2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Slope Handling for Quadruped Robots Using Deep Reinforcement Learning and Toe Trajectory Planning*

Athanasios S. Mastrogeorgiou, Yehia S. Elbahrawy, Andrés Kecskeméthy, and Evangelos G. Papadopoulos, *Fellow, IEEE*

***Abstract*— Quadrupedal locomotion skills are challenging to develop. In recent years, deep Reinforcement Learning promises to automate the development of locomotion controllers and map sensory observations to low-level actions. Moreover, the full robot dynamics model can be exploited, but no model-based simplifications are to be made. In this work, a method for developing controllers for the Laelaps II robot is presented and applied to motions on slopes up to 15°. Combining deep reinforcement learning with trajectory planning at the toe level, reduces complexity and training time. The proposed control scheme is extensively tested in a Gazebo environment similar to the treadmill-robot environment at the Control Systems Lab of NTUA. The learned policies produced promising results.**

I. Introduction

Recently, there has been an increased interest in legged robots. Legged systems continuously interact with their surroundings through multiple permanently changing contact points. Such systems can traverse various terrain types, or handle terrain discontinuities with the use of accurate foot placement making them more versatile than wheeled robots. Yet, quadrupeds have complex dynamics and many degrees of freedom that must be well orchestrated for achieving a robust and dynamically stable locomotion pattern. Dealing with such high-dimensional, non-linear, and underactuated system is a long-standing research challenge.

In most cases, state-of-the-art model-based control approaches require an accurate dynamics model of the robot and include state estimation to contact scheduling, trajectory optimization, and foot placement planning [1, 2, 3, 4]. In contrast, data-driven methods, such as model-free deep Reinforcement Learning (deep RL), already have produced promising results showing that they can overcome the limitations of prior model-based approaches by learning effective controllers directly from experience. Deep RL attempts to automate the development of locomotion controllers and map sensory inputs directly to low-level actions [5]. The main disadvantage that these methods suffer from is the so-called reality gap, when trying to apply the learned policy in a real robot. There are two general approaches for overcoming the reality-gap: either to improve the simulation accuracy as much as possible or employ parameter identification. Very recent promising research results in the field of legged robots demonstrated that learned locomotion policies could be transferred from simulation to reality [6, 7, 8]. To realize this transfer, it was important to use high-fidelity simulations. This was achieved by learning parts of the simulated model from real data [6], or by model parameter estimation [7]. Model-free methods have been applied to bipeds like Cassie from Agility Robotics [9], without the need for model-based simplifications commonly used to realize control policies.

This paper presents a framework for learning trotting controllers on sloped terrain employing a realistic 3D model of the Laelaps II quadruped developed by CSL [10] (Figure 1).

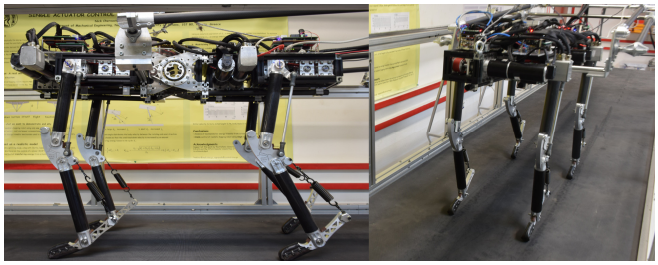


Figure 1. The quadruped robot Laelaps II, built by the Legged Robotics Team at the Control Systems Lab of NTUA, on the lab's treadmill.

The focus of this work is to study whether it is possible to develop a controller using deep RL enabling the Laelaps II quadruped to handle positive and negative slopes, starting from trajectory planning at the toe level and open loop stability. The Laelaps II is an appropriate testbed to evaluate the developed algorithm, since it cannot perform hip abduction which would help to stabilize the robot in cases of increasing body roll/yaw angles.

The proposed controller displayed robustness in environment uncertainties and managed to produce stable gaits, i.e.: bounded body pitch/roll angles. Also, when the quadruped is trotting on slopes up to 15°, it does not drift away from its goal, i.e. its body yaw angles are bounded. The control scheme mainly consists of two parts, of the applied deep RL algorithm, and of the toe level trajectory planning part. The performance of this controller is extensively tested using an accurate Laelaps II robot model in Gazebo, exploiting the full dynamics of the robot.

The paper consists of five sections. Section II presents an overview of the simulation environment and of a detailed Laelaps II 3D model in Gazebo. In Section III, the control architectures including the deep RL algorithm and the semi-elliptic trajectory planner are described. In the last two sec-

* This work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant" (Project Number: 2182, ARGOS).

A. S. Mastrogeorgiou (e-mail: amast@central.ntua.gr), and E. G. Papadopoulos (egpapado@central.ntua.gr) are with the Department of Mechanical Engineering, National Technical University of Athens, 15780 Athens, Greece. Y. S. Elbahrawy and A. Kecskeméthy, are with the Faculty of Engineering, University of Duisburg-Essen, Duisburg, Germany.

tions, successful experiments in the Gazebo environment are presented, followed by a discussion and the future work.

II. Simulation Environment

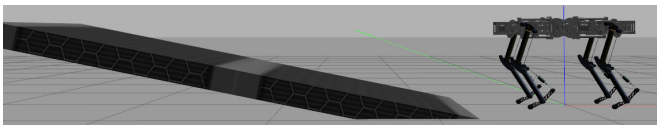
Having the goal of transferring the learned policies to a quadruped, a more convenient approach is to use the same development frameworks commonly used by researchers in legged robotics and deep RL. As a result, tools such as the Gazebo simulator [11], the Gym framework [12] and Robot Operating System (ROS) [13] were employed. In this framework different robot models (described in SDF format [14]) can be loaded and with the appropriate adjustments, can be trained using state-of-the-art deep RL algorithms.

A. Laelaps II

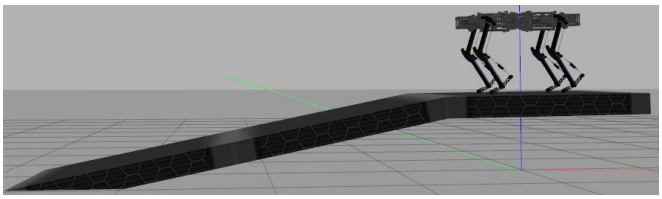
Laelaps II is a quadruped robot built by the Legged Robotics Team at the Control Systems Lab of NTUA [10]. The robot parameters are presented in Table I. The actuation system of each leg comprises a RE50 Maxon motor for the hip and an EC45 Maxon motor for the knee. Both are equipped with gearboxes and belt-pulley transmissions. Since the knee motor is body-mounted, a parallel mechanism is used to drive the distal leg segment (tibia). The maximum torque/angular rate capabilities of the Laelaps II leg are 50 Nm/55 rpm for the hip, and 50 Nm/75 rpm for the knee; exceeding these limits will cause damage to the gearboxes, thus the gearboxes are responsible for the torque/angular rate limitations.

B. Laelaps II in Gazebo

For the needs of this work, a simulation environment was set up in Gazebo. Every parameter of the Laelaps II robot presented in Table I & Figure 3 is used in the Gazebo model in order to build an accurate simulation model/environment. This environment consists of the Laelaps robot, of level terrain and of ramps with inclinations of $\pm 10^{\circ}$, see Figure 2.



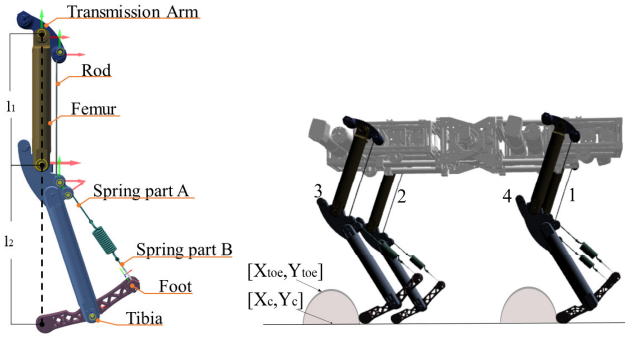
(a)



(b)

Figure 2. Simulation Environment of Laelaps II Quadruped in Gazebo. (a) Positive and (b) negative slopes are used.

For the ground contacts, the coefficient of restitution used by the Open Dynamics Engines was utilized. Concerning the robot model, the original CAD files of the robot's legs and body were used and added in the XACRO description [15], see Figure 3. With xacro, it is possible to construct shorter and more readable XML files using macros that expand to larger XML expressions. All robot parts were extracted as STL or DAE files from the Solidworks CAD design and imported in the xacro description. The mass properties for each part shown with different color in (mass, inertia matrix, CoM, etc.) were used also. The robot description xacro files as well as the Gazebo environment set-up can be found at [16]. The tendon like part of the leg was modeled as a prismatic joint with a spring constant equal to the springs used in Laelaps II robot legs, i.e.: 26,480 N/m. Regarding the max torque and angular rate values, the calculated outputs of the actuation system after the gearbox and the belt-pulley transmission system were used.



(a) (b)

Figure 3. (a) Detailed model of the Laelaps II leg in Gazebo. CAD files were used, and all mass properties were added in the SDF description, (b) Laelaps II Robot in Gazebo. The toes follow semi-elliptical trajectories. Deep RL produces the center of the semi-elliptical trajectories and the trajectory planner produces the exact toe position.

TABLE I. Laelaps II parameters modelled in Gazebo.

Parameter	Value
Body mass	40 kg
Hip to hip distance 1 (body length)	0.6 m
Hip to hip distance 2 (body width)	0.4 m
Body inertia matrix	$\{I_{XX}=0.87, I_{YY}=1.23, I_{ZZ}=2.03\}$ kg·m^2
Femur inertia matrix	$\{I_{XX}= 0.0030216, I_{XY}=0.0, I_{XZ}= 0.0, I_{YY}=0.000606, I_{YZ}= 0.0000508, I_{ZZ}= 0.0030174\}$ kg·m^2
Tibia inertia matrix	$\{I_{XX}= 0.0069971, I_{XY}= -0.0025079, I_{XZ}= -0.0000413, I_{YY}= 0.00113, I_{YZ}= 0.0001357, I_{ZZ}= 0.0080248\}$ kg·m^2
Foot inertia matrix	$\{I_{XX}= 0.000039, I_{XY}= 0.000114, I_{XZ}= 0.000014, I_{YY}= 0.000946, I_{YZ}= 0.000003\ I_{ZZ}= 0.000974\}$ kg·m^2
Transmission arm inertia matrix	$\{I_{XX}=0.000238, I_{XY}=-0.000143, I_{XZ}= 0, I_{YY}=0.000238, I_{YZ}=0\ I_{ZZ}=0.000326\}$ kg·m^2
Spring stiffness	26480 N/m
Max hip torque	50 Nm
Max knee torque	50 Nm
Hip reduction ratio	97.8462
Knee reduction ratio	79.3846
Max hip angular rate	55 rpm
Max knee rate	75 rpm

III. Controller Architecture

The developed control architecture is presented in Figure 4.

First, the deep RL algorithm performs foot placement planning by choosing the center of the semi-elliptical trajectories to be applied, see Figure 3. This is passed to the trajectory planner that produces the exact toe position for every leg. The diagonal legs are phase shifted by φ. Using the toe position as input and the inverse kinematics of the Laelaps leg as presented in Subsection D, the angles for the hip and knee joints are calculated. In turn, they are passed to a PID controller to produce the motor torques to be applied in the simulation environment. The position control model eases the performance and speed of the deep RL algorithm [17].

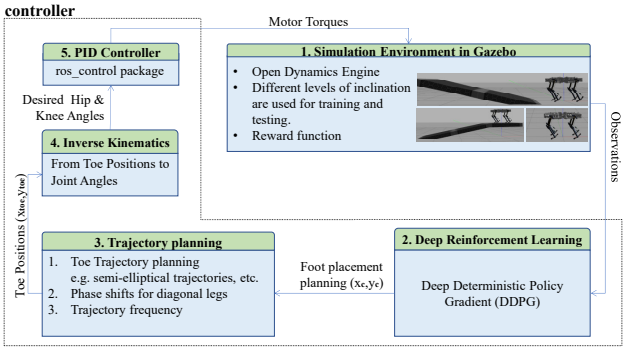


Figure 4. Architecture of the control algorithm.

A. Proposed Reinforcement Learning Scheme

The RL problem is formulated as a Markov Decision Process (MDP) that is described for each time step t with a tuple (s,a,p,r,γ), where s is the current robot state, a is the action applied, p is the transition probability function from the current state s_t to the next state s_{t+1}, r is a reward value obtained due to the transition, and $\gamma \in [0,1]$ is a discount factor for the long term reward [18].

The robot starts by exploring a stochastic environment to find an optimal behavior and increase cumulative reward values over subsequent timestamps t throughout the robot trajectory [18]. Here, the problem is defined in an episodic setting, where the agent follows a trajectory of simulation steps until a predefined termination criterion is reached. The accumulated reward of one episode with m time steps is defined as [18]

$$\sum_{t=0}^{m} \gamma^t r_{a_t}(s_t, s_{t+1}) \tag{1}$$

where r_{a_t} is a reward function under action a.This function is defined so as to promote robot forward motion and punish divergence from its goal (e.g.: climb up a ramp)

$$r_{a_t} = w_f(x_t - x_{t-1}) - w_d(y_t - y_{t-1}), \tag{2}$$

where w_f and w_d are the weights for the forward and drifting terms respectively. Since Laelaps II has a forward motion due to the planned semi-elliptical trajectory, the forward weight is tuned to $w_f = 1$ and the drift weight to $w_d = 2$. Therefore, the reward function focuses on penalizing drifting motions.

The control algorithm receives from Gazebo proprioceptive sensory information s regarding the Laelaps II body, i.e. roll/pitch/yaw angles and angular rates, building a compact observation space. Reducing the observation space has a significant influence in simplifying the reinforcement learning problem [19].

For consistency, all sensor measurements were taken from the same ROS topic, i.e.: */Gazebo/model_states*. It was cross-checked that the measurements from this topic compared to the measurements from an equivalent virtual sensor that Gazebo provides (e.g. IMU) are the equal. Given the roll/pitch/yaw angles and angular rates as input, the algorithm outputs a vector of actions $[x_{C_1}, y_{C_1}, .., x_{C_4}, y_{C_4}]^T$, with (x_c, y_c) the center of the semi-elliptical trajectory (Figure 3) w.r.t. the reference frame fixed at the hip of each leg (c1, c2, c3, c4), where: $x_c \in [-0.1, 0.1]$ and $y_c \in [-0.55, -0.50]$.

To solve the MDP problem, the robot must find a policy μ that maximizes (1). With an optimal policy learned, the agent discovers the interconnection between states, actions, and rewards [18]. In the implemented algorithm, the toe positions (x_{toe}, y_{toe}) are chosen individually for every leg as the robot climbs up/down the ramp. This means that the algorithm is adjusting the CoM height accordingly for slope climbing, it is stabilizing the robot (bounding roll and pitch angles) and it is keeping the robot moving straight (bounding yaw angle). Alternatively, training at the joint level could be performed but this could easily lead to invalid configurations, singularities etc. Defining the toe workspace that the algorithm could choose actions from, guarantees that no invalid actions would be applied to the robot.

B. Deep Deterministic Policy Gradient (DDPG)

The DDPG is a model-free algorithm that requires sensor reading from the environment and maps the current state, s, using a deterministic policy $\mu(s|\theta^\mu): s \rightarrow a$ to a single predicted action, a. The algorithm adjusts the parameter θ towards the direction of the policy gradient, that should increase with better predicted actions corresponding to better total reward values [20].

Deterministic policy algorithms are computationally simpler compared to algorithms with stochastic policy, which output a probability function of all possible actions given a state. Furthermore, their gradients can be approximated efficiently, as the estimation of the gradient involves integration over the state space instead of over a combination of state-action spaces. Therefore, estimation of the stochastic policy gradient requires more data samples [18, 21].

As proposed in [20, 21], the DDPG algorithm is based on actor-critic deep neural network architecture shown in Figure 5. The actor network uses the target policy $\mu(s|\theta^\mu)$ to predict the actions for a given state, while the critic network estimates the action-value function Q, which is the total discounted reward of an action applied on the given state. In other words, the Q value gives an insight for the quality of the action applied.

For training stability, a target network for the actor and critic is used. The target critic network estimates Q_{target} of the next state using the actions obtained from the target actor

network. The Bellman equation using the estimated Q_{target} is defined as [20]:

$$y_t = r(s_t, a_t) + \gamma Q_{\text{target}}\left((s_{t+1}, a_{t+1}) \,|\, \theta^Q\right). \quad (3)$$

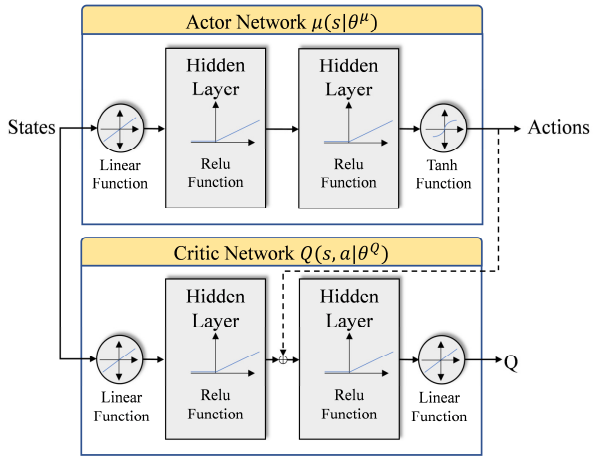


Figure 5. Actor-Critic DNN architecture.

The main critic network loss is determined by the Bellman error, which is a mean square error (MSE) between the main network estimated Q value and the y_t values estimated by the Bellman equation updating the critic network weights θ^Q. The main actor network loss is defined by the negative of the critic output Q-value, which can be seen as the actor network updates its weights θ^μ by backpropagation to maximize the critic output. At the end of every training step the target network weights are updated by soft update strategy that copy a small ratio of the learned updates in the main network [20].

To enhance learning, the training samples are required to be independent and identically distributed. As an off-policy algorithm, DDPG guarantees this requirement by using replay buffer, i.e.: fixed saved experiences $(s_t, a_t, r_t, s_{t+1}, d)$, where d is a Boolean variable, with true/false corresponding to episode termination [20]. Correspondingly, the neural networks learn from uncorrelated training samples using uniformly sampled mini batches from the buffer, whereas the PTAN package introduced in [22] was utilized to save the experiences and fill the buffer. However, off-policy algorithms have the disadvantage of environment exploration. Using an Ornstein-Uhlenbeck process as in [20], the exploration strategy was implemented by adding noise to the actor output actions with temporally correlated values.

C. Semi elliptical trajectories

The semi-elliptical trajectory equations with respect to the frame fixed at the hip is given by

$$x_{toe} = x_c + a \cdot \cos(\omega_{toe} t_{sim} + \varphi)$$
$$y_{toe} = \begin{cases} y_c + b \cdot \sin(\omega_{toe} t_{sim} + \varphi), \; if \; y_c < y_{toe} \\ y_c, \qquad\qquad\qquad otherwise \end{cases} \quad (4)$$

where $[x_{toe}, y_{toe}]^T$ are the coordinates of the elliptical trajectory that the toe should be placed at, $[x_c, y_c]^T$ is the ellipse center, $a = 0.06\ m$ and $b = 0.03\ m$ are the ellipse semi-axis, $\omega_{toe} = 30\ rad/s$ is the angular velocity of the toe motion along the elliptical trajectory, t_{sim} is the simulation time and φ is the phase shift between the diagonal pair legs. The phase shift variable is set to 0 rad for legs 1 and 3 and to π rad for legs 2 and 4, see Figure 3 (b). Last, if the calculated y_{toe} is greater than y_c then y_{toe} is set to be equal to y_c so that the toe follows a semi-elliptical trajectory. DDPG plans the next step by choosing the center $[x_c, y_c]^T$ of the semi-elliptical trajectories that will be applied. The center coordinates are passed to the trajectory planner that produces the exact toe position coordinates $[x_{toe}, y_{toe}]^T$, for every leg. For this part, the ros_control package is utilized. Specifically, the ros_control position controller is running at 1 kHz and produces the torques that the simulated hip and knee motors apply. The gains of the PID controller can be found at [16].

D. Inverse kinematics

Concerning the leg inverse kinematics, the two distal segments of the Laelaps II leg, are considered equivalent to a single virtual rigid segment since the connecting tendon-like spring is very stiff. As a result, given the $[x_{toe}, y_{toe}]^T$ toe position, the virtual links l_1 and l_2 see Figure 3 (a), are used to calculate the hip and knee joint angles. In turn, these angles are translated to actual motor angles that will be applied to the model based on the original leg design. As a result:

$$c = \left(x_{toe}^2 + y_{toe}^2 - l_1^2 - l_2^2\right) / 2 l_1 l_2, \quad s = \pm\sqrt{1 - c^2} \quad (5)$$

$$\theta_2 = \text{atan2}(y_{toe}, x_{toe}) - \text{atan2}(l_1 s, l_2 x_{toe} + l_1 c) \quad (6)$$

$$\theta_1 = \theta_2 + \text{atan2}(s, c) \quad (7)$$

$$\varphi_{hip} = \pi/2 + \theta_2 \quad and \quad \varphi_{knee} = \theta_1 \quad (8)$$

IV. Results

All trainings were carried out with a mid-range desktop computer equipped with Intel® i5-3470 CPU @ 3.20GHz, Nvidia GeForce GTX 950 GPU, and 12GB of DD3 RAM running Ubuntu 18.04 with kernel 5.3 and ROS Melodic Morena installed with Gazebo 9.12.

Controllers and trajectory planning that can produce stable gaits were employed [23]. When tested with the detailed 3D Laelaps II model presented earlier, they could produce forward motion, but the quadruped was diverging from its goal, i.e. increasing yaw angles were observed, Figure 6. More sophisticated solutions were needed for negative or positive slope handling. Foot placement planning should be performed to balance the robot, especially when realistic friction coefficients were used. It turned out that deep RL, i.e. DDPG in this case, can enhance the implementations presented in [23]. Comparisons with and without the DDPG layer are presented in figures 7 & 10. Even though the quadruped physically cannot perform hip abduction, it is trotting stably as presented in Figures 8 & 9.

The algorithm was trained for trotting up and down a slope of ±10°. To evaluate if this control model can learn in different tasks, for both scenarios the same control scheme

and hyperparameters were used. Concerning the DDPG architecture, the actor and critic neural networks have two hidden layers with 500 and 400 hidden neurons respectively and learning rate of 0.0001. To simplify dealing with gradients during training, two optimization steps using ADAM algorithm [24] were used for the actor and critic networks. The training samples were randomly chosen from a replay buffer of 10K transitions. The replay buffer's size was chosen to have stable behavior and avoid overfitting.

One common evaluation metric for RL performance is tracking the average episode reward. In this work, testing the learned policy periodically (every 500 steps) was additionally used. Figures 11 & 12 indicate that the agent learns both tasks in about 3 h. Specifically, it converged to a validation reward of ~3, corresponding to the ramp end. To demonstrate that the trained model is general enough, it was tested on a slope of 15° as shown in Figure 13. Laelaps reaches the top of the ramp in this case also. All tasks were accomplished with a torque limit of 50 Nm (Figures 14 & 15).

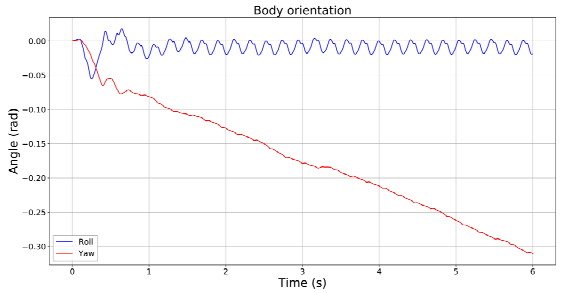


Figure 6. Application of a controller similar to [23] on the Laelaps II 3D model in Gazebo *without* the developed control scheme, and on level terrain. The body roll angle is bounded but yaw angle is increasing.

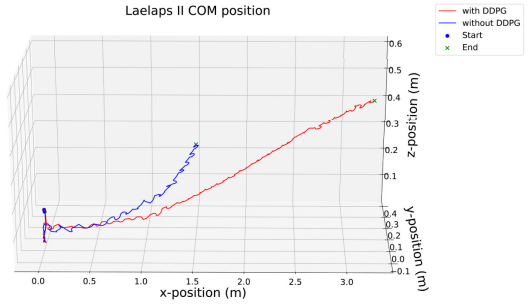


Figure 7. Laelaps CoM while trotting up a ramp of 10°. Without the DDPG layer (blue line), the robot is diverging from its goal and eventually falls. Using the trained policy (red line), the robot manages to reach the end of the 3.2 m ramp.

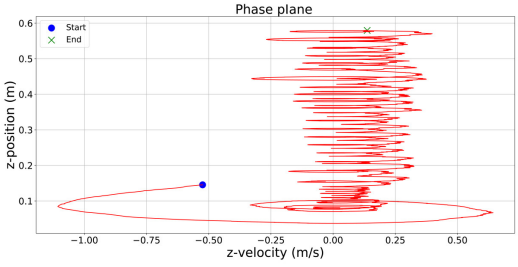


Figure 8. The Laelaps phase plane diagram at trotting up a ramp of 10°.

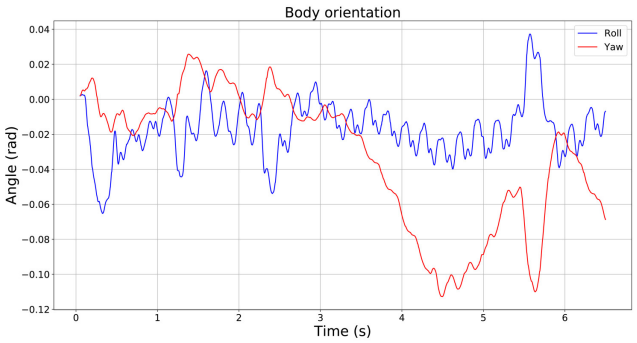


Figure 9. Roll and yaw angles of the Laelaps robot as it is trotting up a ramp of 10° *with* the developed control scheme. The values are bounded for the entire experiment.

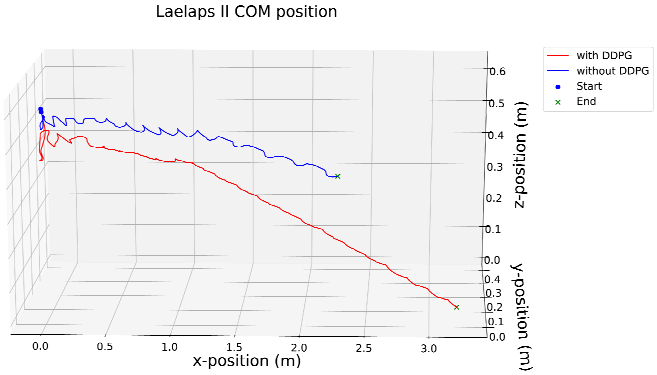


Figure 10. Laelaps CoM position while trotting down a ramp of 10°. Without the developed control scheme, (blue line), the robot is diverging from its goal and eventually falls. Using the trained policy, the robot manages to reach the end of the ramp.

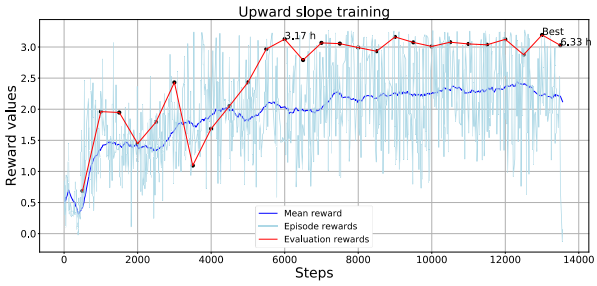


Figure 11. Training and evaluation reward values for inclination of +10°.

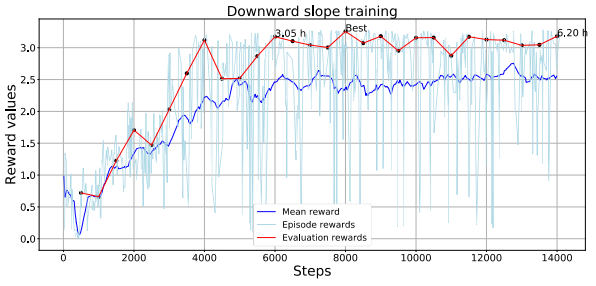


Figure 12. Training and evaluation reward values for inclination of -10°.

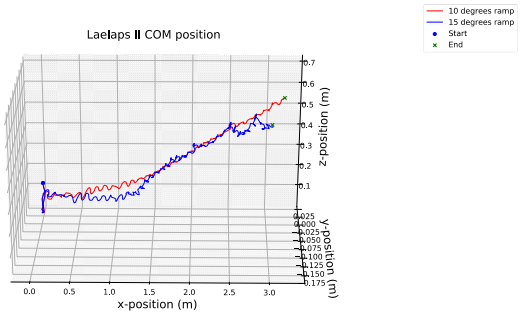


Figure 13. Testing of the learned policy on 10° (red line) and 15° (blue line). In both cases the quadruped reaches the goal. 10° ending point (x= 3.18m, y=-0.0078m, z=0.579m). 15° ending point (x= 3.01m, y=-0.17m, z=0.68m).

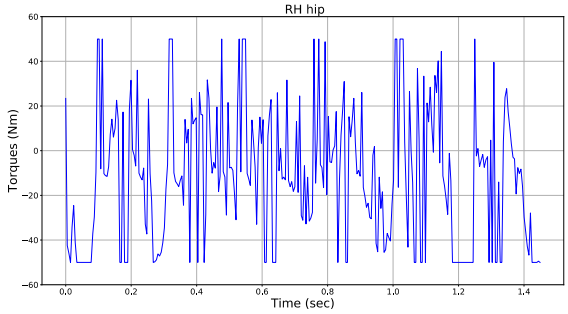


Figure 14. Right hint hip motor torques while Laelaps is trotting up a ramp of 10°. Torques are limited to 50Nm, the same limit that the real Laelaps hip motors are subject to.

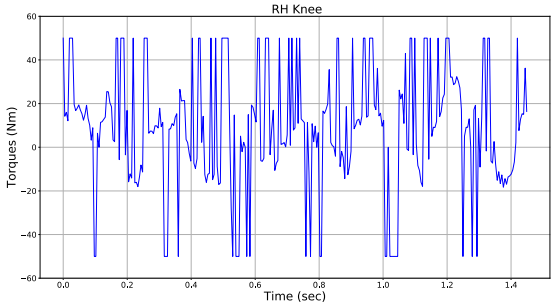


Figure 15. Right hint knee motor torques while Laelaps is trotting up a ramp of 10°. Torques are limited to 50Nm, the same limit that the real Laelaps knee motors are subject to.

V. Conclusion

A new method for developing controllers for the Laelaps II quadruped was proposed. Using a detailed model of the quadruped, its full dynamics were exploited, and no model-based simplifications were made. The training was carried out in a Gazebo simulation environment and tested in various ramp inclinations to prove the generalization of the trained policies, with coefficients of friction close to 1, similar to the foot-treadmill coefficient at CSL. Having the trajectory planning on the toe level as a starting point, the training time was significantly reduced to almost three hours on a mid-range PC. The simulation environment in Gazebo is as close as possible to the treadmill-robot setup at CSL. This will allow us to apply the method on Laelaps II, as soon as it is ready for experiments. In addition, the combination of Gazebo and Gym employed here will enable testing and comparing of various state-of-the-art deep RL algorithms in the future. Last, using the same setup, Laelaps can be trained to perform on more challenging terrains, such as mixed slopes, stairs etc.

References

[1] Taylor A., Patrick C., Kevin G., Alan F., and Jonathan W. H., "Fast online trajectory optimization for the bipedal robot cassie," *Robotics: Science and Systems (RSS),* 2018.

[2] 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," *Proc. International Conference on Intelligent Robots and Systems (IROS*), Madrid, Spain, Oct. 1-5, 2018.

[3] C. Gehring, M. Coros, S. Hutler, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, et al., "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robotics & Automation Magazine*, 23(1):34–43, 2016

[4] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, et al. "Anymal-a highly mobile and dynamic quadrupedal robot," *Proc. International Conference Intelligent Robots and Systems (IROS)*, Daejeon, Korea, October 9-14, 2016, pp. 38–44.

[5] Haarnoja, Tuomas & Zhou, Aurick & Ha, Sehoon & Tan, Jie & Tucker, George & Levine, Sergey, "Learning to Walk via Deep Reinforcement Learning," *arXiv:1812.11103*, 2018.

[6] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter, "Learning Agile and Dynamic Motor Skills for Legged Robots," *Science Robotics*, 4(26), 2019.

[7] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke, "Sim-to-Real: Learning Agile Locomotion for Quadruped Robots," *arXiv:1804.10332*, 2018.

[8] Lee, Joonho, Jemin Hwangbo and Marco Hutter. "Robust Recovery Controller for a Quadrupedal Robot using Deep Reinforcement Learning," *ArXiv abs/1901.07517*, 2019.

[9] Xie, Zhaoming & Berseth, Glen & Clary, Patrick & Hurst, Jonathan & Panne, Michiel. (2018). Feedback Control for Cassie With Deep Reinforcement Learning. 1241-1246. 10.1109/IROS.2018.8593722.

[10] http://nereus.mech.ntua.gr/legged/

[11] http://Gazebosim.org/

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv:1606.01540v1*, 2016

[13] https://www.ros.org/

[14] http://sdformat.org/

[15] https://wiki.ros.org/xacro

[16] https://github.com/mastns/Laelaps2_DDPG

[17] Peng, Xue & Panne, Michiel, "Learning locomotion skills using DeepRL: does the choice of action space matter?" *ACM* 1-13. 10.1145/3099564.3099567, 2017.

[18] Sewak, Mohit, "Deep Reinforcement Learning: Frontiers of Artificial Intelligence," 10.1007/978-981-13-8285-7, 2019.

[19] Kober, Jens & Peters, Jan, *Learning Motor Skills: From Algorithms to Robot Experiments*, 10.1007/978-3-319-03194-1, Springer 2014.

[20] Lillicrap, Timothy & Hunt, Jonathan & Pritzel, Alexander & Heess, Nicolas & Erez, Tom & Tassa, Yuval & Silver, David & Wierstra, Daan, "Continuous control with deep reinforcement learning, CoRR," *arXiv:1509.02971*, 2015.

[21] Silver, David & Lever, Guy & Heess, Nicolas & Degris, Thomas & Wierstra, Daan & Riedmiller, Martin, "Deterministic Policy Gradient Algorithms," *31st Int. Conference on Machine Learning, ICML* 2014.

[22] Maxim Lapan., Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more. Packt Publishing, 2018

[23] Machairas, K. and Papadopoulos, E., "An Active Compliance Controller for Quadruped Trotting," *24th Mediterranean Conference on Control and Automation (MED '16)*, Athens, Greece, June 21-24, 2016.

[24] Kingma, Diederik P. and Ba, Jimmy, "Adam: A Method for Stochastic Optimization", *3rd International Conference for Learning Representations*, San Diego, 2015.