Towards General and Autonomous Learning of Core Skills: A Case Study in Locomotion

Roland Hafner
DeepMind
rhafner@google.com

Tim Hertweck
DeepMind
thertweck@google.com

Philipp Klöppner
TU Darmstadt
philipp.kloeppner@gmx.de

Michael Bloesch
DeepMind
bloesch@google.com

Michael Neunert
DeepMind
neunertm@google.com

Markus Wulfmeier
DeepMind
mwulfmeier@google.com

Saran Tunyasuvunakool
DeepMind
stunya@google.com

Nicolas Heess
DeepMind
heess@google.com

Martin Riedmiller
DeepMind
riedmiller@google.com

Techreport, 30 Aug 2020

Abstract: Modern Reinforcement Learning (RL) algorithms promise to solve difficult motor control problems directly from raw sensory inputs. Their attraction is due in part to the fact that they can represent a general class of methods that allow to learn a solution with a reasonably set reward and minimal prior knowledge, even in situations where it is difficult or expensive for a human expert. For RL to truly make good on this promise, however, we need algorithms and learning setups that can work across a broad range of problems with minimal problem specific adjustments or engineering. In this paper, we study this idea of generality in the locomotion domain. We develop a learning framework that can learn sophisticated locomotion behavior for a wide spectrum of legged robots, such as bipeds, tripeds, quadrupeds and hexapods, including wheeled variants. Our learning framework relies on a data-efficient, off-policy multi-task RL algorithm and a small set of reward functions that are semantically identical across robots. To underline the general applicability of the method, we keep the hyper-parameter settings and reward definitions constant across experiments and rely exclusively on on-board sensing. For nine different types of robots, including a real-world quadruped robot, we demonstrate that the same algorithm can rapidly learn diverse and reusable locomotion skills without any platform specific adjustments or additional instrumentation of the learning setup. (Supplementary video available.[1])

1 Introduction

Robots with legs and hybrid leg-wheel configurations are rapidly gaining popularity as mobile helpers with the potential to navigate challenging, human-built environments. The control of such platforms is a well studied engineering and research problem and the last two decades have seen impressive demonstrations of robot locomotion, with solutions excelling both in speed and robustness [1, 2, 3, 4]. Despite these successes, however, approaches used by classical control engineers and roboticists usually require detailed understanding of and specialization to the platform at hand. Learning based approaches to control, especially Reinforcement Learning (RL) algorithms, have made much progress in the last few years [5, 6, 7, 8]. They hold the promise of solving challenging motor control problems directly from raw sensory inputs, optimizing the perception-action pipeline end-to-end. In particular, they can be a general paradigm that allows us to learn a solution, even if it were difficult or expensive for a human expert, with a well-defined goal but minimal prior knowledge. However, in order for RL to really keep this promise, we need algorithms and learning setups

[1] https://youtu.be/7V0-oj3b5I4

that can function across a wide range of problems with minimal problem-specific adjustments or design. Although the data-efficiency and robustness of RL algorithms has much improved, significant task-specific effort is still required for algorithm tuning, reward design and providing specific hard- and software for reward calculation. This can make the success of learning experiments highly dependent on the availability of RL expert knowledge and limit them to carefully controlled lab settings. Our learning framework relies on a data-efficient multi-task RL algorithm [6]. With a small set of reward functions that are semantically simple and, above all, identical across robots, we show that we can learn sophisticated locomotion behavior for a wide range of robots such as bipeds, tripeds, quadrupeds and hexapods, including wheeled variants. We demonstrate that in our learning framework, the same RL agent, with a single setting of hyperparameters and the same set of reward functions, can learn diverse and reusable locomotion skills for 9 different types of robots. The framework is sufficiently data efficient to enable learning directly on a real-world quadruped without any adjustments. Although the reward semantics are identical across robots the resulting control policies vary significantly in line with the highly diverse dynamic properties of the platforms. Importantly, as it relies exclusively on on-board sensing it does not require any additional instrumentation of the learning setup, neither for state estimation for the controller nor for reward calculation, thus enabling learning experiments beyond a controlled lab setting.

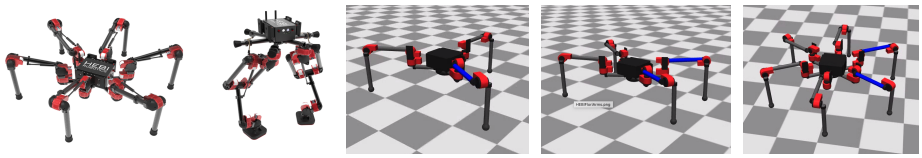
Our results are complementary to other recent results on learning locomotion such as those of [5, 9, 10], which focus primarily on data-efficiency, robustness of the resulting gaits, or the autonomy of the learning process. Our work also addresses these points but specifically emphasizes the generality and robustness of the learning framework. Beyond locomotion, and in combination with the results of [6, 11] the results in the present paper provide another small piece of evidence that the grand vision of general, autonomous robot learning may not be entirely beyond reach.

2 Preliminaries

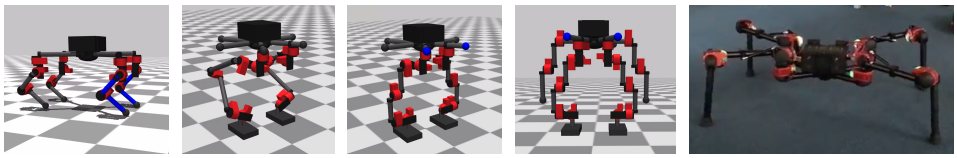
The goal of this paper is to study the generality of learning techniques and we thus want to evaluate our learning framework on a diverse set of robot platforms. To reduce the effort, we will mainly work with platforms that are simulated as true to the original as possible. Unfortunately, even creating and validating a large number of independent simulation models requires a lot of work. We therefore rely on a modular hardware system which allows to construct different robot models from a small number of hardware building blocks. Rather than performing system identification for each robot model separately, we can then identify the properties of the hardware modules in isolation and use these well-calibrated simulation components to build a large number of realistic models with very different morphologies and dynamic properties. Obtaining a good alignment between the learning results in the simulation and the actual hardware on a small number of models can give us some confidence that the results in the simulation are meaningful for other models as well.

HEBI Robotics (www.hebirobotics.com) is a provider of a modular hardware system for robotics. The system is built around series elastic actuator modules that are available with different nominal speed and torque ratings. The series elastic elements allow for accurate torque control and protect the motor from strong impacts. The modules have a rich set of sensors built in: encoders for the motor and the output shaft, temperature sensors, a 3 axis accelerometer, as well as a 3 axis gyroscope (including on board orientation estimation). A low level controller, consisting of integrated control and power electronics, implements different control modes and safety mechanisms, and processes sensor information. In combination with accessories such as brackets and tubes, the modules allow building a wide variety of different robots, including (but not limited to) legged robots. An attractive feature of the system is that we have access to all relevant state variables used for low-level control (actuator position, velocity, deflection, deflection velocity, torque, motor temperature, etc.). In Fig 1b and 1a two existing walker topologies are shown. As Florence is currently only available as a prototype we use the Daisy kit for evaluation of our learning framework in real-world experiments.

For our investigations, we have developed a MuJoCo [12] based simulation of these building blocks. In our simulation we attempt to faithfully reflect the modularity as well as the kinematic and dynamic properties of the system. The latter include not only the properties of the motor, gearbox and serial elastic element, but also the firmware safety features, temperature models and overheating effects. We implemented 7 basic robot models with different morphologies and dynamic properties. An overview can be seen in Fig 1c to 1i).



(a) Real Daisy. (b) Real Florence. (c) Daisy3 (d) Daisy4 (e) Daisy6



(f) Dog (g) Florence (h) Flori (i) FloriArms (j) Real Daisy4

Figure 1: Real and simulated creatures built using the HEBI system.

Fig 1e shows the Daisy hexapod (Daisy6) in the configuration of the original kit. It has two degrees of freedom in each shoulder and one additional in each elbow, resulting in 18 active degrees of freedom. By removing two legs, we get a more challenging to control quadruped robot (Daisy4, see Fig 1d), with 12 active degrees of freedom. It is notable that for a human engineer, these two topologies already differ in important dynamical aspects: e.g. only the hexapod allows for a simple bipartite statically stable gait [13]. To have non-optimal, but still somehow feasible, kinematics, we can remove another leg to get a three legged version of the Daisy (Daisy3, see Fig 1c), with nine active degrees of freedom (which is very difficult to control with standard methods since it will almost inevitably have to rely on friction dynamics).

We also modeled a mammalian leg configuration (see Fig 1f), by changing the orientation of the shoulders and adapting slightly the lengths of the upper and lower legs. This assembly has the same number of active degrees of freedom as Daisy4, but strongly differs in forward kinematics and joint torque loads (it is usually less strenuous for the shoulder joints).

Another important class of robots for locomotion are bipedal robots. Fig 1g shows a simulation model of HEBI Florence. Florence has three active degrees of freedom in each hip, one degree of freedom in the knee and two degrees of freedom in each ankle. It further has long upper and lower leg segments and starts with a backward flexed knee to prevent operation close to kinematic singularities and to make balancing easier. While the latter are design choices that were made with engineered control solutions in mind, we also designed a more human like version of the Florence - that we call Flori (see Fig 1h) - with the same number of active degrees of freedom but different leg configurations. To have a rudimentary example for additional limbs, we also added a Flori version with two arms adding two additional degrees of freedom for each of them (FloriArms, see Fig 1i).

3 A General Framework for Learning Core Locomotion Skills

Our long-term goal is the development of a general and autonomous learning framework. 'General' means that the same framework with minimal modifications can be applied across a broad range of different platforms. 'Autonomous' means, that our system is able to learn with minimal external infrastructure and assistance. In the present work, we focus on general proprioceptive rewards, relying only on on-board sensing, thereby reducing reliance on external sensors, and other elaborate lab settings.

Many contemporary applications of RL require careful, task-specific engineering of the rewards together with expensive additional hardware such as motion capture systems, to enable reward computation. This can make learning experiments expensive and often restricts them to specialized laboratories. Reducing this dependency both for acting and learning will increase the applicability of mobile robots, and, importantly, dramatically simplify the setup of learning experiments, enabling learning and adaptation to proceed after deployment.

Rather than relying on pre-processed position or velocity information from a separate state estimation system, we train agents to act directly from raw sensor values. We further demonstrate how a set of primitive rewards for locomotion can be derived directly from the same on-board sensors also used for acting, and how these rewards can be used to learn diverse and robust locomotion skills.

3.1 Reward Computation for General Locomotion Topologies

Our learning framework relies on a diverse set of basic rewards. The combination of these rewards enables learning a diverse set of locomotion skills which can subsequently help to learn more complex behaviors. The rewards are defined such that they can be computed from limited on-board sensing comprising IMUs and joint encoders but require no contact sensing. To this end we draw on heuristics to obtain rough estimates of helpful quantities such as egocentric velocity. The underlying assumptions of these estimates do not hold at all times, but the agent has access to the full sensory stream and is able to learn robust locomotion skills despite the potentially limited consistency of the rewards. Assuming that the lowest foot is in contact with the ground and not slipping, we estimate the linear velocity of the robot torso and the feet in a coordinate system that is simultaneously aligned with the robots forward direction and gravity (motivated by [14], details see Appendix). Together with the IMU gyroscope measurements in the torso, we can now define various rewards for locomotion.

An important skill in locomotion is to learn to stand upright. We define the ***StandUpright*** reward by keeping the robot torso leveled (reducing the roll and pitch angles) while keeping the torso linear velocity and the torso angular rotation rate small. If we add a component for rewarding height differences of a certain foot i w.r.t the lowest foot, we can define a reward function for standing upright and lifting a certain foot: $\boldsymbol{LiftFoot}_i$. For doing actual locomotion, we can modify the stand upright reward by rewarding rotational velocities around the torso z axis to get a ***Turn*** reward and reward translational velocities of the torso and the feet to get a ***Walk*** reward. While we could have rewards for different velocities, we picked rewards to maximize discrete instances of these rewards for this paper. In consequence we define six distinct locomotion skill rewards: ***TurnLeft***, ***TurnRight***, ***WalkForward***, ***WalkBackward***, ***WalkLeft*** and ***WalkRight***, as well as the ***LiftFoot*** for all feet for each creature (for details, see Appendix).

3.2 Action and Observation Space

The actuation modules offer multiple control modes, including position control, velocity control, torque control and PWM direct control mode. In principle, our learning methods should be able to cope with all of these modes and will learn to make use of them. While each of the modes has its own pros and cons, we picked the position control mode using a low-gain P-controller. The main advantage of the position control mode is that we can enforce certain limits of the joint angles during the execution of our agent while it can still regulate forces indirectly by choosing appropriate position set-points. As an additional safety mechanism, we use a sliding window filter with a width of ν steps for the set-point that is sent to the actuation modules. In consequence, we have an action space where each of the used modules adds one dimension of continuous actions that is bounded by the allowed position set-point for that individual joint.

For a robot that is built from multiple actuator modules, the observation space consists of observations associated with the individual modules, as well as observations from the torso. For a default filter window of width $\nu = 5$, this adds up to a 11 dimensional observation for each of the actuation modules, containing the position and velocity of the joint and elastic element, temperatures and filter state. For the torso observations, we stack measurements of h consecutive time frames to allow the agent to have richer information about the state. As we only use robo-centric measurements, we provide the roll and pitch estimate together with the feet reference points and the measurements of the gyro. Consequently, the range of dimensionality of the action and observation spaces we investigate here ranges from 9 action dimensions with 127 observation dimensions for Daisy3 up to 18 action dimensions for Daisy6 and 282 observation dimensions for FloriArms (details, see Appendix).

3.3 Multi-Task Training of a Locomotion Module

In general, we aim for a capable locomotion module, that can not only solve one task, but is able to perform multiple tasks. This makes the motion module not only more versatile, we also expect syn-

ergies across tasks that will improve data-efficiency in this multi-task learning setting. To this end, we apply the Scheduled Auxiliary Control (SAC-X) [6] framework to the domain of locomotion. The core idea of SAC-X is that we can learn multiple tasks in parallel, switching between different tasks during each episode, and sharing data across tasks for learning. This framework has three potential advantages: (1) switching between tasks forces the agent to visit different parts of the state space and can thus improve exploration (and in consequence data-efficiency); (2) switching between tasks can also improve robustness of policies since behaviors are initiated in a more diverse set of states; (3) sharing data across tasks via off-policy learning can further improve data-efficiency. We expect the resulting controller module to provide a sound basis of finely tuned movement skills that eventually also allow to achieve more high-level goals.

4 Experiments

To investigate the framework outlined in the previous section we conduct a case study that focuses on a set of basic locomotion skills *StandUpright*, *LiftFoot*, *TurnLeft*, *TurnRight*, *WalkLeft*, *WalkRight*, *WalkForward*, *WalkBackward* (see 3.1; note that additional rewards could be easily defined following the same approach). We use the off-policy RL algorithm used in [6], using the very same hyper-parameters that were also used in other domains like manipulation. In each episode the robot starts with all actuators in the default position, feet touching the ground (see Fig 1e to 1g). We run each episode for 800 steps with a control time step duration of 25 ms, which yields episodes of 20 seconds length. We are interested in applying our approach directly on a real robot platform. Our main interest therefore is data-efficiency which we measure by counting the episodes that were required to learn the behaviour(s) (details, see Appendix). This gives us a good estimate of whether learning the tasks has reached a level of efficiency such that it could be trained in the real world.

4.1 Individual Skills

We first investigate the plausibility of our reward definition in a singe-task setting. As Table 1 shows, we can learn the individual locomotion skills on all platforms in a reasonable number of interaction episodes. For instance, starting from a random policy we can successfully learn behaviours like *StandUpright* for creatures like Daisy6, Daisy4 and Daisy3 in less than 20 interaction episodes. This is equivalent to less than 7 minutes of interaction between the agent and the robot. Furthermore, our results for the bipedal robots Florence, Flori and FloriArms show that the very same reward definition can have a very different complexity depending on the configuration we apply it to. Since the static stability of these creatures is strongly impeded by the reduced support polygons, the agent needs to be much more careful when moving its center of mass. Still it can learn the task in less than 4h of interaction time (about 700 episodes) for all robots.

This is even more evident for *LiftFoot*: depending on the structure of the robot platform the same reward definition results in tasks of varying difficulties and leads to very different solution strategies: While we can learn to lift a certain foot for Daisy6, Daisy4 and Dog in less than 20 minutes of interaction time (about 50 episodes), the task is considerably harder for the bipedal robots. Nevertheless, the very same agent and reward learns a balancing policy on one leg in about 5.6h of interaction time (about 1000 episodes). These results highlight that the same simple reward definition can give

	StandUpright	*LiftFoot*$_1$	*TurnLeft*	*WalkLeft*	*WalkForward*	
	Episodes	Episodes	Episodes	Episodes	Episodes	Velocity
Daisy6	<20	60	120	180	170	0.59 m/s
Daisy4	<20	50	90	150	160	0.62 m/s
Dog	30	50	210	520	310	0.70 m/s
Daisy3	<20	N/A	420	250	240	0.01 m/s
Florence	650	1000	1100	1340	1200	1.20 m/s
Flori	710	1000	1220	1220	1120	1.45 m/s
FloriArms	700	320	1400	1210	1100	1.44 m/s

Table 1: Results for learning basic locomotion tasks on different walker platforms in a single-task setting. Shown are the interaction episodes required to learn the task and the final velocity reached for the *Walk* tasks.

rise to very different behaviors. We get comparable results for *TurnLeft* and *TurnRight*, as all of our creatures are symmetric. For Daisy4 and Daisy6 these tasks are considerably more difficult than *StandUpright* and *LiftFoot* and the amount of interaction data that is required to learn the skills roughly doubles. For *Walk* we can learn a reasonable fast solution for Daisy6 and Daisy4 in about an hour of interaction time. The resulting gait looks highly symmetric even though we do not directly encourage this in the reward. Interestingly, the agent also finds a very good gait for the bipeds Florence, Flori and FloriArms in about 7.5h of interaction time (about 1200 episodes). This walking gait looks not only very symmetric but also very dynamic. The learned walking gait for *WalkLeft*, *WalkRight*, *WalkForward* and *WalkBackward* take a comparable amount of interaction episodes to learn, but as can be seen in Table 1 vary widely in the achievable speed.

It is worth noting that we apply exactly the same reward function, agent and hyperparameters to all robot platforms. The characteristics of the resulting behaviors, however, vary widely and are naturally adapted to the morphology and dynamic properties of each platform, e.g. FloriArms learns to use its arms for additional support while lifting a leg and to swing it's arms in a very natural way to keep balance while walking[2].

4.2 Learning a Versatile Motor Module

To obtain a versatile motor module we would like to be able to learn a large number of locomotion skills in parallel. Although learning many individual skills separately is feasible, it is not the most data-efficient way to achieve this. Also, when learning skills separately we are not guaranteed to be able to transition between skills. We therefore switch to the multi-task regime outlined in section 3.3 in which we switch between and share data across tasks [6]. We keep the basic learning algorithm, parameters and the general learning setting from the previous sections. We consider three basic task definitions: *WalkForward*, *WalkBackward* and *StandUpright*. In every episode we execute two sequences of 10 seconds length each, giving a total episode length of 20 seconds as before. In each sequence we randomly execute one of the three tasks to collect data (this corresponds to the SAC-U version of the algorithm described in [6]).

For the quadrupeds and hexapod, we see a small increase in data-efficiency compared to the single-task experiments. For example, we need 360 episodes in total for Daisy6 when learning each task in a separate experiment, while we can learn all skills together in only 300 episodes in the multi-task setting (results are comparable for Daisy4 and Dog). For the bipeds, the differences are much bigger. For example we would need 3050 episodes for Florence to learn all skills separately, while we can learn them in 1590 episodes in the multi-task setting. While this saves only roughly 20 minutes of interaction time for the quadrupeds and hexapod, the savings amount to over 8h of interaction time for the bipeds. Importantly, in the multi-task setting the agent also learns to transition between *WalkForward*, *WalkBackward* and *StandUpright* without falling, which is a very challenging task for itself for many control approaches.

4.3 Learning Higher Level Behaviours: Reaching a Target

In the previous section we have demonstrated that the multi-task regime allows us to learn multiple individual skills more efficiently and robustly than when learning separately. Many more complex tasks, however, cannot reasonably be learned in a single-task setting at all. We now show that the training regime from the previous section enables learning both locomotion skills as well as more complex tasks that build on these skills, including sparse reward tasks that would be hard to learn otherwise. To this end we add a virtual target to the environment that is randomly spawned in a certain range around the robot. We add a sparse *ReachTarget* task to the set of training tasks for our locomotion module. The reward is zero when the target is more than 50 cm away from the robot and one when the distance of the target and the robot torso is zero. In each episode the target is spawned at a distance from 1 to 3 meters around the robot.

Creature	Baseline	SAC-Q
Daisy6	>20k	840
Daisy4	>20k	800
Florence	>20k	2000

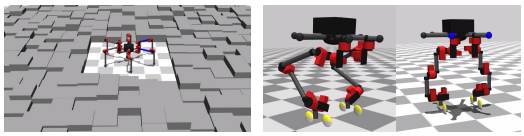
Table 2: Interaction episodes for task *ReachTarget*.

[2] e.g. see supplementary video https://youtu.be/7V0-oj3b5I4

As baseline we attempt to learn the task with only the *ReachTarget* reward and the default settings of our agent. For all creatures this baseline fails to solve the task in the first 20k episodes. As a comparison, we use our motion module in the multi-task setting together with 3 auxiliaries (*WalkForward*, *WalkBackward*, *StandUpright*). As we can see in Table 2, we can learn all skills plus the main task *ReachTarget*, in a reasonable time of about 5h (roughly 800 episodes) for the quadrupeds and hexapods and about 12h of interaction time (roughly 2000 episodes) for the bipeds. In this experiment, we assumed that we train all tasks from scratch, while in practice it would also be possible to pre-train a set of skills and learn only the main task, which would make the motion module even more powerful.

4.4 Robustness of Proprioceptive Reward Definitions

As discussed in section 3.1 the reward calculation is based on simplifying assumptions which may not always hold true. While these may appear restrictive, we do not require them to hold at every point in time in order to promote the emergence of sensible locomotion behaviors. To demonstrate that we can deal with violations of the assumptions and to underline the robustness of our approach, we conduct experiments in an expanded set of tasks. In a first experiment, we let our creatures run over uneven, tiled terrain and can observe that learning for height differences of a few centimeters still works successfully. Moreover we see that platforms with more legs can overcome rougher terrain using the same rewards (e.g. see Fig 2a). In a different experiment, we extend the 7 creatures by 2 more and attach passive wheels to the feet of the bipeds Florence and Flori. Running the same experiments results in a completely different locomotion pattern: dynamic skating (see Fig 2b, for more details, see Appendix).



(a) Rough terrain. (b) Passive skates.

Figure 2: Platforms in the robustness experiments.

5 Real-World Experiments

To verify the results obtained in simulation we conduct learning experiments 'from scratch' on an actual HEBI robot. Instead of the original HEBI Daisy (Daisy6, see Fig 1a), we decided to run the real world experiments on the more challenging quadruped Daisy4 that is shown in Fig 1j. We use the same settings as in simulation (agent, rewards, methods, hyperparameters, etc.). From a control perspective going to a real robot means that the agent now has to deal with additional time delays and noise that makes the control problem more difficult. When we run the single-task experiments of section 4.1, we initialize the robot in each episode to its default pose by a hand designed initialization procedure. Afterwards we can start the episode in the same way as we do in simulation. While the reset of the robot after an episode is not a problem in simulation, we allow more time in between episodes to manually turn the robot around when it used up the available space.

To learn *WalkForward* in the real robot experiment, we need approximately 130 episodes, which is even a bit less compared to the simulation experiments (160 episodes). While this corresponds to approximately 40 minutes of pure interaction time, the full experiment (including resets) runs for about 2h. The resulting walking gait is highly symmetric and achieves a speed of approximately 0.3 m/s. We further conduct a multi-task experiment in the real world with 6 different tasks: $LiftFoot_1$, $LiftFoot_2$, $LiftFoot_3$, $LiftFoot_4$, *WalkForward* and *WalkBackward*. We use the same setting as in section 4.2, but increase the sequence length to 20 seconds. Starting from a random initialisation, the agent is able to learn all the tasks requiring only 225 episodes of robot interaction. This corresponds to approximately 3h of pure interaction time, while the overall experiment (including resets) runs for about 5h. This demonstrates that we can learn robust skills that allow for smooth transitions not just in simulation but also on an actual robot in reasonable time from scratch. It further demonstrates that another core feature of our simulation results holds true on the real robot: specifically, the multi-task setup continues to provide us with increased data efficiency, as we would have to run 460 episodes (10h) to learn all the skills in a single-task setting. Without multi-task training we would have had to wait for additional 5h and would not have learned to transition between skills.

6 Related Work

Legged locomotion has seen significant progress in the last couple of decades with increasingly performing hardware and control approaches [1, 15, 16]. Optimal control approaches, partially implemented as Model Predictive Control (MPC), have gained traction, especially combined with a centroidal dynamics approximation which allows separating the problem into a high-level base motion controller and a low-level contact force controller [17, 18, 19]. But also whole-body approaches have been successfully investigated by various research groups [20, 21]. These approaches can reach an astonishing level of dynamics and agility [22].

On the other hand side in the last few years there has been a growing interest in learning locomotion both in simulation and for real robots. In simulation, especially for simple robot models, basic locomotion behavior can often be achieved with simple reward functions [e.g. 23]. More sophisticated and diverse skills can be obtained through curricula and diverse training conditions (such as different terrains) [7]. However, in general, such skills require carefully chosen shaping or penalty terms [8] or constrained optimization [24] that in turn are time-consuming, may need an iterative process [25] and are specific for a certain platform. However, in general, such skills often lack the naturalness, efficiency, smoothness and other properties that would be essential for deployment on actual robotics hardware. This can be mitigated through carefully chosen shaping or penalty terms [8], or constrained optimization [24]. But designing regularization strategies that shape behaviors in particular ways can be a time-consuming endeavor and may require an iterative process [25].

The results in this paper are also complementary to several recent demonstrations of successful sim-to-real transfer of control policies for legged robots [8, 25, 26]. Although training in simulation offers additional flexibility, successful transfer usually requires detailed knowledge of dynamic properties of the robot of interest to build sufficiently accurate simulation models or additional adaptation of the learned control strategies on the actual hardware [27, 28, 29]. In some cases demonstrations, e.g. from motion capture data [e.g. 30, 31] or other reference motions [25, 27] can be used to directly constrain learned behavior. Yet, such data is not always easily available or may not easily transfer to a particular robot body. Furthermore, composing reference behaviors in a flexible, goal-directed manner can be challenging [e.g. 31, 30]. Our work uses a multi-task learning scheme taken from [6, 11, 32] that employs several simple reward functions with minimal additional shaping terms to obtain well regularized and robust behavior across a number of different bodies.

In some cases, locomotion skills learned in simulation can be transferred to corresponding robotic hardware. This usually requires careful system identification and well matched simulation models [8, 29, 25, 26]. Transfer can be further improved with additional adaptation of the learned control strategies on the actual hardware [27, 28, 29]. Accurate simulation models can, however, be expensive to develop, and some phenomena encountered in the real world (such as sophisticated terrain properties) may be hard to simulate.

Recent improvements in the efficiency of learning algorithms has made it possible to learn locomotion skills directly on the robot. This has been pursued both with model-based [33], and with model-free approaches [5, 29, 9] for quadrupeds [5, 29] and the HEBI Daisy robot which we are considering here [9]. Similar to our work, [29, 9] learn multiple skills that can later be chained to achieve goal directed behaviors. Learning on the hardware requires answering practical questions related to safety, reset, and state-estimation e.g. to compute rewards. For the latter, prior work usually relies on external motion capture systems which can require significant effort to set up. We show that sophisticated skills can be learned from simple rewards computed from on-board sensors only, thus significantly reducing the complexity of the training setup. Furthermore, whereas prior work usually targets a single robot platform, we investigate whether the same setup can be used across a number of different robots.

Our use of multiple simple rewards derived from on-board sensing is closely related to the work of [11] who use a similar scheme to solve difficult tasks with a robotic arm. It also bears similarity to a number of papers who employ learned reward functions, for instance based on an empowerment objective, to discover reusable skills [34, 35, 36, 37, 34], including for legged robots [38]. Our reward functions are hand-crafted, but nevertheless simple and transferable across body morphologies.

7 Conclusion

We have investigated a framework for learning of core locomotion skills for general walker topologies and applied it to a diverse set of robots with very different morphologies and dynamic properties. We have demonstrated that the same set of reward functions and the same learning framework (identical algorithm and hyperparameter settings) can successfully learn a diverse set of robust locomotion skills for all platforms and we can reuse these skills to learn more complex tasks. Even though the rewards are the same for all robots, the resulting skills are naturally adapted to the characteristics of each platform. Our framework is sufficiently data-efficient to learn all tasks in a couple of hours of interaction time, and we have verified some of our results in simulation with matching experiments on real hardware. Our framework and reward definitions further minimize the need for external state estimation and instrumentation of the learning setup by relying only on on-board sensing. This has already made it possible to conduct experiments for some of our robots essentially in the wild, although further work will be necessary for more complicated robots such as the biped Florence, e.g. to ensure their safety during learning.

We believe that learning frameworks that are general enough to work across a wide range of platforms with minimal adjustments and that enable more autonomous learning will be an important step to fully reap the benefits of self-learning systems in robotics (for work similar in spirit in the manipulation domain, see [11]).

Acknowledgements

We thank Mr Florian Enner, HEBI Robotics, for the excellent technical support and expert consultation on Daisy and Florence.

References

[1] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart. Quadrupedal locomotion using hierarchical operational space control. *The International Journal of Robotics Research*, 33(8):1047–1062, 2014.

[2] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson. Optimization-based full body control for the darpa robotics challenge. *Journal of Field Robotics*, 32(2):293–312, 2015.

[3] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[4] M. Bjelonic, C. D. Bellicoso, Y. de Viragh, D. Sako, F. D. Tresoldi, F. Jenelten, and M. Hutter. Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots. *IEEE Robotics and Automation Letters*, 4(2):2116–2123, 2019.

[5] T. Haarnoja, A. Zhou, S. Ha, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. *ICML*, 2018. URL http://arxiv.org/abs/1812.11103.

[6] M. A. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. V. de Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by playing - solving sparse reward tasks from scratch. *ICML*, 2018. URL http://arxiv.org/abs/1802.10567.

[7] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[8] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[9] T. Li, N. Lambert, R. Calandra, F. Meier, and A. Rai. Learning generalizable locomotion skills with hierarchical reinforcement learning. 09 2019.

[10] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan. Learning to walk in the real world with minimal human effort. 2020.

[11] T. Hertweck, M. Riedmiller, M. Bloesch, J. T. Springenberg, N. Siegel, M. Wulfmeier, R. Hafner, and N. Heess. Simple sensor intentions for exploration. *arXiv preprint arXiv:2005.07541*, 2020.

[12] 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*, pages 5026–5033. IEEE, 2012.

[13] U. Saranli, M. Buehler, and D. E. Koditschek. Rhex: A simple and highly mobile hexapod robot. *The International Journal of Robotics Research*, 20(7):616–631, 2001.

[14] M. Bloesch, M. Hutter, P. Vadakkepat, and A. Goswami. Technical implementations of the sense of balance. *Humanoid Robotics: A Reference*, 2017.

[15] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell. Design of hyq–a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225 (6):831–849, 2011.

[16] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.

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

[18] S. Kuindersma, F. Permenter, and R. Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2589–2594. IEEE, 2014.

[19] O. Villarreal, V. Barasuol, P. Wensing, and C. Semini. Mpc-based controller with terrain insight for dynamic legged locomotion. *arXiv preprint arXiv:1909.13842*, 2019.

[20] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard. Whole-body model-predictive control applied to the hrp-2 humanoid. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3346–3351. IEEE, 2015.

[21] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, 2018.

[22] E. Guizzo. By leaps and bounds: An exclusive look at how boston dynamics is redefining robot agility. *IEEE Spectrum*, 56(12):34–39, 2019.

[23] N. Heess, G. Wayne, Y. Tassa, T. Lillicrap, M. Riedmiller, and D. Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

[24] S. Bohez, A. Abdolmaleki, M. Neunert, J. Buchli, N. Heess, and R. Hadsell. Value constrained model-free continuous control. *arXiv preprint arXiv:1902.04623*, 2019.

[25] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *Conference on Robot Learning*, pages 317–329, 2020.

[26] O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv preprint arXiv:1908.05224*, 2019.

[27] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu. Sim-to-real transfer for biped locomotion. *arXiv preprint arXiv:1903.01390*, 2019.

[28] T. Sun, D. Shao, Z. Dai, and P. Manoonpong. Adaptive neural control for self-organized locomotion and obstacle negotiation of quadruped robots. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1081–1086. IEEE, 2018.

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

[30] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. In *Advances in Neural Information Processing Systems*, pages 3686–3697, 2019.

[31] J. Merel, L. Hasenclever, A. Galashov, A. Ahuja, V. Pham, G. Wayne, Y. W. Teh, and N. Heess. Neural probabilistic motor primitives for humanoid control. *arXiv preprint arXiv:1811.11711*, 2018.

[32] M. Wulfmeier, A. Abdolmaleki, R. Hafner, J. T. Springenberg, M. Neunert, T. Hertweck, T. Lampe, N. Siegel, N. Heess, and M. Riedmiller. Compositional transfer in hierarchical reinforcement learning. 2019.

[33] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4754–4765. 2018.

[34] K. Gregor, D. Jimenez Rezende, and D. Wierstra. Variational intrinsic control. *arXiv*, pages arXiv–1611, 2016.

[35] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[36] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[37] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2019.

[38] A. Sharma, M. Ahn, S. Levine, V. Kumar, K. Hausman, and S. Gu. Emergent real-world robotic skills via unsupervised off-policy reinforcement learning. *arXiv preprint arXiv:2004.12974*, 2020.

A Supplementary Material for Submission: Towards General and Autonomous Learning of Core Skills - A Case Study in Locomotion

A.1 Method Details and Hyper Parameters

As defined in [6], the problem of Reinforcement Learning (RL) in a Markov Decision Process (MDP) is considered. Let $\mathbf{s} \in \mathbb{R}^S$ be the state of the agent in the MDP $\mathcal{M}$, $\mathbf{a} \in \mathbb{R}^A$ the continuous action vector and $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ the probability density of transitioning to state $\mathbf{s}_{t+1}$ when executing action $\mathbf{a}_t$ in $\mathbf{s}_t$. All actions are assumed to be sampled from a policy distribution $\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})$, with parameters $\boldsymbol{\theta}$. With these definitions in place, we can define the goal of Reinforcement Learning as maximizing the sum of discounted rewards $\mathbb{E}_\pi[R(\tau_{0:\infty})] = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \mid a_t \sim \pi(\cdot|\mathbf{s}_t), \mathbf{s}_{t+1} \sim p(\cdot|\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_0 \sim p(\mathbf{s})]$, where $p(\mathbf{s})$ denotes the state visitation distribution, and we use the short notation $\tau_{t:\infty} = \{(\mathbf{s}_t, \mathbf{a}_t), \dots\}$ to refer to the trajectory starting in state t.
The main idea of the multi-task RL setting in Scheduled Auxiliary Control (SAC-X) [6] is, that we have a main MDP $\mathcal{M}$ and a set of auxiliary MDPs $\mathcal{A} = \{\mathcal{A}_1, \dots, \mathcal{A}_K\}$. These MDPs share the state, observation and action space as well as the transition dynamics, but have separate reward functions $r_{\mathcal{A}_1}(\mathbf{s}, \mathbf{a}), \dots, r_{\mathcal{A}_K}(\mathbf{s}, \mathbf{a}), r_{\mathcal{M}}(\mathbf{s}, \mathbf{a})$. After executing an action – and transitioning in the environment – the agent now receives a scalar reward of all the auxiliary rewards and the main reward.
Given the set of reward functions we can define intention policies and their return as $\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}, \mathcal{T})$ and

$$\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s},\mathcal{T})}\Big[R_{\mathcal{T}}(\tau_{t:\infty})\Big] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s},\mathcal{T})}\Big[\sum_{t=0}^{\infty} \gamma^t r_{\mathcal{T}}(\mathbf{s}_t, \mathbf{a}_t)\Big], \tag{1}$$

where $\mathcal{T} \in \mathcal{T} = \mathcal{A} \cup \{\mathcal{M}\}$, respectively.
Optimization of the policy is achieved by using an off-policy, model free RL approach, by trying to find an optimal multi-task value function $Q_{\mathcal{T}}(\mathbf{s}_t, \mathbf{a}_t)$ for task $\mathcal{T}$ as

$$Q_{\mathcal{T}}(\mathbf{s}_t, \mathbf{a}_t) = r_{\mathcal{T}}(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\pi_{\mathcal{T}}}\Big[R_{\mathcal{T}}(\tau_{t+1:\infty})\Big], \tag{2}$$

with $\pi_{\mathcal{T}} = \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{x}, \mathcal{T})$. Leading to the the (joint) policy improvement objective as finding $\arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ where θ is the collection of all intention parameters and,

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}; \mathcal{M}) + \sum_{k=1}^{|\mathcal{A}|} \mathcal{L}(\boldsymbol{\theta}; \mathcal{A}_k), \tag{3}$$

$$\text{with } \mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = \sum_{\mathcal{B} \in \mathcal{T}} \mathop{\mathbb{E}}_{p(s|\mathcal{B})}\Big[Q_{\mathcal{T}}(\mathbf{s}, \mathbf{a}) \mid \mathbf{a} \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}, \mathcal{T})\Big]. \tag{4}$$

To optimize the objective a gradient based approach is used. Using a parameterized predictor $\hat{Q}^\pi_{\mathcal{T}}(\mathbf{s}, \mathbf{a}; \phi)$ (with parameters ϕ) of state-action values; i.e. $\hat{Q}^\pi_{\mathcal{T}}(\mathbf{s}, \mathbf{a}; \phi) \approx Q^\pi_{\mathcal{T}}(\mathbf{s}, \mathbf{a})$ and a replay buffer B containing trajectories τ gathered from all policies, the policy parameters $\boldsymbol{\theta}$ can be updated by following the gradient

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \approx \sum_{\substack{\mathcal{T} \in \mathcal{T} \\ \tau \sim B}} \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\substack{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_t, \mathcal{T}) \\ \mathbf{s}_t \in \tau}} \Big[\hat{Q}^\pi_{\mathcal{T}}(\mathbf{s}_t, \mathbf{a}; \phi) - \alpha \log \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}_t, \mathcal{T})\Big], \tag{5}$$

where $\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_t, \mathcal{T})}[-\log \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}_t, \mathcal{T})]$ corresponds to an additional (per time-step) entropy regularization term (with weighting parameter α).
The second step in [6] is to find an optimal schedule during training that allows to learn the main task $\mathcal{M}$ in a data-efficient way by executing the auxiliaries to collect appropriate data and help with exploration. To achieve this, the scheduler divides an episode in a number of subsequent sequences and decides which intention is executed in a certain sequence. In [6] two schedulers are proposed, a pure uniform random scheduler, called SAC-U, and an optimizing scheduler SAC-Q.

To recap, we can apply the approach from [6] in three different ways:

- In a single-task setting e.g. $\mathcal{T} = \mathcal{A} = \{\mathcal{A}_{\textit{WalkForward}}\}$, where the approach simply reduces to an off-policy RL experiment. This is used in the first set of experiments to show the properties of the skill rewards in section (4.1 and 4.4).

- In a multi-task setting with a set of locomotion skills, where we show that we can learn a set of auxiliaries in parallel, e.g. $\mathcal{T} = \{\mathcal{A}_{\textit{WalkForward}}, \mathcal{A}_{\textit{StandUpright}}, \mathcal{A}_{\textit{WalkBackward}}\}$ but without using a main task and the random uniform scheduler (see section 4.2).
- Or in the full setting in section 4.3, where we have the complex and sparse task *ReachTarget* and a set of auxiliaries that will help to learn it e.g. $\mathcal{T} = \{\mathcal{A}_{\textit{WalkForward}}, \mathcal{A}_{\textit{StandUpright}}, \mathcal{A}_{\textit{WalkBackward}}\} \cup \{\mathcal{M}_{\textit{ReachTarget}}\}$, using the full SAC-Q scheduler setup in [6].

We use the same hyper parameters for all experiments. Following [6] the stochastic policy consists of a layer of 256 hidden units with an ELU activation function, that is shared across all intentions. After this first layer a layer norm is placed to normalize activations. The layer norm output is fed to a second shared layer with 256 ELU units. The output of this shared stack is routed to a head network for each of the intentions. The heads are built from a layer of 100 ELU units followed by another layer of ELU units and a final tanh activation with twice the number of action dimension outputs, that determine the parameters for a normal distributed policy (whose variance we allow to vary between 0.3 and 1 by transforming the corresponding tanh output accordingly). For the critic we use the same architecture, but with 400 units per layer in the shared part and a 300-1 head for each intention. Training of both policy and Q-functions was performed via using a learning rate of $2 \cdot 10^{-4}$ (and default parameters otherwise), a discount factor of 0.99 and a replay buffer size of four million.
For each of the simulation experiments the agent interacts with the simulated environment on episodes with a data rate that makes it comparable to experiments on a single real robot (single actor). For all experiments we run two sequences of 400 steps with a step duration (of the simulated physics) of 25 milliseconds. This gives us 20 seconds of simulated interaction in each of the episodes overall. In each episode we measure the accumulated intention reward over the first sequence for the executed intention policy. To measure the performance, we average the accumulated intention reward for the last 10 episodes for which that specific intention was active in the first sequence. In this way we measure the performance from the set of starting states. For each task, we report the average number of episodes we need to have this performance measure exceed a threshold (or convergence, whatever happens first) over three independent seeds. To be able to compare a single reward definition over different robot platforms, we use the same task specific threshold for all platforms. The threshold is chosen so that we see a minimal expected behaviour (average speed of 0.1 m/s for the walk tasks, 0.05 rad/s for the turn task, average height of 1 cm for the feet) without exceeding a roll or pitch angle of ± 0.4 radians. We use the same procedure to report the episodes for the real robot experiments, but we run only one experiment (not several seeds) for each experiment in the real world. It is also important to note, that if we report a certain number of episodes for the multi-task experiments, we report all episodes the agent interacted with environment to learn all the tasks from scratch (not per task).

A.2 Reward Details

We assume that all robots have access to an IMU which allows them to estimate the roll and pitch angles of the robot w.r.t. gravity. In contrast to the roll and pitch angles, which can be reliably estimated from accelerometer and gyroscope data, the absolute yaw angle of the robot is typically estimated based on the earth's magnetic field, which especially indoors is often disturbed by other electromagnetic devices (including the robot's motors themselves) and hence unreliable. However this does not represent a problem since basic locomotion skill should be invariant w.r.t. to the yaw angle.
Using these measurement, this allows us to work in a virtual reference coordinate system F_H that is simultaneously aligned with the robots forward direction and gravity. Hence F_H has the same origin as the torso coordinate frame F_T, has a x-y plane parallel to the worlds x-y plane, and no yaw component w.r.t. F_T. This reference frame F_H allows simple computation of different rewards that can be used over a broad range of different walker topologies. Drawing on the forward kinematics of the walker, we represent each foot of the walker as a set of reference points (1 point for spherical feet, 8 corner points for plate feet). Using the IMU, joint angles and forward kinematics, we can then compute the position of these reference points j in the frame F_H in each time step and for each foot i: $f_{ij}^H(t)$.
We reduce this to a single reference point for each foot i by taking the reference point with the smallest z coordinate: $f_i^H(t) = f_{ij}^H(t)$ with $j = argmin_j(f_{ij}^H(t) \cdot (0, 0, 1))$. We can also define a

translational velocity of the feet reference points as $\delta f_i^H(t) = \frac{f_i^H(t) - f_i^H(t-dt)}{dt}$, where we neglect a small change in yaw between the consecutive coordinate frames.

To make use of these quantities we make the assumption that in each time step the robot is in contact with the ground and that the contact point is close to the lowest reference point. Using this assumption we can make an estimate of the translational torso velocity relative to the world as $\delta T = -\delta f_a^H(t)$ with $a = argmin_i(f_i^H(t) \cdot (0,0,1))$.

A.2.1 *StandUpright*

We first define a reward function that encourages the robot to stay upright and not to fall or lean the torso in any direction. Using our proprioceptive definitions and measurements, we first define a reward term to keep roll and pitch angle small. Given roll angle $\phi(t)$ and pitch angle $\theta(t)$ we define this reward as:

$$r_{up}(t) = 1 - c_{prec}(\sqrt{\phi(t)^2 + \theta(t)^2}, 0.0, 0.4) \tag{6}$$

Given a general precision cost function:

$$\begin{aligned} c_{prec}(v,t,m) &= \tanh |(v-t) * w|^2 \\ w &= \tfrac{atanh(\sqrt{0.95})}{m} \end{aligned} \tag{7}$$

In addition we want to punish movements of the torso relative to the ground. Assuming that we can estimate the torso velocity relative to the ground in the x and y axis of F_H as v_{xy} (taken directly from δT), we have:

$$r_{still}(t) = -|v_{xy}(t)| \tag{8}$$

As a last component of the reward, we want to prevent the torso from rotating. Assuming that we can measure the torso rotation rate directly from the gyroscope as $g_z(t)$, we can formulate this reward component as a negative thresholding of another reward r:

$$r_{rot}(t,r) = min(k * r, r) \tag{9}$$

$$\text{with } k = 1.0 - c_{precise}(\hat{g}_Z(t), 0.0, 0.5) \tag{10}$$

Given these definitions, we can now define the *StandUpright* reward as:

$$r_{\textit{StandUpright}}(t) = r_{rot}(t, r_{still}(t) + r_{up}(t)) \tag{11}$$

A.2.2 *Turn*

For the turn task we expect the robot to rotate as fast as possible around the z axis of the torso while being upright. Using the already given reward terms, we directly increase the gyroscope value $g_z(t)$ (instead of punishing, as we did in the *StandUpright* reward) while still keeping the torso levelled.

$$r_{\textit{Turn}}(t, dir) = dir * g_z(t) + 0.1 * r_{up}(t) \tag{12}$$

$$\tag{13}$$

In this investigation we consider turning left and right:

$$r_{\textit{TurnLeft}}(t) = r_{\textit{Turn}}(t, 1.0) \tag{14}$$

$$r_{\textit{TurnRight}}(t) = r_{\textit{Turn}}(t, -1.0) \tag{15}$$

$$\tag{16}$$

A.2.3 *LiftFoot*

For the task of lifting a certain foot i, *LiftFoot*$_i$, we define a reward, $r_{\textit{LiftFoot}}(t,i)$, that tries to stand still while lifting a certain foot i over a threshold of 5 cm. We use the definitions from before and add an r_{lift} incentive:

$$r_{\textit{LiftFoot}}(t,i) = r_{rot}(t, r_{lift}(t,i) + 0.1 * r_{still} + 0.1 * r_{up}(t)) \tag{17}$$

with r_{lift} being a bounded shaped reward of the height of the foot relative to the stand leg:

$$r_{lift}(t,i) = min(1,h) \tag{18}$$

$$h = (f_i^H(t) \cdot (0,0,1)) - (f_a^H(t) \cdot (0,0,1)) \tag{19}$$

$$\text{a being stand leg id} \tag{20}$$

A.2.4 *Walk*

Finally we define a reward for moving in a certain direction $\hat{v}_{xy}$ (with $\|\hat{v}_{xy}\| = 1$), relative to the x-y-plane of F_H. To compute the full reward for robust locomotion only based on robo-centric measurements, we define a reward term for moving the torso in the desired direction: $r_{torso}(t, \hat{v}_{xy})$.

$$r_{torso}(t, \hat{v}_{xy}) = r_{rot}(t, \hat{v}_{xy} \cdot v_{xy}) \tag{21}$$

As we saw in our experiments in simulation and in the real word, adding another incentive to move legs in the same direction helps to increase robustness and data efficiency. We define a foot swing velocity v^i_{swing} in the frame F_H as:

$$v^i_{swing} = \delta f^H_i(t) + \delta T \tag{22}$$

If we neglect the z coordinate, we can now also define an incentive to move the feet forward:

$$r_{feet}(t, \hat{v}_{xy}) = r_{rot}(t, \frac{1}{|i|} \sum \hat{v}_{xy} \cdot v^i_{swing}) \tag{23}$$

This will cause a small incentive to move feet forward. For all leg in contact with the ground, this will have neither a positive or negative reward. For legs moving with the torso the rewards grows.

Finally the reward for walking is defined by:

$$r_{\textit{Walk}}(t, \hat{v}_{xy}) = r_{torso}(t, \hat{v}_{xy}) + 0.5 * r_{feet}(t, \hat{v}_{xy}) + 0.1 * r_{up}(t) \tag{24}$$

In this investigation we use 4 instances of this reward:

$$r_{\textit{WalkForward}}(t) = r_{\textit{Walk}}(t, (1, 0)) \tag{25}$$
$$r_{\textit{WalkBackward}}(t) = r_{\textit{Walk}}(t, (-1, 0)) \tag{26}$$
$$r_{\textit{WalkRight}}(t) = r_{\textit{Walk}}(t, (0, -1)) \tag{27}$$
$$r_{\textit{WalkLeft}}(t) = r_{\textit{Walk}}(t, (0, 1)) \tag{28}$$
$$\tag{29}$$

A.3 Action and Observation Details

As stated in the main paper, we use the position control mode of the HEBI actuation modules. For convenience, the agent action is constrained to the range $d \in [\delta_{min}, \delta_{max}]$ and transformed to actuator position command $\hat{p}$ by adding an initial position α: $\hat{p} = \alpha + d$ for each of the actuators.

Action	Unit	dim	range
position set point	[rad]	1	$[\alpha + \delta_{min}, \alpha + \delta_{max}]$

Table 3: Action space for each actuator.

In table 4 we summarize the observations that are used for each of the HEBI actuation modules. The raw values are sent with 400 Hz over a ROS node running on the robot, while the filter state of the set-point smoothing window filter (of length $\nu = 5$ steps) is stored in the agent. The commanded action is computed by updating the filter state with the agent action and communicating the mean value over the last ν steps to the actuation module.

Each actuation module provides a filtered orientation estimate, as well as acceleration and gyroscope readings, based on it's own IMU. We use a simple kinematics equation for each of the creatures to compute these values for the torso, based on the estimates of all modules directly attached to it. While already the estimate from a single modules would be sufficient, we can use multiple modules to make it more robust.

For the observation vector of the robot, we use a history of $h = 2$ time steps of the roll and pitch angle estimates to capture also the first derivative of these values in the observation. These values are all independent of the yaw angle, that is also computed by all modules. As the yaw angle has typically not a reliable absolute reference if we only take internal measurements of the robot, we ignore it for the agent observations as well as for the reward calculations. To capture the rotational

Observation	Unit	dim
position	[rad]	1
velocity	[rad/s]	1
elastic element deflection	[rad]	1
elastic element deflection velocity	[rad/s]	1
winding temperature	[°C]	1
housing temperature	[°C]	1
filter state	[rad]	$\nu = 5$

Table 4: Observations for each actuator module.

Observation	Unit	dim
torso roll angle estimate	[rad]	h x 1
torso pitch angle estimate	[rad]	h x 1
feet relative positions estimate	[m]	h x n x $\|j\|$ x 3
torso gyro values	[rad/s]	h x 3

Table 5: Additional robot observations.

velocities of the torso, we have access to the gyroscope readings of the fused IMUs. As we only use robo-centric measurements, we also provide the feet points in the reference frame: $f_{ij}^{H}(t)$ (assuming n feet with $|j|$ reference points each).

Table 6 summarizes the range of different creatures with their respective observation and action dimensions that we use in this work. For the foot reference points, we use a single point in the center of the sphere-like foot and eight points on the corners of the plate-like foot.

Creature	description	act dim	obs dim
Daisy6	hexapod, 6 legs	18	244
Daisy4	quadruped, 4 legs	12	166
Dog	quadruped, 4 legs	12	166
Daisy3	tripod, 3 legs	9	127
Florence	biped, 2 legs	12	238
Flori	biped, 2 legs	12	238
FloriArms	biped, 2 legs	16	282

Table 6: Basic creatures with respective action and observation space ($h = 2$).

A.4 Ablation: Robustness of Reward Definitions

As described in section 3.1 our general reward scheme makes some basic assumptions that may appear to be pretty strict. For example, given the lack of contact detection, we assume that the vertically lowest foot is always in contact with the ground. Another assumption is that this contact point does not slip on the ground. In essence we use these assumptions to give the reward some semantics, while we do not expect them to be fulfilled at each point in time. The central idea is that we also expect our method to still create useful locomotion behaviours if these assumptions are not fulfilled in each time step.

A.4.1 Uneven Terrain

Walking over rough or cluttered terrain is a challenging and important task for locomotion. Especially using only internal sensors and without any additional sensors like cameras, LIDAR or sensorized feet (e.g. with contact sensors). We created an environment with pedestals that will have a random height drawn uniformly between 0 and h_{max} in each episode (see Figure 2a), where the assumption that the lowest foot is always in contact with the ground will inevitably be violated.

The creature starts in the middle of the arena and has to solve the task *WalkForward*, which it can only do by crawling over the pedestals. Using the same reward definitions compared to the flat terrain experiments, the bipeds Florence and Flori are able to handle height differences of about $h_{max} = 1cm$. As one can expect having more legs allows more robust behaviours. Daisy4 can handle height differences of $h_{max} = 4cm$ and gets stuck afterwards (mostly with it's hind feet). Having even more legs helps not only by allowing for simple statically stable gaits, it also help in terms of redundant sensor information. Consequently, Daisy6 shows an even better performance and can handle up to $h_{max} = 20cm$. The learned gate shows an interesting pattern that looks like it would "feel" it's way in the blind.

A.4.2 Hybrid Locomotion

As an additional ablation to show the versatility of our approach, we changed the topology of creatures in our zoo to be even more dynamic. As shown in Figure 2b we replaced the foot plates of the bipeds Florence and Flori with passive skates (similar to inline skates). To allow for the very same rewards and observations as before, we put 4 foot reference points on each of the wheels outer diameter. As each foot has 2 wheels, we have a total of 8 reference points that now rotate with the passive wheels. When we do this, we can run the very same setting as in the previous experiment and do the same computations. The only difference is that we have to measure the passive wheel velocities to compute the location of the reference points. To have a better comparability we don't put these in the observations and only use the reference points as described above. We can learn *WalkForward*, *WalkBackward* and *StandUpright* with a comparable number of interactions for the bipeds in the single and multi-task setting, while the motion of the robot looks completely different. It learns a very dynamic skating behaviour, to stop and turn, just using the simple rewards we used in all of the experiments.