

Decentralized Deep Reinforcement Learning for a Distributed and Adaptive Locomotion Controller of a Hexapod Robot

Malte Schilling¹, Kai Konen¹, Frank W. Ohl², and Timo Korthals³

Abstract—Locomotion is a prime example for adaptive behavior in animals and biological control principles have inspired control architectures for legged robots. While machine learning has been successfully applied to many tasks in recent years, Deep Reinforcement Learning approaches still appear to struggle when applied to real world robots in continuous control tasks and in particular do not appear as robust solutions that can handle uncertainties well. Therefore, there is a new interest in incorporating biological principles into such learning architectures. While inducing a hierarchical organization as found in motor control has shown already some success, we here propose a decentralized organization as found in insect motor control for coordination of different legs. A decentralized and distributed architecture is introduced on a simulated hexapod robot and the details of the controller are learned through Deep Reinforcement Learning. We first show that such a concurrent local structure is able to learn better walking behavior. Secondly, that the simpler organization is learned faster compared to holistic approaches.

I. INTRODUCTION

Over the last years, Deep Reinforcement Learning (DRL) has been established as an exploratory learning approach (Fig. 1 a) that produces effective control solutions in diverse tasks [1]. Currently, the mainstream of DRL research is addressing how to improve Deep Reinforcement Learning and in particular how to make learning more (sample) efficient as well as stabilize training [2]. The initial success of DRL in playing computer games [3] has shaped the field and even though there have been extensions to continuous control domains and applications in many areas [1], the field is still widely dominated by approaches that deal with simulation environments. In the domain of robotics, transfer to real-world problems has proven to be difficult [4] as it appears that the nature of such problems is fundamentally different from those in playing computer games. In most cases, simulation is first used as a tool to produce viable controllers that are only later fine-tuned, for example, on a real robot for a specific task.

It has been argued [5], [6] that some of the problems encountered in applications of DRL to real-world scenarios might be indicative of some in fact rather fundamental aspects of DRL approaches. In particular, two connected problems were identified: First, DRL is geared towards a

¹Malte Schilling and Kai Konen are with the Neuroinformatics Group, Bielefeld University, 33501 Bielefeld, Germany
mschilli@techfak.uni-bielefeld.de

²Frank W. Ohl is with the Department of Systems Physiology of Learning, Leibniz Institute for Neurobiology and with the Institute of Biology, Otto-von-Guericke University, Magdeburg, Germany.

³Timo Korthals is with the Kognitronik and Sensorik Group, Bielefeld University, 33501 Bielefeld, Germany.

reward and it is the explicit goal to exploit the reward structure which leads to overfitting [7]. Secondly, real world application face much more noise compared to simulations which questions the underlying assumption of a stationary Markov Decision Process [8] (or this becomes non-stationary as the environment or other agents are changing themselves). At its core Deep Reinforcement Learning has proven to be able to find viable solutions. But as a disadvantage it has shown that DRL is trying to exploit a reward structure as good as possible and doesn't take into account robustness of the behavior. Therefore, DRL tends to find narrow solutions that can be optimal only for a small and quite specific niche, instead of providing adaptive behavior in the sense that a control approach should be rewarded that is widely applicable in variations of a task [9].

As these problems have hindered application of DRL approaches in robotics when dealing with unpredictable tasks, there is now a growing interest in motor control principles in biological systems and the exploitation of these principles to improve robust adaptive behaviors in robots. For example, Merel et al. [6] propose several such bio-inspired principles and advocate their implementation also into robot architectures. As one example they emphasize the hierarchical structure of motor control (Fig. 1 b). Such hierarchies lend themselves to suitable factorization of all available information to different subsystems and to a coordinated implementation of different levels of abstraction or different timescales on which subsystems operate. It should be noted that while such approaches enfold their beneficial effects by allowing a flexible reuse and transfer of trained competences between distinct subtasks [10] and hence improve flexibility between different (but still highly defined) contexts [11], [12], they still do not provide the robustness in any given context that is typically found in animals (see Fig. 1 a) for a visualization of hierarchical DRL).

Therefore, we here focus on another aspect of bio-inspired mechanisms already mentioned in [6]: while higher-levels modulate the lower-level systems, these are still *partially autonomous* which means that these lower-level systems can act autonomously — for example, realizing a fast reflex pathway (see Fig. 1 b) for a schematic view on biological motor control). This has been further pointed out by Sejnowski who explicitly highlighted distributed control as part of a multi-level hierarchy [18]. Such a local and decentralized control structure is in agreement with what is known on the organization of motor control in animals in general [15] and in particular for locomotion in insects [16] (Fig. 1 b) and c). In this paper, we want to introduce such a decentralized

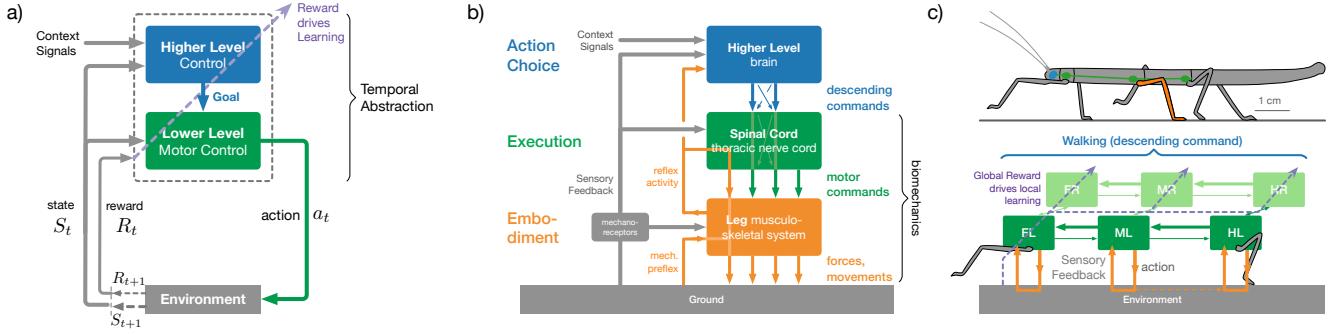


Fig. 1: Visualization of influences for the biological inspired approach: On the left (a) the standard view of interaction with the environment in reinforcement learning [13] is extended to a hierarchical perspective [10] as advocated, for example in [6]. For higher level control (shown in blue) this is in agreement with what we know on the structure of motor control in mammals [14] about descending pathways and modulation of lower level control centers (shown in green) in the spinal cord (see schematic b) in blue and green). Such structures are shared not only in mammals, but also in invertebrates and insects [15], see c). Work in such simpler model systems allows more detailed analysis of interaction with the environment which has stressed the importance of very fast and local reflex activity controlled directly on the lowest level or that are even realized by passive properties as muscle elasticities or preflexes (shown in orange). One important characteristic emphasized by this work is the emergence of behavior as a result of decentralized and locally interacting concurrent control structures (bottom part of c), one example is given by the decentralized control structure found in stick insects [16], but this concurrency is as well assumed in primates [17]. In the presented approach, this decentralized architecture (bottom part of c) is used on a hexapod and learning of the six local control modules (realized as deep neural networks) is driven by a reward signal as in reinforcement learning.

control principle into a control architecture for six-legged walking and want to analyze how learning in general and DRL — as a contribution of this publication — benefits from this structure and how in the end control performance is affected by such a structure. Therefore, we train and compare a decentralized control architecture and compare this to learning of a baseline centralized control approach. As a result, the decentralized controller learns faster and, importantly, is not only able to produce comparable walking behavior, but even results in significantly better performance and is able to generalize as well towards novel environments. Decentralization appears as a viable principle for DRL and an important aspect for adaptive behavior [6] as demonstrated here in our simulation on a six-legged robot.

II. RELATED WORK

As DRL still struggles in high-dimensional problems and when dealing with noisy as well as uncertain environments, most of such work has been applied in simulated environments. One recent example of DRL for locomotion on a real robot was provided by Hwangbo et al. [4]. In their approach, a policy network for a quadruped robot was initially trained in simulation. In a second step, the pretrained control network was transferred to the real robot system and further adapted on that robot. They pointed out as well a tendency of their system to overfit [4] which is a common problem in DRL [7]. As a solution to alleviate this problem they propose a hierarchical structure [6].

In simulation, different approaches to locomotion already applied hierarchical structure [11], [12], [19]. In such cases,

two different levels of control were distinguished and operated on different time scales which realizes a form of temporal abstraction (Fig. 1 a). Such hierarchical approaches allowed to flexibly switch between distinct subtasks and behaviors which allows to deal with a variety of distinct contexts [11], [12]. For example, hierarchical DRL showed well suited for adjusting to severe interventions as the loss of a leg or when switching between obstacle avoidance, following a wall or straight walking [19]. This was successful for solving problems that can be solved through sequences of basic, individually learned actions, as for example a navigational task in which control should be learned from basic movement primitives. Transfer is in such cases realized as a reuse of basic motor primitives and the hierarchical organization is understood in the sense of switching between different motor primitives.

But this still does not show adaptivity within a certain behavior as found in animals which can handle broad variations within a specific context, as in climbing through a twig that only provides sparse footholds and moves unpredictably [16]. Furthermore, these simulated hierarchical approaches only dealt with a small number of degrees of freedom. One exemption is given by an evolutionary approach by Cully et al. who proposed a similar hierarchical structure and explicitly introduced an intermediate behavioral space as a lower dimensional representation for behavioral switching [20]. But again, this approach of switching between different behaviors depending on the current context was tested when loosing a single leg. While this poses a difficult problem and is a severe intervention, it is still a singular event that in the

end required a drastic change or switch of behavior [9]. In contrast, we want to focus on adaptivity as the way how animals deal with perturbations and variability in one given context and when using one behavior.

Here, we take inspiration from biology and in particular research on insect locomotion. This work questions the wide held assumption that for walking there are distinct different behaviors or gaits [21], [22]. Instead, there appears to be a continuum of gaits that emerge from the interaction with the environment. Such emergent behavior allows to constantly adapt to unpredictable environments and adjust the temporal coordination as required. A decentralized control structure appears to be crucial and beneficial for this kind of adaptive locomotion (Fig. 1 c) gives a schematic overview of the walking controller; for details see [16]): There is one local controller for each of the six legs and the overall behavior emerges from the interaction of these six control modules. Importantly, these control modules are interacting: on the one hand, part of these interaction is directly mediated through the body as actions of a leg affect the movement of the whole animal which can be locally sensed by the other legs [23]. On the other hand, there are local coordination rules between neighboring legs through which neighboring controllers can influence the behavior of each other [16]. As such a decentralized organizational structure of motor control in insects is well described [22], [24], we will use this basic organization for our control architecture that consists of six local modules, one for each leg. Such a control structure has already been realized successfully on a six-legged robot and showed adaptivity in particular on a short timescale and when dealing with perturbations in one given context. Robust behavior emerged in such approaches that dealt well with unpredictability and small disturbances [16]. But as these controllers were handcrafted and took detailed inspiration from biology, it appears difficult to scale these towards more difficult problems as is walking through uneven terrain or climbing. Therefore, we will in this article start from such a decentralized organization in a learning based approach. This will allow to scale up to more difficult tasks and later-on to transfer to a real robot.

III. METHODS

This study analyzes learning in architectures for a neural network based control system of a six-legged robot. First, the robot platform and simulation will be described. Second, the employed DRL framework will be referenced. Third, as the main contribution of this article, the decentralized control architecture and a baseline structure will be introduced.

A. PhantomX Robot Simulation

The PhantomX MK-II is the second generation of a six-legged robot distributed by Trossen Robotics [25]. It consists of six legs each actuated by three joints. The main body consists of a rectangular plexiglas base (260 mm by 200 mm) and for each leg, the first joint is attached to the main body in a sandwich type construction (mounted between two such plexiglas sheets). Therefore, the first joints moves the

leg forward and backward, while the other two joints lift respectively extend the leg. For our learning approach, joint limits were restricted to avoid limb- and leg-collisions as well as unpractical leg postures, e.g. postures in which the tibia would touch the top of the torso.

The basic structure is loosely inspired from insects and has comparable degrees of freedom. All joints are common revolute joint servo motors (Dynamixel AX-12A). While the original PhantomX robot can be controlled using a preinstalled control system on a Arduino compatible microcontroller, it has been adapted to the Robotic Operating System (ROS) [26].

In this study, we use as a first step only a simulated robot¹ in the ROS simulation environment *Gazebo* [27] as it is open source and because of the good support of ROS (here ROS 2, version Dashing Diademata [28]). As a physics-engine *Bullet* was used [29]. The simulation required several custom modules (e.g., publishing joint-effort controllers in ROS 2) that have been implemented now for ROS 2². One current difference between simulated robot and the real robot was the implementation of ground contact sensors using data from the simulation environment, but solutions for such sensors have been proposed for the robot as well [30]. Simulator and ROS were run with a fixed publishing-rate of 25 Hz in order to align sensor and actuator data. This guaranteed a stable and constant step-size.

B. Deep Reinforcement Learning Framework

Reinforcement Learning is characterized by an agent that is interacting in an environment (Fig. 1 a). While the agent produces actions, it gets as a response an updated state of the environment and a reward signal. The goal of the agent is to learn a policy $\pi(S)$ for sequential decision making that maximizes the accumulated long-term return. During learning the agent has to explore possible alternatives and while getting more confident should come up with a policy it can exploit (for an introduction see [1], [13]). Sequential decision making can be formalized as a Markov Decision Process which provides and is defined as a tuple of:

- observable states S ,
- possible actions A ,
- a reward signal R providing the immediate reward after choosing an action from the current state,
- transition probabilities P that describe the probability distribution over states after following an action when being in the current specific state, and
- a discount factor γ that describes how to decrease the weights of future rewards.

We are dealing with a continuous control task: the state space as given by the sensory signals is continuous as well as the action space which corresponds to the joint motor signals. Overall, these spaces are high dimensional and continuous which requires to rely on function approximation for learning

¹robot definitions: <https://github.com/kkonen/PhantomX>

²for details see the open repository <https://github.com/kkonen/ros2learn>

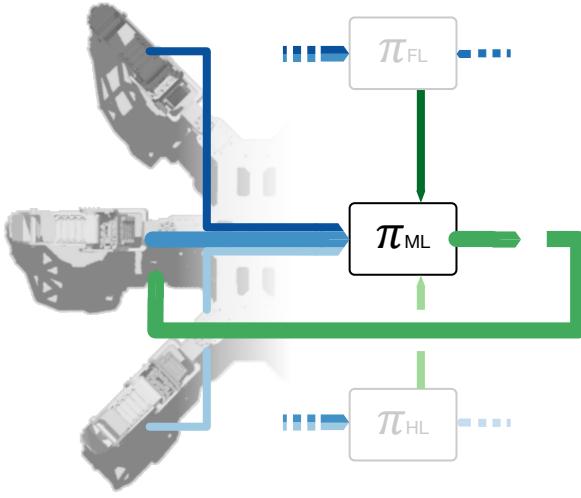


Fig. 2: Overview of the decentralized control architecture, shown for the middle left leg (ML): inputs to the controller are sensory information (blue) from the leg and its' two neighbors and last actions of the neighboring controllers to provide context. Control output (joint activations for only the respective leg) are shown in green. For each leg there is an individual controller in the decentralized structure.

a probabilistic policy. Deep neural networks are used in Deep Reinforcement Learning as function approximators that map observations to actions [1].

As a framework for DRL we employed OpenAI's gym. The connection to the simulator through ROS was wrapped as a new environment through a connector using *ROS2Learn* [31]. This allowed to use the standard baseline implementations of DRL algorithms [32]. In this approach Proximal Policy Optimization (PPO) [33] was used as it has shown to work well on continuous tasks without the need of intensive hyperparameter tuning. As a further advantage, it has a comparably low sampling complexity (amount of training samples required to find usable policies).

The reward function was defined as the mean velocity over an episode (1024 iteration steps, equals nearly 41 seconds). The agent uses the received reward R_{t+1} after executing an action A_t in state S_t in order to directly optimize its' policy π . A neural network was used as a function approximator for the learned policy. The policy networks consisted of two hidden layers of each 64 units with *tanh* activation functions. Updating the neural networks' weights is called an epoch. After training, the policy network should generalize to novel sensed states and modulate motor output in order to achieve a maximal external reward. The next section will describe in detail the two compared architectures that employ such policy networks.

C. Decentralized Control Architecture

The main contribution of this paper is the decentralized control architecture. Importantly, the presented approach is not hierarchically structured using a higher level and switch-

ing between different behaviors (Fig. 1 shown in a) in blue), but realizes the decentralized structure on the lower control level (Fig. 1 c) shown in green). This decentralized structure is embodied in a six-legged robot. There is one local control module for each leg. Each controller consists of a single policy neural network (two hidden layers with 64 hidden units in each of these layers). Input to each of the leg controllers is only local information stemming from that particular leg and the two neighboring legs (Fig. 1 c) shows these influences as green arrows). Furthermore, there is interaction with the environment that can mediate information between legs (Fig. 1 c) orange arrows). The observation space of one controller (Fig. 2 shows as an example the middle left leg controller and its input as well as output space) is 42 dimensional as it receives information from the controlled leg (positions of the three leg segments and ground contact). In addition, a local controller gets information from the two neighboring legs (including positions of leg segments, ground contact, and last action signal from the neighboring legs). Last, local controllers get information on the orientation of the body as a six-dimensional vector [34].

During Reinforcement Learning the six controllers learn in parallel which is comparable to multi-agent learning. But in our decentralized architecture, the individual controllers are part of the same robot and can have access to information from the other control agents. In contrast to a fully centralized approach, this information is restricted in our approach and only neighboring legs provide information.

The decentralized architecture is compared to a centralized (single-agent) DRL approach: for this baseline, a single policy network is used (Fig. 3). All available information is used as an input (overall 84 dimensions) and the task for the policy network is to learn optimal actions for all 18 joints.

As the centralized approach has access to all information, the input space and space of possible decisions is high dimensional. It appears more difficult to explore and find an optimal policy. In contrast, for the decentralized

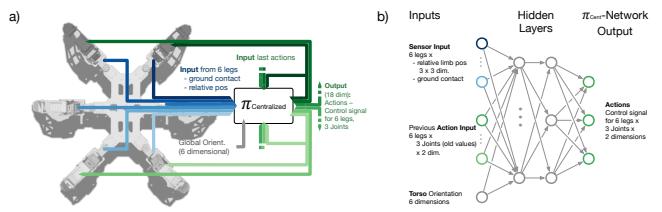


Fig. 3: a) Overview of the centralized architecture that is trained as a single-agent network. All sensory information from the legs (blue) are fed into the policy network which produces joint actions for all 18 joints (shown in green). b) View of centralized policy network: a neural network with two hidden layers (64 hidden units in each hidden layer). Sensory input is given as information from leg and output are joint actions (for the decentralized approach: six such networks are used which each produce joint activations for a single leg and only receives partial sensory inputs).

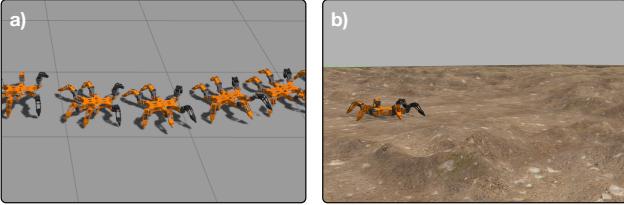


Fig. 4: Visualization of the simulated PhantomX robot. a) shows a walking sequence on flat terrain. b) shows the uneven terrain condition in comparison to the robot.

approach the input and output space of each of the six local control modules is lower dimensional. While this simplifies exploration, it is not clear if all necessary information is available to the local controllers. Our experiments address, first, if the information provided to the local control modules is sufficient to learn competent walking behavior. And, secondly, how the lower dimensional input and control space affect learning and if there is faster learning observable.

IV. RESULTS

We hypothesize, first, that a decentralized architecture consisting of six local leg controllers can produce stable locomotion behavior. Secondly, that a local structure simplifies learning. As a last aspect, we want to analyze how such a control structure can generalize and copes with novel environments. In the following, the learning experiments will be shown and analyzed that compare a centralized structure with the proposed decentralized structure.³

A. Performance of Locomotion Architectures on Flat Terrain

As a first experiment, we studied learning to walk on a flat terrain and compared as the two conditions the two different architectures. The controllers were trained through DRL from scratch. Each of the two architectures was trained 15 times using different random seeds for 5000 epochs.

Both learning approaches were able to learn walking behavior (see Fig. 4 a) and supplemental video) with quite a high velocity that looked well coordinated. The trained controllers from the last epoch were afterwards evaluated. For each controller for over 100 individual episodes the performance (mean velocity over the episode) was collected. Individual results from the different seeds are given in the supplement (see data repository). For the centralized baseline approach the mean performance was 546.70 (standard deviation 131.23). The proposed decentralized architecture consisting of six local control modules reached a performance of 657.11 (std.dev. 68.07). First, we wanted to test that the decentralized architecture does not perform worse. Therefore, we performed a two-tailed Welch t-test (following [35] with the null hypothesis that the two approaches perform on a similar level). The null hypothesis could be rejected

³Simulation environment is made available as a docker. Instructions and all data from the experiments can be found at: https://github.com/malteschilling/ddrl_hexapod

(*p*-value of .011) and we can conclude that the decentralized approach performed significantly better compared to the centralized approach. The mean rewards were actually substantially higher for the decentralized approach with a relative effect size of 1.02 which indicates a large effect size.

The results showed a large variance which is typical for DRL-based solutions. In many cases, the focus is therefore on the best solutions produced by running multiple seeds during learning. Therefore, we compared as well the best performing controllers. Overall, the standard deviation in both conditions were in a similar range, but the distribution of the centralized approaches appeared larger. In table I the ten best ranked controllers are shown. As can be seen from these results, the best performing approach was a centralized approach, but there seems to be an upper limit for maximum mean velocity and multiple approaches converged towards this performance. Overall, we found more decentralized architectures in the best performing approaches which was confirmed by a rank-sum test showing that the decentralized architectures performed significantly better (WilcoxonMann-Whitney test, *p* = .041).

TABLE I: Detailed results after learning for 5000 epochs.

The mean rewards were evaluated for each of the fifteen learned controllers for the two different architectures in 100 simulation runs. Note that standard deviation is taken for the single controller over the 100 repetitions which showed to be low.

Rank	Architecture	Seed	Avg. Reward	Std. Dev.
1.	Central/Baseline	2	770.158	21.52
2.	Decentralized	1	764.209	26.01
3.	Decentralized	5	756.096	27.50
4.	Decentralized	7	733.194	56.82
5.	Decentralized	12	721.685	37.84
6.	Decentralized	2	707.283	33.84
7.	Decentralized	6	695.860	24.94
8.	Decentralized	4	681.352	26.10
9.	Central/Baseline	5	652.852	18.28
10.	Decentralized	14	640.512	30.23

To summarize: As a first result, we found that a decentralized control architecture is able to produce high performance and well coordinated walking behavior. The local controller even showed significantly better performance with a large effect size.

B. Comparison of Learning

The learned architectures were trained for 5000 epochs as initial tests showed that at this point controller performance converged for both cases (running simulations for up to 12000 epochs only showed minor further improvements which appear to confirm the reasonable idea of a maximum reachable velocity). In this section, we will look at the development of training performance over time (Fig. 5).

Mean performance was calculated over the fifteen seeds for each of the two architectures over training. Looking at individual runs showed a high variance during learning. While

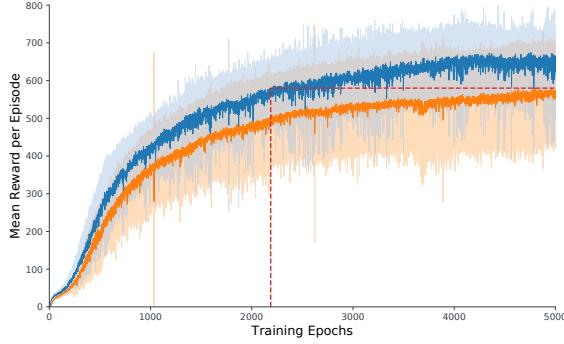


Fig. 5: Comparison of the mean reward during training: mean performance over the fifteen decentralized controllers is shown in blue and mean reward for the baseline centralized approach is shown in orange (shaded areas show standard deviation). Performance is measured as reward per episode. Seeds were only measured up to 5000 epochs points as learning appears to have converged by then. The horizontal red dashed line visualizes the maximum of the centralized approach at the end of training. The vertical red dashed line indicates when this performance level was reached by the decentralized approach (after 2187 epochs already).

the mean calculation smoothed the reward signal, this is still visible. Such jumps in performance are typical for DRL approaches. Learning progressed in several stages (details not shown) from simple crawling towards walking lifted from the ground. Comparing the two different architectures, the difference in performance between these is clearly visible. The decentralized architecture reached a higher reward level earlier and appears to learn much faster. Compared to the mean reward of the centralized approach at the end of training, the decentralized approaches reached such a level already after 2187 epochs.

C. Generalization in Transfer to Uneven Terrain

In order to understand how such a controller deals with unpredictability, we further evaluated the controller for a novel, uneven terrain it hadn't experienced during learning. All controllers trained on the flat terrain were again tested for 100 simulation runs each on uneven terrain. The terrain was generated from height-maps with a maximum height of 0.10 m (Fig. 4 b), further variations can be found in the data repository). Height-maps were generated using the diamond-square algorithm [36]. Detailed results are given in table II and see Fig. 6.

First, there was an expected drop-off in performance for the uneven terrain for both architectures (Fig. 6 shown between left most results and results in the third column that shows distribution of performance on the height map). We further trained fifteen additional controllers for both architectures on the uneven terrain. When comparing results on the uneven terrain, there was no significant difference between controllers that had to generalize (as they were

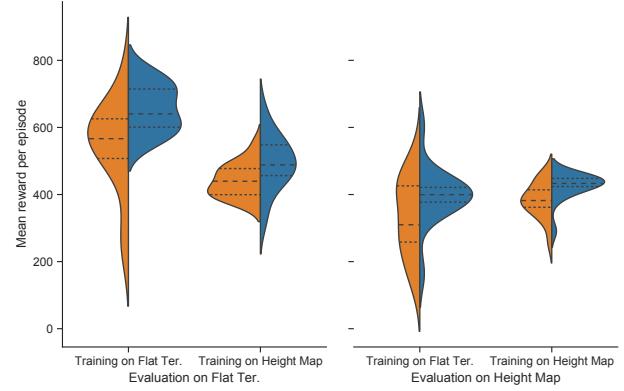


Fig. 6: Comparison of the (approximated) performance distribution for the controllers. Shown are violin plots for two evaluation conditions: left shows the evaluation of walking on flat terrain and right shows walking on uneven terrain (modulated through a height map with maximal height of 0.10 m). Inside the two evaluations, it is further distinguished between controllers trained on the flat terrain (first and third column) and controllers trained on the height map (second and fourth column). The violin plots show approximated distributions (and the quartiles are given inside): orange shows the centralized baseline approach and blue shows the decentralized architecture. Performance for the fifteen different seeds for each condition was measured as mean reward per episode (indicates mean velocity over simulation time).

trained on flat terrain) and the specialized controllers that were trained on the same uneven terrain (Fig. 6 two rightmost columns, comparison of the two different training conditions evaluated on the uneven terrain). Again, we found a significant difference for the expert condition in which the controllers trained on the height map and tested on the height map were compared: in this case, the two-tailed Welch t-test again showed a significant difference ($p = .020$) for mean performance in favor of the decentralized architecture which confirms our earlier results. Inside the generalization condition, difference in performance of the two architectures was reduced to a trend ($p = .102$).

Last, the controllers trained on uneven terrain were evaluated on the flat terrain. Here, the controllers trained on the uneven terrain don't reach the level of the specialist controllers that were trained on the flat terrain and there is a significant performance difference. This was true for decentralized and centralized approaches to the same extent, but again the decentralized architecture performed better. The performance of the controllers was only slightly above their performance on uneven terrain.

To summarize: It appears that decentralized controllers generalize well towards the more difficult task as they perform on the same level as controllers that were explicitly trained on the task. In comparison to the baseline, decentral-

TABLE II: Comparison of rewards between the different control architectures during evaluation. Data was collected during evaluation for 100 episodes for each controller. Given are mean rewards (and standard deviation) for each group of controllers (each group consisted of fifteen individually trained controllers). A Welch t-test was performed to compare the two conditions (decentralized and centralized architecture) with the p-value given in the last column.

Condition	Decentralized Arch.		Centralized Arch.		p-value
	Mean Reward	std.dev.	Mean Reward	std.dev.	
Evaluation on flat terrain					
Trained on flat terrain	657.11	68.07	546.70	131.23	.011
Generalization, trained on height map	492.69	79.22	441.50	48.00	.050
Evaluation on uneven terrain					
Generalization, trained on flat terrain	397.91	83.87	334.06	112.94	.102
Trained on height map	424.30	40.09	382.42	51.23	.023

ized controllers were on the same performance level.

V. DISCUSSION AND CONCLUSION

In this article, we presented a biologically-inspired decentralized control architecture for a hexapod robot. The architecture consists of six local control modules that each control a single leg. As each controller only has access to local information, we wanted to analyze if the local information is sufficient to produce robust walking behavior. Therefore, we tested the decentralized architecture on a six-legged robot in a dynamic simulation and analyzed the learning behavior in a DRL setup. There are two main results from our study: First, the decentralized architecture produced well performing and coordinated controllers that as a group even showed significantly better performance compared to the baseline centralized approach.

Second, the learning task for the decentralized approach appears simplified as the dimensionality (input dimensions as well as control dimensions) is drastically reduced. This lead to faster learning of well performing controllers and earlier convergence towards an assumed maximum velocity. The performance of the centralized approach was reached by decentralized architectures in less than half the training time. Furthermore, the distribution of performances appears narrower. In contrast, for the centralized approach it appears as learning got stuck more often in local minima. Decentralization therefore appears as a viable principle that helps facilitate Deep Reinforcement Learning and in continuous control tasks allow to speed up learning as well as avoid local minima.

Considering generalization towards a novel and supposedly more difficult task, both approaches showed similar good performance that was on the level of controllers trained on the novel task. From behavioral research in insects [16], we would have assumed that decentralization would facilitate adaptive behavior and lead to better generalization capabilities [9]. But we couldn't show such an advantage. This might be explained by the selection of the reward function which aimed only for high velocities. In fast walking in animals, it is assumed that the influence of sensory feedback is reduced in faster walking and running, and coordination is more driven by a simpler synchronization signal [37]. As

future work, it is therefore important to use a wider variety of tasks during training. This is further emphasized by the fact that after training on the uneven and supposedly more difficult terrain, the controller didn't perform as good on the flat terrain. In hindsight, this appears reasonable as such controllers have converged in the more demanding terrain to a different maximum velocity that is appropriate for the uneven terrain and takes the disturbances into account. But after learning to walk on such a terrain, the controller is suddenly tasked with literally running without having to consider possible disturbances.

As a next step, we want to extend the current analysis to further strengthen our results. Furthermore, we want to analyze the structure of the learned controllers. Zahedi et al. [38] used a local control approach for a chain of wheeled robots in a learning approach. They provided a geometric interpretation assuming that such local policies form a low-dimensional subfamily of the family of all possible policies. Our results confirm their findings as our experiments support that such a restriction to a lower-dimensional subclass appeared as not too restrictive even for the much more complicated and higher dimensional case of locomotion. Analyzing the different controllers might shed further light on the underlying geometric principles and how decentralization helps to constrain optimization in a meaningful way.

In the future, we want to extend training to more diverse and demanding tasks, similar to curriculum learning [39]. This will allow us to analyze how decentralized approaches generalize: Does a local structure contribute to the robustness and adaptivity of behavior or does a decentralized approach tend to overfit as well? Furthermore, this will include integrating more task demands into the reward function without running the risk of bias due to reward shaping [40]. In particular, as our long term goal is application on the real robot, the reward function has to reflect physical properties as well, for example penalizing high torques.

ACKNOWLEDGMENT

The authors thank Chris J. Dallmann and Holk Cruse for discussions and helpful comments. Furthermore, we want to thank the Computer System Operators of the Faculty of Technology for their technical support.

REFERENCES

- [1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A Brief Survey of Deep Reinforcement Learning,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017. [Online]. Available: <http://arxiv.org/abs/1708.05866>
- [2] O. Nachum, S. Gu, H. Lee, and S. Levine, “Data-Efficient Hierarchical Reinforcement Learning,” *arXiv:1805.08296 [cs, stat]*, May 2018. [Online]. Available: <http://arxiv.org/abs/1805.08296>
- [3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
- [4] 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*, vol. 4, no. 26, p. eaau5872, Jan. 2019. [Online]. Available: <http://robotics.sciencemag.org/lookup/doi/10.1126/scirobotics.aau5872>
- [5] E. O. Neftci and B. B. Averbeck, “Reinforcement learning in artificial and biological systems,” *Nature Machine Intelligence*, p. 1, Mar. 2019. [Online]. Available: <https://www.nature.com/articles/s42256-019-0025-4>
- [6] J. Merel, M. Botvinick, and G. Wayne, “Hierarchical motor control in mammals and machines,” *Nature Communications*, vol. 10, no. 1, pp. 1–12, Dec. 2019, 00000. [Online]. Available: <https://www.nature.com/articles/s41467-019-13239-6>
- [7] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Perolat, D. Silver, and T. Graepel, “A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning,” *arXiv:1711.00832 [cs]*, Nov. 2017. [Online]. Available: <http://arxiv.org/abs/1711.00832>
- [8] K. Kurach, A. Raichuk, P. Stańczyk, M. Zajkac, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, and S. Gelly, “Google Research Football: A Novel Reinforcement Learning Environment,” *arXiv:1907.11180 [cs, stat]*, Jul. 2019. [Online]. Available: <http://arxiv.org/abs/1907.11180>
- [9] M. Schilling, H. Ritter, and F. W. Ohl, “From Crystallized Adaptivity to Fluid Adaptivity in Deep Reinforcement Learning — Insights from Biological Systems on Adaptive Flexibility,” in *IEEE International Conference on Systems, Man, and Cybernetics 2019*, 2019.
- [10] T. Kulkarni, K. Narasimhan, A. Saedi, and J. B. Tenenbaum, “Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation,” in *Advances in Neural Information Processing Systems*, 2016, pp. 3675–3683.
- [11] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman, “Meta Learning Shared Hierarchies,” *arXiv:1710.09767 [cs]*, Oct. 2017. [Online]. Available: <http://arxiv.org/abs/1710.09767>
- [12] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, “DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning,” *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–13, Jul. 2017. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=3072959.3073602>
- [13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: <http://incompleteideas.net/book/the-book-2nd.html>
- [14] S. Arber and R. M. Costa, “Connecting neuronal circuits for movement,” *Science*, vol. 360, no. 6396, pp. 1403–1404, Jun. 2018. [Online]. Available: <http://www.sciencemag.org/lookup/doi/10.1126/science.aat5994>
- [15] M. H. Dickinson, C. T. Farley, R. J. Full, M. a. R. Koehl, R. Kram, and S. Lehman, “How Animals Move: An Integrative View,” *Science*, vol. 288, no. 5463, pp. 100–106, Apr. 2000. [Online]. Available: <http://science.sciencemag.org/content/288/5463/100>
- [16] M. Schilling, T. Hoinville, J. Schmitz, and H. Cruse, “Walknet, a bio-inspired controller for hexapod walking,” *Biol Cybern*, vol. 107, no. 4, pp. 397–419, Aug. 2013.
- [17] M. Graziano, “The Organization of Behavioral Repertoire in Motor Cortex,” *Annu Rev Neurosci*, Mar. 2006.
- [18] T. J. Sejnowski, “The unreasonable effectiveness of deep learning in artificial intelligence,” *Proceedings of the National Academy of Sciences*, 2020. [Online]. Available: <https://www.pnas.org/content/early/2020/01/23/1907373117>
- [19] N. Heess, G. Wayne, Y. Tassa, T. P. Lillicrap, M. A. Riedmiller, and D. Silver, “Learning and Transfer of Modulated Locomotor Controllers,” *CoRR*, vol. abs/1610.05182, 2016.
- [20] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, “Robots that can adapt like animals.” *Nature*, vol. 521, no. 7553, pp. 503–7, May 2015.
- [21] B. D. DeAngelis, J. A. Zavatone-Veth, and D. A. Clark, “The manifold structure of limb coordination in walking Drosophila,” *eLife*, vol. 8, p. e46409, Jun. 2019. [Online]. Available: <https://elifesciences.org/articles/46409>
- [22] S. S. Bidaye, T. Bockemühl, and A. Büschges, “Six-legged walking in insects: how CPGs, peripheral feedback, and descending signals generate coordinated and adaptive motor rhythms,” *Journal of Neurophysiology*, vol. 119, no. 2, pp. 459–475, 2018.
- [23] R. A. Brooks, “Intelligence Without Reason,” in *inproceedings*, J. Mylopoulos and R. Reiter, Eds. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991, pp. 569–595.
- [24] V. Dürr, J. Schmitz, and H. Cruse, “Behaviour-based modelling of hexapod locomotion: Linking biology and technical application,” *Arthropod Structure and Development*, vol. 33, no. 3, pp. 237–250, 2004.
- [25] T. Robotics. (2020) Phantomx ax hexapod mark ii. [Online]. Available: <http://www.trossenrobotics.com/hex-mk2>
- [26] K. Ochs. (2020) hexapod_ros. [Online]. Available: https://github.com/KevinOchs/hexapod_ros
- [27] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sep. 2004, pp. 2149–2154 vol.3. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/1389727>
- [28] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.
- [29] E. Coumans, “Bullet physics simulation,” in *ACM SIGGRAPH 2015 Courses*, ser. SIGGRAPH ’15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: <https://doi.org/10.1145/2776880.2792704>
- [30] M. Tikam, “Posture control of a low-cost commercially available hexapod robot for uneven terrain locomotion,” Ph.D. dissertation, University of Pretoria, 2018.
- [31] Y. L. E. Nuin, N. G. Lopez, E. B. Moral, L. U. S. Juan, A. S. Rueda, V. M. Vilches, and R. Kojcev, “Ros2learn: a reinforcement learning framework for ros 2,” 2019.
- [32] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, “Openai baselines,” <https://github.com/openai/baselines>, 2017.
- [33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: <http://arxiv.org/abs/1707.06347>
- [34] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, “On the continuity of rotation representations in neural networks,” *CoRR*, vol. abs/1812.07035, 2018. [Online]. Available: <http://arxiv.org/abs/1812.07035>
- [35] C. Colas, O. Sigaud, and P.-Y. Oudeyer, “A Hitchhiker’s Guide to Statistical Comparisons of Reinforcement Learning Algorithms,” *arXiv:1904.06979 [cs, stat]*, Apr. 2019. [Online]. Available: <http://arxiv.org/abs/1904.06979>
- [36] G. S. P. Miller, “The definition and rendering of terrain maps,” *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, pp. 39–48, Aug. 1986. [Online]. Available: <https://doi.org/10.1145/15886.15890>
- [37] A. J. Ijspeert, “Central pattern generators for locomotion control in animals and robots: a review,” *Neural Netw*, vol. 21, no. 4, pp. 642–53, 2008.
- [38] K. Zahedi, N. Ay, and R. Der, “Higher Coordination With Less Control—A Result of Information Maximization in the Sensorimotor Loop,” *Adaptive Behavior*, vol. 18, no. 3–4, pp. 338–355, Jun. 2010, 00088. [Online]. Available: <http://journals.sagepub.com/doi/10.1177/1059712310375314>
- [39] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in *Proceedings of the 26th Annual International Conference on Machine Learning - ICML ’09*. Montreal, Quebec, Canada: ACM Press, 2009, pp. 1–8. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=1553374.1553380>
- [40] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, and D. Silver, “Emergence of locomotion behaviours in rich environments,” *CoRR*, vol. abs/1707.02286, 2017. [Online]. Available: <http://arxiv.org/abs/1707.02286>

Supplement for ‘Decentralized Deep Reinforcement Learning for a Distributed and Adaptive Locomotion Controller of a Hexapod Robot’

Malte Schilling¹, Kai Konen¹, Frank W. Ohl², and Timo Korthals³

OVERVIEW

This report provides supplementary information for the article ‘Decentralized Deep Reinforcement Learning for a Distributed and Adaptive Locomotion Controller of a Hexapod Robot’. It contains detailed results for the different experiments described in the paper and gives information on where to find the source code as well as how to run the provided docker image and the experiments.

I. PERFORMANCE AFTER LEARNING ON FLAT TERRAIN

In the first experiment, controllers were trained on flat terrain for 5000 epochs. We trained 15 controllers of each type.

Detailed results for all training runs are given in table I (for the decentralized architecture) and table II (for the centralized/baseline approach)

TABLE I: Detailed results after learning for 5000 epochs. The mean rewards were evaluated for each of the fifteen learned controllers for the decentralized architectures in 100 simulation runs after training. Note that standard deviation is taken for the single controller over the 100 repetitions which showed to be low.

Seed (Architecture)	Avg. Reward	Std. Dev. (btw. runs)	Overall Rank
Seed 1, Decentralized	764.209	26.01	2.
Seed 2, Decentralized	707.283	33.84	6.
Seed 3, Decentralized	570.693	39.56	21.
Seed 4, Decentralized	681.352	26.10	8.
Seed 5, Decentralized	756.096	27.50	3.
Seed 6, Decentralized	695.860	24.94	7.
Seed 7, Decentralized	733.194	56.82	4.
Seed 8, Decentralized	575.666	48.79	20.
Seed 9, Decentralized	598.666	30.16	18.
Seed 10, Decentralized	617.380	23.71	15.
Seed 11, Decentralized	638.920	38.65	11.
Seed 12, Decentralized	721.685	37.84	5.
Seed 13, Decentralized	551.884	58.02	25.
Seed 14, Decentralized	640.512	30.23	10.
Seed 15, Decentralized	603.263	29.08	16.
Mean performance	657.111		
Std. dev. over controller	68.072		

For statistical analysis, we compared the two architectures using a two-tailed Welch t-test (see [1]), applying the function from the `scipy.stats` package (`ttest_ind`) which returned a p-value of 0.011. The relative effect size was 1.02 which indicates a large effect size.

As the results showed a large variance, we compared as well the best performing controllers. Overall, the standard deviation in both conditions were in a similar range, but the distribution of the centralized approaches appeared larger. While the best performing approach was a centralized approach, there, first, seems to be an upper limit for maximum mean velocity and multiple approaches converged towards this performance. Overall, we found more decentralized architectures in the best performing approaches which was confirmed by a rank-sum test showing a significant difference (WilcoxonMannWhitney test, $p = .041$).

II. LEARNING ON FLAT TERRAIN

The learned architectures were trained for 5000 epochs. Learning curves for the individual seeds are shown in Fig. 1.

¹Malte Schilling and Kai Konen are with the Neuroinformatics Group, Bielefeld University, 33501 Bielefeld, Germany
mschilli@techfak.uni-bielefeld.de

²Frank W. Ohl is with the Department of Systems Physiology of Learning, Leibniz Institute for Neurobiology and with the Institute of Biology, Otto-von-Guericke University, Magdeburg, Germany.

³Timo Korthals is with the Kognitronik and Sensorik Group, Bielefeld University, 33501 Bielefeld, Germany.

TABLE II: Detailed results after learning for 5000 epochs. The mean rewards were evaluated for each of the fifteen learned controllers for the centralized (baseline) architectures in 100 simulation runs after training. Note that standard deviation is taken for the single controller over the 100 repetitions which showed to be low.

Seed (Architecture)	Avg. Reward	Std. Dev. (btw. runs)	Overall Rank
Seed 1, Central/Baseline	504.095	26.81	27.
Seed 2, Central/Baseline	770.158	21.52	1.
Seed 3, Central/Baseline	225.315	86.06	30.
Seed 4, Central/Baseline	566.485	13.29	22.
Seed 5, Central/Baseline	652.852	18.28	9.
Seed 6, Central/Baseline	302.733	5.48	29.
Seed 7, Central/Baseline	565.406	25.68	23.
Seed 8, Central/Baseline	600.676	23.58	17.
Seed 9, Central/Baseline	592.450	21.06	19.
Seed 10, Central/Baseline	557.175	27.35	24.
Seed 11, Central/Baseline	470.521	16.68	28.
Seed 12, Central/Baseline	621.663	26.89	14.
Seed 13, Central/Baseline	629.747	16.32	13.
Seed 14, Central/Baseline	630.497	14.97	12.
Seed 15, Central/Baseline	510.738	14.48	26.
Mean performance	546.701		
Std. dev. over controller	131.229		

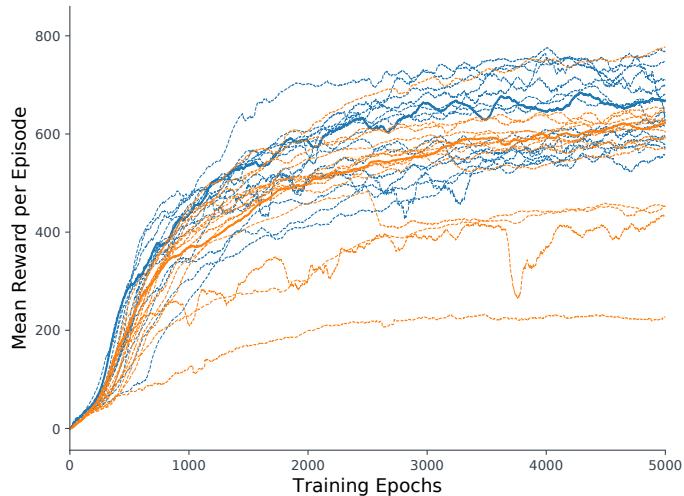


Fig. 1: Showing performance during training on flat terrain for individual seeds for 5000 epochs. Performance is measured as mean reward per episode (calculated as a running average over the last 100 episode in order to smoothen results). Results are given for the fifteen decentralized controllers (blue) and for the baseline centralized approach these are shown in orange. Dashed lines show individual seeds and medians are shown as continuous line.

Seeds were only measured up to 5000 epochs points as learning appears to have converged by then: Initially, we run simulations for up to 10000 epochs, but these only showed minor further improvements. There appears to be a maximum reachable velocity, Fig. 2.

Mean performance (shown in Fig. 3) was calculated over the fifteen seeds for each of the two architectures over training. Looking at individual runs showed a high variance during learning. The decentralized architecture learned significantly faster and reached the maximum performance of the centralized controller much earlier. The horizontal red dashed line visualizes the maximum of the centralized approach after 5000 epochs. The vertical red dashed line indicates when this performance level was reached by the decentralized approach (after 2187 epochs already).

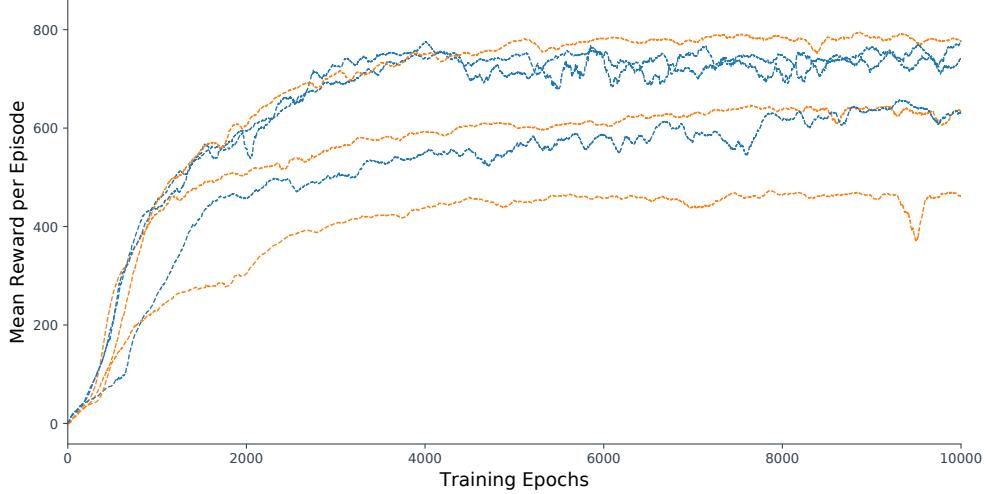


Fig. 2: Showing performance during training on flat terrain for individual seeds for 10000 epochs. Performance is measured as mean reward per episode (calculated as a running average over the last 100 episode in order to smoothen results). Results are given for selected decentralized controllers (blue) and for the baseline centralized approach these are shown in orange.

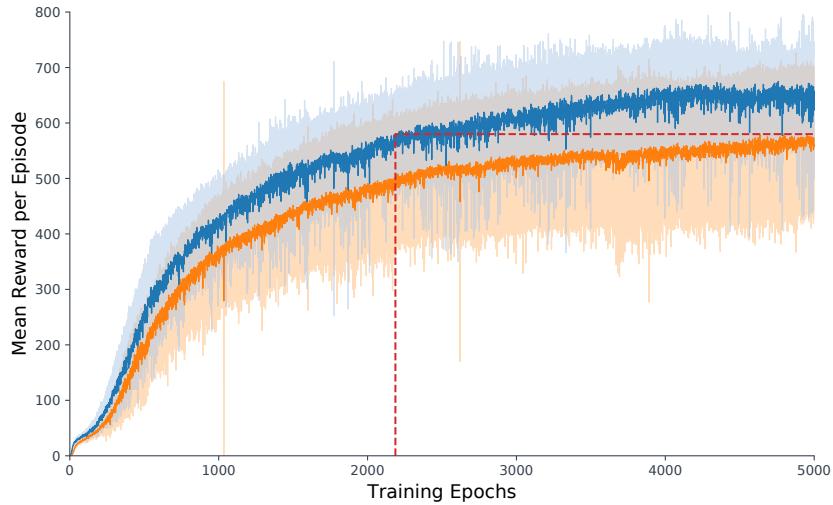


Fig. 3: Comparison of the mean reward during training: mean performance over the fifteen decentralized controllers is shown in blue and mean reward for the baseline centralized approach is shown in orange (shaded areas show standard deviation). Performance is measured as reward per episode. Seeds were only measured up to 5000 epochs points as learning appears to have converged by then. The horizontal red dashed line visualizes the maximum of the centralized approach at the end of training. The vertical red dashed line indicates when this performance level was reached by the decentralized approach (after 2187 epochs already).

III. GENERALIZATION AND LEARNING ON UNEVEN TERRAIN

Controllers were further evaluated on novel, uneven terrain that weren't experienced during learning. All controllers trained on the flat terrain were again tested for 100 simulation runs each on different uneven terrains. The terrains were generated from height-maps, using three different maximum heights of 0.05 m, 0.10 m, and 0.15 m. Height-maps were generated using the diamond-square algorithm [2]. Detailed results are given in table III and see Fig. 4.

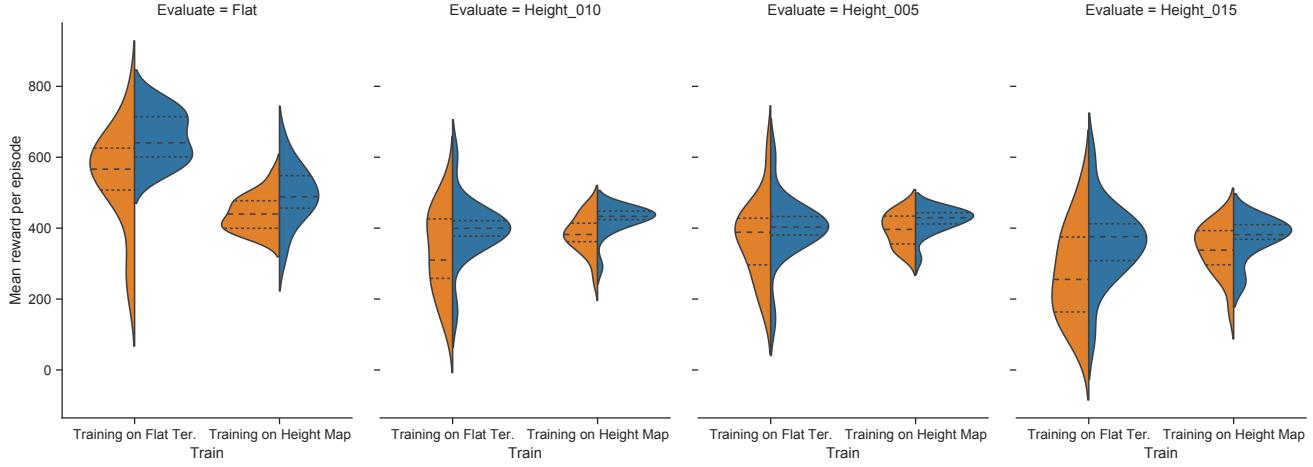


Fig. 4: Comparison of the (approximated) performance distribution for the controllers. Shown are violin plots for four evaluation conditions, from left to right: evaluation of walking on flat terrain; evaluation on three different height maps of maximum height of 0.05 m, 0.10 m, and 0.15 m. Inside the four evaluations, it is further distinguished between controllers trained on the flat terrain (left side) and controllers trained on the height map (right side). The violin plots show approximated distributions (and the quartiles are given inside): orange shows the centralized baseline approach and blue shows the decentralized architecture. Performance for the fifteen different seeds for each condition was measured as mean reward per episode (indicates mean velocity over simulation time).

TABLE III: Comparison of rewards between the different control architectures during evaluation. Data was collected during evaluation for 100 episodes for each controller. Given are mean rewards (and standard deviation in brackets) for each group of controllers (each group consisted of fifteen individually trained controllers). There were two different controller architectures, decentralized and centralized approaches, and two different training conditions, trained on flat terrain or on uneven terrain (using a height map with maximum height of 0.10 m). Values in italic reflect controllers that were evaluated on the same terrain they were trained on.

Condition	Decentralized Arch.		Centralized Arch.	
	Trained on ...	Flat Terrain	Uneven Terrain	Flat Terrain
Evaluation on flat terrain	<i>657.11(68.07)</i>	492.69(79.22)	<i>546.70(131.23)</i>	441.50(48.00)
Evaluation on height map (0.05 m)	400.86(87.15)	421.06(35.53)	369.14(105.84)	404.95(43.93)
Evaluation on height map (0.10 m)	<i>397.91(83.87)</i>	<i>424.30(40.09)</i>	334.06(112.94)	<i>382.42(51.23)</i>
Evaluation on height map (0.15 m)	357.68(104.93)	371.10(55.83)	271.09(125.96)	339.41(67.63)

IV. IMPLEMENTATION

The code of the implementation is open-sourced and available online. The easiest way to reproduce the results is to use the existing docker-image (from `kkonen/ros2_phantomx:master.thesis_final`). Detailed information on how to run the simulation and evaluation can be found in the repository accompanying the supplemental material at https://github.com/malteschilling/ddrl_hexapod.

In the experiments, we used a simulated robot¹ in the ROS simulation environment *Gazebo* [3] as it is open source and because of the good support of ROS (here ROS 2, version Dashing Diademata [4]). As a physics-engine *Bullet* was used [5]. The simulation required several custom modules (e.g., publishing joint-effort controllers in ROS 2) that have been implemented now for ROS 2². One current difference between simulated robot and the real robot was the implementation of ground contact sensors using data from the simulation environment, but solutions for such sensors have been proposed for the robot as well [6]. Simulator and ROS were run with a fixed publishing-rate of 25 Hz in order to align sensor and actuator data. This guaranteed a stable and constant step-size.

As a framework for Deep Reinforcement Learning (DRL) we employed OpenAI's gym. The connection to the simulator through the Robotic Operation System (ROS) was wrapped as a new environment through a connector using *ROS2Learn* [7].

¹robot definitions: <https://github.com/kkonen/PhantomX>

²for details see the open repository <https://github.com/kkonen/ros2learn>

This allowed to use the standard baseline implementations of DRL algorithms [8]. In this approach Proximal Policy Optimization (PPO) [9] was used as it has shown to work well on continuous tasks without the need of intensive hyperparameter tuning. As a further advantage, it has a comparably low sampling complexity.

For details on the architectures and hyperparameters see original paper.

REFERENCES

- [1] C. Colas, O. Sigaud, and P.-Y. Oudeyer, “A Hitchhiker’s Guide to Statistical Comparisons of Reinforcement Learning Algorithms,” *arXiv:1904.06979 [cs, stat]*, Apr. 2019. [Online]. Available: <http://arxiv.org/abs/1904.06979>
- [2] G. S. P. Miller, “The definition and rendering of terrain maps,” *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, pp. 39–48, Aug. 1986. [Online]. Available: <https://doi.org/10.1145/15886.15890>
- [3] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sep. 2004, pp. 2149–2154 vol.3. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/1389727>
- [4] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.
- [5] E. Coumans, “Bullet physics simulation,” in *ACM SIGGRAPH 2015 Courses*, ser. SIGGRAPH ’15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: <https://doi.org/10.1145/2776880.2792704>
- [6] M. Tikam, “Posture control of a low-cost commercially available hexapod robot for uneven terrain locomotion,” Ph.D. dissertation, University of Pretoria, 2018.
- [7] Y. L. E. Nuin, N. G. Lopez, E. B. Moral, L. U. S. Juan, A. S. Rueda, V. M. Vilches, and R. Kojcev, “Ros2learn: a reinforcement learning framework for ros 2,” 2019.
- [8] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, “Openai baselines,” <https://github.com/openai/baselines>, 2017.
- [9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: <http://arxiv.org/abs/1707.06347>