Configuration Path Control

Sergey Pankov

Abstract: Reinforcement learning methods often produce brittle policies – policies that perform well during training, but generalize poorly beyond their direct training experience, thus becoming unstable under small disturbances. To address this issue, we propose a method for stabilizing a control policy in the space of configuration paths. It is applied post-training and relies purely on the data produced during training, as well as on an instantaneous control-matrix estimation. The approach is evaluated empirically on a planar bipedal walker subjected to a variety of perturbations. The control policies obtained via reinforcement learning are compared against their stabilized counterparts. Across different experiments, we find two- to four-fold increase in stability, when measured in terms of the perturbation amplitudes. We also provide a zero-dynamics interpretation of our approach.

Keywords: Biped, configuration path control, reinforcement learning, stability, virtual constraints, zero dynamics.

1. INTRODUCTION

The past decade has seen successful applications of deep neural networks (NN) to various machine learning tasks, such as image classification [1,2], speech recognition [3] and language translation [4,5]. In the field of reinforcement learning (RL), the employment of deep NNs as expressive function approximators has been crucial in tackling many difficult problems involving an agent interacting with its environment with the goal of maximizing its rewards. Among the notable examples of deep RL applications are: playing arcade games [6] and challenging board games, such as Go [7], often at super-human level; solving continuous control tasks [8], such as cart-pole and acrobat; 2D and 3D locomotion [9,10] and manipulation [11] tasks, including complex high-dimensional setups. In many cases, elaborate policies were learned from scratch, with little to no task-specific engineering.

However, despite numerous successes within simulated environments, there is a paucity of the real-world deployment of RL trained robots. Consider, for example, legged robots. The design of control policies of the most agile bipedal [12-14] and quadrupedal [15-17] robots appears to have little to do with RL. Among the main limiting factors, precluding the application of RL to physical robots, are high training sample complexity [18] and policy brittleness (also known as the reality gap) [19,20]. These deficiencies are related to the lack of generalization and over-fitting. A popular way to counter them is to train on more samples and on more diverse samples. Domain randomization [21-24], training with perturbations [25,26] and adversarial training [27,28] are among the methods incor-porating such strategy.

By design, the above approaches improve generalization by collecting more training samples (in a simulated environment). In this work we pursue a very different route: we investigate an approach for stabilization of RL-trained continuous control policies that does not resort to generation of additional training samples. Instead, our algorithm takes the data, produced by a policy during training with a RL algorithm, and converts it to a policy with superior stability.

The fundamental ingredient of our approach is a control policy stabilization around desired configuration paths (time-reparameterized trajectories) that can be rigorously justified in the high gain limit (HGL). We call this approach configuration path control (CPC). It has some overlap with the zero dynamics (ZD) concept [29], which is central to the hybrid zero dynamics (HZD) framework in the context of bipedal walkers [30,31]. We reinterpret the derived CPC control law in the language of the HZD literature, by re-stating it in terms of the reparameterization invariant virtual constraints.

We empirically validate the CPC method by stabilizing RL-trained policies of a bipedal walker. The RL policies are compared with their CPC-stabilized counterparts in a number of stability testing experiments, showing remarkable improvements in stability in all of the tests.

The paper is organized as follows: In Section 2, we derive the control law and present the CPC algorithm. Relation to virtual constraints of ZD is explained in Section 3. The approach is experimentally validated in Section 4, followed by a conclusion in Section 5.
2. CPC FORMULATION

In this section we provide the underlying theory of our approach and present the CPC algorithm.

2.1. Control law derivation

In this section we derive the CPC control law, by deriving and solving a trajectory reachability equation in an underactuated system.

2.1.1 Preliminaries

Let $q$ be the $N$-dimensional vector of configuration variables of the system. We use the dot notation for the time derivative to represent velocity $\dot{q}$ and acceleration $\ddot{q}$. Let $x \equiv [q; \dot{q}]$ denote the state vector. The system dynamics are governed by the equation of motion

$$D(q)\ddot{q} + H(q, \dot{q}) = B_r \tau,$$

where $D(q)$ is the inertia matrix (which is positive definite), $H(q, \dot{q})$ includes the terms dependent on $q$ and $\dot{q}$, such as Coriolis, centrifugal and gravitational forces, $\tau$ is the $M$-dimensional vector of control variables $B_r$ is the control distribution matrix. Through this section we assume the system to be fully actuated. A fully actuated system (full actuation requires $B_r$ to have rank $N$) can be forced to move with an arbitrary acceleration $\dot{a}$ with the following choice of controls (this choice minimizes actuation cost $\approx \tau^T \tau$)

$$\tau_{ff}(q, \dot{q}, \dot{a}) = B^*_r (D(q)\dot{a} + H(q, \dot{q})),$$

which we call the feedforward control term, where $B^*_r = B^*_r (B_r B^*_r)^{-1}$ is the right pseudoinverse. The system in an initial state $x_d(0) = [q_d(0); \dot{q}_d(0)]$ can then in principle follow an arbitrary desired trajectory $q_d(t)$ by setting the controls to $\tau_d = \tau_{ff}(q, \dot{q}, \dot{a}_d)$, where $\dot{a}_d = \dot{q}_d$. In practice, due to accumulation of errors in $\dot{q}$ and $\dot{q}$, the system also needs a stabilizing feedback control term $\tau_{fb}$ to track $q_d(t)$.

In this paper, unless noted otherwise, we use $\Delta$ to indicate an error term, that is the deviation of a quantity from its desired value, thus $\Delta a \equiv a - a_d$ for any quantity $a$, unless specified otherwise. We define the control matrix $B(q) \equiv D^{-1}(q) B_r$, so

$$\Delta \ddot{q} = B(q) \Delta \tau,$$

where $\Delta \ddot{q} \equiv \ddot{q} - \ddot{q}_d$ and $\Delta \tau = \tau - \tau_{ff}(q, \dot{q}, \dot{q}_d)$. Let us define the problem of tracking $q_d(t)$, within the optimal control framework [32], by penalizing $\Delta \ddot{q}$ and $\Delta \dot{u}$ with quadratic cost terms $c_q \Delta \ddot{q}^T \Delta \ddot{q}$ and $c_u \Delta \dot{u}^T \Delta \dot{u}$. The optimal control solution for the feedback term is a linear controller

$$\tau_{fb} = -B^{-1}(q)K \Delta x, \quad K = \begin{bmatrix} k_p & k_d \end{bmatrix} \otimes I_N,$$

where $\Delta x = [\Delta q, \Delta \dot{q}]$, $e = (c_q/c_d)^{1/2}$, $k_p$ and $k_d$ are $O(1)$ and $I_N$ denotes $N \times N$ identity matrix. We see (from (3), (4)) that with the control law $\tau = \tau_{ff} + \tau_{fb}$, $\Delta \ddot{q}$ follows the dynamics of a near-critically damped oscillator:

$$\Delta \ddot{q} + \frac{k_d}{e} \Delta \dot{q} + \frac{k_p}{e^2} \Delta q = 0.$$

It implies that the error terms vanish exponentially fast on the time scale $O(\varepsilon)$, while $\Delta \ddot{q} = O(\Delta \tau)$, $\dot{q}_d = O(\Delta \tau \varepsilon)$ and $\Delta q = O(\Delta \tau \varepsilon^2)$. Note, we are using $O(a)$ notation as synonymous to “of the order of $ca$”, where $c$ is a (dimensional, in general) constant of order one. We see that in the HGL $\varepsilon \to 0$, the terms $\Delta q$ and $\Delta \dot{q}$ can be ignored relative to $\Delta \ddot{q}$ and $\Delta \tau$ to leading order in $\varepsilon$.

We define the configuration path of a trajectory $q(t)$ as the set of all configurations visited by $q(t)$. Thus, two different trajectories $q(t)$ and $q'(t)$, related by a time reparameterization $q(t') = q(t)$, map to the same configuration path. In that case we may say that $q'(t)$ is associated with the path of $q(t)$, and vice versa. In this paper, we use interchangeably the terms: “configuration path” and “path”, as well as, “desired” and “target” trajectory.

2.1.2 Motivations of CPC formulation

Unlike the case of full actuation in the previous section, many important control problems, such as walking on point feet [30], are essentially underactuated, which makes them much more challenging. Motivated by the desire to make the underactuated control problem more tractable, while preserving its safety critical aspects, we will resort to stabilization around a time-independent configuration path, instead of stabilization around a time-dependent trajectory. Giving up the time dependence makes the control problem easier, without affecting the robot’s safety metrics, provided they are defined in terms of the configuration variables. Indeed, a collision indicator — be it collisions with obstacles, the ground or between the robot parts — is typically defined in terms of configuration variables alone.

Another crucial motivation stems from our aim for stabilization of model-free RL policies, in which we at most can afford a quasi-instantaneous estimation of the local dynamics based on the $O(N)$ most recent points along the current trajectory. Without possibility of longer time planning, we will be assuming short stabilization time scale $\varepsilon$. In other words, we derive the CPC control law under the assumption of HGL, in which case we only need to retain the leading terms in $\varepsilon$. When we omit sub-leading terms in an equation, we will either explicitly indicate its precision by adding a precision term to the right hand side of the equation or use the approximately equal sign. See Section 3 for a discussion on the satisfiability of HGL in specific robotic systems.
2.1.3 Reachability equation

We define trajectory reachability in the HGL, specified by the scale $\varepsilon$, as follows: the system can reach a trajectory $x_d(t)$ from a given state if it can be driven to $x_d(t)$ in time $t = O(\varepsilon)$. A fully actuated system can move with an arbitrary acceleration, and therefore, can reach any trajectory. An underactuated system, on the contrary, can only reach a subset of trajectories. In this section we derive the trajectory reachability equation and use it to analyze the configuration path reachability in the underactuated case.

Let $q_d(t)$ be the system’s trajectory under some choice of controls $\tau_d(t)$ starting at $x_d(0)$. Similarly for $q(t), \tau(t)$ and $x(0)$. We subtract from (1) for $q$ the same equation for $q_d$ to find

$$\Delta q = B(q)\Delta \tau + O(\Delta \tau \varepsilon),$$

which is only superficially similar to (3), since $\Delta \tau$ has now a different meaning. Without loss of generality we assume the independence of the first $M$ rows of the control matrix (i.e., $B(q)$ is full rank), while underactuation implies $N > M$. Therefore, we can write $B(q) = [B_\chi(q); B_\psi(q)]$, where $B_\chi(q)$ is invertible. Accordingly, we write $q = [\chi; \psi]$, where $\dim(\chi) = M$. We call $\chi$ and $\psi$ the controlled and free coordinates respectively. Writing (6) as a system of two equations, $\Delta \chi \approx B_\chi(q)\Delta \tau$ and $\Delta \psi \approx B_\psi(q)\Delta \tau$, then using the invertibility of $B_\psi(q)$ in the first equation to eliminate $\Delta \tau$ in the second equation, we obtain

$$b^T(q)\Delta q = 0 + O(\Delta \tau \varepsilon),$$

where

$$b(q) = [B_\psi(q)B_\chi^{-1}(q), -I_{N-M}]^T,$$

so the set of possible accelerations $\Delta q$ is the null-space of $b^T(q)$.

Let us define $\bar{b} \equiv b(q(0))$, and similarly for $B, B_\chi$ and $B_\psi$, so $b = [B_\psi B_\chi^{-1}, -I_{N-M}]^T$. Note that $b = b(q(\varepsilon)) + O(\varepsilon)$. If $q_d$ is reachable, the system can be driven to $q_d$, so $\Delta q(t) = 0$ for $t > \tau_\varepsilon = O(\varepsilon)$. In that case, integrating (7) twice from $0$ to $t > \tau_\varepsilon$, we find $b^T(\Delta q(0) + t\Delta q(0)) \approx 0$. The equality must hold for any $t > \tau_\varepsilon$, leading to the trajectory reachability equation

$$(\text{diag}(1, \varepsilon) \otimes b^T)\Delta v(0) = 0 + O(\Delta \tau \varepsilon^3).$$

Note, the derivation of the reachability equation implies that a target trajectory $q_d(t)$ in (9) must be realizable under the system dynamics by some choice of controls $\tau_d(t)$. The trajectory reachability equation can also be used to determine configuration path reachability. We say the configuration path of $q_d(t)$ is reachable if there exists a reachable trajectory $q_d'(t)$, related to $q_d(t)$ by time reparameterization. Below we will solve the path reparameterization to linear order in time reparameterization.

Before we proceed, let us introduce some shorthand notations. We will optionally use a superscript in the error term, to differentiate between different target trajectories, so $\Delta' a \equiv a - a_d$. Unless specified otherwise, $\Delta q \equiv q - q_d$, where $q_d$ is defined above. We reserve 0 superscript to indicate a value at $t = 0$, so $a^0 \equiv a(t = 0)$. We also define $\bar{a} \equiv b^T a$. We may use $O(a_1, a_2, ..., a_n)$ notation to represent $\sum_{i=1}^{n} O(a_i)$.

Let $q_d(t) = q_d(t'(t))$ and $t_0$ be the root of $t'(t) = 0$. The linear order expansion of $t'(t)$ about $t' = 0$ is $t'(t) \approx (t - t_0)/s$, where $s = (\partial t'(t = 0)/\partial t)^{-1}$. Therefore, $q_d(t) = q_d^0 + q_d^0(t - t_0)/s + O(t'^2 + t_0^2)$. Substituting $q_d^0$ in the reachability equation (9) we obtain a linear system of $2(N-M)$ equations on $t_0$ and $s$. In the case of one degree of underactuation, $N - M = 1$, it is satisfied exactly by

$$t_0 = \frac{q_d^0 - q_0^0}{q_0^0}, \quad s = \frac{\bar{q}_d^0}{q_0^0}.$$

The considered linear approximation is justified provided $s = O(1)$ and $|t_0| \ll O(q_0^0)$. In practice, (in the discrete time domain and with an abundance of target points), provided a system stays close to its target trajectory, these conditions are either always satisfied or only briefly violated (when the sign of $q_0^0$ flips). Therefore, we limit our CPC algorithm implementation, as well as the theoretical investigation in the rest of the paper, to the above linear analysis, which assumes $\bar{q}_d^0 \neq 0$ and $q_0^0 \neq 0$.

Let us define the renormalized target trajectory $q_d'$ as the linear part of a reachable trajectory $q_d$, corresponding to the path of $q_d$.

$$q_d'(t) = q_0^0 + \frac{\bar{q}_d}{s}(t - t_0),$$

where $t_0$ and $s$ are given by (10). Our analysis implies that for one degree of underactuation, to linear order in $t$ and $t_0$, the renormalized target trajectory $q_d'$ depends on $q$ and the configuration path of $q_d$, but not on the time parameterization of $q_d$. Indeed, from (10) and (11), one finds to linear order

$$\bar{q}_d'(t) = \bar{q}(t).$$

Our analysis also implies that both $q_d'$ and the path of $q_d$ are reachable with precision $O(\Delta \tau \varepsilon^3, \varepsilon^2 + t_0^2)$, for $N - M = 1$. It is therefore desirable to select target points $x_d$ with $t_0 \approx 0$. To that end, we will be evaluating the quality of target points using a certain score function measuring how close $t_0$ is to 0, among other criteria. Note, for $N - M > 1$, the reachability equation can only be solved approximately for a general $q_d$. In that case we can include a loss term in the score function, quantifying the error of (9). The reparameterization parameters $t_0$ and $s$ can then be determined by minimizing the score function. Assuming the loss term in the form $f_0(||\Delta' \bar{q}(0)||) + f_1(||\Delta' \bar{q}(0)||)$,
with the only requirement for \( f_r(y) \) to have a global minimum at \( y = 0 \), we find a generalization of (10) to \( N - M > 1 \)

\[
t_0 = \frac{\hat{q}_d^T (\hat{q}_d^2 - \hat{q}_d^0)}{\hat{q}_d^2}, \quad s = \frac{\hat{q}_d^T \hat{q}_d^0}{\hat{q}_d^2}
\]

(13)

A more detailed consideration of the higher degree of underactuation case \( N - M > 1 \) is beyond the scope of this paper.

2.1.4 CPC control law

Let \( x(t) = [x; \dot{x}] \) and \( \chi^0(t) = \chi^0(t) + \chi^0(t-t_0)/s \), where \( t_0 \) and \( s \) are given by (13), or by (10) for the special case of \( N - M = 1 \), for the current state \( x = [q^0, \dot{q}^0] \) and a target state \( x_d = [\dot{q}^0, \dot{q}^0_d] \). We define the CPC control law as (cf. (4))

\[
\tau_{\text{CPC}}(t) = \tau_d - B^{-1} \Delta x(t), \quad K = \left[ \begin{array}{cc} k_p & k_d \end{array} \right] \otimes I_M.
\]

(14)

If \( \Delta x \) satisfies the reachability equation, the CPC control law will drive the system to the configuration path of \( q_d \) despite underactuation. Moreover, the control law is invariant with respect to the choice of the controlled coordinates. These two statements are formalized and proven (see Appendix A for proofs) in the following theorems.

Theorem 1 (Path convergence theorem): Assume \( \dot{q} \neq 0 \) and \( \dot{q}_d \neq 0 \). Let \( \tau_{\text{CPC}}(t) \) be defined by (14). If \( \Delta x(0) \) satisfies the reachability equation (9), then under \( \tau_{\text{CPC}}(t) \), within the HGL, \( \varepsilon \rightarrow 0 \), the error term \( \Delta q(t) \) reduces to \( O(\varepsilon^2 + t_0^2) \) on the time scale \( O(\varepsilon) \).

Theorem 2 (Reparameterization invariance theorem): Let all the conditions of Theorem 1 hold. Let \( \tilde{q} = [\tilde{x}, \tilde{\dot{x}}] = Cq, \tilde{B} = [\tilde{B}_x, \tilde{B}_y] = CB \), where \( C \) is a linear invertible transformation and \( \tilde{B} \) is invertible. Other quantities defined through \( q \) and \( B \) change correspondingly, e.g., \( x \rightarrow \tilde{x} = [\tilde{q}, \tilde{\dot{q}}] \) and \( b \rightarrow \tilde{b} = [\tilde{B}_y \tilde{B}_x^{-1}, -I_N-M]^T \). Let \( \tau_{\text{CPC}}(t) \) be defined by (13) and (14), with \( q, \dot{x}, B, b \) replaced by the corresponding transformed quantities \( \tilde{q}, \tilde{x}, \tilde{\dot{x}}, \tilde{\bar{b}} \). Then, within the HGL, \( \tau_{\text{CPC}}(t) = \tau_{\text{CPC}}(t) \).

The reparameterization invariance theorem and its proof may appear obvious. However, this invariance is specific to our formulation and does not generally hold for the ZD control law designs proposed in the HZD literature, which we denote \( \tau_{\text{ZD}} \). Rather, \( \tau_{\text{ZD}} \) is formulated in terms of virtual constraints \([30,31]\), requiring manual selection of a gait phasing parameter and ensuring that it is monotonically increasing with time, for the type of motion under consideration. Generally, \( \tau_{\text{ZD}} \) is sensitive to the choice of free coordinates. In our approach, on the contrary, the selection of free coordinates has no bearing on \( \tau_{\text{CPC}} \), as long as \( B_x \) is full rank. Due to this invariance, the definition of \( \tau_{\text{CPC}} \) does not require a foresight into the type of motion considered, (unlike \( \tau_{\text{ZD}} \)), and is therefore suitable for stabilization of a generic RL-learned policy. A rigorous connection between our approach and HZD is provided in Section 3, where we express the CPC control law in the language of virtual constraints.

2.2. Selection of target points

In the previous section we have introduced the CPC control law (see (14)), that specifies controls \( \tau_{\text{CPC}} \) for a given target point \( x_d \), that drives the system to a configuration path containing \( q_d^0 \). While \( \tau_{\text{CPC}} \) is formally defined for any \( x_d \), the best possible target points should be selected for computing \( \tau_{\text{CPC}} \), because the quality of the adopted approximations strongly depends on the choice of \( x_d \) as we explain below.

Firstly, the error term in the statement of Theorem 1 reduces to \( O(t_0^2) \) in the limit \( \varepsilon/t_0 \rightarrow 0 \). Therefore, one should select those target points that minimize \( t_0^2 \). Secondly, the feedforward term \( \tau_d(t) \) depends on \( q \) (see (30)), while in our application of CPC the feedforward term \( \tau_d(t) \) is taken from available RL samples. Therefore, one should generally expect \( \tau_d \approx \tau_d^{\text{RL}} \) only for \( s \approx 1 \). Thus, one should typically select target points with \( s \approx 1 \).

The path tracking precision is not the only criterion one may need to consider in the context of RL policy stabilization. In the tasks with stochastic environments or variable goals one also faces the problem of selecting the best path out of multiple candidates of generally unequal value, which is encoded in the RL value function. Therefore, the value function must be incorporated in the process of the target point selection. The selection procedure that addresses both issues is presented in the next two subsections.

2.2.1 Selection by time reparameterization parameters with efficient ball-tree search

We assume that the training data of an RL algorithm is stored as a set \( X_d \) of data points \( (x, \tau, G) \), representing an instance of the system’s state \( x \), controls \( \tau \) and return \( G \) (total future discounted reward, see Subsection 2.2.2) recorded at time \( t \). For brevity, we will often use the state notation \( x \) alone, implying an access to the corresponding \( \tau \) and \( G \).

Generally, a suitable target point \( x_d \in X_d \) is selected based on the proximity of \( (t_0, s) \) to \((0, s_d)\) (where usually, but not always, \( s_d = 1 \)) and on the maximization of the expected return. We found it convenient to separate the optimization over \( (t_0, s) \) from the return optimization. The main reason is that the loss function for \( t_0 \) and \( s \) can be very efficiently optimized by storing \( X_d \) in a ball-tree search data structure, as we explain below.

Let us introduce a proximity loss

\[
L^{\text{prox}} (t_0, s) = (\omega t_0)^2 + (s - s_d)^2,
\]

(15)
where \( \omega \) balances the relative importance of \(|t_0|\) versus \(|s - s_g|\) smallness. The set \( X_\delta \), stored on a ball-tree, can be searched for best performing points, that minimize the proximity loss, by a standard branch and bound method. The central element of such a method is the bounding of the optimization function on a given tree node. The maximum efficiency is reached when the bounds are tight. The tight bounds can indeed be computed in closed form, as shown in Appendix B.1.

### 2.2.2 Selection by expected return

Depending on the problem setup, different target points may vary greatly in terms of how valuable it is for the system to visit them. The cost of getting to those points from the current state may vary greatly as well. We will quantify the utility of choosing \( x_d \) as a target point in the current state \( x_0 \) by estimating the corresponding expected return.

In a discrete time formulation, we define return \( G_t \) at time \( t \) as

\[
G_t = \sum_{i=0}^m \gamma r(x_{t+i}, \tau_{t+i}),
\]

where \( r(x, \tau) \) is the reward function and \( \gamma \) is the discount factor. In RL the goal is to maximize the expected return.

The value function \( V(x) \) is defined as the expected return of the system in state \( x \)

\[
V(x) = E_{[x_0=x]}[G_0],
\]

The expectation is taken over trajectories \( x_0 \tau_1 x_1 \tau_1 \ldots \), sampled under a given policy and the problem dynamics, starting from \( x_0 = x \). For simplicity, we will treat both the policy and environment as deterministic.

Let a discrete time step \( \Delta t \) correspond to incrementing \( i \) by 1 in 16. Consider \( \gamma \) and \( r(x, \tau) \) in the form

\[
\gamma = 1 - \frac{\Delta t}{T_\gamma},
\]

\[
r(x, \tau) = \frac{\Delta t}{T_\gamma} \left( \tau^T C_{r} \tau + r(x) \right), \quad C_{r} = C_{r}^T \leq 0,
\]

where the parameter \( T_\gamma \) sets the time scale of the reward discounting. This choice ensures that \( V(x) \) is well behaved in the continuous time limit \( \Delta t \to 0 \), in which

\[
V(x) = T_\gamma^{-1} \int_0^{\infty} \left( \tau(t)^T C_{r} \tau(t) + r(x(t)) \right) e^{-\frac{t}{T_\gamma}} dt.
\]

We use this equation to estimate (in the HGL, and assuming \( \langle t_0, s \rangle \approx (0, 1) \)) the value function \( V_{x_0}(x_0) \) of the system in state \( x(0) = x_0 \) that reaches the target trajectory \( q_0 \) at \( x'_0(0) \) (we call it stage I) and then follows it under the CPC control law (we call it stage II). The value function has two corresponding contributions

\[
V_{x_0}(x_0) = V_I + V_{II},
\]

where \( V_I \) is dominated by the work penalty during the transition from \( x(0) \) to \( x'_0(0) \)

\[
V_I \approx T_\gamma^{-1} \left( 2\tau_0^T C_{\tau} \int_0^{\infty} \Delta \tau(t) dt + \int_0^{\infty} \Delta \tau(t)^T C_{r} \Delta \tau(t) dt \right),
\]

\[
V_{II} \approx V(x_0'(0)) = V(x_d) + \frac{t_0}{T_\gamma} \left( \tau_0^T C_{\tau} \tau_d + r(x_d) - V(x_d) \right).
\]

In this section, we omit CPC subscript for brevity, so above \( \Delta \tau \equiv \Delta \tau_{CPC} \). In the HGL that we consider, the integral in (22) can be evaluated exactly, as \( \Delta \tau(t) = -B_x^{-1} K \Delta x(t) \), where

\[
\Delta x(t) = e^{\kappa t} \Delta x(0), \quad F = \begin{bmatrix} 0 & 1 \\ -k_x & -k_x \end{bmatrix} \otimes I_M.
\]

Below, we present the result for a critically damped controller case, \( k_d = 2\sqrt{K_p} \), which we adopt through the rest of the paper. Denoting \( \kappa = \sqrt{K_p}/\varepsilon \), we find

\[
V_I = -\frac{2}{T_\gamma} \tau_0^T C_{\tau} Z_1 \Delta x(0) + \frac{\kappa}{4T_\gamma} \Delta x(0)^T \left( Z_1 C_{r} Z_1^T + Z_2 C_{r} Z_2^T \right) \Delta x(0),
\]

\[
Z = \begin{bmatrix} \tau \otimes I_M \end{bmatrix} B_x^{-1\top}, \quad Z_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad Z_2 = \begin{bmatrix} \kappa \\ 2 \end{bmatrix}.
\]

Equations (21), (23), (25), and (26) provide a closed-form expression for estimating \( V_{x_0}(x_0) \). Note (23) contains \( V(x_0) \), which in this work is simply approximated by a recorded return \( V(x_d) \approx G_d \) from the corresponding triplet \((x_d, \tau_d, G_d)\).

### 2.3. Algorithm

We are now in a position to present the algorithm that combines all the necessary ingredients derived in Subsections 2.1 and 2.2. The function CPCLOOP(), shown in Algorithm 1, is called at every cycle of the robot controller’s

**Algorithm 1: CPCLOOP(** \( x_0, B, X_d, \omega, s_g, n_d, k_0, \tau, k_c \))

1. \( Y \leftarrow \) CANDIDATES \( (x_0, B, X_d, \omega, s_g, n_d) \)
2. \( k \leftarrow k_0 \)
3. \( y^* \leftarrow \) argmin \( y \in Y \) COST \( (y, k, x_0, B) \)
4. \( \tau \leftarrow \) CONTROLS \( (y^*, k, x_0, B) \)
5. \( k \leftarrow k/2 \)
6. until \( ||\tau|| < \tau_c \) or \( k < k_c \)
7. return \( (\tau) \)
loop. It is supplied with the current state $x_0$ and the control matrix estimate $B$, (in addition to a number of static parameters). It then selects the best possible target point and returns corresponding CPC control values $\tau_{PC}$.

The selection of the best target point $x_d \in X_d$ is done in two stages. In the first stage, the proximity loss $L^{prox}$ (see (15)) is optimized to select $n_d$ best performing target point candidates. This is done by the function $\text{CANDIDATES}()$ that returns a set of candidates $Y$ containing tuples $y = \left(x^d_0, t_0, s\right)$, with $t_0$ and $s$ computed for corresponding $x^d_0 \in X_d$ and the current state $x_0$, (see (10)). The function can be implemented efficiently as described in Subsection 2.2.1.

In the second stage, out of $n_d$ preselected candidates, the best $x^d$ is selected based on the current state $x$.

The controls $\tau$ are computed by the function $\text{CONTROLS}()$ that returns $\tau_{PC}(0)$, (see (14)). In the second stage we also verify that the computed controls stay within certain bounds, so not to exceed reasonable limits of actuation. Starting from $k = k_0$, if $||\tau||$ exceeds $\tau_c$, we keep reducing $k$ (and recomputing $\tau$) until either $||\tau||$ falls within the bounds or the gain becomes unreasonably small $k \leq k_c$.

All functions in Algorithm 1 rely on the control matrix $B$, which in this work is estimated on-line. While a higher accuracy of $B$ estimation may be beneficial for the algorithm’s performance, it was not critical in our experiments. The control matrix was crudely estimated from (6) using a standard linear regression technique. Namely, $B$ is the least-square error solution over the last $n$ data points $(\tau^i, u^i)$, $i = 1..n$, along the current trajectory, with $\tau_d$ and $u_d$ set to 0. We choose $n$ to be slightly larger than $N$.

3. RELATION TO ZD VIRTUAL CONSTRAINTS

We briefly overview the ZD approach [30,31]. The ZD are a restriction of the full dynamics (1) to a submanifold of the configuration space manifold, defined by means of a virtual constraint

$$y = h(q) \equiv \chi - h_d(\theta(q)),$$

where $y$ is the system’s output that needs to be driven to zero by a ZD controller, and $h_d$ is the desired value of the controlled variable $\chi$ at the gait phasing parameter $\theta$. In this section we assume $N = M$ and a scalar $\theta(q)$. For the constraint to represent the configuration path of $q_d$, one needs $h_d(\theta(q_d)) = \chi_d$ and $\theta$ to be monotonic along the trajectory $q_d$. Differentiating $h(q)$ twice with respect to time and using (1) to eliminate $\dot{q}$ in the result, one finds for $\ddot{y}$

$$\ddot{y} = A(q) \left(\tau - \tau^{ZD}_d(q, \dot{q})\right),$$

where

$$A(q) = \frac{\partial h(q)}{\partial q} B(q),$$

and

$$\tau^{ZD}_d(q, \dot{q}) = A^{-1}(q) \left(\frac{\partial h(q)}{\partial q} D^{-1}(q) H(q, \dot{q}) - \frac{\partial}{\partial q} \left(\frac{\partial h(q)}{\partial q}\right) \dot{q}\right).$$

We define the ZD controller

$$\tau_{ZD} = \tau^{ZD}_d(q, \dot{q}) - A^{-1}(q) \left(k_p \frac{\dot{y}}{\varepsilon} + \frac{k_d}{\varepsilon^2} \ddot{y}\right),$$

which drives the system to $y = 0$ submanifold, as

$$\ddot{y} + \frac{k_d}{\varepsilon^2} \ddot{y} + \frac{k_p}{\varepsilon} \dot{y} = 0.$$
is similar to ZD under HVC (as follows from Theorem 2),
the effective velocity adaptation arises thanks to the target
point selection procedure, that favors trajectories with \( s \approx 1 \) that are picked (using efficient search) from a multitude
of possibilities.

We view HGL primarily as a methodological approach
that allows us to make a number of rigorous statements,
proven in the theorems. In practical terms, the proofs also
provide precision terms, quantifying the degree of approx-
imation. The limit \( \epsilon \rightarrow 0 \) is an idealization, that is neither
necessary nor possible to attain in a real system. The pa-
rameter \( \epsilon \) regulates the stiffness of the feedback controller.
In practice, it is limited by various factors, such as the con-
troller bandwidth, torque limits, ground friction and so on.
In our numerical experiments in Section 4, the initial stiff-
ness is well below the bandwidth limitation (determined
by the time discretization), and is adaptively attenuated to
stay within reasonable torque limits, see Algorithm 1.

4. EXPERIMENTS

While our approach is very general, in this work we test
it on two distinct stability-critical tasks: 1) bipedal walk-
ing under strong disturbances, 2) acrobot balancing de-
\begin{itemize}
\item[1] https://youtu.be/6OG8x-iSWAE
\end{itemize}

different experiments. To facilitate the comparison of CPC-

dom sparse moments of time, representing random blows
to the walker. The three cases correspond to the sustained,
static and punctuated perturbations respectively. In each
experiment a perturbation amplitude keeps increasing un-
til the walker falls at time \( t_f \). The stability of a controller
is measured by how long the walker can withstand an in-
creasing perturbation (see accompanying video).

4.1.2 Walker model

We considered one of the simplest possible kneed
bipedal walkers – a planar four-link walker with (nearly)
point feet, (see Fig. 1 and Appendix B.2). The joint an-
gles are unrestricted. It is rewarded for walking with a cer-
tain velocity and penalized for excessive control or if its
COM drops too low. We trained 10 NN controllers using a
variant of the natural policy gradient approach [35], most
closely related to PPO with adaptive KL penalty [36].

To avoid the complications of dealing with different gait
classes (such as backward- vs. forward-bending knees),
the controller policies are initialized by an example of a
humanlike-gait policy using imitation learning and ran-
dom initialization of NN weights. Due to the inherent lim-
itations of imitation learning, the initial policies are quite
unstable and require substantial amount of RL training.
Upon completion of training, visual inspection revealed
very similar gaits in all the controllers. We verified that
the mean returns and the cost of transport (COT), were
very similar gaits in all the controllers. We verified that
the mean returns and the cost of transport (COT), were
within a relatively narrow range (with COT about 0.08),
suggesting that the controllers converged to a near opti-

4.1.3 Results

The stability testing results are summarized in Fig. 2.
It shows \( \langle t_f \rangle \) (mean \( t_f \)) for each controller in three dif-
ferent experiments. To facilitate the comparison of CPC-
4.2. Acrobot

Our second example illustrates the application of CPC to a popular class of control problems: balancing of an underactuated system around an unstable equilibrium point. In this experiment, however, no reference trajectories are provided by a functioning controller, as was the case with the planar walker. Instead, we use the CPC algorithm to construct a successful controller purely from the examples of uncontrolled failures. It is only required that the examples start at the equilibrium state, with a small random noise added to the controls to trigger a failure. To appreciate the difficulty of this task, note that it is akin to learning to balance on a tightrope simply by observing the failures of a person with no balancing skills.

The main idea is to set \( s_\epsilon = -1 \), which corresponds to reversing the time of reference trajectories, thus directing the system toward the equilibrium point. The idea of using a time reversal for stabilizing an unstable equilibrium is only straightforward for a fully actuated system. We investigate how well it works in practice for an underactuated system in the case of an acrobot - a double inverted pendulum with an actuated elbow, see Fig. 1.

A CPC controller is constructed from \( N_f \) fall trajectories recorded for the duration of one second (100 target points) starting from a completely upright configuration under a small Gaussian noise \( \sigma_0 \) of the joint torque, see Appendix B.2 for details. A handful of fall examples (\( N_f \approx 10^3 \)) is sufficient to produce (with a high probability) a controller stable against the same level of the torque noise \( \sigma_0 \). To achieve stability against a stronger noise of \( 6\sigma_0 \), it takes about \( N_f \approx 10^2 \) failure samples (corresponding to \( |X_d| = 10^3 \)). This is illustrated in Fig. 3, displaying the mean fall time \( \langle t_f \rangle \) for a given \( N_f \), averaged across independent controllers, when measured to the maximum time of \( T = 30 \) s. Note that for large \( T \) the controllers mostly fall into two categories: stable controllers with...
$t_f = T$ or unstable controllers with $t_f \ll T$. Therefore, $(T - (t_f))/T$ shows the fraction of unstable controllers. We can see that the probability of an unstable controller appears to quickly vanish as $N_f$ increases.

5. CONCLUSION

In this paper we made three contributions: 1) we introduced the CPC formalism and a CPC-based algorithm for controlling underactuated systems, 2) we elucidate its relation to the ZD approach, 3) we empirically demonstrated that CPC leads to a significant enhancement of stability in RL trained policies of a bipedal walker.

We provided an explicit connection between CPC and ZD, particularly in the context of the HZD literature [30], by identifying the zero-dynamics virtual constraints that give rise to a HZD controller identical to the CPC controller in the HGL. There are substantial differences between the two approaches as well, which we summarize below, that make CPC especially suitable for stabilizing RL-learned policies.

In CPC one does not need to construct virtual constraints and to choose a gait phasing parameter, ensuring its monotonicity, as one does in HZD. This may not even be feasible for an unknown policy. Also, CPC is invariant with respect to the choice of free coordinates, while HZD is generally not. In practice, HZD is often used in a trajectory-centric manner, that is for periodic motion on a one-dimensional manifold. CPC, on the other hand, relies on a cloud of target points and can naturally handle a wide distribution of the initial state. Also, for the target point selection, CPC incorporates the expected return foresight, either from the RL value function or from the recorded data.

The validity and utility of our approach was verified empirically by stabilizing RL-trained policies of a bipedal walker. The original policies were trained with an unbiased policy gradient method, similar to PPO. The RL policies were then compared with their CPC-stabilized counterparts in a number of stability testing experiments. In all the experiments, remarkable improvements in stability were observed, indicating a promising direction of research.

Off-line RL is an active area of research [37] with the potential of greatly facilitating RL, as it does not rely on expensive on-line data collection. Our approach addresses the same problem as off-line RL: converting off-line training data into a viable policy. This problem is challenging, as the off-line trained policies, with rare exception, are outperformed by the on-line trained policies [37]. It is all the more impressive that our approach produces superior policies greatly outperforming the original on-line RL policies in terms of stability, while relying on a small amount (only $10^4$ data points) of the off-line data. In the future, we would like to apply CPC to the standard off-line RL data sets, such as D4RL [37].

APPENDIX A: THEOREM PROOFS

A.1. Path convergence theorem

Proof: Throughout the proof we assume $0 \leq t < O(\varepsilon)$. Consider a system under the CPC control law $\tau_{CPC}(t)$. Let $\Delta \tau = \tau_{CPC} - \tau_t$. From (7) and the definition of the renormalized target trajectory $q_\tau^*(t)$ (see (11)) it follows that

$$\Delta^r \tilde{q} = 0 + O(\Delta \varepsilon, 1),$$

and $\chi(t)$ satisfies

$$\Delta^r \chi = B_\chi \Delta \tau + O(\Delta \varepsilon, 1).$$

Let us define: $a^{(k)}(t)$ to be the $k$-th time derivative of $a(t)$, and $z = BB^{-1} = [\mathcal{M}_B B_\chi B_\chi^{-1}]$. By integrating (A.1) one can see that, if $\Delta^r \chi(0)$ satisfies the reachability equation, then

$$\Delta^r q^{(k)}(t) = 0 + O(\Delta \varepsilon^{3-k}, \varepsilon^{2-k} + \hat{r}_0^{-2-k}),$$

and

$$z\Delta^r \chi^{(k)}(t) = \Delta^r q^{(k)}(t) + O(\Delta \varepsilon^{3-k}, \varepsilon^{2-k} + \hat{r}_0^{-2-k}),$$

for $k \in \{0, 1, 2\}$. Plugging in $\tau_{CPC}$ from (14) into (A.2), multiplying the result by $z$ from the left and using (A.4), we find

$$\Delta^r \tilde{q} + \frac{k_d}{\varepsilon} \Delta^r \tilde{\tau} + \frac{k_\delta}{\varepsilon^2} \Delta^r q = 0 + O(\Delta \varepsilon, 1 + \hat{r}_0^2 \varepsilon^2).$$

(A.5)

Note that $\Delta \varepsilon(t)$ is $O(\Delta \varepsilon(0)/\varepsilon^2)$, and therefore $O(\Delta \varepsilon) = O(\Delta \varepsilon^2)/\varepsilon$. It then follows from (A.5), that both $\Delta q(t)$ and $\Delta q(t)$ reduce down to $O(\Delta \varepsilon q(0), \varepsilon^2 + \hat{r}_0^2)$ on the time scale $O(\varepsilon)$.

A.2. Reparameterization invariance theorem

Proof: We can repeat every step of the proof of Theorem 1 till (A.5), but with the transformed quantities. We then arrive at

$$\Delta^r \tilde{q} + \frac{k_d}{\varepsilon} \Delta^r \tilde{\tau} + \frac{k_\delta}{\varepsilon^2} \Delta^r q = 0 + O(\Delta \varepsilon, 1 + \hat{r}_0^2 \varepsilon^2).$$

(A.6)

Because $C$ is invertible, we see that under $\tau_{CPC}(t)$ the system follows the exact same trajectory as under $\tau(t)$. Since the controls are assumed to be independent, it implies $\tau_{CPC}(t) = \tau(t)$.

A.3. CPC-ZD correspondence theorem

Proof: Note that $\Delta^{ZD} q$ satisfies

$$c^T \Delta^{ZD} q^{(k)} = 0.$$  

(A.7)
If $c \propto b$, then from (A.3) and (A.7) it follows that
\[ q_d^{zd} \approx q_d. \quad \text{(A.8)} \]

Differentiating $h(q)$ with respect to $q$ we find
\[ \frac{\partial h(q)}{\partial q} = [L_{\alpha}, \partial_{\alpha}] - \frac{\partial h_d(\theta(q))}{\partial \theta} c^T, \quad \text{(A.9)} \]

where $\partial_{\alpha}$ is a vector of $M$ zeros. Then from (A.7)-(A.9) it follows:
\[
\frac{\partial h(q)}{\partial q} \Delta^q (\xi) = \frac{\partial h(q)}{\partial q} \Delta^{zd} (\xi) = \Delta^{zd} \chi(q) - \frac{\partial h_d(\theta(q))}{\partial \theta} c^T \Delta^{zd} (\xi) = y(q). \quad \text{(A.10)}
\]

Let us define $A$
\[ A = \frac{\partial h(q)}{\partial q} B. \quad \text{(A.11)} \]

In the HGL $B(q) \rightarrow B$ and $A(q) \rightarrow A$. We define $K_q = \left( \kappa_q, \kappa_q \right) \otimes L_q$. Using (14), (31), (A.4), (A.10), and (A.11), we can now prove the theorem
\[
\Delta \tau_{cpc} = -B^{-1}_x K_M \Delta x = -A^{-1} AB^{-1}_x K_M \Delta x
\]
\[ = -A^{-1} \frac{\partial h(q)}{\partial q} z K_M \Delta x
\]
\[ \approx -A^{-1} \frac{\partial h(q)}{\partial q} K_N \Delta x
\]
\[ \approx -A^{-1} \left( \frac{k_q}{\epsilon} \xi + \frac{k_d}{\epsilon} \eta \right) \rightarrow \Delta \tau_{zd}. \quad \text{(A.12)}
\]
as $\epsilon \rightarrow 0$.

APPENDIX B: DETAILS OF EFFICIENT SEARCH AND EXPERIMENTAL SETUP

B.1. Tight bounds of efficient ball-tree search

In a ball tree, each node is associated with a sphere containing all the points of a subtree rooted at this node. Let a sphere of radius $\rho$ be centered at some point $[q_c; \dot{q}_c]$. A point $x_0$ inside the sphere can be parameterized by $(\xi, \eta) = (q_0 - q_c, \dot{q}_0 - \dot{q}_c)$, with $\xi^2 + \eta^2 \leq \rho^2$. As follows from (10) and (15), the proximity loss has the form
\[ L^{\text{prox}} = \left( \alpha_\xi + \beta_\xi^T \xi \right)^2 + (\alpha_\eta + \beta_\eta^T \eta)^2, \quad \text{(B.1)} \]

where
\[ \beta_\xi = \frac{\omega b}{q_0}, \quad \alpha_\xi = \beta_\xi^T (q_c - q_0), \]
\[ \beta_\eta = \frac{b}{\dot{q}_0}, \quad \alpha_\eta = \beta_\eta^T q_c - s_\eta. \quad \text{(B.2)} \]

To construct bounds we need to only consider $\xi \propto \beta_\xi$ and $\eta \propto \beta_\eta$, because a component of $\xi$ orthogonal to $\beta_\xi$ increases $\xi^2 + \eta^2$ without affecting $L^{\text{prox}}$, and similarly for $\eta$. The lowest possible loss value $L_d = 0$ is reached at $(\xi, \eta) = (\xi_0, \eta_0)$
\[ (\xi_0, \eta_0) = \left( -\alpha_\xi \beta_\xi, -\frac{\alpha_\eta \beta_\eta}{\beta_\eta^2 + \beta_\eta^2} \right), \quad \text{(B.3)} \]

provided $\xi_0^2 + \eta_0^2 \leq \rho^2$. Otherwise, the lower bound $L_d > 0$ and it should be looked for on the sphere $\xi^2 + \eta^2 = \rho^2$. In that case we parameterize $\xi$ and $\eta$ as
\[ (\xi, \eta) = \left( \frac{\beta_\xi}{\beta_\eta} \rho \cos \theta, \frac{\beta_\eta}{\beta_\eta} \rho \sin \theta \right). \quad \text{(B.4)} \]

The extrema of $L^{\text{prox}}$ are given by $\partial L^{\text{prox}}/\partial \theta = 0$
\[ \rho (\beta_\xi^2 - \beta_\eta^2) \cos \theta \sin \theta + \alpha_\xi \beta_\xi \sin \theta \]
\[ = \alpha_\eta \beta_\eta \cos \theta. \quad \text{(B.5)} \]

Squaring both sides of (B.5) we obtain a fourth degree polynomial in $\cos \theta$
\[ \left[ \rho (\beta_\xi^2 - \beta_\eta^2) \cos \theta + \alpha_\xi \beta_\xi \sin \theta \right]^2 (1 - \cos^2 \theta)
\]
\[ - (\alpha_\eta \beta_\eta \cos \theta)^2 = 0, \quad \text{(B.6)} \]

that can be solved analytically. The lowest and highest values of the loss corresponding to the real roots $\in [-1, 1]$ of the polynomial, that also satisfy (B.5), are then the lower and upper bounds $L_d$ and $L_u$.

B.2. Details of the experimental setup

Our experiments were simulated using open dynamics engine (ODE). The walker is modeled as a planar chain of 4 identical segments (capped ODE cylinders of length 1 radius 0.1 and density 1), connected by powered joints. We set the gravitational acceleration to 10, the simulation time step $\Delta t$ to 0.005 and the ODE friction coefficient parameter to 1. The control matrix $B$ is estimated from the last $n = 9$ points of the current trajectory.

We have not searched extensively for the algorithm hyperparameter values, settling instead with the values that appeared reasonable after a few trials: $\omega = 10$, $s_\xi = 1$, $n_d = 20$, $k_0 = 2000$, $\tau_\xi = 2$ and $k_\dot{q} = 2$.

Within the RL framework, the walker is controlled by a Gaussian policy $\tau \sim N(\mu(x), \sigma^2(x))$ with fixed noise strength $\sigma_0 = 0.02$ and $\mu(x)$ represented by a 5-layer NN with 4860 weights. The walker’s goal is to walk efficiently with a velocity close to $v_f = 1$. To increase the gait stability, the reward function includes a penalty term for swinging a foot too close to the ground.

In the acrobot case, the parameters that differ from the walker setup are: $s_\xi = -1$, $\Delta t = 0.01$ and $n = 7$. Also, we set $\tau_\xi = V(x_d) = r(x_d) = 0$ for the purpose of computing $V_f$ and $V_{it}$ in (22) and (23), as no good reference policy is provided in this task.
REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems, vol. 25, pp. 1097-1105, 2012.

[2] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, pp. 436-444, 2015.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82-97, 2012.

[4] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent trends in deep learning based natural language processing,” IEEE Computational Intelligence Magazine, vol. 13, no. 3, pp. 55-75, 2018.

[5] S. P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi, and S. Jain, “Machine translation using deep learning: An overview,” Proc. of International Conference on Computer, Communications and Electronics (Comptelix), pp. 162-167, IEEE, 2017.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” arXiv preprint arXiv:1312.5602, pp. 1-9, 2013.

[7] E. Gibney, “Google AI algorithm masters ancient game of Go,” Nature, vol. 529, no. 7587, pp. 445-446, 2016.

[8] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, “Benchmarking deep reinforcement learning for continuous control,” Proc. of International Conference on Machine Learning, vol. 48, pp. 1329-1338, 2016.

[9] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, “Trust region policy optimization,” Proc. of the 32nd International Conference on Machine Learning, vol. 37, pp. 1889-1897, 2015.

[10] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, “High-dimensional continuous control using generalization advantage estimation,” Proc. of 4th International Conference on Learning Representations, pp. 1-14, 2016.

[11] S. Gu, E. Holly, T. Lillicrap, and S. Levine, “Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates,” Proc. of IEEE International Conference on Robotics and Automation, pp. 3389-3396, IEEE, 2017.

[12] S. Kuindersma, R. Deits, M. F. 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, vol. 40, no. 3, pp. 429-455, 2016.

[13] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, “Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway,” Proc. of American Control Conference (ACC), pp. 4559-4566, IEEE, 2019.

[14] S. Shigemi, “ASIMO and humanoid robot research at Honda,” Humanoid Robotics: A Reference, pp. 55-90, 2018.

[15] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, “Bigdogg, the rough-terrain quadruped robot,” IFAC Proceedings Volumes, vol. 41, no. 2, pp. 10822-10825, 2008.

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

[17] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, “Anymal-A highly mobile and dynamic quadrupedal robot,” Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 38-44, IEEE, 2016.

[18] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection,” The International Journal of Robotics Research, vol. 37, no. 4-5, pp. 421-436, 2018.

[19] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” The International Journal of Robotics Research, vol. 32, no. 11, pp. 1238-1274, 2013.

[20] J. P. Hanna and P. Stone, “Grounded action transformation for robot learning in simulation,” Proc. of the AAAI Conference on Artificial Intelligence, vol. 31, pp. 3834-3840, 2017.

[21] F. Sadeghi and S. Levine, “CAD2RL: Real single-image flight without a single real image,” Robotics: Science and Systems XIII, pp. 1-12, 2017.

[22] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” Proc. of International Conference on Intelligent Robots and Systems (IROS), pp. 23-30, IEEE, 2017.

[23] S. James, A. J. Davison, and E. Johns, “Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task,” Proc. of Conference on Robot Learning, pp. 334-343, PMLR, 2017.

[24] F. Muratore, M. Gienger, and J. Peters, “Assessing transferability from simulation to reality for reinforcement learning,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 4, pp. 1172-1183, 2021.

[25] J. M. Wang, D. J. Fleet, and A. Hertzmann, “Optimizing walking controllers for uncertain inputs and environments,” ACM Transactions on Graphics (TOG), vol. 29, no. 4, pp. 1-8, 2010.

[26] H. K. Venkataraman and P. J. Seiler, “Recovering robustness in model-free reinforcement learning,” Proc. of American Control Conference (ACC), pp. 4210-4216, IEEE, 2019.
[27] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, “Robust adversarial reinforcement learning,” Proc. of International Conference on Machine Learning, pp. 2817-2826, PMLR, 2017.

[28] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, “Adversarially robust policy learning: Active construction of physically-plausible perturbations,” Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3932-3939, IEEE, 2017.

[29] C. I. Byrnes and A. Isidori, “Asymptotic stabilization of minimum phase nonlinear systems,” IEEE Transactions on Automatic Control, vol. 36, no. 10, pp. 1122-1137, 1991.

[30] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, Feedback Control of Dynamic Bipedal Robot Locomotion, CRC press, 2018.

[31] J. W. Grizzle and C. Chevallereau, “Virtual constraints and hybrid zero dynamics for realizing underactuated bipedal locomotion,” arXiv preprint arXiv:1706.01127, pp. 1-17, 2017.

[32] B. D. Anderson and J. B. Moore, Optimal Control: Linear Quadratic Methods, Courier Corporation, 2007.

[33] B. Griffin and J. Grizzle, “Nonholonomic virtual constraints and gait optimization for robust walking control,” The International Journal of Robotics Research, vol. 36, no. 8, pp. 895-922, 2017.

[34] J. C. Horn, A. Mohammadi, K. A. Hamed, and R. D. Gregg, “Hybrid zero dynamics of bipedal robots under nonholonomic virtual constraints,” IEEE Control Systems Letters, vol. 3, no. 2, pp. 386-391, 2018.

[35] S. M. Kakade, “A natural policy gradient,” Advances in Neural Information Processing Systems, pp. 1531-1538, 2002.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” arXiv preprint arXiv:1707.06347, pp. 1-12, 2017.

[37] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, “D4RL: Datasets for deep data-driven reinforcement learning,” arXiv preprint arXiv:2004.07219, pp. 1-19, 2020.

Sergey Pankov received his Ph.D. degree in physics from Rutgers University in 2003. His research interests include legged locomotion control, reinforcement learning, and deep learning.

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.