Reactive Stepping for Humanoid Robots using Reinforcement Learning: Application to Standing Push Recovery on the Exoskeleton Atalante

Alexis Duburcq, Fabian Schramm, Guilhem Boéris, Nicolas Bredeche, Yann Chevaleyre

Reactive Stepping for Humanoid Robots using Reinforcement Learning: Application to Standing Push Recovery on the Exoskeleton Atalante

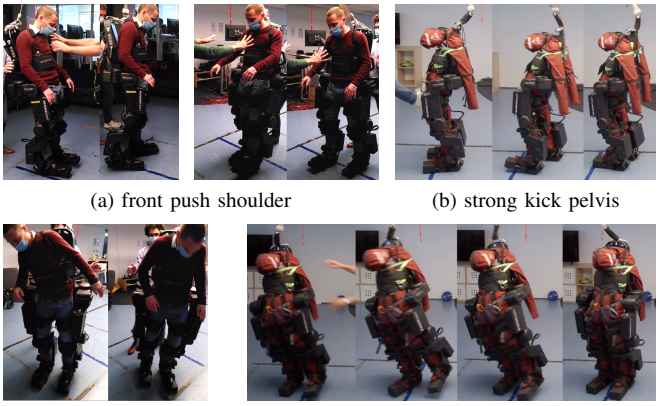
Alexis Duburcq[1,2,3], Fabian Schramm[1], Guilhem Boéris[1], Nicolas Bredeche[2] and Yann Chevaleyre[3]

***Abstract*— State-of-the-art reinforcement learning is now able to learn versatile locomotion, balancing and push-recovery capabilities for bipedal robots in simulation. Yet, the reality gap has mostly been overlooked and the simulated results hardly transfer to real hardware. Either it is unsuccessful in practice because the physics is over-simplified and hardware limitations are ignored, or regularity is not guaranteed, and unexpected hazardous motions can occur. This paper presents a reinforcement learning framework capable of learning robust standing push recovery for bipedal robots that smoothly transfer to reality, providing only instantaneous proprioceptive observations. By combining original termination conditions and policy smoothness conditioning, we achieve stable learning, sim-to-real transfer and safety using a policy without memory nor explicit history. Reward engineering is then used to give insights into how to keep balance. We demonstrate its performance in reality on the lower-limb medical exoskeleton Atalante.**

I. Introduction

Achieving dynamic stability for bipedal robots is one of the most complex tasks in robotics. Continuous feedback control is required to keep balance since the vertical posture is inherently unstable [1]. However, hybrid high-dimensional dynamics, kinematic redundancy, model and environment uncertainties, and hardware limitations make it hard to design robust embedded controllers. Offline trajectory planning for bipedal robots has been solved successfully through whole-body optimization [2], [3]. In particular, stable walking on flat ground and without disturbances was achieved on the exoskeleton Atalante [4]. Yet, online re-planning, robustness to uncertainties and emergency recovery in case of external perturbations are still very challenging.

Modern control approaches require a lot of expert knowledge and effort in tuning because of discrepancies between approximate models and reality. Solutions are mainly task-specific, and improving versatility is usually done by stacking several estimation and control strategies in a modular hierarchical architecture [5]–[7]. Though efficient in practice, it makes the analysis and tuning increasingly difficult and thereby limits its capability. In contrast, deep reinforcement learning (RL) methods require expert knowledge and extensive efforts to tailor the optimization problem, rather than structuring explicit controllers and defining good approximate models. RL aims at solving observation, planning and control as a unified problem by end-to-end training of a policy [8]. Tackling the problem globally maximizes the potential of this method, but state-of-the-art algorithms still face learning instabilities and difficulties in discovering satisfactory behaviors for practical applications.



(a) front push shoulder (b) strong kick pelvis

(c) random user moves (d) side push hip

Fig. 1. Scenarios showing the robustness of the push recovery policy: smoothly transferred to different users, exoskeleton setups and force profiles.

A ubiquitous problem of controllers trained with deep RL is the lack of safety and smoothness. This is dangerous for real-life deployment as human beings cannot anticipate future motions. Without special care, the control varies discontinuously like a bang-bang controller, which can result in a poor transfer to reality, high power consumption, loud noise and system failures [9]. Despite those potential limitations, a robust gait and push recovery for the bipedal Cassie robot was recently learned in simulation using deep RL and transferred successfully to the real device [10]. Concurrently, several works on standing push recovery for humanoid robots trained in simulation suggest that the approach is promising [11], [12], although the same level of performance has not been achieved on real humanoid robots yet.

Our main contribution is the development of a purely reactive controller for standing push recovery on legged robots using deep RL, which is used as the last resort fallback in case of emergency. Precisely, we design an end-to-end policy featuring a variety of balancing strategies from the latest proprioceptive sensor data only, while promoting predictable, safe and smooth behavior. Our method combines carefully designed termination conditions, reward shaping and so-called policy smoothness conditioning [9]:

- Termination conditions improve sample efficiency and learning stability by limiting the search space. Besides, they can enforce hard constraints such as hardware limitations and safety. We can generate sensible policies regardless of the reward function.

[1]Wandercraft, Paris, France. <alexis.duburcq@gmail.com>

[2]Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France.

[3]Université Paris-Dauphine, PSL, CNRS, Laboratoire d'analyse et modélisation de systèmes pour l'aide à la décision, Paris, France.

- Reward engineering improves exploration efficiency and promotes more natural behaviors.
- Smoothness conditioning favors few versatile strategies over many locally optimal behaviors. Slow motions are less sensitive to modeling errors including the internal dynamics of the feedback loop, so it alleviates the need for transfer learning approaches known for trading performance over robustness, e.g. domain randomization.

Some emerging strategies would be very challenging to reproduce using classical model-based control. Moreover, the policy can be directly transferred to a real robot. We demonstrate experimentally safe and efficient recovery behaviors for strong perturbations on the exoskeleton Atalante carrying different users, as shown in the supplementary video[1].

II. Related Work

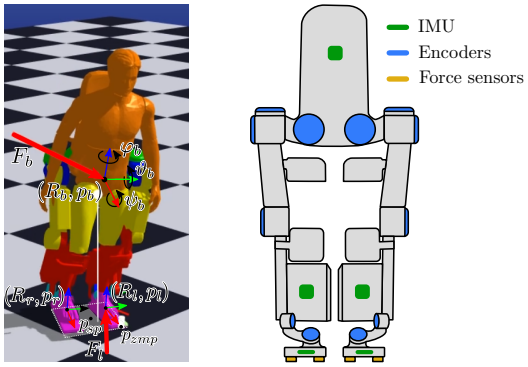
A. Classical Non-Linear Control

Upright standing offers a large variety of recovery strategies that can be leveraged in case of emergency to avoid falling down, among them: ankle, hip, stepping, height modulation and foot-tilting for any legged robot, plus angular momentum modulation for humanoid robots [13]. For small perturbations, in-place recovery strategies controlling the Center of Pressure (CoP) [14], the centroidal angular momentum [15], or using foot-tilting [16], [17] are sufficient. To handle stronger perturbations, controllers based on Zero-Moment Point (ZMP) trajectory generation have been proposed [18], along with Model Predictive Control (MPC) methods controlling the ZMP [19], but in practice the efficiency was limited. More recently, approaches based on the Capture Point (CP) showed promising results on real robots [1]. The Linear Inverted Pendulum (LIP) model [20] is used to make online computations tractable. However, due to partial model validity, it tends to be restricted to moderately fast and conservative motions.

B. Deep Reinforcement Learning

Several ground-breaking advances were made in deep RL during the last decade. It has shown impressive effectiveness at solving complex continuous control tasks for toy models [21], [22], but real-world applications are still rare. Lately, deep RL was used to learn locomotion policies for dynamic and agile maneuvers on the quadruped ANYmal, which were smoothly transferred to reality [23], [24]. Besides, extremely natural motions were obtained on various simplified models by leveraging reference motion data from motion capture in an imitation learning framework [25], [26]. Walking on flat ground, learned without providing any reference trajectories, was demonstrated on a mid-size humanoid [27]. However, the motion was slow and unnatural, with limited robustness to external forces. Promising results were achieved in simulation by several authors concurrently regarding standing push recovery [11], [28], [29]. Yet, robust locomotion and standing push recovery for humanoid robots using deep RL falls short from expectations on real devices. The emerging behaviors are often unrealistic or hardly transfer to reality.

[1] https://youtu.be/HLx6CHfpmBM



(a) Jiminy simulator (b) Actuators and sensors

Fig. 2. Atalante in learning environment and hardware overview.

C. Simulation to Real World Transfer

Various policies have been trained in simulation for the bipedal robot Cassie and successfully transferred to reality. In [30], stable walking has been achieved without disturbances based on reference trajectories generated using a traditional model-based method. Domain randomization is not needed because the policy predicts targets for a low-level PD controller and the simulation is faithful. The latter cannot be expected for an exoskeleton because of the unknown user dynamics. Concurrently, it was generalized to a whole gait library [10] and no reference at all [31]. Domain randomization enables to deal with disturbances for which it was never trained, including pushes. Although effective, it leads to more conservative behaviors than necessary. It can even prevent learning anything if the variability is not increased progressively [10]. Alternatively, the stability can be improved by predicting high-level features for a model-based controller [32], but it bounds the overall performance. Besides, a memory network or a history of previous timesteps is often used to thwart partial observability of the state [10], [31]. It improves robustness to noise and model uncertainty, but it makes the training significantly more difficult [24].

The efficiency of RL-based controllers is limited in practice by the lack of smoothness in the predicted actions, which can be mitigated by filtering. Although it can be sufficient [10], it is known to make policies unstable or underperforming [9]. In contrast, the Conditioning for Action Policy Smoothness (CAPS) method adjusts the smoothness of the policy itself [9], by combining temporal and spatial regularization of the learning problem. It enables transfer to reality without domain randomization. Nevertheless, the formulation [9] has flaws that we address in the current work: First, it penalizes exploration, impeding the learning algorithm to converge to the optimal solution. Second, it prevents bursts of acceleration which are necessary to recover from strong pushes, leading to an underperforming policy.

III. Background

A. The Reinforcement Learning Problem

RL methods aim at training an agent to solve an episodic task through an iterative process of trial and error. The agent interacts with its environment in a discrete-time setting: at

each time step t, the agent observes o_t based on the state of the system s_t, receives a reward r_t and subsequently performs an action a_t given this information. It then moves to the next state s_{t+1} and receives the reward r_{t+1} associated with the transition (s_t, a_t, s_{t+1}). The interaction generates trajectories $\tau = \{(s_i, a_i, r_i)\}_{i=0}^{T}$ whose length T depends on termination conditions. Formally, this problem can be described as an infinite-horizon Markov Decision Process (MDP) [8]. It is defined by a state space $S \in \mathbb{R}^n$, observation space $O \in \mathbb{R}^l$, valid action space $A \in \mathbb{R}^m$, transition function $p : S \times A \times S \to [0,1]$, observation function $o : S \times A \to O$, reward function $r : S \times A \times S \to \mathbb{R}$ and discount factor $\gamma \in [0,1]$. The transition function p encodes the dynamics of the agent in the world.

The agent's behavior is determined by a stochastic policy $\pi(\cdot|s_t)$ mapping states to distributions over actions $Pr(A)$. The goal is to find a policy π^* maximizing the return $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$, i.e. the discounted cumulative reward, in expectation over the trajectories τ induced by the policy:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} J(\pi) \text{ s.t. } J(\pi) = \mathbb{E}_{\tau \sim \pi}\left[R(\tau)\right].$$

The discount factor enables a trade-off between long-term vs. short-term profit, which is critical for locomotion tasks.

B. Policy Gradient Methods

In deep RL, the policy is a neural network with parameters θ. A direct gradient-based optimization of θ is performed over the objective function $J(\pi_\theta)$. The analytical gradient of the return is unknown, so it computes an estimate of $\nabla_\theta J(\pi_\theta)$ and applies a stochastic gradient ascent algorithm. It yields

$$\nabla_\theta J(\pi_\theta) = \hat{\mathbb{E}}_{\tau \sim \pi_\theta}\left[\nabla_\theta \log(\pi_\theta(a_t|s_t)) A_t\right],$$

where $\hat{\mathbb{E}}$ is the expectation over a finite batch of trajectories and $A_t = R_t - V(s_t)$ is the advantage function corresponding to the difference between the future return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ and the value function $V(s_t) = \hat{\mathbb{E}}_{\tau \sim \pi}[R_t|s_t]$.

Usually, the value function $V(s_t)$ is a neural network itself, trained in conjunction with the policy. Using it as a baseline reduces the variance of the gradient estimator. However, the gradient estimator still suffers from high variance in practice, weakening the convergence and asymptotic performance.

Up to now, Proximal Policy Optimization (PPO) [33] is the most reliable policy gradient algorithm for continuous control tasks. It is an actor-critic algorithm, which means that both the policy and the value function are trained. The policy is normally distributed with parametric mean μ_θ but fixed standard deviation σ. It tackles learning instability by introducing a surrogate that gives a conservative gradient estimate bounding how much the policy is allowed to change:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the likelihood ratio and $\hat{A}_t$ is the Generalized Advantage Estimator (GAE) [34]. The latter offers an additional parameter λ over the naive estimate A_t to adjust the bias vs. variance trade-off.

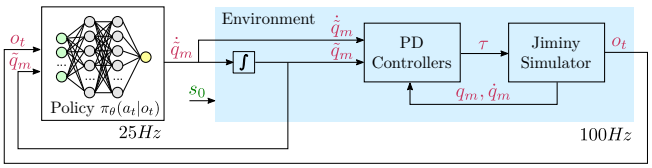


Fig. 3. Overview of the proposed control system.

IV. Methodology

A. Learning Environment

1) Atalante Model: Atalante is a crutch-less lower-limb exoskeleton for people with disabilities. It has 6 actuators on each leg and a set of basic proprioceptive sensors, see Fig. 2b. The patient is rigidly fastened to the exoskeleton, which features dimensional adjustments to fit the individual morphology. In this regard, the system exoskeleton-patient can be viewed as a humanoid robot after merging their respective mass distributions. Contrary to usual humanoids, the control of the upper body is partial. For learning, a single average patient model is used.

We use the open-source simulator Jiminy [35] based on Pinocchio [36], see Fig. 2a. It was originally created for classic control robotic problems and realistic simulation of legged robots. Jiminy includes motor inertia and friction, sensor noise and delay, constraint-based contact forces [37], as well as an advanced mechanical flexibility model [38].

2) Action Space: A distinctive feature in RL is the slow update frequency in contrast to classic control approaches, giving enough time for the effect of the actions to build up. First, it is critical for exploration efficiency, which relies on random action sampling. Second, it reduces the variance of the gradient estimate by increasing the signal-to-noise ratio.

The policy is trained to predict targets that are forwarded to decentralized low-level PD controllers running at higher frequency. Decoupled control of each joint is well-known to be robust to model uncertainties. Thus, this hybrid control architecture improves robustness and transferability compared to predicting motor torques u directly. Moreover, these controllers can be tuned to trade off tracking accuracy versus compliance, smoothing out vibrations and errors in the predicted actions. In this work, the policy predicts target motor velocities at 25 Hz, which are further integrated to target motor positions to enforce a smooth behavior. We forward consistent target motor positions and velocities to the low-level PD controllers running at 100 Hz, see Fig. 3. It is necessary for accurate tracking, especially for dynamic motions. This consistency issue was always disregarded in related works since it does not affect performance but rather predictability and safety.

3) State Space and Observation Space: The user is not modeled explicitly but rather considered as an external disturbance. Hence, the theoretical state of the system s_t is fully determined by the position p_b, roll-pitch-yaw orientation ψ_b, θ_b, ϕ_b, linear velocity v_b and angular velocity ω_b of the pelvis of the robot, plus the motor positions q_m and velocities $\dot{q}_m$. Even so, the state s_t is only partially observable. The observation $o_t \in \mathcal{O} \in \mathbb{R}^{49}$ is computed from low-cost

proprioceptive sensors: one inertial measurement unit (IMU) estimating the roll, pitch and angular velocity of the pelvis, 8 vertical force sensors in both feet $F^z_{r,l}$ and 12 motor encoders measuring the motor positions. The motor velocities $\dot{q}_m$ are obtained by numerical differentiation. Additionally, the target motor positions $\tilde{q}^{t-1}_m$ from the last time step are included in the observation. No window over several time steps is aggregated. The quantities are independently normalized over training batches. Their distributions change over training iterations since it depends on the current policy, hence it would be hazardous to do it manually.

Some insightful quantities cannot be reliably estimated without exteroceptive sensors, e.g. the pelvis height z_b and linear velocity v_b. They are not included in the observation space because any significant mismatch between simulated and real data may prevent transfer to reality. Although the odometry pose $p_o = [x_b, y_b, \varphi_b]^T$ is not observable, it is not blocking as the recovery strategies should be invariant to it.

4) External Disturbances Modeling: To learn sophisticated recovery strategies, the external pushes in the learning environment need to be thoughtfully scheduled. They must be strong enough to sometimes require stepping, but pushing too hard would prohibit learning. As suggested in [11], we apply forces of constant magnitude for a short duration periodically on the pelvis, where the orientation is sampled from a spherical distribution. In this work, the pushes are applied every 3s, with a jitter of 2s to not overfit to a fixed push scheme and learn recovering consecutive pushes. The pushes are bell-shaped instead of uniform for numerical stability. They have a peak magnitude of $\max \|F_b\|_2 = 800$N and are applied during 400ms. Premature convergence to a suboptimal policy was observed if the force is gradually increased from 0 to 800N over the episode duration instead.

B. Initialization

The initial state distribution $\rho_0(s_0) : S \to [0,1]$ defines the probability of an agent to start the episode in state s_0. It must ensure the agent has to cope with a large variety of situations to promote robust recovery strategies. Ideally, it should span all recoverable states, but this set is unknown as it depends on the optimal policy. Naive random sampling would generate many unrecoverable states or even trigger termination conditions instantly. Such trajectories have limited value and can make learning unstable.

We propose to sample the initial state uniformly among a finite set of reference trajectories $t \to (\hat{q}(t), \hat{\dot{q}}(t))$, and then to add Gaussian noise to increase the diversity. The trajectories correspond to the range of motions in everyday use: standing, walking forward or backward at different speeds and turning. They are generated beforehand using a traditional model-based planning framework for the average patient morphology [4]. Therefore, the corresponding states are feasible and close to dynamic stability.

C. Termination Conditions

Bounding quantities in simulation is not helping to enforce hard constraints in reality unlike termination conditions. The latter stop the reward from accumulating. Assuming the reward must be always positive, it caps the return to a fraction of its maximum. The agent becomes risk-averse: being confident about preventing critical failure in a stochastic world requires extra caution. Otherwise, it is detrimental because the agent tends to kill itself on purpose. Terminal conditions are the counterpart to log barrier penalties in constrained optimization. It is complementary to strict safety guarantees at runtime and does not allow for getting rid of them.

One key contribution of this article is a set of carefully designed termination conditions. First, they ensure transfer to reality and safety. Secondly, they reduce the search space to sensible solutions only. Some local minima of poor recovery strategies are strongly discouraged from the start, which leads to more stable learning and faster convergence [39]. Numerical values are robot-specific. Unless stated otherwise, they were obtained by qualitative study in simulation.

1) Pelvis Height and Orientation: We restrict the pose of the pelvis to avoid frightening the user and people around,

$$z_b > 0.3, \quad -0.4 < \psi_b < 0.4, \quad -0.25 < \vartheta_b < 0.7 \, .$$

2) Foot Collisions: For safety, foot collision needs to be forestalled as it can hurt the patient and damage the robot,

$$\mathscr{D}(CH_r, CH_l) > 0.02,$$

where CH_r, CH_l are the convex hulls of the right and left footprints respectively, and $\mathscr{D}$ is the euclidean distance.

3) Joint Bound Collisions: Hitting the mechanical stops q^-, q^+ is inconvenient but forbidding it completely is not desirable. Considering PD controllers and bounded torques, it induces safety margins that constrain the problem too strictly. It would impede performance while avoiding falling is the highest priority. Still, the impact velocity must be restricted to prevent destructive damage or injuries. An acceptable threshold has been estimated from real experimental data,

$$|\dot{q}_i| < 0.6 \text{ or } q_i^- < q_i < q_i^+ .$$

4) Reference Dynamics: Long-term drift of the odometry position is inevitable, but it must be limited. The user is only supposed to rectify it occasionally and the operational space is likely to be confined in practice. We restrict the odometry displacement over a time window Δp_o instead of its instantaneous velocity [39]. It limits the drift while allowing a few recovery steps. In general, the expected odometry displacement must be subtracted if non-zero.

$$|\Delta p_o - \Delta \hat{p}_o| < [2.0, 3.0, \pi/2],$$

where $\Delta\star = \star(t) - \star(t - \Delta T)$ and $\Delta T = 20$s.

5) Transient Dynamics: The robot must track the reference if there is no hazard, only applying minor corrections to keep balance. Rewarding the agent for doing so is not effective as favoring robustness remains more profitable. Indeed, it would anticipate disturbances, lowering its current reward to maximize the future return, primarily averting termination. We need to allow large deviations to handle strong pushes but also urge to quickly cancel it afterwards.

$$\min_{t' \in [t-\Delta T, t]} \left\| q_m(t') - \hat{q}_m(t') \right\|_2 < 0.3 \ , \ \Delta T = 4\text{s}.$$

6) Power Consumption: We limit the power consumption to fit the hardware capability and avoid motor overheating,

$$P = \langle u, \dot{q}_m \rangle \ < 3\text{kW}.$$

D. Reward Engineering

We designed a set of cost components that provides insight into how to keep balance. Additionally, we use them as a means to trigger reactive steps only if needed, as it is a last resort emergency strategy. We aim to be generic, so they can be used in conjunction with any reference trajectory to enhance stability and provide recovery capability. The total reward is a normalized weighted sum of the individual costs

$$r = \sum\nolimits_i w_i K(c_i),$$

where c_i is a cost component, w_i its weight and K a kernel function that scales them equally. We use the Radial Basis Function (RBF) with cutoff parameters κ_i

$$K(c_i) = \exp(-\kappa_i c_i^2) \ \in [0,1].$$

The gradient vanishes for both very small and large values as a side effect of this scaling. The cutoff parameter κ is used to adjust the operating range of every reward component.

1) Reference Dynamics: We encourage tracking a specific reference trajectory, which boils down to a stable resting pose for standing push recovery. Evaluating the tracking discrepancy by computing the absolute motor position error wrt a reference has several limitations:

- Tendency to diverge because of high-dimensionality. One large scalar error is enough for the whole gradient to vanish because of the kernel.
- Inability to enforce high-level features independently, e.g. the position and orientation of the feet.
- Quantities like p_o are allowed to drift, which requires using velocity error instead of position.

To overcome them, we extract a set of independent high-level features and define reward components for each of them. As they are unrelated, even if one feature is poorly optimized, it is possible to improve the others.

Odometry. The real environment may be cluttered. For the recovery strategies to be of any use, the robot must stay within a radius around the reference position in world plane,

$$\|\bar{v}_o - \hat{\bar{v}}_o\|_2,$$

where $\bar{\star}$ denotes the average since the previous step.

Reference configuration. The robot should track the reference when no action is required to keep balance. Besides, it must not try any action if falling is inevitable at some point. It is essential to prevent dangerous motions in any situation.

$$\|q_m - \hat{q}_m\|_2.$$

Foot positions and orientations. The feet should be flat on the ground at a suitable distance from each other. Without promoting it specifically, the policy learns to spread the legs, which is both suboptimal and unnatural. One has to work in a symmetric, local frame, to ensure this reward is decoupled from the pelvis state, which would lead to unwanted side effects otherwise. We introduce the mean foot quaternion $q_{r+l} = (q_r + q_l)/2$ and define the relative position and orientation of the feet in this frame,

$${}^{r+l}p_{r\text{-}l} = R_{r+l}^T(p_r - p_l), \ {}^{r+l}R_{r\text{-}l} = R_{r+l}^T(R_r R_l^T).$$

The two rewards for the foot positions and orientations are

$$\|{}^{r+l}p_{r\text{-}l} - {}^{r+l}\hat{p}_{r\text{-}l}\|_2, \ \|\log({}^{r+l}R_{r\text{-}l}\,{}^{r+l}\hat{R}_{r\text{-}l}^T)\|_2.$$

2) Transient Dynamics: Following a strong perturbation, recovery steps are executed to prevent falling in any case.

Foot placement. We want to move the feet as soon as the CP goes outside the support polygon. We encourage moving the pelvis toward the CP to get it back under the feet,

$$\|{}^b p_{cp} - {}^b\hat{p}_{cp}\|_2,$$

where ${}^b p_{cp}$ is the relative CP position in odometry frame.

Dynamic stability. The ZMP should be kept inside the support polygon for dynamic stability,

$$\|p_{zmp} - p_{SP}\|_2,$$

where *SP* is the center of the vertical projection of the footprints on the ground. It anticipates future impact with the ground and is agnostic to the contact states.

3) Safety and Comfort: Safety is critical during the operation of a bipedal robot. Comfort is also important for a medical exoskeleton to enhance rehabilitation.

Balanced contact forces. Distributing the weight evenly on both feet is key in natural standing,

$$|F_r^z \hat{\delta}_r + F_l^z \hat{\delta}_l - mg|,$$

where $\hat{\delta}_r, \hat{\delta}_l \in \{0,1\}$ are the right and left contact states and F_r^z, F_l^z are the vertical contact forces.

Ground friction. Reducing the tangential contact forces limits internal constraints in the mechanical structure that could lead to overheating and slipping. Moreover, exploiting too much friction may lead to unrealistic behaviors,

$$\|F_{r,l}^{x,y}\|_2,$$

where $F_{r,l}^{x,y}$ gathers the tangential forces on the feet.

Pelvis momentum. Large pelvis motion is unpleasant. Besides, reducing the angular momentum helps to keep balance,

$$\|\bar{\omega}_b - \hat{\bar{\omega}}_b\|_2.$$

E. Policy Conditioning for Smoothness and Safety

Safe and predictable behavior is critical for autonomous systems evolving in human environments such as bipedal robots. Beyond this, jerky commands are not properly simulated and hardly transfer to reality, let alone it is likely to damage the hardware or shorten its lifetime. Ensuring smoothness of the policy during the training mitigates these issues without performance side effects. In the case of a medical exoskeleton, smoothness is even more critical. Vibrations can cause anxiety, and more importantly, lead to injuries over time since patients have fragile bones.

In RL, smoothness is commonly promoted through reward components, e.g. the total power consumption or the norm of

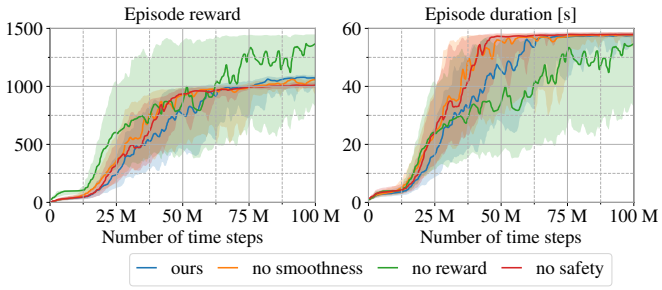


Fig. 4. Ablation study of our proposed reward formulation, policy regularization for smoothness and termination conditions for safety. The standard deviation is taken over episodes per batch.

the motor velocities [11], [28]. This approach is unreliable because it is actually the return that is maximized and not the instantaneous reward. The agent would disregard any reward component that increases the likelihood of falls. Adding up regularization terms to the loss function directly gives us control over how much it is enforced during the learning.

We use temporal and spatial regularization to promote smoothness of the learned state-to-action mappings of the neural network controller. The temporal regularization term

$$L^T(\theta) = \|\mu_\theta(s_t) - \mu_\theta(s_{t+1})\|_1,$$

with weight λ_T as well as the spatial regularization term

$$L^S(\theta) = \|\mu_\theta(s_t) - \mu_\theta(\tilde{s}_t)\|_2^2,$$

where $\tilde{s}_t \sim \mathcal{N}(s_t, \sigma_S)$ with weight λ_S are added to the original surrogate loss function of the PPO algorithm L^{ppo},

$$L(\theta) = L^{ppo}(\theta) + \lambda_T L^T(\theta) + \lambda_S L^S(\theta).$$

It was suggested in [9] to choose σ_S based on expected measurement noise and/or tolerance. However, it limits its effectiveness to robustness concerns. Its true power is unveiled when smoothness is further used to enforce regularity and cluster the behavior of the policy [40]. By choosing the standard deviation properly, in addition to robustness, we were able to learn a minimal yet efficient set of recovery strategies, as well as to adjust the responsiveness and reactivity of the policy on the real device. A further improvement is the introduction of the L1-norm in the temporal regularization. It still guarantees that the policy reacts only if needed and recovers as fast as possible with minimal action, but it also prevents penalizing too strictly short peaks corresponding to very dynamic motions. It is beneficial to withstand strong pushes and reduce the number of steps. Finally, it is more appropriate to penalize the mean of the policy $\mu_\theta(s_t)$ instead of the actions $\pi_\theta(s_t)$ as originally stated. It provides the same gradient wrt the parameters θ but is independent of the standard deviation σ, which avoids penalizing exploration.

F. Actor and Critic Network Architecture

The policy and the value function have the same architecture but do not share parameters. The easiest way to ensure safety is to limit their expressiveness, i.e. minimizing the number of parameters. It enhances the generalization ability by mitigating overfitting, and thereby leads to more predictable and robust behavior on the real robot, even for unseen or noisy observations. However, it hinders the overall performance of the policy. Different networks with varying depth and width have been assessed by grid search. The final architecture has 2 hidden layers with 64 units each. Below this size, the performance drops rapidly.

Fig. 5. Strong impact kick, 1 recovery step per frame in highlighted section.

V. Results

A. Training Setup and Performance

We train our policy using the open-source framework RLlib [41] with refined PPO parameters. The episode duration is limited to $60s$, which corresponds to $T = 1500$ time steps. In practice, 100M steps are necessary for asymptotic optimality under worst-case conditions, corresponding to roughly four months of experience on a real robot. This takes 6h to obtain a satisfying and transferable policy, using 40 independent simulation workers on a single machine with 64 physical cores and 1 GPU Tesla V100.

The training curves of the average episode reward and duration in Fig. 4 show the impact of our main contributions:

- Smoothness conditioning slightly slows down the convergence but does not impede the asymptotic reward.
- Without reward engineering, i.e. systematically $+1$ per step, similar training performance can be observed until 25M steps thanks to the well-defined termination conditions. After this point, the convergence gets slower, and the policy slightly underperforms at the end. This result validates our convergence robustness and that our reward provides insight into how to recover balance.
- Without the termination conditions for safety and transferability, faster convergence in around 30M steps is achieved. It is consistent with [10]–[12]. However, using such a policy on the real device would be dangerous.

B. Closed-loop Dynamics

Fig. 6 shows that smoothness conditioning improves the learned behavior, cancels harmful vibrations and preserves dynamic motions. Moreover, it also recovers balance more efficiently, by taking shorter and minimal actions.

1) Vibrations in Standing: Our regularization is a key element in avoiding vibrations in standing on the real device and promoting smooth actions. The effect is clearly visible in the target velocity predicted from our policy network, see Fig. 6. The target velocity without regularization leads to noisy positions and bang-bang torque commands, whereas our proposed framework learns an inherently smooth policy. Moreover, using L1 norm for the temporal regularization enables us to preserve the peaks at $13.8s$ and $17.5s$, which is critical to handle pushes and execute agile recovery motions.

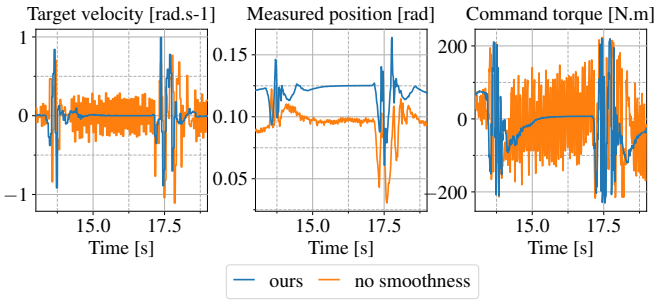


Fig. 6. Comparison of predicted target velocity, measured position and command torque from the PD controller of the left knee joint with and without smoothness conditioning.

2) Steady-state Attractiveness and Hysteresis: Once pushed, the robot recovers balance with minimal action, see Fig. 6. It quickly goes back to standing upright close to the reference. A small hysteresis is observed. It does not affect the stability and avoids doing an extra step to move the feet back in place, see Fig. 1a. It will vanish after the next push.

C. Analysis of Learned Behaviors

Different recovery strategies are generated and tested in simulation for horizontal pushes on the pelvis. We build an interactive playground in Jiminy, to test the trained policy with pushes from all sides, see Fig. 2a. Depending on the direction and application point, a specific recovery is triggered and a different magnitude of force can be handled, see Fig. 7. In-place strategies, like ankle or hip strategy, are sufficient for small pushes from any direction. Strong front, back and diagonal pushes are handled with reactive stepping strategies, and even jumps, while side pushes activate a different behavior performed with the ankles, see Table I. For side pushes, the robot twists the stance foot alternating around the heel and the toes while balancing the opposite leg, Fig. 1d. This dancing-like behavior avoids weight transfer, which was found the most difficult to learn. A larger variety of push recovery strategies on the real device are displayed in the supplementary video[1].

TABLE I

OVERVIEW OF EMERGING STRATEGIES FOR PUSHES FROM ALL SIDES

Emerging strategy	front	back	diagonal	side	small
Ankle control	✗	✗	✗	✓	✓
Hip control	✗	✗	✗	✓	✓
Stepping	✓	✓	✓	✗	✗
Jumping	✓	✗	✓	✗	✗
Foot tilting	✓	✓	✗	✗	✓
Angular momentum	✗	✗	✗	✓	✗

Recent results in simulation on the Valkyrie robot [28] invite for comparison. The humanoid has roughly the same weight as the exoskeleton Atalante carrying a dummy ($135kg$) and height of an average human person ($1.8m$). To compare impulses, the applied forces are put in relation to the duration of the push. We can handle impulses of about $190Ns$ for sagittal pushes on the pelvis with our safe policy shown in Fig. 7, compared to $240Ns$ for Yang *et al.* [28]. This is satisfactory considering that the motor torque limits are about 50% lower on Atalante and knowing that the safety constraints are limiting the performance of our policy.

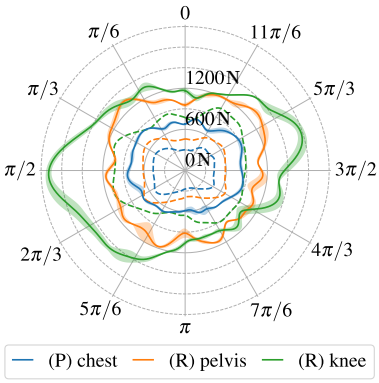


Fig. 7. Maximum recoverable force magnitude F, applied from any direction in plane $(\cos(\phi), \sin(\phi), 0)$ at several locations on the patient (P) and the left side of the robot (R). Solid and dashed lines are respectively associated with our policy and PID tracking of the reference standing pose.

D. Experimental Validation on the Exoskeleton Atalante

The trained control policy is evaluated qualitatively for both valid users and dummies of different masses on several Atalante units in the Wandercraft offices. Even though the robot is only pushed at the pelvis center during the learning, Fig. 7 strongly suggests that the policy can handle many types of external disturbances. We push the robot in reality at multiple application points and obtain impressive results, see Fig. 1 and 5. The recovery strategies are reliable for different push variations and pulling. The transfer to Atalante works out-of-the-box despite wear of the hardware and discrepancy to simulation, notably ground friction, mechanical flexibility and patient disturbances.

VI. CONCLUSION AND FUTURE WORKS

We obtain a controller that provides robust and versatile recovery strategies. Several techniques are combined to promote smooth, safe and predictable behaviors, even for unexpected events and unrecoverable scenarios. As theoretical guarantees are limited, our method was only verified empirically in simulation and reality. The policy was easily transferred to a medical exoskeleton. Even though trained for a single average patient model, our policy was validated experimentally with different users and dummies. It performed successfully a variety of recovery motions against unknown perturbations at various locations. To our knowledge, we are the first to demonstrate reactive push recovery in standstill on a real humanoid robot using deep RL.

Our framework being generic, theoretically it could be applied to any reference motion to stabilize it and provide recovery capability. For now, walking was put aside because it is more challenging than standing. Indeed, the stability is precarious during single support phases and the mechanical deformation of the structure becomes problematic. We are planning to unify walking and push recovery in future works. Besides, our framework can be adapted to other bipedal robots, and it would be interesting to compare the performance on other platforms. Further research directions include switching to more sample-efficient off-policy algorithms and enhancing exploration via curiosity-based intrinsic reward.

VII. Acknowledgment

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] S. Caron, A. Kheddar, and O. Tempier, "Stair climbing stabilization of the hrp-4 humanoid robot using whole-body admittance control," *International Conference on Robotics and Automation (ICRA)*, pp. 277–283, 2019.

[2] S. Dalibard, A. El Khoury, F. Lamiraux, A. Nakhaei, M. Taïx, and J.-P. Laumond, "Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach," *The International Journal of Robotics Research (IJRR)*, vol. 32, no. 9-10, pp. 1089–1103, 2013.

[3] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous robots*, pp. 429–455, 2016.

[4] T. Gurriet, S. Finet, G. Boeris, A. Duburcq, A. Hereid, O. Harib, M. Masselin, J. Grizzle, and A. D. Ames, "Towards restoring locomotion for paraplegics: Realizing dynamically stable walking on exoskeletons," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2804–2811.

[5] F. L. Moro and L. Sentis, "Whole-body control of humanoid robots," in *Humanoid robotics: a reference*. Springer, 2019, pp. 1–23.

[6] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, 2014.

[7] D. Kim, S. J. Jorgensen, J. Lee, J. Ahn, J. Luo, and L. Sentis, "Dynamic locomotion for passive-ankle biped robots and humanoids using whole-body locomotion control," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 936–956, 2020.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[9] S. Mysore, B. Mabsout, R. Mancuso, and K. Saenko, "Regularizing action policies for smooth control with reinforcement learning," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1810–1816, 2021.

[10] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2811–2817, 2021.

[11] D. Ferigo, R. Camoriano, P. M. Viceconte, D. Calandriello, S. Traversaro, L. Rosasco, and D. Pucci, "On the emergence of whole-body strategies from humanoid robot push-recovery learning," *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2020.

[12] D. C. Melo, M. R. O. A. Máximo, and A. M. da Cunha, "Push recovery strategies through deep reinforcement learning," in *2020 Latin American Robotics Symposium (LARS)*, 2020, pp. 1–6.

[13] K. Yuan, C. McGreavy, C. Yang, W. Wolfslag, and Z. Li, "Decoding motor skills of artificial intelligence and human policies: A study on humanoid and human balance control," *IEEE Robotics & Automation Magazine (RA-M)*, vol. 27, no. 2, pp. 87–101, 2020.

[14] S.-H. Hyon, R. Osu, and Y. Otaka, "Integration of multi-level postural balancing on humanoid robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 1549–1556.

[15] B. J. Stephens and C. G. Atkeson, "Dynamic balance force control for compliant humanoid robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 1248–1255.

[16] Z. Li, C. Zhou, Q. Zhu, and R. Xiong, "Humanoid balancing behavior featured by underactuated foot motion," *IEEE Transactions on Robotics (T-RO)*, vol. 33, no. 2, pp. 298–312, 2017.

[17] S. Caron, "Biped stabilization by linear feedback of the variable-height inverted pendulum model," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9782–9788, 2020.

[18] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1620–1626, 2003.

[19] P.-b. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2006.

[20] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2001, pp. 239–246.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2016.

[22] N. Heess *et al.*, "Emergence of locomotion behaviours in rich environments," 2017. [Online] https://arxiv.org/abs/1707.02286

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

[24] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, 2022.

[25] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, pp. 1–14, 2018.

[26] X. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *Robotics: Science and Systems (RSS)*, 2020.

[27] D. Rodriguez and S. Behnke, "Deepwalk: Omnidirectional bipedal gait by deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 3033–3039.

[28] C. Yang, K. Yuan, W. Merkt, T. Komura, S. Vijayakumar, and Z. Li, "Learning whole-body motor skills for humanoids," in *IEEE/RAS International Conference on Humanoid Robotics (Humanoids)*, 2018.

[29] C. Yang, K. Yuan, S. Heng, T. Komura, and Z. Li, "Learning natural locomotion behaviors for humanoid robots using human bias," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, pp. 2610–2617, 2020.

[30] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 100. PMLR, 2020, pp. 317–329.

[31] J. Siekmann, K. Green, J. Warila, A. Fern, and J. W. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," in *Robotics: Science and Systems (RSS)*, 2021.

[32] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5136–5143, 2021.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online] https://arxiv.org/abs/1707.06347

[34] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *International Conference on Learning Representations (ICLR)*, 2016.

[35] A. Duburcq, "Jiminy: a fast and portable python/c++ simulator of poly-articulated systems for reinforcement learning," 2019. [Online] https://github.com/duburcqa/jiminy

[36] J. Carpentier, F. Valenza, N. Mansard *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," https://stack-of-tasks.github.io/pinocchio, 2015–2021.

[37] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

[38] M. Vigne, A. E. Khoury, F. D. Meglio, and N. Petit, "State estimation for a legged robot with multiple flexibilities using imus: A kinematic approach," *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[39] L.-K. Ma, Z. Yang, X. Tong, B. Guo, and K. Yin, "Learning and exploring motor skills with spacetime bounds," pp. 251–263, 2021.

[40] Q. Shen, Y. Li, H. Jiang, Z. Wang, and T. Zhao, "Deep reinforcement learning with robust and smooth policy," in *International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 8707–8718.

[41] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "Rllib: Abstractions for distributed reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 3053–3062.