

THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Learning Whole-body Motor Skills for Humanoids

Citation for published version:
Yang, C, Yuan, K, Merkt, W, Komura, T, Vijayakumar, S & Li, Z 2019, Learning Whole-body Motor Skills for Humanoids. in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids).* Institute of Electrical and Electronics Engineers (IEEE), Beijing, China, pp. 270-276, 2018 IEEE-RAS 18th International Conference on Humanoid Robots, Beijing, China, 6/11/18.
https://doi.org/10.1109/HUMANOIDS.2018.8625045

Digital Object Identifier (DOI):
10.1109/HUMANOIDS.2018.8625045

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

OPEN ACCESS

Download date: 29. Mar. 2024

Learning Whole-body Motor Skills for Humanoids

Chuanyu Yang, Kai Yuan, Wolfgang Merkt, Taku Komura, Sethu Vijayakumar, Zhibin Li

***Abstract*— This paper presents a hierarchical learning framework that can learn a wide range of push recovery and balancing behaviors, i.e., ankle, hip, foot tilting, and stepping strategies. The policy is trained in a realistic physics simulator using the robot model in a setup designed to be able to easily transfer and deploy synthesized control policies to real-world platforms. The advantage over traditional methods that integrate high-level planner and feedback control is that one single coherent policy network is generic for generating versatile, unprogrammed balancing and recovery motions against unknown perturbations at arbitrary locations (e.g., legs, torso). Furthermore, the proposed framework allows the policy to be learned with any state-of-the-art learning algorithm. By comparing the proposed approach with other methods in literature, we found the performance of learning is similar in terms of disturbance rejection with additional benefits of generating generic and versatile behaviors.**

I. Introduction

Legged robots have great potential for being deployed in environments where wheeled robots are limited, such as obstacle obstructed terrain as well as narrow and elevated surfaces (e.g., stairs). However, in contrast to wheeled or tracked robots, humanoids are intrinsically unstable and require active control to balance due to their limited support area, high center of mass, and limited actuator capabilities. Therefore, the range of possible scenarios in which humanoids can be deployed is mostly limited by the humanoids' ability to maintain balance and deal with disturbances and uncertainties. Balance can therefore be considered as one of the core skills for humanoid robots and locomotion.

Classical control methods propose a wide range of balance recovery algorithms, which however lack in the universality of their application. In order to deal with a wide range of pushes, different control strategies need to be applied and traditionally a switching between controllers for the given situation is needed. Generally, different sets of parameters are used for the four main push recovery strategies: ankle, hip, foot-tilting, and stepping (cf. Fig. 1).

Methods from machine learning on the other hand provide a promising alternative as they can incorporate multiple push recovery skills without the need for hand-tuned gains. Their use is motivated by three main factors:

First, supplementing existing control strategies with learning methods allows dealing with scenarios that are hard to engineer in a traditional sense such as sudden, high impact forces and discrete, sudden switches of contact.

This research is supported by the EPSRC UK RAI Hub in Future AI and Robotics for Space (FAIR-SPACE, project ID: EP/R026092/1) and EU H2020 project Memory of Motion (MEMMO, project ID: 780684).

All authors are with the Institute for Perception, Action, and Behaviour, School of Informatics, The University of Edinburgh (Informatics Forum, 10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom). Email: chuanyu.yang@ed.ac.uk.

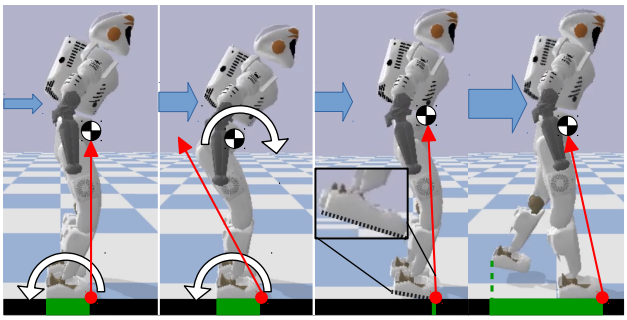


Fig. 1: Learned push recovery behavior: (a) ankle strategy, (b) hip strategy, (c) foot-tilting strategy, (d) stepping strategy.

Second, in contrast to planning and control algorithms that demand high computational power to run at or close to real-time, e.g. Model-Predictive Control, the computation for machine learning approaches can be outsourced offline [1]. I.e., the computation for Deep Reinforcement Learning (DRL) can be off-loaded into the neural network training phase. By doing so, faster online performance for high dimensional control systems such as humanoids can be achieved.

Last, in recent years, DRL has been shown to be capable of solving complex manipulation and locomotion tasks that involve learning a control policy in high-dimensional continuous observation and action spaces [2]–[4]. Instead of manually tuning the control parameters, a feasible policy is learned through interaction with the environment.

While there exist various studies using DRL to learn bipedal locomotion for humanoids [5]–[7], the used robot models leverage simplified dynamic and collision models and environments in order for faster than real-time simulations at the cost of less realistic simulations. The motivation of this paper is to learn locomotion skills using a realistic robot model obtained from system identification in a realistic simulation environment in order to apply the learned skills on the real robot. We leverage recent advances in DRL to design a unified balance recovery controller which is able to generate sequences of actions that perform similar to or even exceed traditional methods with respect to their disturbance rejection ability. Our work has the following contributions:

1) Application of DRL on an accurate robot model of the Valkyrie platform with realistic settings for simulation.
2) Design of a learning framework that generates a generic policy. This policy captures a variety of sensor-motor synergies and various control strategies emerge in a

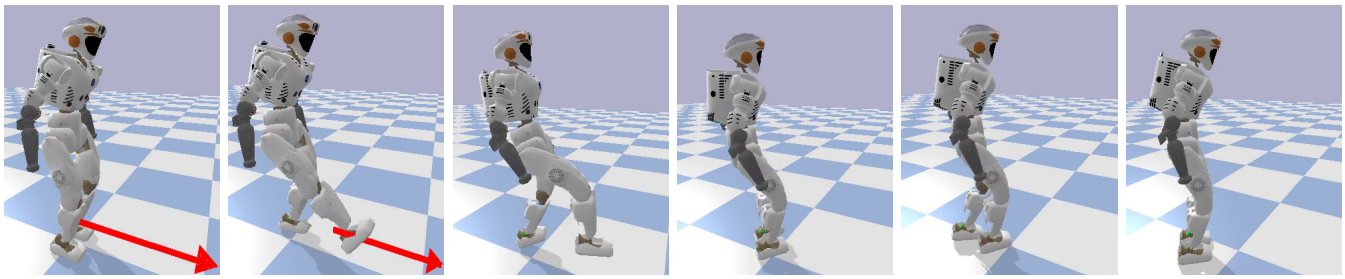


Fig. 2: Snapshots of Valkyrie recovering from an impulse at the shin of $108Ns$, which is a test scenario not encountered during training. The learned policy automatically generates a stepping behavior (cf. `https://youtu.be/43ce2cLV0ZI`).

unified manner without the need of multiple controllers and the related switching mechanism.

3) Proposal of a balance recovery specific reward design and training settings of disturbances. These allow the exploration of versatile motions and as a result human-like balancing behaviors, such as foot-tilting and stepping, emerge naturally.
4) Benchmarking the learned policy against control methods. The learned policies generate balance recovery strategies which reject impulses in a similar (or even superior) magnitude as traditionally designed controllers.

This paper is organized as follows. A brief review on conventional push recovery methods and DRL is presented in Section II. Background information on some concepts of DRL and push recovery is explained in Section III. The proposed methodology is elaborated in Section IV. The obtained results are demonstrated and discussed in Section V. Finally, a conclusion is drawn in Section VI.

II. Related Work

A. Conventional Push Recovery methods

Over the past two decades, remarkable progress in the field of push recovery for humanoid robots has been made. Without the use of arms, humanoids can leverage four lower body balancing strategies: ankle, hip, stepping, and foot-tilting. The first three strategies, controlling ankle torque, angular momentum around the Center of Mass (CoM), and the timing and position of steps, are analyzed with respect to their ability to reject disturbances in [8]. A control framework for the foot-tilting strategy has been proposed in [9] demonstrating a humanoid's ability to use foot-tilting for push recovery. Traditionally control schemes can be divided into predictive schemes which calculate reference motions, and reactive schemes which respond to sudden disturbances.

A Model Predictive Control (MPC) scheme that constrains the Center of Pressure to be within the Support Polygon has been proposed in [10]. Strategies involving modulating the Angular Momentum to reactively deal with disturbances have been formally analyzed in [11], [12]. Due to the limited size of the contact area, i.e. the foot size, stepping strategies have been proposed [13], [14]. This idea was formalized as the Capture Point (CP), the point on which one needs to step in order to come to a complete halt [11]. Enlarging the support area by stepping has been extended to multi-contact push recovery scenarios in [15], [16]. Methods for balancing on inclined slopes has also been proposed [17]. Lastly, strategies modulating the height of the CoM in order to compensate for disturbances have been proposed in [18]. This CoM height modulation can be achieved by either lengthening the leg or in form of foot tilting [9].

B. Deep Reinforcement Learning of Locomotion

There exists various successful studies using model-free DRL to solve bipedal locomotion tasks in 3D simulation environments. Schulman et al. proposed a DRL algorithm, Proximal Policy Optimization (PPO), which was applied to successfully learn a locomotion policy that is capable of heading towards a target location in the Roboschool humanoid simulation environment [4]. PPO, together with Deep Deterministic Policy Gradient (DDPG) [3] and Trust Region Policy Optimization (TRPO) [2], are the most commonly used state-of-the-art DRL algorithms for continuous observation-action space control. Further extensions include a parallel computing version of PPO, Distributed Proximal Policy Optimization (DPPO) [19], which was applied on a humanoid and successfully learned dynamic and diverse parkour movements for the humanoid character.

Various frameworks have been proposed to allow the DRL agent to learn a policy that generates human-like locomotion behavior for bipedal locomotion tasks. Merel et al. proposed a framework that uses generative adversarial imitation learning [20] to enable the network to learn a policy that produces human-like locomotion gait using limited demonstrations from motion capture data [21]. Peng et al. proposed a framework that incorporated imitation learning by reshaping the reward through the introduction of an imitation term that provides higher reward when the motion is closer to the reference motion capture data [22].

III. Background

A. Software Setup

The simulation environment of the Valkyrie robot is built using PyBullet [23] (Fig. 2). The robot model used in the simulation is the NASA Valkyrie robot [24] with realistic inertia, center of mass, and joint actuation limits. Self-collisions are enabled in the simulation. The DRL algorithm is built using Tensorflow [25].

B. Deep Reinforcement Learning

For learning a suitable policy, Deep Reinforcement Learning, particularly model-free policy gradient methods, are

used. Policy gradient algorithms operate by maximizing the direct sum of rewards with reference to a stochastic policy. The policy gradient algorithms used in this work are the TRPO [2], PPO [4], and DDPG [3]. Due to resulting in the best and most robust policy, the TRPO algorithm will be outlined in the following.

1) Trust Region Policy Optimization: In practice, policy gradient methods suffer from high variance which can lead to fluctuations in the performance of the policy between iterations. This problem of instability during training is remedied by introducing a trust region to the numerical optimization which takes a step in the improving direction within a determined trust region. By constraining the amount of changes to the parameters, measured by the Kullback-Leibler (KL) divergence, TRPO guarantees a theoretical monotonic performance improvement of the reward.

For every parameter update iteration, TRPO performs several rollouts and stores the state s_t, action a_t and reward r_t into a batch $\mathcal{D}$ until enough data samples are collected, which will then start the update process. During the update process, TRPO updates the policy parameters by minimizing a surrogate loss function while constraining the KL divergence between the new and old policies π_θ, $\pi_{\theta_{old}}$ to remain within a trust region:

$$\min_\theta L_{\theta_{old}}(\theta) = -\mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t\right] \tag{1}$$

$$\text{subject to } \mathbb{E}_t\left[KL\left[\pi_{\theta_{old}}(\cdot|s_t)\,\pi_\theta(\cdot|s_t)\right]\right] \leq \delta, \tag{2}$$

where δ is the hyperparameter that determines the trust region, $A_t = R_t - V(s_t)$ is the advantage which is calculated by subtracting the return with a baseline. A value estimation $V(s_t)$ provided by a critic is used as the baseline.

2) Discounted return: The total return is used as an evaluation of performance and is determined by calculating the discounted reward,

$$R_t = \sum_{l=0}^{T-t}\gamma^l r_{t+l}, \tag{3}$$

where T is the total number of samples in an episode and γ is the discount factor. The half-life of future rewards is used as a reference to decide the value of the discount factor γ. For balancing, a time horizon between 0.5s and 2s is close. With a frequency of 25Hz, 0.5s equates to 13 time steps. We choose the discount value in a way that the half-life of the future reward occurs at 0.5s, meaning that the accumulated discount factor equates to 0.5 at 13 time step $\gamma^{13} = 0.5$, hence $\gamma \approx 0.95$.

3) Generalized Advantage Estimation: With the policy gradient method and a stochastic policy, we obtain an unbiased estimate of the gradient of the expected total reward, however the estimated policy gradient has high variance. An effective variance reduction scheme for policy gradients called Generalized Advantage Estimator (GAE) was proposed in [26]. GAE interpolates between a high bias and low bias estimator through the parameter $\lambda \in [0, 1]$. One can

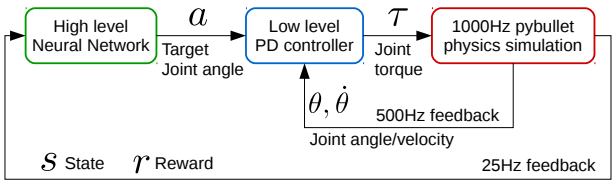


Fig. 3: Hierarchical control system overview [27]

adjust the bias/variance trade-off by tuning λ. The GAE for the parameters γ, λ at time t is:

$$\begin{aligned} A_t^{GAE(\gamma,\lambda)} &:= \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+1}^V \\ \delta_{t+1}^V &= r_t + V(s_{t+1}) - V(s_t). \end{aligned} \tag{4}$$

C. Capture point

The Capture point (CP) describes the point on the ground on which the robot should step on in order to come to a complete rest [11], and is defined as:

$$x_{\text{CP}} = x_{\text{CoM}} + \dot{x}_{\text{CoM}}\sqrt{\frac{z_c}{g}}, \tag{5}$$

where x_{CP} is the CP, $x_{\text{CoM}}, \dot{x}_{\text{CoM}}$ is the CoM position and velocity, z_c its height, and g the gravitational constant.

When the CP is within the support polygon, the robot does not need to perform any footstep to maintain balance. Knowing the feet dimensions and therefore the support polygon, the theoretical maximum impulse which can be rejected without foot-stepping can be approximated as follows [11]:

$$J_{\text{reject}} = m\sqrt{\frac{g}{z_c}}\Delta_{\text{COP}}, \tag{6}$$

where Δ_{COP} is the distance between the CoM and the closest border of the Support Polygon in the direction of the push. For the nominal, upright-standing pose the dimensions of the Support Polygon of Valkyrie is $0.26m \times 0.38m$, the CoM height is at $1.1m$, the mass of the Valkyrie robot is $137kg$. Equation (6) yields an approximate maximal impulse of $53Ns$ in the sagittal plane and $78Ns$ in the lateral plane for $\Delta_{\text{COP}} = [0.13m, 0.19m]$.

IV. Methodology

A. Hierarchical control framework

We designed our control framework to have a hierarchical structure (Fig. 3). A hierarchical structure allows implementation of two (high and low-level) layers that are independent from each other and can be designed and calibrated separately. The high-level control works under a frequency of 25Hz while the low-level control works at 500Hz. The high-level control is responsible for generating joint angles for a desired motion and the low-level control is responsible for translating the joint angles into joint torques.

B. Joint-level control of the robot

Instead of directly controlling the joint motor torque, a PD controller is used to translate joint angles into joint torques. A PD controller resembles the biomechanics of a system in

a sense that it has spring damping properties. A comparison between directly using torques and a PD controllers to compute the torques for certain motor tasks was conducted in [28] and showed that using a PD controller improved learning speed and overall performance. The resulting torque is computed as:

$$u = K_p(q_{\text{target}} - q_{\text{measured}}) - K_d\dot{q}_{\text{measured}}, \tag{7}$$

where K_p, K_d are the PD gains respectively, q_{target} is the targeted joint angle, and $q_{\text{measured}}, \dot{q}_{\text{measured}}$ are the measured joint angles and velocities respectively..

C. Observation space and action space

Input states are chosen in a way such that they can be acquired by sensors on the robot with minimal amount of computation. Immeasurable states are inferred or estimated by the Neural Network. All the sensory information provided as the observational input for the policy is heading-invariant. For balancing, the rotation along the direction of the gravity vector is irrelevant to the balancing state, therefore information about the heading is not needed as feedback, i.e. the policy will perform the same action regardless of global yaw orientation. In order to make the state observation heading invariant, we preprocessed the state by performing transformation of the observations along the gravity axis.

The state $\mathcal{S} \in \mathbb{R}^{47}$ consists of joint angle and velocity, pelvis states (translational and angular velocity, orientation), CoM states (translational velocity and position w.r.t. pelvis), ground contact force, torso position w.r.t. pelvis, and foot position w.r.t pelvis. The observation states are sampled at a frequency of 500Hz and are filtered by a first-order Butterworth filter with a cut-off frequency of 10Hz.

Under consideration of computation efficiency, we minimize the size of action space. A minimum of 11 joints that includes only roll and pitch joints are sufficient for balancing. The action space $\mathcal{A} \in \mathbb{R}^{11}$ of the policy describes the motion of the joint angles. The upper body joints are locked in a nominal position, while for the lower body, only the pitch joint and roll joint are controlled. The controlled joints therefore are: torso pitch, left and right hip pitch & roll, knee pitch, and ankle pitch & roll.

D. Design of reward function

The design of the reward function is a crucial part in reinforcement learning as the reward governs the outcome behavior. The reward design follows a similar design rule as in [27]. Balancing can be divided into four subtasks: regulating upper body pose, regulating CoM position, regulating CoM velocity, and regulating ground contact force. The individual reward is calculated using $\tilde{r}_i = \exp(-\alpha_i(x_{target} - x)^2)$, with x_{target} as the desired value, x as the real value, and α_i as the normalization factor. These are then weighted by w_i. Furthermore, additional penalty terms are added: ground contact regulation, loss of contact with the ground, and when other parts of the body other than the foot make contact with the ground. We also apply a penalty for the control effort used. The overall reward can be viewed as a sum of the individual reward terms:

$$\begin{aligned} r =& r_{pose} + r_{CoM_pos} + r_{CoM_vel} + r_{GRF} + \\ & r_{contact} + r_{power}. \end{aligned} \tag{8}$$

1) Upper body pose modulation: The upper body pose is represented by the pitch and roll angle of the torso and pelvis. The desired orientation for the pitch roll angle for both pelvis and torso is 0, which is the orientation of the upper body when it is upright:

$$\begin{aligned} r_{pose} =& w_{\phi_{\text{torsoPitch}}}\tilde{r}_{\phi_{\text{torsoPitch}}} + w_{\phi_{\text{pelvisPitch}}}\tilde{r}_{\phi_{\text{pelvisPitch}}} + \\ & w_{\phi_{\text{torsoRoll}}}\tilde{r}_{\phi_{\text{torsoRoll}}} + w_{\phi_{\text{pelvisRoll}}}\tilde{r}_{\phi_{\text{pelvisRoll}}}. \end{aligned} \tag{9}$$

2) CoM position modulation: The reward term for CoM modulation is decoupled into horizontal and vertical components. For the horizontal CoM position, the target position is the center of the support polygon to provide maximum disturbance compensation. For the vertical CoM position, the robot should stand upright and maintain a certain height,

$$r_{CoM_pos} = w_{xy_{\text{CoM}}}\tilde{r}_{xy_{\text{CoM}}} + w_{z_{\text{CoM}}}\tilde{r}_{z_{\text{CoM}}}. \tag{10}$$

3) CoM velocity modulation: Similar to the CoM position, the reward for CoM velocity is decoupled into two components: velocity in the horizontal and vertical planes. The CoM velocity is represented in the world frame. The desired vertical CoM velocity is 0 as we want to minimize vertical movement, while the desired velocity for horizontal CoM velocity is derived from capture point. The Capture Point is only valid when the robot has contact with the ground with no slipping, therefore when the robot is in the air, the reward term for horizontal CoM velocity $\tilde{r}_{\dot{x}\dot{y}_{\text{CoM}}}$ is deemed invalid and is set to 0:

$$r_{CoM_vel} = \begin{cases} w_{\dot{x}\dot{y}_{\text{CoM}}}\tilde{r}_{\dot{x}\dot{y}_{\text{CoM}}} + w_{\dot{z}_{\text{CoM}}}\tilde{r}_{\dot{z}_{\text{CoM}}}, & \text{foot contact} \\ w_{\dot{z}_{\text{CoM}}}\tilde{r}_{\dot{z}_{\text{CoM}}}, & \text{no foot contact.} \end{cases} \tag{11}$$

4) Contact force modulation: The force has to be evenly distributed between both feet for a stable robust balance. The total mass of 137kg yields a force of 671.3N for each foot:

$$r_{GRF} = w_{F_{left}}\tilde{r}_{F_{left}} + w_{F_{right}}\tilde{r}_{F_{right}}. \tag{12}$$

5) Ground contact: When the robot is standing, only the feet are in contact with the ground, therefore a penalty is introduced whenever both feet lose contact with the ground or body parts other than the feet make contact with the ground:

$$r_{contact} = \begin{cases} -2, & \text{if no foot contact with ground} \\ -10, & \text{if upper body contact with ground.} \end{cases} \tag{13}$$

6) Power consumption: The power consumption is calculated as follows:

$$r_{power} = w_{power} \cdot \sum_{j=0}^{11} \left\| \tau^j \cdot \dot{q}^j \right\|, \tag{14}$$

with τ^j is the torque applied on individual joints, and $\dot{q}^j$ is the joint velocity.

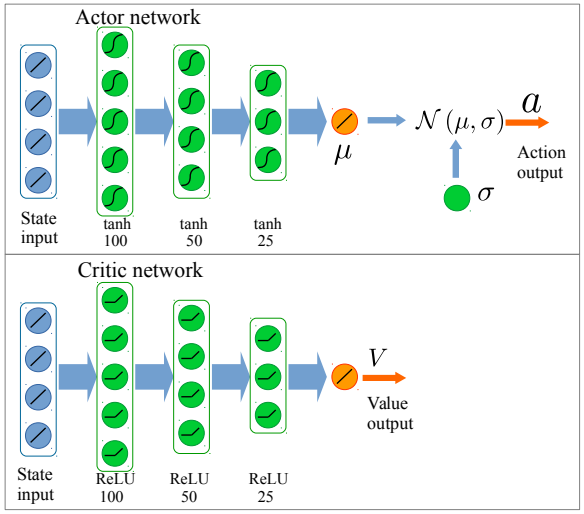


Fig. 4: Overview of neural network structure.

E. Network structure

The stochastic policy $\pi_\theta(a|s)$ is represented as a conditional Gaussian policy $\pi_\theta(a|s) \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta)$. The mean of the Gaussian policy is parametrized by a neural network with parameters θ, the covariance of the Gaussian policy is independent from the neural network and is maintained by a separate set of parameters σ_θ.

The critic V_ϕ parametrizes the value function with a separate set of neural networks using parameters ϕ. Both the actor and the critic are parametrized by a fully connected feedforward neural network that consists of 3 hidden layers with 100, 50 and 25 neurons for each layer. The actor network uses $\tanh$ activation for the hidden layers while the critic uses ReLU activation for the hidden layers. The output of both network is produced by linear activation.

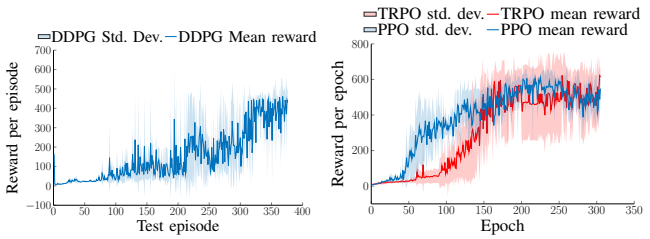
The actor network is trained to maximize the reward function (section IV-D), while the critic network is trained by minimizing the loss function $L_\text{V}(\phi)$:

$$L_\text{V}(\phi) = \mathbb{E}_t \left[(V(s_t) - y_t)^2 \right], \tag{15}$$

with the discounted Return y_t (eq. 3), value function $V(s_t)$.

F. Exploration during training

In order to learn a policy capable of withstanding large push disturbances, sufficient exploration during the training phase needs to be provided. Therefore, in addition to the stochastic policy, random forces are applied on the pelvis during the training. From Capture Point theory, the maximum disturbance in the sagittal plane without foot-stepping is $53Ns$. The bounds of the training disturbances is chosen to be $[53 \times 0.5Ns, 53 \times 2Ns]$. The orientation of the force in the horizontal plane and the disturbance in the bound are randomized. Disturbances are applied to the robot multiple times during each trial, with 5s interval between subsequent push disturbances for push recovery.



(a) DDPG (b) TRPO & PPO

Fig. 5: Learning curves for DDPG, PPO, and TRPO. The performance are evaluated using the deterministic policy. The mean of the Gaussian policy learned by PPO/TRPO is used for evaluation. The results are averaged over 7 learning trials.

G. Learning Algorithm

Due to the structure and choice of our framework, the learned policy is independent of the type of learning algorithm. We trained a policy for maintaining balance via TRPO, PPO, and DDPG, and found similar resulting behavior (cf. Table I). All four balancing strategies (Fig. 1) emerges regardless of the DRL algorithm used. However, from our simulations, TRPO is able to achieve higher rewards and is able to withstand higher impulses. Figure 5 shows the learning curves for the policies learned in Table I. DDPG is trained off-policy and utilizes a replay buffer, whereas TRPO and PPO are trained on-policy batch-wise, which makes it difficult to directly compare.

All three DRL algorithms are able to learn a feasible balancing policy. The difference in performance can be attributed to the randomness in different trials of training and hyperparameters. Training is performed entirely on a single Intel Core i7-6700K with 4.0 GHz and converges in two days.

TABLE I: Maximal rejectable impulses for the various learning algorithms without taking steps.

	Maximal disturbance in Ns	
Learning algorithm	Sagittal	Lateral
TRPO	240	78
DDPG	75	160
PPO	192	36
Baseline from (6)	53	78

V. Results

In the following, a series of test scenarios are presented to evaluate the performance of the control policy acquired by the deep reinforcement learning agent. Furthermore, we show its robustness to external disturbances, as well as noise in the observation (measurement) and action (actuation) spaces. Next, a comparison against traditional methods from other works is made. Lastly, the physical validity of the generated motions is analysed and verified. Please refer to the accompanying video for the results (cf. `https://youtu.be/43ce2cLV0ZI`).

A. Horizontal push on pelvis

Horizontal disturbances were applied on the pelvis during the training phase, and therefore the DRL agent should be able to learn to withstand such type of disturbances. The robot exhibits different behavior depending on the amount of disturbance applied (Table II, Fig. 6). Different control strategies emerge and range from generating ankle torque to shift the COP (ankle strategy), generating angular momentum (hip strategy), over tilting the foot to dissipate the disturbance (foot tilt strategy), to taking a step to recover from the large push (stepping strategy). The magnitude of the lateral pushes, for which the robot is capable of withstanding, is significantly smaller than in the sagittal plane. This is due to the fact that the support leg will block the swing leg in the lateral direction, limiting the range for leg movement. Dealing with this problem involves either jumping to take a step, or crossing the legs, which, due to the kinematic constraints, is not possible for Valkyrie. These jumping manoeuvres were not learned by the policy, as high velocities in the CoM resulted in lower rewards.

From the CoM position (Fig. 6a) and the pelvis orientation (Fig. 6b) it can be inferred that the robot is standing still after 3s. As can be seen by the eight flat plateaus in Figure 6(c), eight steps are taken in order to deal with a impulse disturbance of $240Ns$ at the pelvis. After the eighth step the robot stands still in the nominal pose. For other horizontal pushes, the stepping behavior is similar.

TABLE II: Emerging behavior for impulse disturbances of different magnitudes. A checkmark indicates that the respective strategy is applied in addition to the other marked strategies.

	Impulse disturbance in Ns			
Emerging behaviour	$24Ns$	$72Ns$	$240Ns$	$78Ns$
Push direction	sagittal	sagittal	sagittal	lateral
Ankle strategy	✓	✓	✓	✓
Hip strategy	✓	✓	✓	✓
Foot tilt strategy	✗	✓	✓	✓
Stepping strategy	✗	✗	✓	✗

B. Force disturbance on other body segments

During the training phase only horizontal disturbances were applied on the pelvis. It is well known in machine learning that the test set should vary from the training set. Therefore, we also designed test scenarios which the DRL agent has never encountered before during training to see how well the policy generalizes.

In push recovery studies, the disturbance is usually applied near the CoM to avoid introducing any torque into the system, as it is more challenging to balance a robot with high amount of angular momentum. We are interested in how well the policy will perform when disturbance is applied on other parts of the body far away from the CoM. We chose three body parts for which a large torque and angular movements would result when force is applied on: the upper torso, leg thigh, and leg shank (Table III). The resulting motion for being pulled at the shank can be seen in Figure 7. In Figure 7a) six steps for recovering balance are observed. The support foot height, and roll and pitch angle relative to the ground can be seen in Figure 7b). Finally, the angular movement of the pelvis Euler angles (Fig. 7c) show that the robot recovers into an almost nominal pose after six steps.

TABLE III: Maximum rejected impulse for different body parts.

Body part	Max. impulse in $[Ns]$	Lever in $[m]$	Torque in $[Nm]$	Amount of steps
Upper torso	120	0.32	320	6
Leg thigh	108	0.50	450	4
Leg shank	108	0.70	630	6

C. Landing from height

A different type of impact is applied to the robot by dropping it from a height above the ground with the objective to land stably. Landing involves high impact and sudden changes in ground contact. Various control strategies involving the Zero-Moment Point (ZMP) or CP assume steady contact with the ground and, without a switching mechanism, fail to perform when the robot is in the air. This test scenario is used to test whether the policy can handle high impact and sudden change in ground contact and land stably.

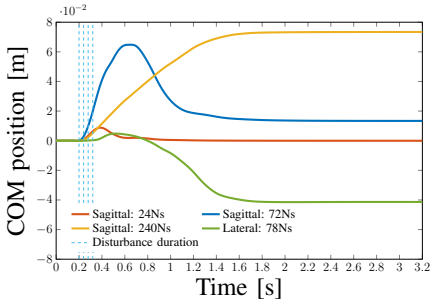
The robot is capable of landing after being dropped from a maximum height of $0.55m$ above the ground. Furthermore, it is able to handle randomly initialized orientations of the pelvis within $[-5°, 5°]$ under a dropping height of 0.4m (with a leg length of $1m$, this is a displacement of $0.087m = 1m \cdot \sin(5\frac{\pi}{180})$ compared to the initial position).

D. Combined test case

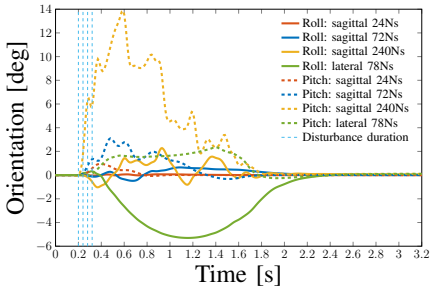
Lastly, a combined test case involving linear and angular momentum disturbances is designed. In this test scenario, we first apply a vertical force upward to lift the robot off the ground and then apply a horizontal force when the robot is in the air. As a result, the robot has to handle linear and angular momentum and sudden high impacts at the same time. The robot is able to recover for a vertical impulse of $500Ns$, and a horizontal impulse of $70Ns$. In order to deal with this combined disturbance, the policy used all four push recovery strategies: ankle, hip, toe, step. Apart from the physical impossibility of throwing a 137kg robot, the resulting joint torques, velocities are within the joint limits of the real Valkyrie (cf. Section V-G).

E. Robustness against noise in observation and action space

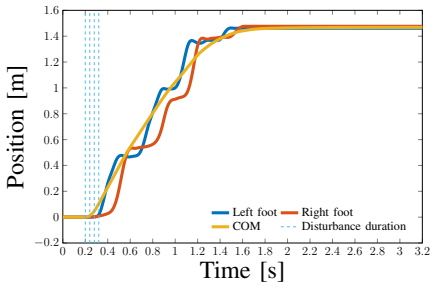
In real world applications, noisy measurements and actuations signals are a main contributor to the discrepancy between simulations and reality. In order to test our policy's ability to be applied in the real world, its ability to handle noise both in action and observation space is tested. For this, a Gaussian distributed noise $d \sim \mathcal{N}(\mu, \sigma)$ is added to both action space ($\mu = 0, \sigma = 0.1$), and state space



(a) CoM motion over time for multiple forces. The CoM position of the $2000N$ is scaled by 0.05

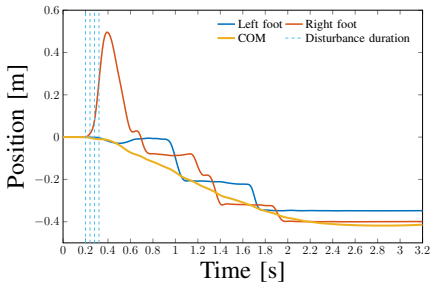


(b) Pelvis orientation over time.

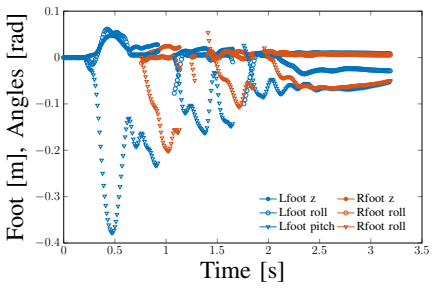


(c) Foot and CoM movement for sagittal push of $2000N$.

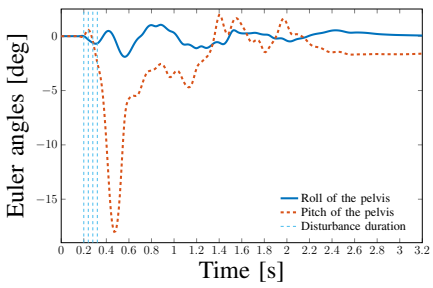
Fig. 6: Resulting motions from impulse disturbance and balance recovery.



(a) CoM and feet motion.



(b) **Support** foot behavior (swing foot not depicted).



(c) Angular movement of the pelvis.

Fig. 7: Resulting motions from an impulse disturbance at the shank. The robot takes 6 steps before standing stably.

($\mu = 0, \sigma = 0.5$). We found that the policy is able to handle both. The observation noise is filtered by the Butterworth filter, while noise in the action space is handled by the robustness of the policy.

F. Comparison against other control methods

In order to compare the results obtained from the policy with other works, the disturbance is normalized. The applied force needs to be put in relation with the duration of the push, resulting in the impulse acting on the system. Furthermore, the mass and the resulting inertia is a crucial variable to a robot's ability to deal with disturbances. Therefore, the impulse is normalized by the weight of the robot.

The normalized impulse is used for comparison between the controllers of other works [10], [13], [29], [30] and the learned policy (Table IV). We compared sagittal and lateral pushes. For the sagittal push, two impulses are chosen such that foot stepping occurs for the larger impulse ($1.73Ns/kg$), while the smaller impulse ($0.57Ns/kg$) will result in a strategy without stepping. By comparing the rejectable normalized impulse of the strategies not taking a step (A, B, E, G), it can be seen that our policy performs similar (E: $0.57Ns/kg$, G: $0.56Ns/kg$) to the other controllers (A: $0.6Ns/kg$, B: $0.52Ns/kg$). For the strategies taking a step (C, D, F), our policy is able to perform better (F: $1.73Ns/kg$) than the stepping controllers C ($0.52Ns/kg$) and D ($0.6Ns/kg$). Albeit our results are obtained from simulation whereas C and D are obtained from real experiments, we show in the next section that the generated motions are realistic and within the real physical constraints.

G. Realism of generated motions

Despite learning is trained in a simulator, we emphasize realistic motions by enforcing joint angle, velocity, and torque limits, which are the same as on the real Valkyrie robot. Therefore, the learned motion could be applied on the real Valkyrie robot without violating physical constraints. Table V compares the peak torques and velocities for different scenarios. The chosen scenarios require the largest joint torque for dropping and the largest joint velocities for taking multiple steps due to large pushes at the pelvis. All other presented test cases required less joint torque and velocity than the the ones presented in Table V.

VI. Conclusion

In this work we proposed a learning framework which is able to learn a versatile unified control policy via Deep Reinforcement Learning. We found that the policy is able to deal with different types of disturbances and has comparable performance to conventional controllers. The policy acquired is capable of functioning in unseen situations, demonstrating that it is generalizing well over tasks. Furthermore, the proposed learning framework is learning algorithm independent. We showed successful balance recovery with a policy trained with three of the state-of-the-art DRL algorithms: TRPO, PPO, and DDPG. The emerging motions for push recovery

TABLE IV: Push disturbance from various push recovery studies

	A: Wieber 2006a [10]	B: Wieber 2006b [29]	C: Stephens 2010 [30]	D: Urata 2011 [13]	E: Sagittal push w/o foot stepping	F: Sagittal push w/ foot stepping	G: Lateral push w/o foot stepping
Robot	HRP-2	Biped model	Sarcos Primus	HRP3L-JSK	Valkyrie	Valkyrie	Valkyrie
Robot height $[m]$	1.539	1.425	1.575	N/A	1.8	1.8	1.8
CoM height $[m]$	N/A	N/A	N/A	0.803	1.1	1.1	1.1
Mass $[kg]$	58	40	92	53	137	137	137
Force $[N]$	1500	750.0	N/A	597	600	2000	650
Interval $[s]$	0.025	0.025	N/A	0.05	0.12	0.12	0.12
Impulse $[Ns]$	37.5	18.8	42.0	29.9	72.0	240.0	78.0
Normalized impulse $[\frac{Ns}{kg}]$	0.6	0.52	0.52	0.6	0.57	1.73	0.56
Stepping	No	No	Yes	Yes	No	Yes	No
Simulated	Yes	Yes	No	No	Yes	Yes	Yes

TABLE V: Peak torques and velocities for different scenarios.

	Peak joint torque $[N]$						Peak joint velocity $[rad/s]$					
	Torso pitch	Hip pitch	Hip Roll	Knee pitch	Ankle Pitch	Ankle Roll	Torso pitch	Hip pitch	Hip Roll	Knee pitch	Ankle Pitch	Ankle Roll
Joint limit	150	350	350	350	205	205	9.00	6.11	6.11	11.00	11.00	11.00
Nominal standing	68.4	39.5	57.1	122	44.9	46.2	0.0	0.0	0.0	0.0	0.0	0.0
0.55m Drop	150	221	350	350	205	205	4.69	2.01	6.11	9.95	11.0	11.0
$78Ns$ Pelvis (lateral)	104	147	103	264	117	106	0.15	0.58	0.46	1.09	0.96	11.0
$240Ns$ Pelvis (sagittal)	150	187	350	350	205	205	6.49	1.47	6.11	5.69	11.0	11.0

are similar to human motions demonstrating the ankle, hip, foot-tilting, and stepping strategy. We compared the learned policy against traditional push recovery controllers and found similar disturbance rejection capabilities.

REFERENCES

[1] K. Yuan and Z. Li, "An improved formulation for model predictive control of leggedrobots for gait planning and feedback control," in *IROS*, 2018.

[2] J. Schulman, *et al.*, "Trust region policy optimization," in *ICML*, 2015.

[3] T. P. Lillicrap, *et al.*, "Continuous control with deep reinforcement learning," *arXiv:1509.02971*, 2015.

[4] J. Schulman, *et al.*, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[5] C. Liu, *et al.*, "Implementation of deep deterministic policy gradients for controlling dynamic bipedal walking," in *Conference on Biomimetic and Biohybrid Systems*, 2018.

[6] X. B. Peng, *et al.*, "DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *Transactions on Graphics*, 2017.

[7] W. Yu, *et al.*, "Learning symmetry and low-energy locomotion," *arXiv:1801.08093*, 2018.

[8] B. Stephens, "Humanoid push recovery," in *Humanoids*, 2007.

[9] Z. Li, *et al.*, "Humanoid balancing behavior featured by underactuated foot motion," *T-RO*, 2017.

[10] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *Humanoids*, 2006.

[11] J. Pratt, *et al.*, "Capture point: A step toward humanoid push recovery," in *Humanoids*, 2006.

[12] T. Komura, *et al.*, "Simulating pathological gait using the enhanced linear inverted pendulum model," *Transactions on Biomedical Engineering*, 2005.

[13] J. Urata, *et al.*, "Online decision of foot placement using singular lq preview regulation," in *Humanoids*, 2011.

[14] W. Hu, *et al.*, "Comparison study of nonlinear optimization of step durations and foot placement for dynamic walking," in *ICRA*, 2018.

[15] W. Han and R. Tedrake, "Feedback design for multi-contact push recovery via lmi approximation of the piecewise-affine quadratic regulator," in *Humanoids*, 2017.

[16] T. Marcucci, *et al.*, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments," in *Humanoids*, Nov 2017.

[17] Z. Li, *et al.*, "Stabilizing humanoids on slopes using terrain inclination estimation," in *IROS*, 2013.

[18] T. Koolen, *et al.*, "Balance control using center of mass height variation: limitations imposed by unilateral contact," in *Humanoids*, 2016.

[19] N. Heess, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv:1707.02286*, 2017.

[20] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NIPS*, 2016.

[21] J. Merel, *et al.*, "Learning human behaviors from motion capture by adversarial imitation," *arXiv:1707.02201*, 2017.

[22] X. B. Peng, *et al.*, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *arXiv:1804.02717*, 2018.

[23] E. Coumans, *et al.*, "Pybullet physics engine."

[24] N. A. Radford, *et al.*, "Valkyrie: NASA's first bipedal humanoid robot," *Journal of Field Robotics*, 2015.

[25] M. Abadi, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/

[26] J. Schulman, *et al.*, "High-dimensional continuous control using generalized advantage estimation," *arXiv:1506.02438*, 2015.

[27] C. Yang, *et al.*, "Emergence of human-comparable balancing behaviours by deep reinforcement learning," in *Humanoids*, 2017.

[28] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: does the choice of action space matter?" in *SIGGRAPH*, 2017.

[29] P.-B. Wieber and C. Chevallereau, "Online adaptation of reference trajectories for the control of walking systems," *Robotics and Autonomous Systems*, vol. 54, no. 7, pp. 559 – 566, 2006.

[30] B. J. Stephens and C. G. Atkeson, "Push recovery by stepping for humanoid robots with force controlled joints," in *Humanoids*, 2010.