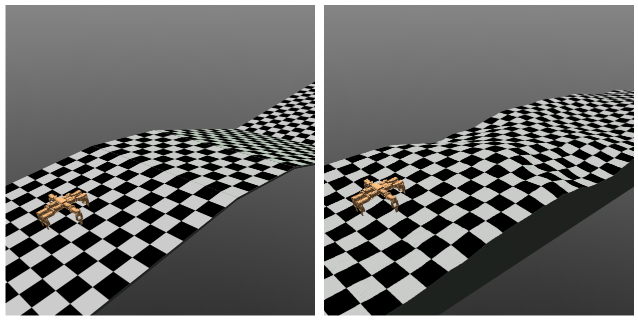

(a) Sequence of tilted planes (b) Natural terrain

Fig. 4: MuJoCo testing environments

Two key metrics were identified which well represent the performance of the different policies. These are:

- Distance - The distance from the origin after 1000 timesteps, where 1 timestep is 0.05 seconds
- Control cost per unit distance - The total control costs over 1000 timesteps divided by the distance covered.

Covering a large distance while exerting little energy, which we define as control cost, usually indicates an effective and efficient gait. A timestep of 0.05 seconds was chosen as this is approximately the amount of time spent communicating with the actuators on the real robot during each iteration of the control loop. In the simulator, this is broken down into 5 discrete steps of 0.01 seconds. This means that the simulator internally takes 5 steps with the same action before returning a new state observation.

The distance, in meters, was obtained by taking the y position of the hexapod's torso after 1000 timesteps. The control cost was defined using the angular positions slice of the state (i.e. the first 18 values), via the following equation

$$Cost_{ctrl} = \sum_{t=0}^{T} \sum_{i=1}^{18} (s_t - s_{t-1})^2, \tag{8}$$

where T is the number of timesteps (1000) and $s_t - s_{t-1}$ is an angular displacement in radians. The experiments were repeated 100 times and the average of each metric is presented in Table I and Table II respectively. It can be seen from Tables I and II that the performance of BeeTLe is better than that of the baselines for all of the more complex terrains, these being the tilted plane, the sequence of tilted planes and the natural hilly terrain. In each instance, greater progress was made in the forward direction while using less energy per unit distance than the baselines. Some of the baselines would result in the robot walking or sliding in the incorrect direction, down a slope, leading to negative distance scores.

TABLE I: Simulation test results - Distance (m)

Policy	Terrain			
	Flat Plane	Tilted Plane	Sequence	Natural Hilly
Mfr. gait	3.85	2.51	3.89	-2.69
Imitation	3.76	2.23	3.67	-2.43
PPO [17]	346.64	88.65	20.62	4.74
SAC [22]	**425.30**	137.33	34.20	9.74
BeeTLe (Ours)	339.49	**276.89**	**78.25**	**28.10**

TABLE II: Simulation test results - Ctrl cost / unit distance

Policy	Terrain			
	Flat Plane	Tilted Plane	Sequence	Natural Hilly
Mfr. gait	5.18	8.12	6.49	23.48
Imitation	4.97	8.56	6.73	25.61
PPO [17]	6.67	9.01	8.01	26.30
SAC [22]	**4.85**	8.82	7.28	20.73
BeeTLe (Ours)	5.00	**4.13**	**5.82**	**12.16**

The imitation learning baseline obtained very similar results to the manufacturer's gait on all of the terrains. This shows that the imitation learning procedure was successful in training a neural network to imitate the output of the manufacturer's gait.

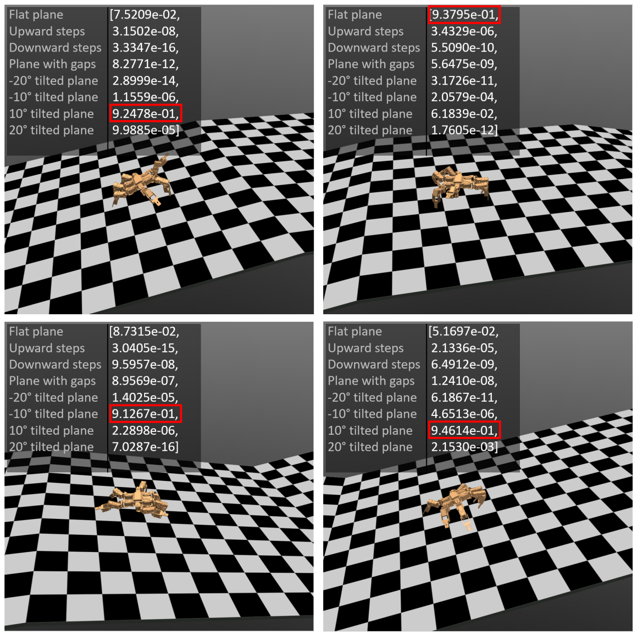
The distance performance of BeeTLe on the flat plane is lower than that of the RL baselines. This implies that there is still a trade off between the generality of the locomotion policy, and its effectiveness. The SAC baseline resulted in the greatest distance covered on the flat plane test. It can also be seen that the SAC baseline covered a greater distance then the PPO baseline in all of the environments, while also using less energy per unit distance, thus demonstrating that SAC is superior to PPO by all testing metrics.

During the natural hilly terrain test, BeeTLe approaches terrain which it has not seen during training, but it is still able to successfully traverse it. This is significant as it shows that BeeTLe is able to use expertise from its bank of experts to derive an effective new gait for the unseen terrain.

The performance of the attention network on live data was also separately verified by testing a BeeTLe checkpoint, which was saved after stage 2 of curriculum learning, on the sequence of tilted planes test. By inspection of the attention matrix throughout the run, it was able to successfully identify the different terrains on-the-fly, while traversing the different portions of the sequence. This is shown in Figure 5. After stage 3 of curriculum learning, the direct expertise of each expert moves away from how they were initialised, and thus it is no longer possible to directly interpret the output of the attention network.

C. Real World Validation

To validate that BeeTLe also works in the real world, our translation system was used to test the policy on a real hexapod robot. The policy was run both in simulation and then on the real robot, and the behaviour was compared. It can be seen from Figure 6 that the real robot is able to behave similarly to how its model does in simulation, despite



```
Flat plane         [7.5209e-02,
Upward steps        3.1502e-08,
Downward steps      3.3347e-16,
Plane with gaps     8.2771e-12,
-20° tilted plane   2.8999e-14,
-10° tilted plane   1.1559e-06,
10° tilted plane    9.2478e-01,
20° tilted plane    9.9885e-05]
```

```
Flat plane         [9.3795e-01,
Upward steps        3.4329e-06,
Downward steps      5.5090e-10,
Plane with gaps     5.6475e-09,
-20° tilted plane   3.1726e-11,
-10° tilted plane   2.0579e-04,
10° tilted plane    6.1839e-02,
20° tilted plane    1.7605e-12]
```

```
Flat plane         [8.7315e-02,
Upward steps        3.0405e-15,
Downward steps      9.5957e-08,
Plane with gaps     8.9569e-07,
-20° tilted plane   1.4025e-05,
-10° tilted plane   9.1267e-01,
10° tilted plane    2.2898e-06,
20° tilted plane    7.0287e-16]
```

```
Flat plane         [5.1697e-02,
Upward steps        2.1336e-05,
Downward steps      6.4912e-09,
Plane with gaps     1.2410e-08,
-20° tilted plane   6.1867e-11,
-10° tilted plane   4.6513e-06,
10° tilted plane    9.4614e-01,
20° tilted plane    2.1530e-03]
```

Fig. 5: Attention matrix at 4 points in time on the sequence of tilted planes test

never being fine-tuned on real-life data. Next, the policy was tested on the real robot for a variety of other terrain types including: on a laboratory floor, tilted board, lumpy carpet and grass. These tests can be seen in the supplementary video and in Figure 1. It can be seen from the video that BeeTLe was able to effectively traverse the various terrains that it encountered. This is good evidence that the policies learned in simulation generalise, and that the translation system is effective at running these learned policies on real hardware.

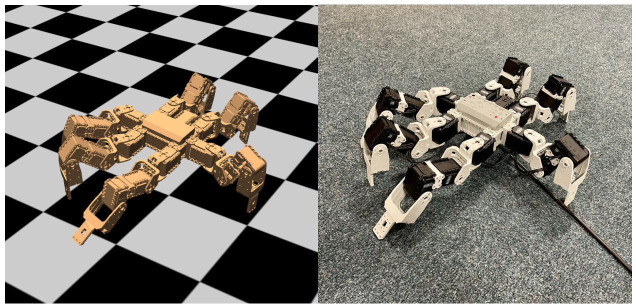


Fig. 6: Hexapod robot walking in simulation (left) and in the real world (right)

V. CONCLUSIONS

The work undertaken has led to BeeTLe, which has clear benefits over the baselines on more complex terrains. This makes it more suitable for deployment in the real world, where terrains are often not simple. Each individual section of the framework was built and verified separately, before being merged into the multi-expert framework. This includes the attention network, which was shown to have an accuracy of 94% on unseen data after pre-training, and the individual experts, which were each shown to have effective gaits for the environment they were set to specialise in. The verification of the individual portions of the network helped to ensure good performance from the full framework, and show that none of the portions are redundant.

REFERENCES

[1] 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," *Robotics: Science and Systems*, 2018.

[2] P. Böhm, P. Pounds, and A. C. Chapman, "Feature extraction for effective and efficient deep reinforcement learning on real robotic platforms," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[3] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, 2020.

[4] H. Lai, W. Zhang, X. He, C. Yu, Z. Tian, Y. Yu, and J. Wang, "Sim-to-real transfer for quadrupedal locomotion via terrain transformer," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[5] G. Christmann, Y.-S. Luo, J. H. Soeseno, and W.-C. Chen, "Expanding versatility of agile locomotion through policy transitions using latent state representation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[6] W. Ubellacker and A. D. Ames, "Robust locomotion on legged robots through planning on motion primitive graphs," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[7] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, 2020.

[8] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Transactions on Robotics*, 2022.

[9] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, 2022.

[10] F. Acero, K. Yuan, and Z. Li, "Learning perceptual locomotion on uneven terrains using sparse visual observations," *IEEE Robotics and Automation Letters*, 2022.

[11] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *Robotics: Science and Systems*, 2021.

[12] I. M. Aswin Nahrendra, B. Yu, and H. Myung, "Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[13] X. Zhang, Z. Xiao, Q. Zhang, and W. Pan, "Synloco: Synthesizing central pattern generator and reinforcement learning for quadruped locomotion," 2023. arXiv:2310.06606 [cs.RO].

[14] G. Bellegarda and A. Ijspeert, "Cpg-rl: Learning central pattern generators for quadruped locomotion," *IEEE Robotics and Automation Letters*, 2022.

[15] L. Smith, J. C. Kew, X. Bin Peng, S. Ha, J. Tan, and S. Levine, "Legged robots that keep on learning: Fine-tuning locomotion policies in the real world," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022.

[16] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *6th Conference on Robot Learning (CoRL)*, 2023.

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

[18] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[19] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016. arXiv:1606.01540 [cs.LG].

[20] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," in *Journal of Machine Learning Research 22*, 2021.

[21] G. Leclerc, A. Ilyas, L. Engstrom, S. M. Park, H. Salman, and A. Madry, "FFCV: Accelerating training by removing data bottlenecks," in *Computer Vision and Pattern Recognition (CVPR)*, 2023.

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *35th International Conference on Machine Learning (ICML)*, 2018.