Data-Efficient and Safe Learning for Humanoid Locomotion Aided by a Dynamic Balancing Model

Junhyeok Ahn, Jaemin Lee, and Luis Sentis

***Abstract*—In this letter, we formulate a novel Markov Decision Process (MDP) for safe and data-efficient learning for humanoid locomotion aided by a dynamic balancing model. In our previous studies of biped locomotion, we relied on a low-dimensional robot model, commonly used in high-level Walking Pattern Generators (WPGs). However, a low-level feedback controller cannot precisely track desired footstep locations due to the discrepancies between the full order model and the simplified model. In this study, we propose mitigating this problem by complementing a WPG with reinforcement learning. More specifically, we propose a structured footstep control method consisting of a WPG, a neural network, and a safety controller. The WPG provides an analytical method that promotes efficient learning while the neural network maximizes long-term rewards, and the safety controller encourages safe exploration based on step capturability and the use of control-barrier functions. Our contributions include the following (1) a structured learning control method for locomotion, (2) a data-efficient and safe learning process to improve walking using a physics-based model, and (3) the scalability of the procedure to various types of humanoid robots and walking.**

***Index Terms*—Humanoid and bipedal locomotion, deep learning in robotics and automation, model learning for control.**

I. Introduction and Related Work

HUMANOID robots are advantageous for mobility in tight spaces. However, fast bipedal locomotion requires precision control of the contact transition process. Many studies have successfully addressed agile and versatile legged locomotion. Analytic approaches have employed differential dynamics of robots to synthesize locomotion controllers. Data-driven approaches have leveraged the representational power of neural networks and designed locomotion policies in an end-to-end manner. Our work combines the advantages of these approaches to achieve locomotion behaviors both safely and efficiently.

Manuscript received December 19, 2019; accepted April 5, 2020. Date of publication April 27, 2020; date of current version June 5, 2020. This letter was recommended for publication by Associate Editor R. Cisneros Limon and Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported in part by the Office of Naval Research, under Grant #N000141512507 and in part by the National Science Foundation, under Grant #1724360. *(Corresponding author: Luis Sentis.)*

Junhyeok Ahn and Jaemin Lee are with the Department of Mechanical Engineering, the University of Texas at Austin, Austin, TX 78712 USA (e-mail: junhyeokahn91@utexas.edu; jmlee87@utexas.edu).

Luis Sentis is with the Department of Aerospace Engineering and Engineering Mechanics, the University of Texas at Austin, Austin, TX 78712 USA (e-mail: lsentis@austin.utexas.edu).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.2990743

Analytic approaches decouple the problem into two subproblems: (1) reducing the complexity of full-body dynamics via simplified models, such as the inverted pendulum [1]–[4] or the centroidal model [5]–[7], to generate high-level walking patterns, and then (2) computing feedback joint commands at every control loop so that the robot tracks the behavior of the simplified models. In our recent studies [8], [9], we achieved unsupported passive ankle dynamic locomotion via two computational elements: (1) a high-level footstep planner, called the Time-to-Velocity-Reversal (TVR) planner, based on the Linear Inverted Pendulum Model (LIPM) and (2) a low-level Whole Body Controller (WBC) that tracks the desired trajectories. Although abstractions based on simplified models enable Walking Pattern Generators (WPGs) to provide computational efficiency and tools for stability analysis, they have a limited ability to incorporate complicated physical effects, such as angular momentum and limb dynamics. As a result, using WPGs cause significant footstep tracking errors, requiring arduous parameter tuning [10]. In this letter, we propose and train a policy that compensates for the limited representation accuracy of WPGs and generates practical walking patterns by incorporating simple physical models.

On the other hand, data-driven approaches have demonstrated the possibility of robust and agile locomotion control through Reinforcement Learning (RL). Model-free RL learns a walking policy via explicit trial and error without using knowledge of the dynamics of the robots. In [11], [12], locomotion policies were trained for various environments and achieved robust locomotion behaviors. In contrast, model-based RL learns a model of a robot through interactions with the environment and leverages the constructed model for planning. The approach in [13] iteratively fitted a local model for a planar walker and performed trajectory optimization, which demonstrated the ability to learn a walking policy efficiently. However, most data-driven approaches for locomotion do not consider the underlying physics of the robot nor prior knowledge and instead train policies from sensor data to joint commands in an end-to-end manner. Therefore, they require substantial training data and often result in unnatural jerky motions, which make the methods challenging to deploy in real hardware. In contrast to these end-to-end methods, our framework learns a policy from sensor data to footstep locations (instead of joint torques) and utilizes a whole-body controller to track the desired trajectories. In the footstep decision making, we rely on a LIPM and a TVR planner to encourage safe and efficient exploration in policy training.

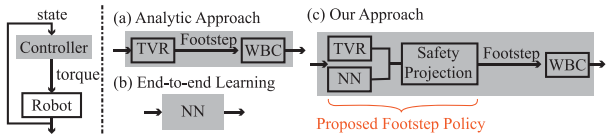

Fig. 1. The left figure illustrates the control structure of a robot, where a controller takes the robot's states and computes joint torques in an end-to-end manner. (a) Analytic approaches compose the controller with a WPG (e.g., a TVR Planner) and a feedback controller (e.g., WBC), whereas (b) end-to-end learning methods train a neural network to compute the joint torques. (c) Our controller includes the footstep policy learning algorithm and WBC, where the footstep policy has three components.

There have been few works that incorporate physical insight and stable feedback control to learn biped locomotion. In [14], an RL agent learns a set of trajectory parameters instead of joint commands, followed by a feedback controller to stabilize the robot along the resulting trajectory. However, the policy is trained in a model-free manner and can yield infeasible trajectories that make the robot explore unsafe state-space regions. In contrast, our work proposes a structured policy with a safety mechanism as well as a TVR planner and a neural network to foster safe and efficient policy search.

Previous works have explored the idea of learning residual actions combined with analytical models. In [15], a ball-throwing action is adjusted by a trained neural network to mitigate model discrepancies for a ball-tossing robot. In [16], swing foot trajectories generated by a feedback controller are modulated to improve performance and data efficiency for a quadruped. Our proposed algorithm can be seen as an extension of these ideas to bipedal robots that also includes safety considerations. Compared to robot manipulators or quadrupeds, biped robots fall more often and benefit from the use of physics-based models to guide the learning process.

In this letter, we devise a Markov Decision Process (MDP) that combines analytic models and data-driven approaches to achieve agile and robust locomotion. In contrast to end-to-end learning such as in [11], [13], [17], [18], whose learning techniques take joint information and map them to joint torque commands, our method learns a policy to make high-level decisions in terms of desired footstep locations. It then uses a feedback whole-body controller to generate locomotion behaviors and the desired reward signals. Our structured footstep control methods includes a TVR planner, a neural network, and a safety controller. The TVR planner provides feasible sub-optimal guidance, the neural network maximizes the long-term reward, and the safety controller encourages safe exploration during the learning process. This safety controller learns the residual dynamics of the LIPM and projects the action onto safe regions considering a walking capturability metric. The overall structures of our method and those of related works are shown and compared in Fig. 1.

The proposed MDP formulation has the following advantages: (1) it bridges the gap between analytic and data-driven approaches, which mitigates the limited effect of using simple models; (2) it allows data efficiency and safe learning; and (3) it can be used for different types of locomotion and in different types of robots.

The remainder of this letter is organized as follows. Section II describes an analytic approach for biped locomotion and RL with safety guarantees. Section III proposes an MDP formulation for humanoid locomotion tasks. Section IV shows the design of a footstep policy that allows safe exploration and data-efficient training. Section V evaluates the effectiveness and generalization of the proposed framework in simulation, and Section VI concludes the letter.

II. Preliminaries

A. An Analytic Approach to Locomotion

We define a *Locomotion State* and a state machine with simple structures to represent general locomotion behaviors.

Definition 1: (**Locomotion State**) A locomotion state is defined as a tuple, $\mathfrak{L} := (\mathcal{L}, T_{\mathcal{L}})$.

- $\mathcal{L}$ represents a semantic expression of locomotion behaviors: $\mathcal{L} \in \{\mathcal{L}_{DS[r/l]}, \mathcal{L}_{LF[r]}, \mathcal{L}_{LF[l]}, \mathcal{L}_{LN[r]}, \mathcal{L}_{LN[l]}\}$.
- The subscripts $(.)_{DS[r/l]}$, $(.)_{LF[r/l]}$, and $(.)_{LN[r/l]}$ describe locomotion states for double support, lifting the right/left leg, and landing the right/left leg, respectively.
- $T_{\mathcal{L}}$ is a time duration for $\mathcal{L}$ and can be chosen based on the desired stepping frequency.

Definition 2: (**State Machine**) We define a state machine as a sequence of Locomotion States:

$$\mathcal{SM} := \{\mathfrak{L}_{DS[r/l]}, \mathfrak{L}_{LF[r/l]}, \mathfrak{L}_{LN[r/l]}\}$$

- The list above is sequential in the order shown.
- The *Locomotion State* $\mathfrak{L}_{LN[r/l]}$ terminates when a contact is detected between the swing foot and the ground.

Definition 3: (**Apex Moment** and **Switching Moment**) Given the $\mathcal{SM}$ defined above, an Apex Moment defines the switch between $\mathfrak{L}_{LF[r/l]}$ and $\mathfrak{L}_{LN[r/l]}$, and we label it as t_a. A Switching Moment defines the middle of $\mathfrak{L}_{DS[r/l]}$, and we label it as t_s.

Let us consider the LIPM for our simplified model. We define the *LIPM state* as the position and velocity of the Center of Mass (CoM) of the robot on a constant height surface with an expression, $\mathbf{x} = [x, y, \dot{x}, \dot{y}]^\top \in \mathbb{R}^4$. The *LIPM stance* is defined as the location of the pivot and represented by $\mathbf{p} = [p_x, p_y]^\top \in \mathbb{R}^2$. We define the *LIPM input* as the desired location of the next stance with an expression $\mathbf{a} = [a_x, a_y]^\top \in \mathbb{R}^2$. We use the subscript k to represent properties in the kth step, for example, $\mathbf{x}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]^\top$, $\mathbf{p}_k = [p_{k,x}, p_{k,y}]^\top$, and $\mathbf{a}_k = [a_{k,x}, a_{k,y}]^\top$. We further use the subscripts k, a, and k, s to denote the properties of the robot at the *Apex Moment* and the *Switching Moment* at the kth step. For example, $\mathbf{x}_{k,*} = [x_{k,*}, y_{k,*}, \dot{x}_{k,*}, \dot{y}_{k,*}]^\top = \mathbf{x}_k(t_{k,*})$, where $* \in \{a, s\}$ represents the *LIPM state* evaluated at the *Apex Moment* and *Switching Moment* at the kth step. Because the *LIPM stance* and *LIPM input* are invariant during the step, $\mathbf{p}_{k,a}$ and $\mathbf{a}_{k,a}$ are interchangeable with $\mathbf{p}_k$ and $\mathbf{a}_k$. We also use these subscripts to describe the properties of a robot. For instance, $\boldsymbol{\phi}^{\text{bs}}_{k,a} \in SO(3)$ and $\mathbf{w}^{\text{bs}}_{k,a} \in \mathbb{R}^3$ represent the orientation and angular velocity of a base link, respectively, and $\boldsymbol{\phi}^{\text{pv}}_{k,a} \in SO(3)$ represents the orientation of a stance foot (a pivot) with respect to the world

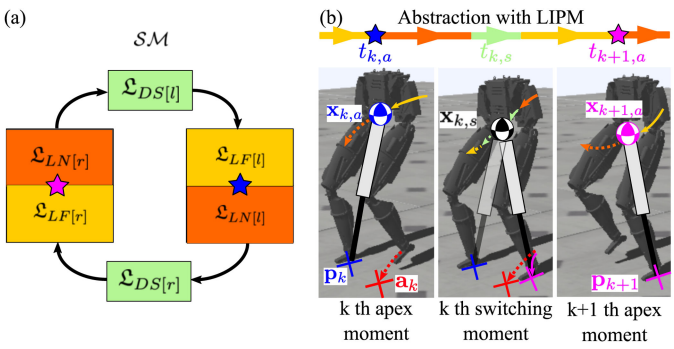

Fig. 2. (a) shows the $\mathcal{SM}$ for locomotion behaviors. The blue and pink stars represent the kth and $k+1$th *Apex Moments*. (b) shows the walking motion with the $\mathcal{SM}$ and its abstraction using the LIPM.

frame at the *Apex Moment* at the kth step. Fig. 2 illustrates the $\mathcal{SM}$ and the abstraction of the locomotion behavior with the LIPM.

The goal of the WPG is to generate $\mathbf{a}_k$ and the CoM trajectory based on $\mathbf{x}_{k,a}$ and $\mathbf{p}_k$ at the *Apex Moment*. From the walking pattern, the low-level WBC provides the computation of sensor-based feedback control loops and torque command for the robot to track the desired location of the next stance and the CoM trajectory. Note that the WPG designs the pattern at the *Apex Moment* at each step, while the WBC computes the feedback torque command at every control loop.

B. TVR Planner

The differential equation of the LIPM is represented as follows:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ g/h & 0 & 0 & 0 \\ 0 & g/h & 0 & 0 \end{bmatrix} \mathbf{x}(t) - \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ g/h & 0 \\ 0 & g/h \end{bmatrix} \mathbf{p}, \quad (1)$$

where g is the gravitational constant and h is the constant height of the CoM of the point mass.

At the kth step, given an initial condition $\mathbf{x}_k(0) = \mathbf{x}_{k,0}$ and a stance position $\mathbf{p}_k$, the solution of Eq. (1) yields a state transition map Ψ, with the expression

$$\mathbf{x}_k(t) = \Psi(t\,;\, \mathbf{x}_{k,0}, \mathbf{p}_k) = f_\Psi(t)\mathbf{x}_{k,0} + g_\Psi(t)\mathbf{p}_k, \quad (2)$$

where

$$f_\Psi(t) := \begin{bmatrix} C_1(t) & 0 & C_2(t) & 0 \\ 0 & C_1(t) & 0 & C_2(t) \\ C_3(t) & 0 & C_1(t) & 0 \\ 0 & C_3(t) & 0 & C_1(t) \end{bmatrix},$$

$$g_\Psi(t) := \begin{bmatrix} 1 - C_1(t) & 0 \\ 0 & 1 - C_1(t) \\ -C_3(t) & 0 \\ 0 & -C_3(t) \end{bmatrix},$$

$C_1(t) := \cosh(\omega t)$, $C_2(t) := \sinh(\omega t)/\omega$, and $C_3(t) := \omega \sinh(\omega t)$, and $\omega := \sqrt{g/h}$, respectively.

Because the TVR planner determines the desired location of the next stance at the *Apex Moment* (i.e., $t = t_{k,a}$), we set the initial condition as $\mathbf{x}_k(0) = \mathbf{x}_{k,a}$. With pre-specified time duration $T_{\mathcal{L}_{LN[r/l]}}$, we compute the state at the *Switching Moment* as

$$\mathbf{x}_{k,s} = \mathbf{x}_k(T_{\mathcal{L}_{LN[r/l]}}) = \Psi(T_{\mathcal{L}_{LN[r/l]}}\,;\, \mathbf{x}_k(t_{k,a}), \mathbf{p}_k). \quad (3)$$

From $\mathbf{x}_{k,s}$, the TVR planner computes $\mathbf{a}_k$, such that the sagittal velocity $\dot{x}$ (and lateral velocity $\dot{y}$, respectively) of the CoM is driven to zero at the predefined time intervals $T_{x'}$ (and $T_{y'}$, respectively) after the LIPM switches to the new stance. These constraints are expressed as

$$0 = \left\langle \boldsymbol{\xi}_j \,,\, \Psi(T_j\,;\, \mathbf{x}_{k,s}, \mathbf{a}_k) \right\rangle, \quad j \in \{x', y'\}, \quad (4)$$

where $\boldsymbol{\xi}_{x'} := [0, 0, 1, 0]^\top$ and $\boldsymbol{\xi}_{y'} := [0, 0, 0, 1]^\top$. From Eq. (4), $\mathbf{a}_k$ is computed with an additional bias term κ_x and κ_y as

$$\mathbf{a}_k^{\text{TVR}} = \Phi(\mathbf{x}_{k,s}) = f_\Phi(T_{x'}, T_{y'})\mathbf{x}_{k,s} + g_\Phi, \quad (5)$$

where

$$f_\Phi(T_{x'}, T_{y'}) := \begin{bmatrix} 1 - \kappa_x & 0 & C_4(T_{x'}) & 0 \\ 0 & 1 - \kappa_y & 0 & C_4(T_{y'}) \end{bmatrix},$$

$$g_\Phi := \begin{bmatrix} \kappa_x & 0 \\ 0 & \kappa_y \end{bmatrix} \begin{bmatrix} x^d \\ y^d \end{bmatrix},$$

$C_4(T) := \frac{e^{wT} + e^{-wT}}{w(e^{wT} - e^{-wT})}$ and $[x^d, y^d]^\top \in \mathbb{R}^2$ represents a desired position for the CoM of the robot. Note that Eq. (5) is a simple proportional-derivative controller and that $T_{x'}, T_{y'}, \kappa_x$, and κ_y are the gain parameters used to keep the CoM converging to the desired position. A more detailed derivation of the LIPM was described in [19].

C. Reinforcement Learning With Safe Exploration

Consider an infinite-horizon discounted MDP with control-affine, deterministic dynamics defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho_0, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{T} : S \mapsto S$ is the deterministic dynamics, in our case affine in the controls, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \mapsto \mathbb{R}$ is the distribution of the initial state, and $\gamma \in (0, 1)$ is the discount factor. The control affine dynamics are written as

$$\mathbf{s}_{k+1} = f(\mathbf{s}_k) + g(\mathbf{s}_k)\mathbf{a}_k + d(\mathbf{s}_k), \quad (6)$$

where $\mathbf{s}_k \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$, and $\mathbf{a}_k \in \mathcal{A} \subset \mathbb{R}^{n_a}$ represent a state and input, respectively. $f : \mathcal{S} \mapsto \mathcal{S}$, and $g : \mathcal{S} \mapsto \mathbb{R}^{n_s \times n_a}$ are the analytic underactuated and actuated dynamics, respectively, while $d : \mathcal{S} \mapsto \mathcal{S}$ is the unknown part of the system dynamics. Moreover, let $\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})$ represent a stochastic control policy parameterized by a vector $\boldsymbol{\theta}$. $\pi_{\boldsymbol{\theta}} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}_{\geq 0}$ maps states to distributions over actions, and $V_{\pi_{\boldsymbol{\theta}}}(\mathbf{s})$ represents the policy's expected discounted reward with the expression

$$V_{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_k) = \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}} \left[\sum_{i=0}^{\infty} \gamma^i r(\mathbf{s}_{k+i}, \mathbf{a}_{k+i}) \right], \quad (7)$$

where $\tau \sim \pi_{\boldsymbol{\theta}}$ is a trajectory drawn from the policy $\pi_{\boldsymbol{\theta}}$ (e.g., $\tau = [\mathbf{s}_k, \mathbf{a}_k, \ldots, \mathbf{s}_{k+n}, \mathbf{a}_{k+n}]$).

For safe exploration in the learning process under uncertain dynamics, the work in [20] employed a Gaussian Process (GP) to approximate the unknown part of the dynamics from the dataset by learning a mean estimate $\boldsymbol{\mu}_d(\mathbf{s})$ and an uncertainty $\boldsymbol{\sigma}_d^2(\mathbf{s})$ in tandem with the policy update with probability confidence intervals on the estimation,

$$\boldsymbol{\mu}_d(\mathbf{s}) - k_\delta \boldsymbol{\sigma}_d(\mathbf{s}) \leq d(\mathbf{s}) \leq \boldsymbol{\mu}_d(\mathbf{s}) + k_\delta \boldsymbol{\sigma}_d(\mathbf{s}), \tag{8}$$

where k_δ is a design parameter indicating a confidence. Then, the control input is computed to keep the following state within a given invariant set $\mathcal{C} = \{\mathbf{s} \in \mathcal{S} \mid h(\mathbf{s}) \geq 0\}$ by computing

$$\sup_{\mathbf{a_k} \in \mathcal{A}} \left[h\left(f(\mathbf{s}_k) + g(\mathbf{s}_k)\mathbf{a}_k + d(\mathbf{s}_k)\right) + (\eta - 1)h(\mathbf{s}_k) \right] \geq 0, \tag{9}$$

where $\eta \in [0, 1]$.

III. MDP Formulation

We define a set of states $\mathcal{S}$ and a set of actions $\mathcal{A}$ associated with the *Apex Moment* at each step:

$$\mathcal{S} := \left\{ (\mathbf{x}_{k,a}, \mathbf{p}_{k,a}, \boldsymbol{\phi}^{\text{bs}}_{k,a}, \mathbf{w}^{\text{bs}}_{k,a}, \boldsymbol{\phi}^{\text{pv}}_{k,a}) \mid \forall k \in [1, m]_{\mathbb{N}} \right\},$$

$$\mathcal{A} := \left\{ \mathbf{a}_{k,a} \mid \forall k \in [1, m]_{\mathbb{N}} \right\},$$

where m can be set as $+\infty$ when considering the infinite steps of the locomotion. Recall from the nomenclatures in Section II-A that $\mathbf{x}_{k,a}, \mathbf{p}_{k,a}$ and $\mathbf{a}_{k,a}$ are the expressions of the *LIPM state*, *LIPM stance*, and *LIPM input* evaluated at the *Apex Moment*. Note that $\mathbf{p}_{k,a}$ and $\mathbf{a}_{k,a}$ are interchangeable with $\mathbf{p}_k$ and $\mathbf{a}_k$. Moreover, $\boldsymbol{\phi}^{\text{bs}}_{k,a}$ and $\mathbf{w}^{\text{bs}}_{k,a}$ represent the orientation and angular velocity of a base link and $\boldsymbol{\phi}^{\text{pv}}_{k,a}$ expresses an orientation of the stance foot at the *Apex Moment*. We divide the state into two parts as

$$\begin{aligned} \mathbf{s}_{k+1} &= \left[\mathbf{s}^u_{k+1} \; \mathbf{s}^l_{k+1} \right]^\top \\ &= \left[\mathbf{x}_{k+1,a} \; \mathbf{p}_{k+1} \; \boldsymbol{\phi}^{\text{bs}}_{k+1,a} \; \mathbf{w}^{\text{bs}}_{k+1,a} \; \boldsymbol{\phi}^{\text{pv}}_{k+1,a} \right]^\top \end{aligned} \tag{10}$$

and define a transition function for the upper part of the state based on Eq. (2) as

$$\begin{aligned} \mathbf{s}^u_{k+1} &= f(\mathbf{x}_{k,a}, \mathbf{p}_k) + g\mathbf{a}_k + d(\mathbf{x}_{k,a}, \mathbf{p}_k), \\ f(\mathbf{x}_{k,a}, \mathbf{p}_k) &:= \begin{bmatrix} f_\Psi(T_{LF})\Psi(T_{LN}; \mathbf{x}_{k,a}, \mathbf{p}_k) \\ \mathbf{0}_{2\times 1} \end{bmatrix}, \\ g &:= \begin{bmatrix} g_\Psi(T_{LF}) \\ \mathbf{I}_{2\times 2} \end{bmatrix}. \end{aligned} \tag{11}$$

$d(\mathbf{x}_{k,a}, \mathbf{p}_k)$ represents the unknown part of the dynamics fitted via Eq. (8).[1] The uncertainties are attributed to the discrepancies between the simplified model and the actual robot. Note that the dynamics of the lower part of the states, $\mathbf{s}^l_{k+1}$, cannot be expressed in closed form. Therefore, we optimize our policy in a model-free sense, but utilize the LIPM to provide safe exploration and data efficiency in the learning process.

[1]We use a squared exponential kernel for the GP process in later experiments.

To train a policy for a locomotion behavior, we adapt a reward function from [18], widely-used for locomotion tasks:

$$r(\mathbf{s}_k, \mathbf{a}_k) = r_a + r_b(\mathbf{s_k}) + r_t(\mathbf{s_k}) + r_s(\mathbf{s_k}) + r_c(\mathbf{a_k}). \tag{12}$$

Given $\mathbf{w}^{\text{bs}}_{k,a} = [w_{k,x}, w_{k,y}, w_{k,z}]^\top$, the Euler ZYX representation $[\boldsymbol{\phi}^{\text{bs}}_{k,x}, \boldsymbol{\phi}^{\text{bs}}_{k,y}, \boldsymbol{\phi}^{\text{bs}}_{k,z}]^\top$ of $\boldsymbol{\phi}^{\text{bs}}_k$ and $[\boldsymbol{\phi}^{\text{pv}}_{k,x}, \boldsymbol{\phi}^{\text{pv}}_{k,y}, \boldsymbol{\phi}^{\text{pv}}_{k,z}]^\top$ of $\boldsymbol{\phi}^{\text{pv}}$, r_a is an alive bonus, $r_b(\mathbf{s}_k) := -w_b \|(\boldsymbol{\phi}^{\text{bs}}_{k,x}, \boldsymbol{\phi}^{\text{bs}}_{k,y})\|^2$ penalizes the roll and pitch variation to keep the body upright, $r_t(\mathbf{s}_k) := -w_t \|(x^d_{k,a}, y^d_{k,a}, \boldsymbol{\phi}^{\text{bs,d}}_{k,z}, \boldsymbol{\phi}^{\text{pv,d}}_{k,z}) - (x_k, y_k, \boldsymbol{\phi}^{\text{bs}}_{k,z}, \boldsymbol{\phi}^{\text{pv}}_{k,z})\|^2$ penalizes divergence from the desired CoM positions and the heading of the robot, $r_s(\mathbf{s}_t) := -w_s \|(\dot{x}^d_{k,a}, \dot{y}^d_{k,a}, w^d_{k,z}) - (\dot{x}_{k,a}, \dot{y}_{k,a}, w_{k,z})\|^2$ is for steering the robot with a desired velocity, and $r_c(\mathbf{a}_k) := -w_c \|\mathbf{a}_k\|^2$ penalizes excessive control input.

IV. Policy Representation and Learning

Our goal is to learn an optimal policy for desired foot locations. We use the Proximal Policy Optimization (PPO) [17] to learn the policy iteratively. PPO defines an advantage function $A_{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_k, \mathbf{a}_k) := Q_{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_k, \mathbf{a}_k) - V_{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_k)$, where $Q_{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_k, \mathbf{a}_k)$ is the state-action value function that evaluates the return of taking action $\mathbf{a}_k$ at state $\mathbf{s}_k$ and following the policy π thereafter. By maximizing a modified objective function

$$L_{\text{PPO}}(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}} \left[\min\left(r_k A_k, clip(r_k,\, 1-\epsilon,\, 1+\epsilon) A_k \right) \right],$$

where $r_k := \frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_k|\mathbf{s}_k)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(\mathbf{a}_k|\mathbf{s}_k)}$ is the importance resampling term that allows us to use the dataset under the old policy $\pi_{\boldsymbol{\theta}_{\text{old}}}$ to estimate for the current policy $\pi_{\boldsymbol{\theta}}$. A_k is a short notation for $A_{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_k, \mathbf{a}_k)$. The $\min$ and $clip$ operator ensures that the policy $\pi_{\boldsymbol{\theta}}$ does not change excessively from the old policy $\pi_{\boldsymbol{\theta}_{\text{old}}}$.

A. Safe Set Approximation

The work in [21] introduced an instantaneous capture point that enables the LIPM to come to a stop if it places and maintains its stance there instantaneously. Here, we consider i-step capture regions for the LIPM at the *Apex Moment:*

$$\left\| \begin{bmatrix} 1 & 0 & 1/\omega & 0 & -1 & 0 \\ 0 & 1 & 0 & 1/\omega & 0 & -1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{k,a} \\ \mathbf{p}_k \end{bmatrix} \right\| \leq \mathcal{CP}_i, \tag{13}$$

where

$$\begin{aligned} \mathcal{CP}_1 &:= l_{\max} e^{-T_{\mathcal{L}_{LN}[r/l]}}, \\ \mathcal{CP}_2 &:= l_{\max} e^{-T_{\mathcal{L}_{LN}[r/l]}} (1 + e^{-T_{\mathcal{L}_{LN}[r/l]}}), \end{aligned}$$

$\omega = \sqrt{g/h}$, and $l_{\max}$ is the maximum step length that the LIPM can reach. Both ω and $l_{\max}$ are achieved from the kinematics of a robot. $T_{\mathcal{L}_{LN}[r/l]}$ is a predefined temporal parameter that represents the time period until the robot lands its swing foot. We conservatively approximate the ellipsoid of Eq. (13) with a polytope and define a safe set of states as

$$\mathcal{C} = \left\{ (\mathbf{x}_{k,a}, \mathbf{p}_k) \mid h(\mathbf{x}_{k,a}, \mathbf{p}_k) \geq \mathbf{0}_{4\times 1}, \forall k \in [1, m]_{\mathbb{N}} \right\}, \tag{14}$$

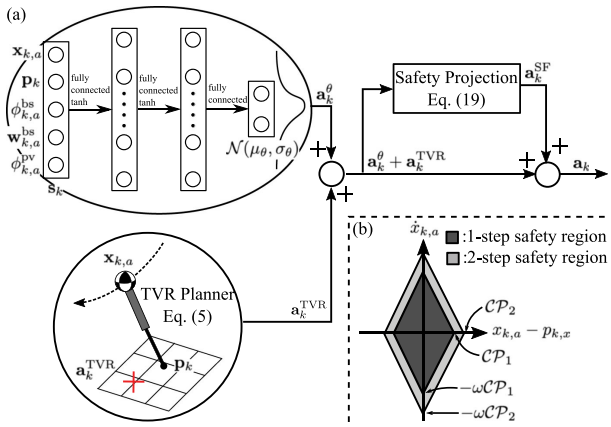

Fig. 3. (a) The design of the safety-guaranteeing policy, $\mathbf{a}_k$. (b) The projection onto the x and $\dot{x}$ plane of the one- and two-step capture regions of the LIPM.

where

$$h(\mathbf{x}_{k,a}, \mathbf{p}_k) := \mathbf{A}_{\mathcal{C}} \begin{bmatrix} \mathbf{x}_{k,a} \\ \mathbf{p}_k \end{bmatrix} + \mathbf{b}_{\mathcal{C}},$$

$$\mathbf{A}_{\mathcal{C}} := \frac{1}{\mathcal{CP}_i} \begin{bmatrix} -1 & -1 & -1/\omega & -1/\omega & 1 & 1 \\ -1 & 1 & -1/\omega & 1/\omega & 1 & -1 \\ 1 & -1 & 1/\omega & -1/\omega & -1 & 1 \\ 1 & 1 & 1/\omega & 1/\omega & -1 & -1 \end{bmatrix},$$

$$\mathbf{b}_{\mathcal{C}} := \mathbf{1}_{4\times 1}. \tag{15}$$

The safe set of states in Eq. (14) represents the set of the *LIPM state* and *LIPM stance* pairs that could be stabilized without falling by taking i-step. In other words, if an *LIPM state* and *LIPM stance* pair is inside the safe set at the kth step, there is always a location for the next stance $\mathbf{a}_k$ (and the following stance $\mathbf{a}_{k+1}$ in the case of two-step capture region) that stabilizes the LIPM. The projection onto the x and $\dot{x}$ plane of capture regions is represented in Fig. 3(b).

B. Safety Guaranteeing Policy Design

For data-efficient and safe learning, we design our control input with three components:

$$\mathbf{a}_k = \mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}} + \mathbf{a}_k^{\mathrm{SF}}(\mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}}), \tag{16}$$

where $\mathbf{a}_k^{\mathrm{TVR}} = \Phi(\Psi(T_{LN}, 0\,;\, \mathbf{x}_k, \mathbf{p}_k))$ is computed by the TVR planner and $\mathbf{a}_k^{\boldsymbol{\theta}}$ is drawn from a parameterized Gaussian distribution, $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}_{\boldsymbol{\theta}})$, where $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\sigma}_{\boldsymbol{\theta}}$ denote a mean vector and covariance matrix parameterized by $\boldsymbol{\theta}$,[2] respectively. $\mathbf{a}_k^{\mathrm{SF}} : \mathcal{A} \mapsto \mathcal{A}$ is the safety projection that takes the sum of $\mathbf{a}_k^{\mathrm{TVR}}$ and $\mathbf{a}_k^{\boldsymbol{\theta}}$ and computes a compensation to make the action safe. Given arbitrary $\mathbf{a}_k^{\mathrm{TVR}}$ and $\mathbf{a}_k^{\boldsymbol{\theta}}$, the safety-guaranteeing controller $\mathbf{a}_k^{\mathrm{SF}}$ ensures the following *LIPM state* and *LIPM stance* pair $(\mathbf{x}_{k+1,a}, \mathbf{p}_{k+1})$ steered by the final control input $(\mathbf{a}_k)$ stays inside the safe set $\mathcal{C}$. In our problem, Eq. (9) is modified as

$$\sup_{\mathbf{a}_k^{\mathrm{SF}}} [h(\mathbf{x}_{k+1,a}, \mathbf{p}_{k+1}) + (\eta - 1)h(\mathbf{x}_{k,a}, \mathbf{p}_k)] \geq \mathbf{0}_{4\times 1}. \tag{17}$$

[2]In the implementation, we choose two fully connected hidden layers with the $\tanh$ activation function.

Algorithm 1: Policy Learning Process.

Data: M episodes, K data samples
Result: $\pi_{\boldsymbol{\theta}}$
Initialize $\pi_{\boldsymbol{\theta}}$, $\mathbf{s}_0 \sim \rho_0$, data array D;
for $m = 1 : M$ **do**
  for $k = 1 : K$ **do**
    $\mathbf{a}_k^{\boldsymbol{\theta}} \sim \pi_{\boldsymbol{\theta}}$, $\mathbf{a}_k^{\mathrm{TVR}} \leftarrow$ Eq. (5);
    $\mathbf{a}_k^{\mathrm{SF}} \leftarrow$ Eq. (19);
    $\mathbf{a}_k \leftarrow$ Eq. (16);
    $r_k \leftarrow$ Eq. (12);
    $\mathbf{s}_{k+1,a} \leftarrow$ WBC stabilizes the robot and brings it to the next *Apex Moment*;
    store $(\mathbf{s}_k, \mathbf{a}_k, \mathbf{s}_{k+1}, r_k)$ in D;
  end
  $\pi_\theta \leftarrow$ Optimize L_{PPO} with D w.r.t $\boldsymbol{\theta}$;
  Update GP model with D;
  clear D;
end

Substituting Eq. (11) and Eq. (15) into Eq. (17), and choosing the worst case for the uncertain dynamics in Eq. (8) yields the following inequality constraint:

$$\mathbf{A}_{\mathcal{C}} \left(f(\mathbf{x}_{k,a}, \mathbf{p}_k) + g(\mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}} + \mathbf{a}_k^{\mathrm{SF}}) + \boldsymbol{\mu}_d(\mathbf{x}_{k,a}, \mathbf{p}_k) \right.$$
$$\left. - |k_\delta \boldsymbol{\sigma}_d(\mathbf{x}_{k,a}, \mathbf{p}_k)| \right) + \mathbf{b}_{\mathcal{C}} \geq (1 - \eta)(\mathbf{A}_{\mathcal{C}} \begin{bmatrix} \mathbf{x}_{k,a} \\ \mathbf{p}_k \end{bmatrix} + \mathbf{b}_{\mathcal{C}}). \tag{18}$$

Considering the safety constraint in Eq. (18) and input boundaries, the optimization problem is summarized in the following Quadratic Programming (QP) and efficiently solved for the safety compensation as

$$\min_{\mathbf{a}_k^{\mathrm{SF}}, \epsilon} \quad \|\mathbf{a}_k^{\mathrm{SF}}\| + K_\epsilon \epsilon$$
$$\text{s.t.} \quad \begin{bmatrix} \mathbf{A}_{\mathrm{qp}}^{(11)} & \mathbf{A}_{\mathrm{qp}}^{(12)} \\ \mathbf{A}_{\mathrm{qp}}^{(21)} & \mathbf{A}_{\mathrm{qp}}^{(22)} \\ \mathbf{A}_{\mathrm{qp}}^{(31)} & \mathbf{A}_{\mathrm{qp}}^{(32)} \end{bmatrix} \begin{bmatrix} \mathbf{a}_k^{\mathrm{SF}} \\ \epsilon \end{bmatrix} \leq \begin{bmatrix} \mathbf{b}_{\mathrm{qp}}^{(1)} \\ \mathbf{b}_{\mathrm{qp}}^{(2)} \\ \mathbf{b}_{\mathrm{qp}}^{(3)} \end{bmatrix}, \tag{19}$$

where ϵ is a slack variable in the safety constraint, and K_ϵ is a large constant to penalize safety violation. Here,

$$\mathbf{A}_{\mathrm{qp}}^{(11)} = -\mathbf{A}_{\mathcal{C}} g, \quad \mathbf{A}_{\mathrm{qp}}^{(12)} = -\mathbf{1}_{4\times 1}, \quad \mathbf{A}_{\mathrm{qp}}^{(21)} = \mathbf{I}_{2\times 2},$$
$$\mathbf{A}_{\mathrm{qp}}^{(22)} = \mathbf{0}_{2\times 1}, \quad \mathbf{A}_{\mathrm{qp}}^{(31)} = -\mathbf{I}_{2\times 2}, \quad \mathbf{A}_{\mathrm{qp}}^{(32)} = \mathbf{0}_{2\times 1},$$

and

$$\mathbf{b}_{\mathrm{qp}}^{(1)} = \mathbf{A}_{\mathcal{C}} \left(f(\mathbf{x}_{k,a}, \mathbf{p}_k) + \boldsymbol{\mu}_d(\mathbf{x}_{k,a}, \mathbf{p}_k) + g(\mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}}) \right.$$
$$\left. - (1 - \eta)\mathbf{A}_{\mathcal{C}} \begin{bmatrix} \mathbf{x}_{k,a} \\ \mathbf{p}_k \end{bmatrix} - k_\delta |\mathbf{A}_{\mathcal{C}} \boldsymbol{\sigma}_d(\mathbf{x}_{k,a}, \mathbf{p}_k)| + \eta \mathbf{b}_{\mathcal{C}}, \right.$$
$$\mathbf{b}_{\mathrm{qp}}^{(2)} = -\left(\mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}}\right) + \mathbf{a}_{\max}$$
$$\mathbf{b}_{\mathrm{qp}}^{(3)} = \left(\mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}}\right) - \mathbf{a}_{\min}.$$

The first segment of the inequality represents a constraint for the safety, and the last two are for the input constraints. The design of the safety-guaranteeing policy is illustrated in Fig. 3(a).

TABLE I
SIMULATION PARAMETERS

	LIPM	$\mathcal{SM}$	$\mathbf{a}^{\mathrm{TVR}}$	$\mathbf{a}^{\boldsymbol{\theta}}$	$\mathbf{a}^{\mathrm{SF}}$	Reward					Behavior
	$[h, l_{\max}]$	$[T_{LN}, T_{LF}]$	$[T_{x'}, T_{y'}, \kappa_x, \kappa_y]$	Layer	$[K_\epsilon, \eta]$	r_a	w_b	w_t	w_s	w_c	$[\dot{x}^d, \omega_z^d]$
DRACO Walking	$[0.93, 0.7]$	$[0.16, 0.16]$	$[0.22, 0.22, -0.18, -0.18]$	$[64, 64]$	$[10^5, 0.8]$	5.0	3.0	3.0	1.0	1.0	$[0.3, 0]$
ATLAS Walking	$[0.82, 0.55]$	$[0.23, 0.23]$	$[0.15, 0.15, -0.16, -0.16]$	$[64, 64]$	$[10^5, 0.8]$	5.0	3.0	3.0	1.0	1.0	$[0.15, 0]$
ATLAS Turning	$[0.82, 0.55]$	$[0.23, 0.23]$	$[0.15, 0.15, -0.16, -0.16]$	$[64, 64]$	$[10^5, 0.8]$	5.0	5.0	5.0	3.0	1.0	$[0, 0.09]$

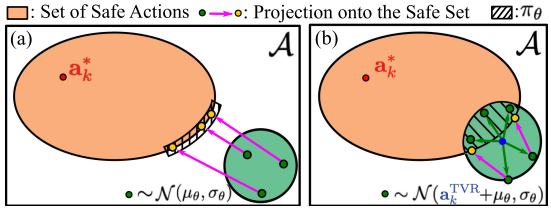

Fig. 4. The safety compensation process. $\mathbf{a}_k^*$ denotes an optimal control input and the orange area represents a set of safe actions that ensures that the state at the next time step stays inside the safe set $\mathcal{C}$. (a) and (b) represent two different instances of feedforward exploration.

Based on the MDP formulation and the policy design, the overall algorithm for efficient and safe learning for locomotion behaviors is summarized in Algorithm 1.

C. Further Details

It is worth taking a look at each of the components in the final action described by Eq. (16). $\mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}}$ provides a "feedforward exploration" in the state space, where the stochastic action explores the TVR planner and optimizes the long-term reward. $\mathbf{a}_k^{\mathrm{SF}}$ projects $\mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}}$ onto the safe set of policies and furnishes "safety compensation".

Particularly, $\mathbf{a}_k^{\mathrm{TVR}}$ in the feedforward exploration provides learning guidance and resolves two major issues in the safety projection: (1) inactive exploration and (2) the credit assignment problem. Consider, for example, two cases with different feedforward explorations, as illustrated in Fig. 4, whose final control policies are: (a) $\mathbf{a}_k = \mathbf{a}_k^{\boldsymbol{\theta}} + \mathbf{a}_k^{\mathrm{SF}}(\mathbf{a}_k^{\boldsymbol{\theta}})$ and (b) $\mathbf{a}_k = \mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}} + \mathbf{a}_k^{\mathrm{SF}}(\mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}})$.

In the case of (a) (and (b), respectively), the cyan area represents feedforward exploration expressed by a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}_{\boldsymbol{\theta}})$ (and $\mathcal{N}(\mathbf{a}_k^{\mathrm{TVR}} + \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\sigma}_{\boldsymbol{\theta}})$, respectively), and the green dots are its samples. The pink arrow represents the safety compensation $\mathbf{a}_k^{\mathrm{SF}}(\mathbf{a}_k^{\boldsymbol{\theta}})$ (and $\mathbf{a}_k^{\mathrm{SF}}(\mathbf{a}_k^{\mathrm{TVR}} + \mathbf{a}_k^{\boldsymbol{\theta}})$, respectively). The black striped area is a distribution of the final action $\mathbf{a}_k$, and the yellow dots are its sample.

As Fig. 4(a) shows, there is no intersection between the set of safe actions and the possible feedforward exploration and the feedforward explorations are all projected onto the safe action set. The projection does not preserve the volume in the action space, and it hinders active explorations in the learning. However, Fig. 4(b) leverages the TVR planner as learning guidance and retains the volume in action space to explore over. When it comes to computing a gradient of the long-term reward, the projected actions make it difficult to evaluate the resulting trajectories and assign the credits in the $\boldsymbol{\theta}$ space. In other words, as Fig. 4(a) shows, three compensated samples (yellow dots) do not roll out different trajectories, which prevents the gradient descent and results in a local optimum.

V. SIMULATION RESULTS

We execute a series of experiments with our 10-DoF DRACO biped [9] and the 23-DoF Boston Dynamic's ATLAS humanoid using the DART simulator [22] to evaluate the proposed MDP formulation and policy design. The parameters used in the simulations are summarized in the Table I. The goal of the experiments is three-fold: (1) How does the proposed method learn locomotion better than the baseline approaches (i.e., end-to-end policy search in [17], DeepLoco in [12], and the TVR planner) in terms of data-efficiency, safety, and the quality of the walking behavior? (2) How does each policy component in Eq. (16) contribute to the learning process? (3) Could the proposed approach be generalized to various types of walking (e.g., turning, walking over irregular terrain, and walking given random disturbances)?

A. Forward Walking

1) Experiment Setup: We include eight MDPs with different states and actions, and train policies for forward walking to demonstrate the effectiveness of our method. As baselines, two MDPs are based on an end-to-end model free learning: each policy learns joint torques $\boldsymbol{\tau}_k$ either from joint positions and velocities $\mathbf{q}_k, \dot{\mathbf{q}}_k$ or from the LIPM described in Eq. (10). We implement and adapt another baseline MDP from DeepLoco [12], where a walking policy is composed of a high-level footstep planner and a low-level feedback controller. Note that DeepLoco trains the networks to generate footsteps and joint commands in an end-to-end manner, whereas our method incorporates a simplified model to train footstep policy efficiently. Another difference is that we consider a model-based feedback controller (i.e., WBC) to compute joint commands. The other baseline MDP is set up based on the deterministic policy shown Eq. (5) that maps LIPM information to footstep locations. Finally, we formulate four variations of our proposed MDP by alternating the components of the actions shown in Eq. (16). Fig. 5 summarizes different states and actions for our experiments. We solve the MDPs using the policy search method described in [17] with the reward defined in Eq. (12) except for the DeepLoco baseline where we follow the reward function and the actor-critic method described in [12]. Experiments whose policy is a footstep location are followed by a low-level WBC. For example, a cubic spline trajectory is generated with a footstep decision made by the MDPs and converted to an operational space task. At the same time, a CoM position and torso orientation task are also specified with different priorities to maintain the robots upright.

Experiments	State	Action
End-to-end learning	Eq. (10)	(i) $\boldsymbol{\tau}_k$
	$\mathbf{q}_k, \dot{\mathbf{q}}_k$	(ii) $\boldsymbol{\tau}_k$
DeepLoco	$\mathbf{q}_k, \dot{\mathbf{q}}_k$	$\mathbf{a}_k$ (high-level)
	$\mathbf{q}_k, \dot{\mathbf{q}}_k, \mathbf{a}_k$	(iii) $\boldsymbol{\tau}_k$ (low-level)
Analytic method	$\mathbf{x}_{k,a}, \mathbf{p}_k$	(iv) Eq. (5)
Proposed MDP with different components	Eq. (10)	(v) Eq. (16) with $\mathcal{CP}_2$ (vi) Eq. (16) with $\mathcal{CP}_1$ (vii) $\mathbf{a}_k^{\text{TVR}} + \mathbf{a}_k^{\theta}$ $(viii)$ $\mathbf{a}_k^{\theta} + \mathbf{a}_k^{\text{SF}}$ with $\mathcal{CP}_1$

Fig. 5. Eight MDPs with different states and actions for the forward walking.

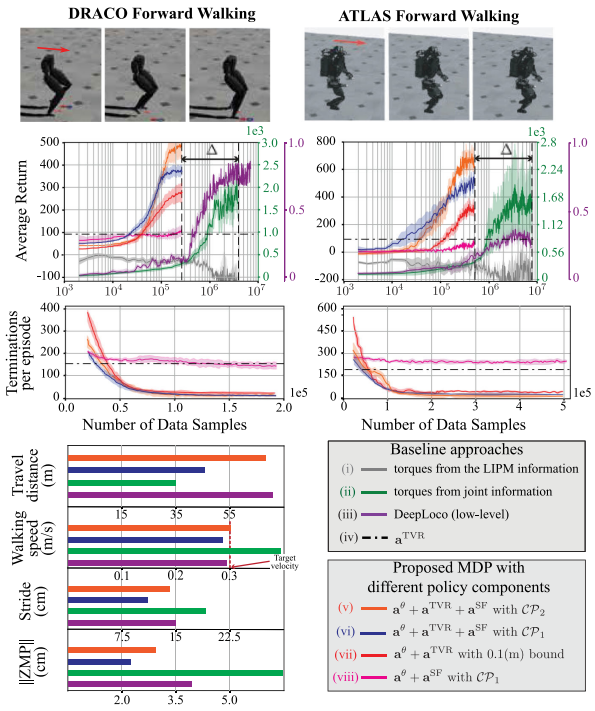

Fig. 6. Learning curves for the experiments are shown here demonstrating learning performance for forward walking. The average return, the number of terminations per episode are shown throughout the training. Each of the curves is plotted with its mean and standard deviation across five runs. Note that in the average return plot, the green and gray curves use the green vertical axis, and the purple curve uses the purple vertical axis on the right side. At the bottom, we run an episode with trained policies from different setups and show the travel distance, walking speed, walking stride, and the average of the two-norm of the ZMP in the local frame.

2) Analysis: Multiple policies are trained in each setup to regulate forward walking. The learning processes of the proposed MDP, as well as the baseline performance of the TVR and end-to-end learning, are illustrated in Fig. 6 with some useful metrics.

In the average return plot, the end-to-end learning with the LIPM information (gray curve) cannot achieve the motion of walking, whereas the other end-to-end learning with the joint information (green curve) shows a convergence of the walking behavior to unnatural motions. This shows that the LIPM information itself is not informative enough to calculate joint torques in an end-to-end manner. It is worth mentioning that the end-to-end learning with joint state information takes a more substantial dataset (denoted by Δ) to generate desired locomotion behavior than using the proposed MDP. DeepLoco (purple curve) shows a faster convergence rate than the end-to-end learning in the case of DRACO thanks to the hierarchical policy structure. However, DeepLoco requires more data than our approach since it has to train the low-level feedback controller instead of using the WBC. Furthermore, DeepLoco does not scale well to ATLAS in our implementation.

The proposed MDP using the conservative one-step capture region (blue curve) helps to accelerate the learning at the beginning phase, but the one using the relaxed two-step capture region (orange curve) eventually achieves a better walking policy in terms of the average return. Training with a heuristic bound (0.1 m) instead of using the safety projection (red curve) exhibits relatively good performance, whereas the one without the TVR planner (pink curve) rarely improves throughout the updates. The results reflect the issues addressed in Section IV-C. The number of terminations per episode decays as the uncertain parts of the dynamics are revealed throughout the training.

We evaluate the quality of walking resulting from different setups. In Fig. 6, the trained policy from the blue curve uses conservative safety criteria, which results in smaller strides and slower walking speed than for the other methods. The walking behavior resulting from the green curve policy takes longer strides with faster walking speeds. However, as we can infer from the ZMP graph, the policy from the green curve shows unnatural walking motions and yields a short travel distance per episode.

B. Generalization to Various Types of Locomotion

1) Experiment Setup: We consider three additional experiments in simulation to show that our proposed formulation can be generalized to various types of locomotion: turning, walking over irregular terrain, and walking given random disturbances. For the turning experiment, the low-level WBC controls the robot's torso, pelvis, and feet orientation. We consider irregular terrains including tilted ground at angles of between -10° and 10°. In addition, random disturbances are applied at intervals of 0.1 sec in the lateral and sagittal directions with a magnitudes between -600 N and 600 N. Note that we apply the disturbances both before and after the *Apex Moment*. For all experiments, the states, actions, and reward function are identical to the MDP formulation we described in Section III. We train the policies using the policy search method described in [17].

2) Analysis: The learning processes for each experiment, as well as the baseline performance of the TVR planner, are illustrated in Fig. 7(a). Our proposed approach succeeds in achieving locomotion behaviors based on average returns. The stance foot orientation captures the heading of the robot and the terrain information, and the neural network used in the learning process adapts to walking in new environments.

The experiment of walking with random disturbances shows an increment of the average return with high variance. It demonstrates robust walking under mild disturbances, but the average return is not as high as the return without disturbances shown in Fig. 6. For further analysis, we evaluate a trained policy in the presence of three different types of disturbances: (1) a disturbance with a magnitude of 600 N on both the sagittal and the lateral directions before the *Apex Moment* (i.e., in $\mathfrak{L}_{LF}$), (2) a disturbance with a magnitude of 300 N on both the sagittal and the lateral directions after the *Apex Moment* (i.e., in $\mathfrak{L}_{LN}$), and (3) a disturbance with a magnitude of 600 N on both the sagittal and the lateral directions after the *Apex Moment* (i.e., in $\mathfrak{L}_{LN}$).

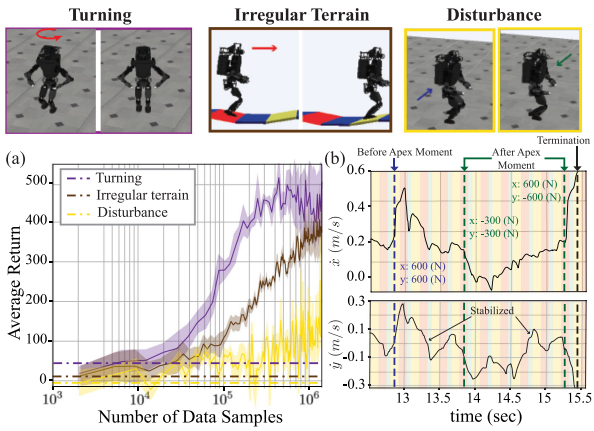

Fig. 7. Various types of locomotion behaviors: turning, walking over irregular terrains, and walking with random disturbances. (a) The average return is shown throughout the training. (b) The trained policy for the disturbance experiments is evaluated. The CoM velocities are plotted when three different disturbances occur. Note that the background colors represent the locomotion states in Fig. 2.

The velocity profiles of the CoM are shown in Fig. 7(b). The first type of disturbance is dealt with by the robot using a single footstep. In the figure, one can see the CoM velocity in the lateral direction ($\dot{y}$) being directed back to near zero in the next double support phase, even with the large disturbances. The second type of disturbance cannot be rejected by using a single step because that footstep is determined at the *Apex Moment* that precedes the disturbance. However, this can still be compensated for in future walking steps, unless the magnitude of the disturbance is significant enough to make the robot fall immediately. In the last case, the magnitude of the disturbance is 600 N and makes the robot fall right away. Future work includes incorporating a disturbance observer and continuous disturbance detection during the swing motion as described in [21]. In such a case, the policy described in Eq. (16) will re-calculate new footstep locations to reject all disturbance within the first footstep.

VI. Concluding Remarks

In this letter, we describe an MDP formulation for data-efficient and safe learning for locomotion. Our formulation combines analytic and data-driven approaches to make high-level footstep decisions based on the LIPM. The proposed policy includes a TVR planner, a neural network, and a safety controller. The TVR planner computes achievable sub-optimal guidance, the neural network modulates the guidance to maximize the long-term reward, and the safety controller facilitates safe exploration during the learning process. The safety controller learns the unknown part of dynamics in tandem with the policy updates and compensate for unsafe actions from the neural network based on the capturability metric and the use of control-barrier function. We thoroughly evaluate the effectiveness of the proposed method show how it could be generalized for various types of walking with two humanoids. Our contributions include: (1) a structured learning control method that mitigates the limited effect of using simple models and generates agile and robust locomotion, (2) a data-efficient and safe learning process to reinforce walking using a physics-based model, and (3) the scalability of the method to various types of humanoid robots and walking. In the near future, we plan to implement this framework into a real bipedal robot called DRACO. In the past, we have encountered many problems using the LIPM without a learning process causing complicated tuning procedures. We believe that the policy learning technique presented here will automatically determine the gap between the model and reality and will adjust the policy accordingly with minimal tuning.

References

[1] S. Kuindersma *et al.*, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, Mar 2016.
[2] S. Rezazadeh *et al.*, "Spring-mass walking with ATRIAS in 3D: Robust gait control spanning zero to 4.3 KPH on a heavily underactuated bipedal robot," in *Proc. ASME Dyn. Syst. Control Conf.*, Columbus: ASME, Oct. 2015, Art. no. V001T04A003.
[3] S. Caron, A. Kheddar, and O. Tempier, "Stair climbing stabilization of the HRP-4 humanoid robot using whole-body admittance control," in *Proc. IEEE Int. Conf. Robot. Automat.*, Montreal, QC, Canada, 2019, pp. 277–283, doi: 10.1109/ICRA.2019.8794348.
[4] S. Kajita *et al.*, "Biped walking pattern generation by using preview control of zero-moment point," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, Sep. 2003, pp. 1620–1626 vol.2.
[5] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1441–1460, Dec. 2018.
[6] A. Herzog *et al.*, "Structured contact force optimization for kino-dynamic motion generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 2703–2710.
[7] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Auton. Robots*, vol. 35, no. 2, pp. 161–176, Oct. 2013.
[8] D. Kim *et al.*, "Dynamic locomotion for passive-ankle biped robots and humanoids using whole-body locomotion control," 2019, *arXiv:1901.08100*.
[9] J. Ahn, D. Kim, S. Bang, N. Paine, and L. Sentis, "Control of a high performance bipedal robot using viscoelastic liquid cooled actuators," in *Proc. IEEE-RAS 19th Int. Conf. Humanoid Robots*, Oct. 2019, pp. 146–153.
[10] Y.-M. Chen and M. Posa, "Optimal reduced-order modeling of bipedal locomotion," 2019, *arXiv:1909.10111*.
[11] Heess *et al.*, "Emergence of locomotion behaviours in rich environments," 2017, *arXiv:1707.02286*.
[12] X. B. Peng, G. Berseth, K. Yin, and M. van de Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017. [Online]. Available: https://doi.org/10.1145/3072959.3073602
[13] S. Levine and V. Koltun, "Guided policy search," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, vol. 28, pp. III-1–III-9.
[14] G. A. Castillo *et al.*, "Hybrid zero dynamics inspired feedback control policy design for 3d bipedal locomotion using reinforcement learning," 2019, *arXiv:1910.01748*.
[15] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," in *Proc. Robot.: Sci. Syst.*, FreiburgimBreisgau, Germany, Jun. 2019.
[16] A. Iscen *et al.*, "Policies modulating trajectory generators," in *Proc. 2nd Conf. Robot Learn.* (Proceedings of Machine Learning Research), vol. 87, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds. PMLR, 29–31 Oct. 2018, pp. 916–926. [Online]. Available: http://proceedings.mlr.press/v87/iscen18a.html
[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[18] G. Brockman *et al.*, "Openai gym," 2016, *arXiv:1606.01540*.
[19] J. Ahn *et al.*, "Fast kinodynamic bipedal locomotion planning with moving obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 177–184.
[20] R. Cheng, G. Orosz, R. Murray, and J. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, vol. 33, pp. 3387–3395.
[21] T. Koolen, T. de Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *Int. J. Robot. Res.*, vol. 31, no. 9, pp. 1094–1113, 2012.
[22] J. Lee *et al.*, "Dart: Dynamic animation and robotics toolkit," *J. Open Source Softw.*, vol. 3, no. 22, 2018, Art. no. 500. [Online]. Available: https://doi.org/10.21105/joss.00500