On the Emergence of Whole-Body Strategies From Humanoid Robot Push-Recovery Learning

Diego Ferigo, Raffaello Camoriano, Paolo Maria Viceconte, Daniele Calandriello, Silvio Traversaro, Lorenzo Rosasco, and Daniele Pucci

***Abstract*—Balancing and push-recovery are essential capabilities enabling humanoid robots to solve complex locomotion tasks. In this context, classical control systems tend to be based on simplified physical models and hard-coded strategies. Although successful in specific scenarios, this approach requires demanding tuning of parameters and switching logic between specifically-designed controllers for handling more general perturbations. We apply model-free Deep Reinforcement Learning for training a general and robust humanoid push-recovery policy in a simulation environment. Our method targets high-dimensional whole-body humanoid control and is validated on the iCub humanoid. Reward components incorporating expert knowledge on humanoid control enable fast learning of several robust behaviors by the same policy, spanning the entire body. We validate our method with extensive quantitative analyses in simulation, including out-of-sample tasks which demonstrate policy robustness and generalization, both key requirements towards real-world robot deployment.**

Manuscript received December 21, 2020; accepted April 5, 2021. Date of publication April 30, 2021; date of current version September 24, 2021. This work was supported in part by Machine Learning Genoa Center, Universitá di Genova. D. P. acknowledges the support by the An.Dy project, by the European Union's Horizon 2020 Research and Innovation Programme under Grant 731540. L. R. acknowledges the financial support of the European Research Council under Grant 819789, by the Center for Brains, Minds and Machines, funded by NSF STC award CCF-1231216, by AFOSR projects FA9550-18-1-7009, FA9550-17-1-0390, and BAA-AFRL-AFOSR-2016-0007 (European Office of Aerospace Research and Development), by EU H2020-MSCA-RISE Project MADS - DLV-777826, by the NVIDIA Corporation for the donation of the Titan Xp GPUs, and in part by the Tesla k40 GPU used for this research. This letter was recommended for publication by Associate Editor J. Garcia and D. Kulic upon evaluation of the reviewers' comments. D. Kulic. *(Diego Ferigo and Raffaello Camoriano contributed equally to this work.) (Corresponding author: Diego Ferigo.)*

Diego Ferigo is with Dynamic Interaction Control, Istituto Italiano di Tecnologia, Genoa 16163, Italy, and also with the Machine Learning and Optimisation, University of Manchester, Manchester M13 9PL, U.K. (e-mail: diego.ferigo@iit.it).

Raffaello Camoriano and Daniele Calandriello are with the Laboratory for Computational and Statistical Learning - IIT@MIT, Istituto Italiano di Tecnologia, Genoa 16163, Italy, and also with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: raffaello.camoriano@iit.it; daniele.calandriello@mail.polimi.it).

Paolo Maria Viceconte is with Dynamic Interaction Control, Istituto Italiano di Tecnologia, Genoa 16163, Italy, and also with DIAG, Sapienza Università di Roma, Roma 00185, Italy (e-mail: paolomaria.viceconte@uniroma1.it).

Silvio Traversaro and Daniele Pucci are with Dynamic Interaction Control, Istituto Italiano di Tecnologia, Genoa 16163, Italy (e-mail: silvio.traversaro@iit.it; Daniele.Pucci@iit.it).

Lorenzo Rosasco is with the Laboratory for Computational and Statistical Learning - IIT@MIT, Istituto Italiano di Tecnologia, Genoa 16163, Italy, and with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA, and with the MaLGa and DIBRIS, Università degli Studi di Genova, Genova 16126, Italy, and also with the Center for Brains, Minds, and Machines, MIT, Cambridge, MA 02139 USA (e-mail: lrosasco@mit.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3076955, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3076955

***Index Terms*—Robotics, humanoids, reinforcement learning, whole-body control.**

I. Introduction

BIPEDS are those creatures that make use of two legs for moving while maintaining static or dynamic equilibrium. Balancing is a key prerequisite for any kind of locomotion bipeds may achieve. Human evolution determined highly robust bipedal locomotion, providing enhanced environmental adaptability and fitness with respect to other species. Humanoid robots are actuated mechanisms sharing many structural similarities with the human body. In a world largely crafted by and for humans, they also need to balance for effective operation. The challenges posed by bipedal dynamics are manifold. Bipeds, compared to other morphologies, are inherently unstable. Control actions need to account for a narrow support surface and a sparse mass distribution. Nonetheless, bipedal balancing and locomotion successfully established themselves in nature. Therefore, it is reasonable to expect comparable proprioceptive signals to be sufficient for the emergence of similar motor capabilities.

A great variety of methods aiming to solve similar sequential decision-making problems has recently been proposed. Deep Reinforcement Learning (DRL) is among the most promising [1]. Complex locomotion behaviors can be synthesized by policies trained on sequential interactions with the environment [2]. However, this approach poses fundamental challenges when applied to robotics [3]. In particular, collecting the amount of example trajectories required by most state-of-the-art model-free DRL algorithms is unfeasible for current robots [4]. A common solution consists in resorting to synthetic data based on rigid-body dynamics, addressing the mismatch introduced by the sim-to-real gap in a subsequent stage [5], [6]. Nonetheless, learned behaviors often display unnatural characteristics, such as asymmetric gaits, abrupt motions of the body and limbs, or even unrealistic motions exploiting imperfections and glitches in the physical simulator of choice. These issues significantly limit generalization and transferability to real-world robots.

State-of-the-art methods for bipedal robot control [7] are rooted in control theory and optimal control. Control architectures are often organized as hierarchies composed of trajectory optimization [8], simplified model control, and whole-body quadratic programming [9], [10]. While such approaches have achieved considerable results both on simulated and real humanoid robots, they:

1) Rely on an accurate description of the robot dynamics;
2) Require hand-crafted features for online execution [11];
3) Present challenges when simultaneously facing different tasks.

As concerns push recovery, switching between different strategies (e.g., ankle, hip, stepping, and momentum) is not trivial.

Compared to previous results [12], this work offers the following main contributions:

- Demonstration of the emergence of robust momentum-based whole-body push-recovery strategies in addition to ankle, hip, and stepping ones;
- Design of reward components to guide learning towards steady-state balancing, with transient push-recovery strategies;
- Definition of a state space – inspired by floating-base dynamics – encoding sufficient information for solving the task with no prior knowledge about the desired trajectories.

II. Related Work

A. Control-Theoretic Approaches

Humanoid locomotion control has traditionally been tackled by resorting to simplified models. In particular, the 3D Linear Inverted Pendulum (LIP) model is among the most widely employed ones [13]. Its simplified dynamics proved effective and efficient for trajectory generation in walking, balancing, and push-recovery methods. In the presence of limited perturbations, in-place recovery strategies regulating the Center of Pressure (CoP) [14] or the centroidal angular momentum [15] can be sufficient for recovery. These include ankle, hip, and foot-tilting strategies [16], [17]. An alternative method, modulating the Center of Mass (CoM) height was recently proposed [18]. Stronger perturbations require the support surface to be enlarged or shifted to ensure that the CoP is kept enclosed in it [16]. A natural way to achieve this is by means of stepping strategies. To this end, push-recovery stepping controllers based on Zero-Moment Point (ZMP) [19] trajectory generation have been proposed [20], along with Model Predictive Control (MPC) methods controlling the ZMP while rejecting strong external disturbances [21]. Alternatively, footstep planning strategies based on the Capture Point (CP) [22], [23] have been employed for position-controlled [24], [25] and torque-controlled [9] humanoids. Control-theoretic methods significantly improved the state-of-the-art push-recovery performances of humanoids. Still, they present several limitations:

1) Controllers usually encode a single behavior. Being robust to a wide range of perturbations requires complex controller switching;
2) Robot- and task-specific tuning of the controllers and switching system is a costly trial-and-error procedure;
3) Simplified models and hard-coded strategies often constrain the attainable behaviors;
4) MPC-based methods are computationally expensive, hindering real-time deployment.

B. Deep Reinforcement Learning Approaches

In recent years, DRL has been successfully applied to synthesize computationally efficient controllers for complex robotic tasks in a data-driven way, both in simulation and in the real world. Quadrupeds have drawn considerable attention in DRL locomotion research, also due to their relatively lower dimensionality and greater stability with respect to bipeds. Policies trained in simulation have been transferred to real robots via accurate system identification and domain randomization [26], [27], while the data-efficient Soft Actor-Critic algorithm has been shown to learn robust gait policies from few real-quadruped trials [28]. Remarkably, DRL can also train walking policies for non-humanoid bipedal robots [29], including real-world deployment without dynamics randomization [30].

Other works focus on learning locomotion policies for humanoids. This setting is more challenging, due to the complex and redundant body structure. The potential of DRL in this domain was first demonstrated on walking tasks in simulation [31]. Other methods improve the human-likeness of the behaviors by introducing motion imitation [32], [33]. Still, these methods are more targeted towards benchmarking model-free DRL for continuous control and realistic animation of simplified characters rather than applicability to real humanoid robots.

More recent work has been devoted to training push-recovery [12] and walking [34] controllers for accurate humanoid robot models using principles from robot control and transferable observation and reward designs. The latter approaches, although demonstrating diverse effective behaviors emerging from a single policy, control only the lower body joints. DRL-based methods for whole-body humanoid control remain an open problem and have the potential for learning high-dimensional locomotion policies, further improving humanoid capabilities to recover from external perturbations.

III. Background

A. Notation

- W and B denote the world (inertial) frame and the base frame of the robot; R and L denote the frames of the right and left feet.
- Given two frames A and B, $A[B]$ denotes a new frame with the origin of A and the orientation of B.
- $G := G[W]$ denotes the frame with origin on the robot's CoM and orientation of the world frame.
- n denotes the robot's Degree of Freedom (DoF).
- ${}^A\boldsymbol{p}_B \in \mathbb{R}^3$ denotes the coordinates of point B in frame A. Superscripts, e.g. ${}^A\boldsymbol{p}_B^{xy}$, extract specific coordinates.
- Given two frames A and B and a point C, the matrix ${}^AR_B \in SO(3)$ is such that ${}^A\boldsymbol{p}_C = {}^AR_B{}^B\boldsymbol{p}_C + {}^A\boldsymbol{p}_B$.
- Given ${}^AR_B$, the triplet ${}^A(\psi, \rho, \phi)_B$ denotes the Euler angles of the z-x-y sequence of intrinsic rotations.
- Given $\boldsymbol{w}, \boldsymbol{u} \in \mathbb{R}^3$, we define $\boldsymbol{w}^\wedge = W \in \mathbb{R}^{3\times3}$ as the skew-symmetric matrix such that $\boldsymbol{w}^\wedge\boldsymbol{u} = \boldsymbol{w} \times \boldsymbol{u}$, and $W^\vee = \boldsymbol{w}$ its inverse.
- Given ${}^A\boldsymbol{p}_B$ and three frames A, B and C, the velocity of the point B w.r.t. the origin of frame A, expressed in frame C, is ${}^C\boldsymbol{v}_{A,B} = {}^CR_A{}^A\dot{\boldsymbol{p}}_B$.
- Given three frames A, B and C, the angular velocity of frame B w.r.t. frame A, expressed in frame C is ${}^C\boldsymbol{\omega}_{A,B} = {}^CR_A({}^A\dot{R}_B{}^AR_B^\top)^\vee$.
- ${}^C\mathbf{v}_{A,B} = ({}^C\boldsymbol{v}_{A,B}, {}^C\boldsymbol{\omega}_{A,B})$ denotes the 6D velocity of frame B w.r.t. A expressed in frame C.
- $\boldsymbol{s}, \dot{\boldsymbol{s}} \in \mathbb{R}^n$ denote the joint positions and velocities.
- $\mathbf{q} = ({}^W\boldsymbol{p}_B, {}^WR_B, \boldsymbol{s}) \in \mathbb{R}^3 \times SO(3) \times \mathbb{R}^n$ denotes the configuration of the floating-base robot.
- $\boldsymbol{\nu} = ({}^B\mathbf{v}_{W,B}, \dot{\boldsymbol{s}}) \in \mathbb{R}^{6+n}$ denotes the system velocity, where the base is represented as *body-fixed* velocity [15].
- ${}_A\mathbf{f}_F = ({}_A\boldsymbol{f}, {}_A\boldsymbol{m})_F \in \mathbb{R}^6$ denotes the 6D force acting on frame F expressed in frame A.

In the above definitions, the world frame W is implicitly assumed when A is omitted.

B. Reinforcement Learning (RL)

We formulate balancing and push recovery as a discrete-time Reinforcement Learning (RL) problem modelled as an infinite Markov Decision Process (MDP) with a discounted expected return [35], [36]. In this setting, an agent interacts with an environment following a control policy. At each time step t, the agent collects data from the environment in the form of a state $\mathbf{x}_t$. The control policy $\pi(\mathbf{a}_t|\mathbf{x}_t)$ selects an action $\mathbf{a}_t$ whose application results in a new state $\mathbf{x}_{t+1}$ and a scalar reward $r_t = r(\mathbf{x}_t, \mathbf{a}_t, \mathbf{x}_{t+1})$ encoding the immediate value of the experienced transition towards solving the target task. The interaction generates several trajectories $\tau = \{(\mathbf{x}_0, \mathbf{a}_0, r_0), (\mathbf{x}_1, \mathbf{a}_1, r_1), \ldots\}$. The agent's goal is to learn a policy π maximizing its expected return $J(\pi) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{T}\gamma^t r_t]$ over all possible trajectories τ induced by the policy, where T is the trajectory length and γ the discount factor.

C. Policy Gradient (PG) Methods

A popular class of algorithms addressing expected return maximization for continuous-control tasks is provided by model-free PG methods [37]. Given a parameterized policy $\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{x}_t)$, PG methods perform direct gradient-based optimization of $\boldsymbol{\theta}$ over the scalar performance measure :

$$L^{PG}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t \left[\log(\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{x}_t))\hat{A}_t\right]$$

where $\hat{\mathbb{E}}_t$ denotes the empirical mean over a finite batch of trajectories. The advantage function $\hat{A}_t = R_t - \hat{V}(\mathbf{x}_t)$ evaluates the advantage of taking action $\mathbf{a}_t$ at state $\mathbf{x}_t$, defined as the difference between the actual return $R_t = \sum_{k=0}^{T-k}\gamma^k r_{t+k}$ collected from $\mathbf{x}_t$ in the sampled trajectory and the current estimate of the value function $\hat{V}(\mathbf{x}_t)$. Using on-policy samples only, at each iteration of the optimization the gradient of the expected return is estimated by differentiating $L^{PG}(\boldsymbol{\theta})$ and used to update $\boldsymbol{\theta}$. Among the available PG algorithms, we employ Proximal Policy Optimization (PPO) [31], which tackles the instability characterizing the training process in presence of large policy updates by maximizing the objective $L^{CLIP}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t \min(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t,\mathbf{x}_t)}{\pi_{\boldsymbol{\theta}_{old}}(\mathbf{a}_t,\mathbf{x}_t)}\hat{A}_t, \text{clip}(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t,\mathbf{x}_t)}{\pi_{\boldsymbol{\theta}_{old}}(\mathbf{a}_t,\mathbf{x}_t)}, 1 - \epsilon, 1+\epsilon)\hat{A}_t)$ where $\boldsymbol{\theta}_{old}$ are the pre-update policy parameters and ϵ the hyperparameter used to clip the policy update. Maximizing $L^{CLIP}(\boldsymbol{\theta})$ maintains new policies close to old ones while optimizing the objective.

IV. Environment

The environment is structured as a continuous control task with early termination conditions. Its dynamics runs in the Ignition Gazebo simulator embedded into the gym-ignition framework [38], compatible with OpenAI Gym [39]. The enabled physics engine is DART [40]. We selected iDynTree [14] for calculating rigid-body dynamics quantities, using an accurate model of the robot's kinematics and dynamics represented in the following form [15]:

$$M(\mathbf{q})\dot{\boldsymbol{\nu}} + \boldsymbol{h}(\mathbf{q},\boldsymbol{\nu}) = B\boldsymbol{\tau} + \sum_{k=1}^{n_c} J_k^\top \mathbf{f}_k$$

where $M(\mathbf{q})$ is the mass matrix, $\boldsymbol{h}(\mathbf{q},\boldsymbol{\nu})$ the Coriolis and gravity term, B a selector matrix, $\boldsymbol{\tau}$ the joint torques, n_c the number of contacts, J_k and $\mathbf{f}_k$ respectively the Jacobian and the 6D force of the k-th contact.

The environment receives actions and provides observations and rewards at 25 Hz. The physics and the low-level PIDs run at 1000 Hz. During training, some properties of the environment are randomized (see Sec. IV-D).

A. Action

The separation between agent and environment is defined by the action selection. In our nested structure, the policy generates an action $\mathbf{a} \in \mathbb{R}^{23}$ composed of the reference velocities for a large subset of the robot joints (controlled joints), which are then integrated and fed to the corresponding PID position controllers. The controlled joints belong to the legs, torso, and arms. Hands, wrists, and neck, which arguably play a minor role in balancing, are locked in their natural positions. The policy computes target joint velocities bounded in $[-180180]$ deg/s at 25 Hz. Commanding joint velocities rather than joint positions prevents target joint positions from being too distant from each other in consecutive steps. Especially at training onset, this would lead to jumpy references that cannot be tracked by the PID controllers, affecting the discovery of the relation between $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$. The integration process, instead, enables to use a policy that generates discontinuous actions while maintaining continuous PID inputs with no need for additional filters.

B. State

The state of the MDP contains information about the robot's kinematics and dynamics, since no perception is involved. It is defined as the tuple $\mathbf{x} := \langle \mathbf{q}, \boldsymbol{\nu}, \mathbf{f}_L, \mathbf{f}_R \rangle \in \mathcal{X}$. The observation, computed from the state $\mathbf{x}$, is defined as the tuple $\boldsymbol{o} := \langle \mathbf{o}_s, \mathbf{o}_{\dot{s}}, \mathbf{o}_h, \mathbf{o}_R, \mathbf{o}_c, \mathbf{o}_f, \mathbf{o}_F, \mathbf{o}_v \rangle \in \mathcal{O}$, where $\mathcal{O} := \mathbb{R}^{62}$.

The observation consists of the following terms: $\mathbf{o}_s$ are the controlled joints angles in radians, normalized with the hard limits defined in the model description; $\mathbf{o}_{\dot{s}}$ are the velocities of the controlled joints, normalized in $[-\pi, \pi]$ rad/s; $\mathbf{o}_h$ is the height of the base frame, normalized in $[0, 0.78]$ m; $\mathbf{o}_R$ is a tuple containing the roll and pitch angles of the base frame w.r.t. the world frame, normalized in $[-2\pi, 2\pi]$ rad; $\mathbf{o}_c$ is a tuple defining whether the feet are in contact with the ground; $\mathbf{o}_f$ is a tuple containing the vertical forces applied to the local CoP of the feet, normalized in [0, 3, 3, 0] N, i.e. the nominal weight force of the robot; $\mathbf{o}_F$ is a tuple containing the positions of the feet w.r.t. the base frame, normalized in $[0, 0.78]$ m; $\mathbf{o}_v$ is the linear velocity of the CoM expressed in G, normalized in $[0, 3]$ m/s. The exact definition of all the observation terms is reported in Table I.

Although the agent is trained in simulation, we design it for real-time execution on actual robots. We carefully select state components that can be either measured or estimated on-board [14]. To promote policy transfer, we avoid measurements from noisy sensors and values that cannot be estimated with sufficient accuracy. In fact, any significant mismatch between

TABLE I
OBSERVATION COMPONENTS

Name	Value	Set	Range
Joint positions	$\mathbf{o}_s = \boldsymbol{s}$	$\mathbb{R}^n$	$[\boldsymbol{s}_{lb}, \boldsymbol{s}_{ub}]$
Joint velocities	$\mathbf{o}_{\dot{s}} = \dot{\boldsymbol{s}}$	$\mathbb{R}^n$	$[-\pi, \pi]$
Base height	$\mathbf{o}_h = \boldsymbol{p}_B^z$	$\mathbb{R}$	$[0, 0.78]$
Base orientation	$\mathbf{o}_R = (\rho, \phi)_B$	$\mathbb{R}^2$	$[-2\pi, 2\pi]$
Contact configuration	$\mathbf{o}_c = (c_L, c_R)$	$\{0, 1\}^2$	-
CoP forces	$\mathbf{o}_f = (f_L^{CoP}, f_R^{CoP})$	$\mathbb{R}^2$	$[0, mg]$
Feet positions	$\mathbf{o}_F = ({}^B\boldsymbol{p}_L, {}^B\boldsymbol{p}_R)$	$\mathbb{R}^6$	$[0, 0.78]$
CoM velocity	$\mathbf{o}_v = {}^G\boldsymbol{v}_{CoM}$	$\mathbb{R}^3$	$[0, 3]$

simulated and real data would hinder transfer, increasing the reliance on policy robustness. We select minimal state components encoding the environment dynamics without affecting learning performance.

C. Reward

The reward is a weighted sum of terms that can be categorized as regularizers, steady-state, and transient. *Regularizers* are terms often used in optimal control for the minimization of control action and joint torques. *Steady-state* components help to obtain the balancing behavior in the absence of external perturbations, and are active only in Double Support (DS). Finally, the *transient* components favor the emergence of push-recovery whole-body strategies.

The total reward is composed of a weighted sum of scalar components $\sum_i \omega_i r_i$, where r_i is the reward term and w_i its weight. In order to provide a similar scale for each of them, and therefore improving the interpretability of the total reward, we process the real and vector components with a Radial Basis Function (RBF) kernel [41] with a dimension given by a cutoff parameter calculated from the desired sensitivity. Appendix A provides a more detailed description of the kernel. Table II includes the weights of each reward component and the kernel parameters, if active.

Regularizers. **Joint torques** r_τ**.** Torques applied by the PID controllers are penalized. The environment runs at 25 Hz and the low-level controllers at 1000 Hz. Therefore, for each of the 23 joints, 40 torques are actuated between two consecutive environment steps. We collect all these torques in a single vector $\boldsymbol{\tau}_{step} \in \mathbb{R}^{23 \cdot 40}$ and average its elements. **Joint velocities** $r_{\dot{s}}$**.** Our control scheme ensures that joint position references are continuous. However, PPO explores the action space of joint velocities following the active distributions. To promote smoother trajectories, we penalize the norm of the latest action. It can be seen as the minimization of the control effort.

Steady-state. **Postural** r_s**.** Whole-body humanoid control schemes apply different weights to various control objectives. The postural is notably one of the most used [42], although it is usually assigned a low priority. A postural reward term helps to reach a target posture during balancing instead of relying on local minima found by the learning process. This component penalizes the mismatch between the sampled joint configuration and the reference configuration shown in Figure 4(a). **CoM projection** r_G**.** Statically balanced robots, in order to maintain stability, keep the CoM within the Support Polygon (SP), defined as the Convex Hull (CH) of their contact points with the ground. With the same aim, we introduce a Boolean component rewarding the agent if its CoM ground projection is within the SP induced by the feet. For additional safety, we shrink the SP by a 2.5 cm margin all along its perimeter. **Horizontal CoM velocity** r_v^{xy}**.** We define a target horizontal velocity for the CoM as a vector pointing from the CoM projection to the center of the SP $\bar{\boldsymbol{p}}_{hull}^{xy}$. In order to promote faster motions if the CoM is relatively close to the ground, the magnitude of the target is amplified by a factor $w_0 = \sqrt{g/\boldsymbol{p}_G^z}$ derived from the LIP model [13], where g is the standard gravity. This component encourages the motion of the CoM projection towards the center of the SP.

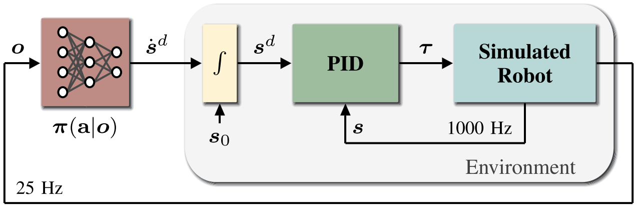

Fig. 1. The proposed control system.

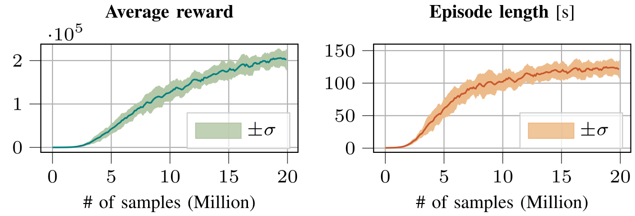

Fig. 2. Learning curves over 11 training runs.

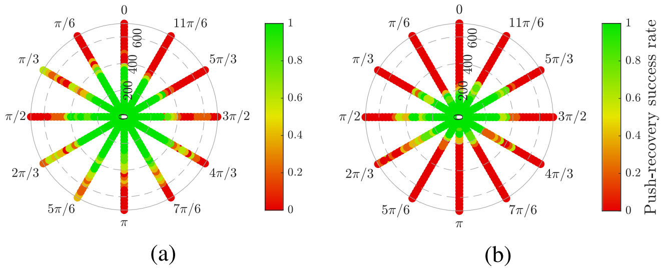

Fig. 3. (a) Push-recovery success rates on the horizontal plane (forward push: 0 rd, $\mu_c = 1$). (b) Results with $\mu_c = 0.2$.

Transient. **Feet in contact** r_c**.** The feet are encouraged to stay on the ground. In order to promote steps and increase movement freedom, we add a Boolean term marking whether any foot is in contact with the ground. **Links in contact** r_l**.** If any link excluding feet is in contact with the ground, the episode terminates with a negative reward of -10 for the terminal state. **Whole-body momentum** r_h**.** Our policy also controls joints belonging to the torso and the arms. The momentum generated by the upper body can, therefore, be exploited for balancing and push recovery. This term minimizes the sum of the norms of the linear and angular components of the robot's total centroidal momentum ${}_G\mathbf{h}$ [15]. **Feet contact forces** r_f**.** This reward term pushes the transient towards a steady-state pose in which the vertical forces at feet's CoP (f_L^{CoP}, f_R^{CoP}) assume the value of half of the robot's weight, distributing it equally on the two feet. **Feet CoP** r_p**.** Beyond the force at the feet CoP, we also promote

TABLE II
REWARD FUNCTION DETAILS. TERMS WITH A DEFINED CUTOFF ARE PROCESSED BY THE RBF KERNEL

Name	Symbol(s)	Weight	Value $\mathbf{x}$	Target $\mathbf{x}^*$	Cutoff x_c		SS	DS
Joint torques	r_τ	5	$\boldsymbol{\tau}_{step}$	$\mathbf{0}_n$	10.0	Nm	✓	✓
Joint velocities	$r_{\dot{s}}$	2	$\boldsymbol{a}$	$\mathbf{0}_n$	1.0	rad/s	✓	✓
Postural	r_s	10	$\boldsymbol{s}$	$\boldsymbol{s}_0$	7.5	deg		✓
CoM z velocity	r_v^z	2	$\boldsymbol{v}_G^{xy}$	0	1.0	m/s	✓	✓
CoM xy velocity	r_v^{xy}	2	$\boldsymbol{v}_G^z$	$\omega_0(\boldsymbol{p}_G^{xy} - \bar{\boldsymbol{p}}_{hull}^{xy})$	0.5	m/s		✓
Feet contact forces	$\{r_f^L, r_f^R\}$	4	$\{f_L^{CoP}, f_R^{CoP}\}$	$mg/2$	$mg/2$	N	✓	✓
Centroidal momentum	r_h	1	$\\|{}_G\mathbf{h}_l\\|^2 + \\|{}_G\mathbf{h}_\omega\\|^2$	0	50.0	kg m^2/s	✓	✓
Feet CoPs	$\{r_p^L, r_p^R\}$	20	$\{\boldsymbol{p}_{L,CoP}, \boldsymbol{p}_{R,CoP}\}$	$\{\bar{\boldsymbol{p}}_{L,hull}^{xy}, \bar{\boldsymbol{p}}_{R,hull}^{xy}\}$	0.3	m	✓	✓
Feet orientation	$\{r_o^L, r_o^R\}$	3	$\{\mathbf{r}_L^{(z)} \cdot \mathbf{e}_z, \mathbf{r}_R^{(z)} \cdot \mathbf{e}_z\}$	1	0.01	-	✓	✓
CoM projection	r_G	10	$\boldsymbol{p}_G^{xy}$	$\in$ CH of support polygon	-	-		✓
Feet in contact	r_c	2	$c_L \wedge c_R$	1	-	-	✓	✓
Links in contact	r_l	-10	c_l	0	-	-	✓	✓

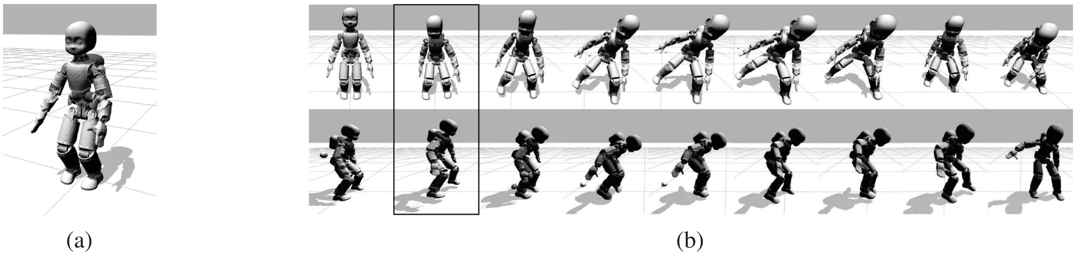

Fig. 4. (a) The initial joint configuration $\boldsymbol{s}_0$. (b) Sequences showing ankle, step, and momentum push-recovery strategies. The robot is pushed by a sphere shot from the left side of the image. Impact takes place in the second frame.

their positions to be located at the center of the corresponding sole $\bar{\boldsymbol{p}}_{foot,hull}^{xy}$. **Vertical CoM velocity** r_v^z**.** This reward component discourages vertical motion of the CoM of the base link, promoting instead the usage of the horizontal component. **Feet orientation** r_o**.** In early experiments, the policy was converging towards feet tipping behaviors, i.e. the feet were not in full contact with the ground. Since the terrain is flat by assumption, we discourage tipping by promoting a feet orientation with the soles parallel to the ground. If ${}^W R_{foot} = [\mathbf{r}^{(x)}, \mathbf{r}^{(y)}, \mathbf{r}^{(z)}]$ is the rotation between the foot frame and the world, this term promotes the alignment of its third column with the world frame.

D. Other Specifications

Initial State Distribution. The initial state distribution $\rho(\mathbf{x}_0) : \mathcal{X} \to \mathcal{O}$ defines the value of the observation in which the agent begins each episode. Sampling the initial state from a distribution with small variance, particularly regarding joint positions and velocities, positively affects exploration without degrading the learning performance. At the beginning of each episode, for each joint j we sample its position $s_{j,0}$ from $\mathcal{N}(\mu = s_0, \sigma = 10\,\text{deg})$, where s_0 represents the fixed initial reference, and its velocity $\dot{s}_{j,0}$ from $\mathcal{N}(\mu = 0, \sigma = 90\,\text{deg/s})$. As a result, the robot may or may not start with the feet in contact with the ground, which encourages the agent to learn how to land and deal with impacts.

Exploration. In order to promote exploration beyond the initial state distribution and favor the emergence of push-recovery strategies, we apply external perturbations in the form of a 3D force to the base frame of the robot. The applied force vector has a fixed magnitude of 200 N and is applied for 200 ms. Considering the weight of the iCub, approximately 33 kg, the normalized impulse sums up to 1.21 Ns/Kg. We sample the direction of the applied force from a uniform spherical distribution. The frequency of the application is defined as average applications per second, again sampling from a uniform distribution. We apply a force on average every 5 simulated seconds.

Early Termination. The balancing and push-recovery objectives for a continuous-control task are characterized by an infinite-horizon discounted MDP. During training, however, episodes should stop as soon as the state reaches a subspace from which either it is not possible to recover or it is uninteresting to explore, following an early-termination criterion. The state space interesting for our work is where the robot is – almost – standing on its feet, therefore we terminate the episodes as soon as it falls to the ground. We detect the falling condition when any link but the feet touches the ground plane.

Domain Randomization. During the training process, at the beginning of each new episode, the environment performs a domain randomization step. The masses of the robot's links are sampled from a normal distribution $\mathcal{N}(\mu = m_0, \sigma = 0.2 m_0)$, where m_0 is the nominal mass of the link defined in the model description. To avoid making assumptions on the material properties of the feet and the ground, we randomize the Coulomb friction μ_c of the feet by sampling it from $\mathcal{U}(0.5, 3)$. Finally, since the simulation does not include the real dynamics of the actuators, to increase robustness we apply a delay to the position references that are fed to the PID controllers, sampled from $\mathcal{U}(0, 20)$ ms.

TABLE III
PPO, POLICY, AND TRAINING PARAMETERS

Parameter	Value
Discount rate γ	0.95
Clip parameter ϵ	0.3
Learning rate α	0.0001
GAE parameter λ	1.0
Batch size	10000
Minibatch size	512
Number of SGD epochs	32
Number or parallel workers	32
Value function clip parameter	1000

V. AGENT

The agent receives the observation $\boldsymbol{o}$ from the environment and returns the action $\mathbf{a}$ defining the reference velocities of the controlled joints. The parameters of the agent are reported in Table III and further explained below.

Learning Algorithm. We select PPO as candidate learning algorithm, in the variant with both the classic gradient clipping and the minimization of the KL divergence.

Policy and Value Function. The stochastic policy $\pi(\mathbf{a}|\boldsymbol{o})$ selects which action to take given a state. The value function $\hat{V}(\boldsymbol{o}_t)$, instead, estimates the average return when starting from the state $\boldsymbol{o}_t$ and then following the policy for the next steps. We represent both the policy and the value function with two different neural networks composed of two fully connected layers, with 512 and 128 units each, followed by a linear output layer. The hidden units use a ReLU activation function. The networks do not share any layer.

Distributed Setup. The chosen PPO algorithm scales gracefully to a setup where the batch samples are collected from multiple workers in parallel. Our training setup is formed by 32 workers with an independent copy of the environment, and a trainer. After collecting a batch of 10 000 on-policy transitions, we train the neural networks with stochastic gradient descent. The optimizer uses minibatches containing 512 samples and performs 32 epochs per batch. The learning rate is $\lambda = 0.0001$. Each trial is stopped once it reaches 20 M agent steps, roughly equivalent to 7 days of experience on a real robot. Worker nodes run only on CPU resources, while the trainer has access to the GPU for accelerating the optimization process. We use the RLlib [43] framework, OpenAI Gym, and distributed training.

VI. RESULTS

A. Training Performance

Fig. 2 reports the learning curves of the average reward and episode duration over 11 independent agent training runs. Average reward across trials exhibits consistent growth and low variance (Fig. 2, left). We have also observed increasing values for all individual reward elements during training. Episode duration improves as well across trials and displays low variance (see Fig. 2, right), approaching maximum episode length more frequently as training progresses.

B. Emerging Behaviors

Controlling the upper body enables rich recovery behaviors that involve the control of the total momentum of the kinematic structure. We succeed in triggering such behaviors applying external forces during policy training. To make force profiles more realistic, instead of applying constant forces for a fixed interval as during training, we throw high-speed objects towards the balanced robot. Figure 4(b) shows two characteristic sequences. A larger variety of push-recovery strategies are displayed in the supplementary video: `https://dic-iit.github.io/emergence-push-recovery-icub/`.

C. Deterministic Planar Forces

We evaluate the push-recovery performance from horizontal forces. Forces are applied for 0.2 s after 3 s from the simulation start, when the robot is stably standing still and front-facing. Success is defined if the robot is still standing after 7 s. In Fig. 3(a), success rates for forces pointing in 12 directions are reported. Magnitudes increase from 50 N to 700 N at 25 N intervals. 5 repetitions are performed for each magnitude and direction, randomizing the initial joints configuration by adding zero-mean Gaussian noise ($\sigma = 2$ deg). Magnitudes within the training range (0-200 N) are counteracted successfully. Remarkably, the policy is also robust to out-of-sample forces in all directions in (200-300 N), up to 400 N in some directions. Moreover, it successfully recovers from pushes in the training range (0-200 N) even with an out-of-sample test friction coefficient $\mu_c = 0.2$ (Fig. 3(b)).

D. Random Spherical Forces on the Base Links

We evaluate policy robustness in challenging scenarios involving sequences of random forces with different combinations of magnitude and duration. Forces are applied to the base in a random direction more frequently than during training, on average every 3 s. For each combination, 50 reproducible episodes with different seed initialization and no domain randomization are executed. Episodes terminate if the robot falls or after 60 s, averaging 20 applications in a full episode. Our evaluation metric is the number of consecutive forces endured by the robot. Fig. 5 reports aggregate results for each combination of magnitude and duration. No matter their magnitude, forces lasting 0.1 s are properly balanced. As expected, performances decrease with growing magnitude and duration. Nevertheless, the agent is able to withstand repeated applications of out-of-sample forces. For instance, on average it withstands 9 consecutive 300 N 0.2 s applications.

E. Random Spherical Forces on the Chest and Elbow Links

We also evaluate robustness of the learned policy to previously unseen forces applied to other links. Fig. 5 shows the results obtained on the chest and elbow links. As expected, forces applied on links which are far from the CoM turn out to be more challenging. Nevertheless, the policy is able to withstand a good number of them and generalize with good performances. For instance, it is on average able to recover from 10 consecutive 200 N 0.2 s forces on the elbow link, as opposed to an average of 17 for the base link. The average number of consecutive counter-balanced forces with the same magnitude and duration decreases to 5 for the chest link. Notice that the randomness of the interval between two subsequent forces applications leads sometimes to very challenging scenarios in which multiple forces are applied in a very short time span.

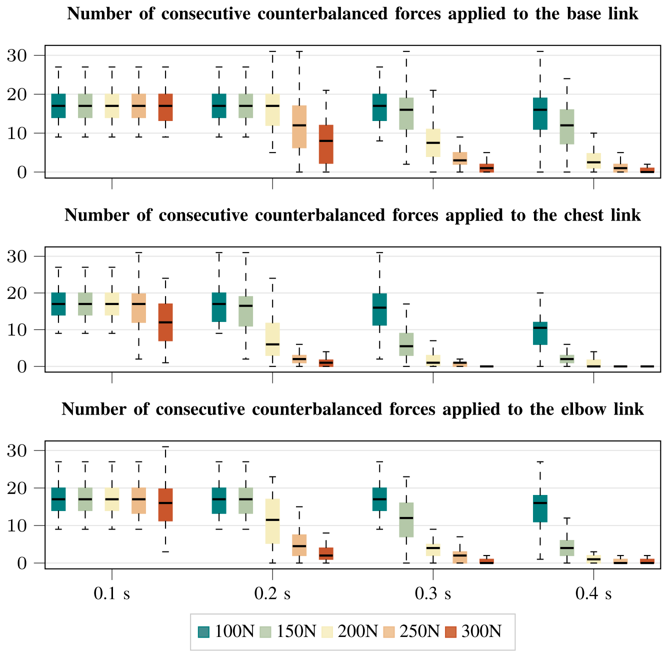

Fig. 5. Consecutive counterbalanced forces in random directions over 50 trials for each combination of magnitude and duration. Forces are applied to the base, chest, and elbow links for an increasing duration.

VII. Discussion

Learning efficiency. The overall experience for a single policy training lasts approximately 7 simulated days. As for other continuous control tasks, model-free PG methods lack sample efficiency. There is plenty of room for robot learning research to bridge this efficiency gap. Indeed, floating-base robots such as iCub can be modeled quite accurately with rigid body dynamics. Most robots used in research are provided with a dynamic model accurate enough to be exploited as a powerful prior. The community has recently proposed interesting model-based algorithms [44] with the potential to improve efficiency and leverage decades of robotics research.

Low-level control. Low-level position control is widely adopted in other similar works. PID controllers have the advantage of being independent of each other and requiring single-joint signals. However, besides being difficult to tune, they trade off tracking accuracy with compliance. A stiff robot, in the presence of high perturbations, is less robust because even if the planner is whole-body, low-level control is not. Whole-body and intrinsically more compliant low-level controllers could be beneficial, although they often operate on the entire underactuated floating-base system. Properly handling the base references from the policy point of view is yet an uncharted domain.

Natural behaviour and sim-to-real. The emerged push-recovery strategies are not as natural as human ones. The policy tends to promote small jumps to full steps, probably due to two factors: the stiffness given by the low-level PIDs, and the difficulty of accurate contact modeling. As concerns low-level control, actuator dynamics plays a vital role. Our simulations introduce variable delay but do not saturate joint torques. Their minimization in the reward does not prevent occasional high torque spikes synthesized by the PIDs. The integration of more realistic actuator models will be explored in future work. Regarding contacts, modeling differences between physics engines notably make policies hardly transferable to different engines or the real world. The simulator we adopt, Ignition Gazebo, will soon provide a transparent physics engine switch, enabling randomization of the entire engine beyond the common physics parameters.

VIII. Conclusion

We present a DRL-based control architecture capable of learning whole-body balancing and push recovery for simulated humanoids. We promote exploration by applying random forces to the kinematic structure, leading to the emergence of a variety of push-recovery behaviors. Compared to previous works, our policy controls most of the robot's joints, and we show that this contributes to extending the space of recovery motions to whole-body strategies. We have shown the results of our architecture controlling 23 DoF of the iCub robot, and showing that our policy can withstand repeated applications of strong external pushes.

Our approach shows different types of limitations. The PID controllers, while providing a simple low-level control, introduce a stiffness that can prevent natural motion and introduce a joint dynamics that differs from the real platform. The learning efficiency of model-free algorithms is pretty low and requires days of simulations for a complex behaviour to emerge. Finally, relying only on state space exploration for finding the expected behaviours requires a carefully designed reward function, that might require a significant effort. These limitations could be mitigated by introducing prior knowledge in the training scheme, like the usage of model-based whole-body controllers for the low-level and more accurate actuator modeling in simulation, and model-based reinforcement learning. We plan to explore some of these directions in future work with the aim to bring our policies to the real robot.

Appendix A
RBF Reward Kernel

Radial basis function (RBF) kernels are widely employed functions in machine learning, defined as

$$K(\mathbf{x}, \mathbf{x}^*) = \exp\left(-\tilde{\gamma}||\mathbf{x} - \mathbf{x}^*||^2\right) \quad \in [0, 1],$$

where $\tilde{\gamma}$ is the kernel bandwidth hyperparameter. The RBF kernel measures similarities between input vectors. This can be useful for defining scaled reward components. In particular, if $\mathbf{x}$ is the current measurement and $\mathbf{x}^*$ is the target, the kernel provides a normalized estimate of their similarity. $\tilde{\gamma}$ can be used to tune the bandwidth of the kernel, i.e. its sensitivity. In particular, we use $\tilde{\gamma}$ to select the threshold from which the kernel tails begin to grow. Introducing the pair (x_c, ϵ), with $x_c, \epsilon \in \mathbb{R}^+$ and $|\epsilon| \ll 1$, we can parameterize $\tilde{\gamma} = -\ln(\epsilon)/x_c^2$. This formulation results in the following properties:

1) $K(\mathbf{x}^*, \mathbf{x}^*) = 1$, i.e. when the measurement reaches the target, the kernel outputs 1;
2) Given a measurement $\mathbf{x}_m$ such that $||\mathbf{x}_m - \mathbf{x}^*|| = x_c$, the kernel outputs $K(\mathbf{x}_m, \mathbf{x}^*) = \epsilon$.

In practice, ϵ can be kept constant for each reward component. The sensitivity of individual components are tuned by adjusting

x_c. We refer to x_c as *cutoff* value of the kernel, since each norm of the distance in the input space bigger than x_c yields output values smaller than ϵ. This formulation eases the composition of the total reward r_t when reward components are calculated from measurements of different dimensionalities and scales. In fact, once the sensitivities have been properly tuned for each component, they can simply be weighted differently as $r_t = \sum_i w_i K(\mathbf{x}_t^{(i)}, \mathbf{x}^*) \in \mathbb{R}$ where $\mathbf{x}_t^{(i)}$ is the i-th measurement sampled at time t, and $w_i \in \mathbb{R}$ the weight corresponding to the i-th reward component.

References

[1] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," ICLR, 2019.

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

[3] G. Dulac-Arnold *et al.*, "An empirical investigation of the challenges of real-world reinforcement learning," 2020, *arXiv:2003.11881*.

[4] O. Sigaud and F. Stulp, "Policy search in continuous action domains: An overview," *Neural Netw.*, vol. 113, pp. 28–40, 2019. https://www.sciencedirect.com/science/article/abs/pii/S089360801930022X

[5] P. Christiano *et al.*, "Transfer from simulation to real world through learning deep inverse dynamics model," 2016, *arXiv:1610.03518*.

[6] F. Muratore, M. Gienger, and J. Peters, "Assessing transferability from simulation to reality for reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 4, pp. 1172–1183, Apr. 2021.

[7] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the atlas robot," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 120–127.

[8] S. Kuindersma *et al.*, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Auton. Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[9] S. Dafarra, F. Romano, and F. Nori, "Torque-controlled stepping-strategy push recovery: Design and implementation on the iCub humanoid robot," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots*, 2016, pp. 152–157.

[10] G. Romualdi, S. Dafarra, Y. Hu, and D. Pucci, "A benchmarking of DCM based architectures for position and velocity controlled walking of humanoid robots," in *Proc. IEEE-RAS 18th Int. Conf. Humanoid Robots*, 2018, pp. 1–9.

[11] S. Dafarra *et al.*, "A control architecture with online predictive planning for position and torque controlled walking of humanoid robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[12] C. Yang, K. Yuan, W. Merkt, T. Komura, S. Vijayakumar, and Z. Li, "Learning whole-body motor skills for humanoids," in *Proc. IEEE-RAS 18th Int. Conf. Humanoid Robots*, 2018, pp. 270–276.

[13] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Expanding Societal Role Robot. Next Millennium (Cat. No.01CH37180)*, 2001, pp. 239–246.

[14] F. Nori, S. Traversaro, J. Eljaik, F. Romano, A. Del Prete, and D. Pucci, "iCub whole-body control through force regulation on rigid non-coplanar contacts," *Frontiers Robot. AI*, vol. 2, p. 6, 2015, https://www.frontiersin.org/article/10.3389/frobt.2015.00006

[15] S. Traversaro, D. Pucci, and F. Nori, "A unified view of the equations of motion used for control design of humanoid robots," 2017, https://www.researchgate.net/publication/312200239.

[16] B. Stephens, "Humanoid push recovery," in *Proc. 7th IEEE-RAS Int. Conf. Humanoid Robots*, 2007, pp. 589–595.

[17] Z. Li, C. Zhou, Q. Zhu, and R. Xiong, "Humanoid balancing behavior featured by underactuated foot motion," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 298–312 Apr. 2017.

[18] T. Koolen, M. Posa, and R. Tedrake, "Balance control using center of mass height variation: Limitations imposed by unilateral contact," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots*, 2016, pp. 8–15.

[19] M. Vukobratovic, A. A. Frank, and D. Juricic, "On the stability of biped locomotion," *IEEE Trans. Biomed. Eng.*, vol. BME-17, no. 1, pp. 25–36, Jan. 1970.

[20] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba, "Online decision of foot placement using singular LQ preview regulation," in *Proc. 11th IEEE-RAS Int. Conf. Humanoid Robots*, 2011, pp. 13–18.

[21] P.-b. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *Proc. 6th IEEE-RAS Int. Conf. Humanoid Robots*, 2006, pp. 137–142.

[22] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Proc. 6th IEEE-RAS Int. Conf. Humanoid Robots*, 2006, pp. 200–207.

[23] T. Koolen, T. de Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models," *Int. J. Robot. Res.*, vol. 31, no. 9, pp. 1094–1113, 2012.

[24] J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schäffer, and G. Hirzinger, "Bipedal walking control based on capture point dynamics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 4420–4427.

[25] M. Morisawa, S. Kajita, F. Kanehiro, K. Kaneko, K. Miura, and K. Yokoi, "Balance control based on capture point error compensation for biped walking on uneven terrain," in *Proc. 12th IEEE-RAS Int. Conf. Humanoid Robots*, 2012, pp. 734–740.

[26] J. Tan *et al.*, "Sim-to-Real: Learning Agile Locomotion for Quadruped Robots," 2018, *arXiv:1804.10332*.

[27] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, 2019https://robotics.sciencemag.org/content/4/26/eaau5872.

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[29] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Hybrid zero dynamics inspired feedback control policy design for 3D bipedal locomotion using reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 8746–8752.

[30] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative reinforcement learning based design of dynamic locomotion skills for cassie," 2019, *arXiv:1903.09537*.

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

[32] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "DeepMimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–14, Jul. 2018.

[33] K. Bergamin, S. Clavet, D. Holden, and J. R. Forbes, "DReCon: Data-driven responsive control of physics-based characters," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–11, 2019.

[34] C. Yang, K. Yuan, S. Heng, T. Komura, and Z. Li, "Learning natural locomotion behaviors for humanoid robots using human bias," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2610–2617, Apr. 2020.

[35] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction,* 2nd Ed.*, Ser. Adaptive Computation and Machine Learning Series*. Cambridge, MA: MIT Press, 2018, Chapter 3, pp. 47–71, http://incompleteideas.net/book/RLbook2020.pdf.

[36] D. P. Bertsekas, "Reinforcement learning and optimal control," *Belmont, MA: Athena Scientific*, 2019.

[37] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Adv. Neural Inf. Process. Syst.*, vol. 99, pp. 1057–1063, 1999.

[38] D. Ferigo, S. Traversaro, G. Metta, and D. Pucci, "Gym-Ignition: Reproducible Robotic Simulations for Reinforcement Learning," in *Proc. IEEE/SICE Int. Symp. Syst. Integration*, 2020, pp. 885–890.

[39] G. Brockman *et al.*, "OpenAI gym," 2016, *arXiv:1606.01540*.

[40] J. Lee *et al.*, "DART: Dynamic animation and robotics toolkit," *J. Open Source Softw.*, vol. 3, no. 22, 2018, Art. no. 500.

[41] C. Yang, T. Komura, and Z. Li, "Emergence of human-comparable balancing behaviours by deep reinforcement learning," in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robotics*, 2017, pp. 372–377.

[42] G. Nava, F. Romano, F. Nori, and D. Pucci, "Stability analysis and design of momentum-based controllers for humanoid robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 680–687.

[43] E. Liang *et al.*, "RLlib: Abstractions for distributed reinforcement learning", *Int. Conf. Mach. Learn.* arXiv, pp. 3053–3062, 2018.

[44] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," 2020, *arXiv:2006.16712*.