2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)
Nancy, France. November 22-24, 2024

Learning Generic and Dynamic Locomotion of Humanoids Across Discrete Terrains

Shangqun Yu[1], Nisal Perera[1], Daniel Marew [1], and Donghyun Kim[1]

***Abstract*— This paper addresses the challenge of terrain-adaptive dynamic locomotion in humanoid robots, traditionally tackled by optimization-based methods or reinforcement learning (RL). Optimization-based methods, such as model-predictive control, excel in finding optimal reaction forces and achieving agile locomotion, but struggle with the nonlinear hybrid dynamics of legged systems and the real-time computation of step location, timing, and reaction forces. Conversely, RL-based methods show promise in navigating dynamic and rough terrains but are limited by their extensive data requirements. We introduce a novel locomotion architecture that integrates a neural network policy, trained through RL in simplified environments, with a state-of-the-art motion controller combining model-predictive control (MPC) and whole-body impulse control (WBIC). The policy efficiently learns high-level locomotion strategies, such as gait selection and step positioning, without the need for full dynamics simulations. This control architecture enables humanoid robots to dynamically navigate discrete terrains, making strategic locomotion decisions (e.g., walking, jumping, and leaping) based on ground height maps. Our results demonstrate that this integrated control architecture achieves dynamic locomotion with significantly fewer training samples than conventional RL-based methods and can be transferred to different humanoid platforms without additional training. The control architecture has been extensively tested in dynamic simulations, accomplishing terrain height-based dynamic locomotion for three different robots.**

I. Introduction

The control of legged locomotion boils down to three fundamental questions: when and where to step, and how to adjust the reaction force. Recently, two major approaches have been extensively studied to address these challenges.

The first approach is optimization-based methods, most notably model-predictive control, which focus on finding optimal reaction forces. This typically involves solving for contact forces based on a single rigid body [1], [2] or centroidal momentum model [3], then computing joint commands by solving inverse kinematics or dynamics. The optimization variables can range from reaction force only [1], [2], to both foot position and reaction force [4], or to full joint torque based on multi-body dynamics [5]. These approaches have achieved highly stable and dynamic locomotion [6], [7]. However, optimization-based algorithms struggle with hybrid dynamics, making it challenging to optimize contact timing, location, reaction forces (equivalently, CoM trajectory) simultaneously. Although several contact-implicit trajectory optimization efforts have aimed to tackle this issue [8]–[11], no real-time optimization-based controller has succeeded in selecting both step location and timing along with reaction forces given terrain information.

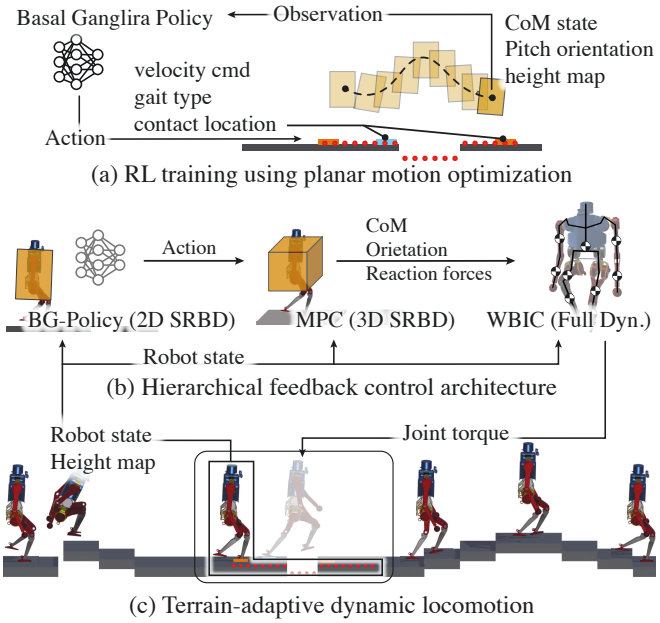


Fig. 1. **The proposed learning framework and control architecture.** (a) We first train a policy using a single rigid body dynamics (SRBD) model in the sagittal plane and trajectory optimization. (b) The trained policy (BG-policy) is integrated into the motion controller (MPC + WBIC) to compute the final joint commands for a humanoid robot. (c) Our control architecture commands a humanoid robot to walk, leap over gaps, jump onto platforms, and navigate stairs based on vision-based data.

Another prevailing approach is reinforcement learning (RL), where neural networks are trained through interaction with the environment. These policies take observations of the robot state and output joint commands (desired position [12]–[14] or torque [15], [16]), bypassing the need to explicitly consider step timing, location, and reaction force. Recent strides in end-to-end RL-based methods have been impressive – quadruped robots can now walk and run over rough terrains [12], [13], [17], and bipeds show significant robustness and terrain adaptation [14], [18], [19]. However, end-to-end RL strategies require massive data sets and long training hours [20], especially when incorporating perception data. Recent RL studies in perception-based locomotion required long training hours ($10 \sim 20$ h) [21]–[23]. Additionally the trained policy is bound to a specific system, making it difficult to adapt to variations like longer legs, body mass changes, or different joint arrangements.

Authors are with the [1] Manning College of Information and Computer Sciences at University of Massachusetts Amherst, Amherst, MA, 140 Governors Dr, Amherst, MA 01002, USA. Corresponding Author: robot.dhkim@gmail.com

In summary, both optimization-based and RL-based methods have limitations in handling terrain-adaptive dynamic locomotion. Although their challenges appear different, they stem from the difficulty of gradient descent-based optimization in large, nonlinear search spaces. RL-based methods perform better with contact decisions due to their exploration capabilities but require massive data. Attempts to merge these methods often rely on heuristic algorithms for contact sequence or location, [24]–[26] or they exclude perception data [27]–[30]. To our knowledge, no successful demonstrations exist for simultaneous selection of both contact location and sequence given terrain information, nor for perception-based dynamic locomotion involving jumping and leaping in a complete humanoid system.

We present a new locomotion architecture combining a neural network policy for high-level decisions and a low-level optimization-based motion controller. The policy, trained through RL in a simple environment with a single rigid body and contact points in planar space, outputs the gait type (i.e., contact timing), contact location, and forward speed. This is followed by trajectory optimization solving a convex problem. In essence, the policy views locomotion abstractly and makes high-level decisions similar to our brain's basal ganglia, hence the name Basal Ganglia-policy (BG-policy) [31]. In deployment, the trained BG-policy is combined with a low-level controller consisting of convex MPC and whole-body impulse controller (WBIC) [6], considered one of the state-of-the-art controllers that have demonstrated various agile behaviors in the MIT mini-cheetah [6], [10] and humanoid robots [32]. The architecture, comprising BG-policy, MPC, and WBIC, enables humanoid robots to dynamically navigate discrete terrains by observing terrain height, selecting appropriate locomotion strategies (e.g., walking, jumping, leaping), and coordinating full-body movements.

Thanks to our efficient learning framework, we trained a BG-policy with just a million samples, significantly fewer than other learning based methods [25], [33]. Additionally, the model-based motion controller allows the same BG-policy to be deployed across various humanoid robots without retraining and to handle tasks beyond locomotion (e.g., object carrying, head rotation). The main contributions of this paper are summarized as follows:

1) A novel control architecture with a BG-policy and optimal motion controller that selects step locations and gait types (e.g., walking, leaping, jumping) based on robot states and terrain height maps.
2) An efficient learning framework focused on sagittal motion, not requiring full dynamics simulations, with the policy handling only high-level decisions, making it much more sample-efficient than traditional end-to-end methods.
3) A generic locomotion controller offering three key benefits: 1) a robot-agnostic approach for zero-shot transfer to various robots, 2) flexibility to add extra tasks (e.g., object carrying) without additional training, and 3) support for omni-directional walking with an additional yaw-rate command.

II. Related work

A. Contact Implicit Trajectory Optimization (CI-TO)

Contact, because of its binary nature, introduces discrete jumps in the gradient when incorporated into optimization variables. Common techniques like mixed-integer programming (MIP) [34] is suffered from exponentially growing computational time. On the other hand, contact implicit trajectory optimization (CI-TO), as initially presented in [35] and later in [36], use complementarity constraints to mitigate this issue. However, real-time control using CI-TO faces the risk of converging to local minima. Soft contact constraints have been suggested as a solution [37]. More recently, advances in differential dynamic programming (DDP)-based approaches have demonstrated experimental success with quadruped robots but require starting near the solution due to DDP's local search nature [9], [11]. Alternatively, solving LCP with a pre-trained warm starting point generator has reduced computation time but performed poorly beyond the trained data set [10]. [38] circumvented the gradient discontinuity issue by representing the contact sequence with parameterized polynomial functions. However, this method necessitates specifying the number of contact points and has not been tested in full dynamics simulations.

Currently, no established CI-TO solution exists beyond relaxing the contact constraint, and CI-TO algorithms haven't been integrated with real-time perception-based locomotion. Therefore, we confined the optimization problem to convex or quasi-convex forms, delegating nonlinear decision variables to the BG-policy outputs.

B. Training efficiency of Reinforcement Learning

Training perception-based locomotion for discrete terrains presents significant challenges, even for quadruped systems, due to the enlarged observation space and difficulties in controlling accurate step locations. For example, [22] required 18 hours to train depth image-based parkour locomotion on an Nvidia 3090, and [39] used a teacher-student training scheme that consumed 12 million samples for student training alone. This challenge is greater for humanoid robots due to their larger action spaces and inherent instability. For instance, [40] spent 30 hours training blind walking for humanoid robots in the Isaac Gym [41] using an Nvidia 3090Ti.

In contrast, our BG-policy efficiently learns dynamic locomotion for humanoid systems with just 1 million samples in under an hour on a desktop with an 8-core AMD Ryzen 7 CPU and an RTX 3080 GPU. Despite the optimization running on the CPU across only 12 parallel environments using Casadi [42] and the Ipopt solver, our approach demonstrates superior efficiency compared to existing methods.

III. Basal Ganglia Policy Training

For efficient training, we distilled the environment to essential elements for learning high-level decisions. In our training setup, a humanoid robot is substituted by a planar

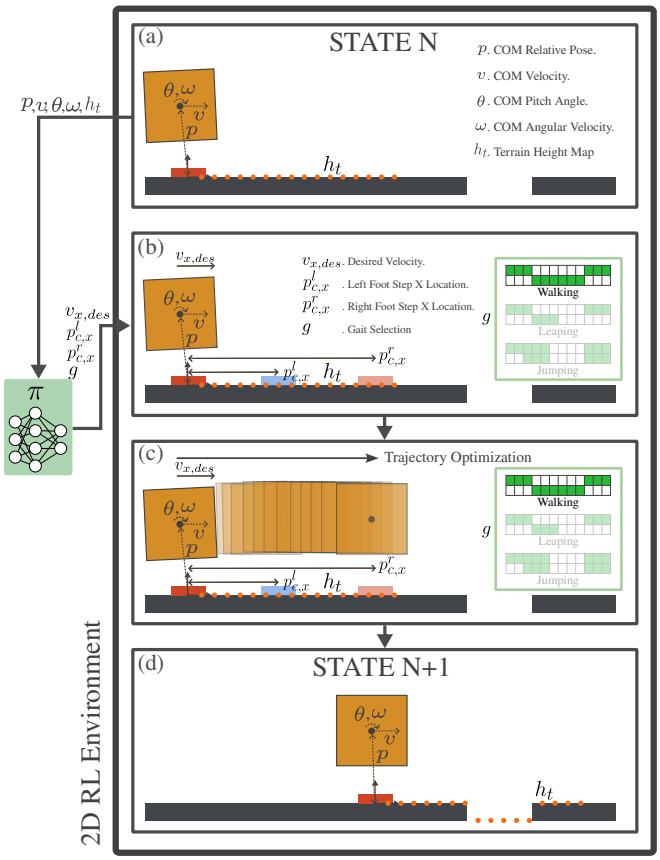


Fig. 2. **State Transition in the 2D Environment.** We designed a 2D environment which makes the policy focus on only the information that matters. The simplification leads to exceptional efficient training of the policy. This environment enables the policy to be effectively trained using no more than 1 million samples, a quantity several magnitude smaller than what is typically required in end-to-end methods with vision based data.

single rigid-body and two line contacts, representing the left and right feet. A line contact is implemented with two paired point contacts for simplicity and to allow for future extensions to various gait types, such as heel-toe transitions. The observation for the environment is

$$\mathbf{obs} = \begin{bmatrix} \mathbf{p}^\top & \theta & \mathbf{v}^\top & \omega & \mathbf{h_t}^\top \end{bmatrix}^\top, \tag{1}$$

where $\mathbf{p} \in \mathbb{R}^2$ and $\mathbf{v} \in \mathbb{R}^2$ are the center of mass position and velocity, respectively, with respect to the stance foot frame. $\theta \in \mathbb{R}$ and $\omega \in \mathbb{R}$ are the pitch angle and angular velocity, respectively, and $\mathbf{h_t} \in \mathbb{R}^{20}$ is the height map of the terrain (Fig. 2(a)). Our BG-policy makes an action when the right foot is under contact, where the origin of the local frame is attached to. The policy outputs the desired forward velocity, the contact locations of the next two steps (left foot and right foot) in the local frame, and the gait type (Fig. 2(b)). Subsequently, a trajectory optimisation seeks the optimal trajectory for the upcoming two steps (Fig. 2(c)) considering the contact constraints set by given action along with the current state and other constraints (e.g., reaction force and kinematics limit). The last state of the optimal trajectory, with some random Gaussian noise added, returns to the policy as the next state of the environment (Fig. 2(d)).

The state of the planar single rigid body model is given by

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}^\top & \theta & \mathbf{v}^\top & \omega \end{bmatrix}^\top, \tag{2}$$

where $\mathbf{p} \in \mathbb{R}^2$ is the position of CoM, θ is the pitch angle of the rigid body, $\mathbf{v} \in \mathbb{R}^2$ is the CoM velocity, and ω is the angular velocity. The derivative of the state $\dot{x}$ is given by

$$\dot{\mathbf{x}} = \frac{d}{dt}\begin{bmatrix} \mathbf{p} \\ \theta \\ \mathbf{v} \\ w \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ w \\ \frac{1}{m}\sum_{i=0}^{n_c} \mathbf{f}_i - \mathbf{g} \\ \frac{1}{I}\sum_{i=0}^{n_c} (\mathbf{p_c}_i - \mathbf{p}) \times \mathbf{f}_i \end{bmatrix}, \tag{3}$$

where $\mathbf{f}_i \in \mathbb{R}^2$ is the force of the contact points. Both feet have two contact points, thus $n_c = 4$ is the total number of contact points. $\mathbf{p_c}_i \in \mathbb{R}^2$ is the position of the i-th contact point. m and $I \in \mathbb{R}$ are the mass and approximated inertia of a robot. To determine the value of I, we initially compute the inertial tensor of the robot in its nominal pose (standing position) and then project it onto the sagittal plane. The optimization is formulated by

$$\min_{\mathbf{x}_k, \mathbf{f}_k} \sum_{k=1}^{N} (\mathbf{x}_k - \mathbf{x}_k^{des})^\top \boldsymbol{Q_x} (\mathbf{x}_k - \mathbf{x}_k^{des}) + \mathbf{f}_k^\top \boldsymbol{Q_f} \mathbf{f}_k$$

$$\begin{aligned}
\text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{x}_k + \dot{\mathbf{x}}_k \triangle t, && \text{(dynamics)} \\
& -\mu f_{i,y} \le f_{i,x} \le \mu f_{i,y}, && \text{(friction)} \\
& 0 \le f_{i,y} \le f_{max}, && \text{(reaction force)} \\
& l_{xmin} \le |p_{i,x} - p_{c_i,x}| \le l_{max,x}, && \text{(kinematics)} \\
& l_{ymin} \le |p_{i,y} - p_{c_i,y}| \le l_{max,x}, && \text{(kinematics)} \\
& f_{i,y}(1 - c_i) = 0, && \text{(contact)}
\end{aligned}$$

where $\boldsymbol{Q_x} \in \mathbb{R}^{6\times6}$ and $\boldsymbol{Q_f} \in \mathbb{R}^{8\times8}$ are weight matrices. $\mathbf{f}_k \in \mathbb{R}^8$ is the reaction force vector for the four contact points. $\mathbf{x}_k^{des} \in \mathbb{R}^6$ is the reference trajectory, which is generated by integrating the desire forward velocity output by the BG-policy. The reference trajectory's height and angle are set by nominal robot height from the ground, zero pitch angle, and zero angular velocity. $p_{i,x}$, $p_{i,y}$ represent the horizontal and vertical position of the CoM, respectively, while $p_{c_i,x}$, $p_{c_i,y}$ denotes the position of the contact point. The selection of $p_{c_i,x}$ is determined by the policy, and $p_{f_i,y}$ is determined by terrain height corresponding to $p_{c_i,x}$.

The kinematics constraint are derived based on the leg length of the robot to ensure the foot step position is realizable in the actual system. $c_i \in \{0, 1\}$ indicates whether the current contact point is in contact or not, which made based on the gait selected by the policy. The reward is given by

$$r = w_{vel} r_{vel} + w_{opt} r_{opt} + w_{gait} r_{gait}, \tag{4}$$

where w_{vel}, w_{opt} and w_{gait} are the weights for each term. The first term is a velocity reward given by

$$r_{vel} = a e^{-b(v_x - v_{nominal})^2}, \tag{5}$$

Algorithm 1 RL Environemnt Step Function

Input: $v_{x,des}$, $p^l_{c,x}$, $p^r_{c,x}$, g
Output: out

$\mathbf{p_c}$, $d_{violate}$ = getContactLoc($p^l_{c,x}$, $p^r_{c,x}$)
$\mathbf{c}$ = getContactSeqFromGait(g)
$\mathbf{x}^{des}_{1..n}$ = getRef($v_{x,des}$)
$\mathbf{x}_{1..n}$, l_{cost}, $d_{terminate}$ = optimize($\mathbf{x}^{des}_{1..n}$, $\mathbf{p_{c,1..n}}$, $\mathbf{c}$)
$done = d_{violate} || d_{terminate}$
r = calcualteReward(l_{cost}, g, x_n)
$\mathbf{x}_{obs}$ = $\mathbf{x}_n$ + $N(0, \sigma)$
$\mathbf{h_t}$ = getHeightMap(x_{obs})
return x_{obs}, $\mathbf{h_t}$, r, $done$

where $a > 0$ and $b > 0$ are hyper parameters, and $v_{nominal}$ is the user-specified nominal velocity. The second term is the optimization reward $r_{opt} = c/l_{cost}$, where l_{cost} is the cost from the trajectory optimization, and $c > 0$ is another hyper parameters. The lower the cost optimization found, the higher the r_{opt} reward. The last term is the gait selection reward, which encourages RL to choose the appropriate gait based on terrain information. Building upon the described reward function, RL aims to sustain the nominal speed, minimize the resultant cost of trajectory optimization, and select an appropriate gait for diverse situations.

Algorithm 1 describes the environment step function design. The BG-policy chooses the desired forward velocity ($v_{x,des}$), horizontal step locations for the next two steps ($p^l_{c,x}$, $p^r_{c,x}$), and gait type (g). The environment finds the vertical step locations based on the terrain and check step feasibility. If a step location is infeasible (e.g., stepping into a pit), $d_{violate}$ is set to true. Contact sequence $\mathbf{c}$ and reference trajectory $\mathbf{x}^{des}_{1..n}$ are based on g and $v_{x,des}$ from the policy. The optimization then searches for the optimal trajectory. The environment is set to "done" if the step location is invalid or optimization fails. The reward is computed using Eq. (4), and Gaussian noise is added to the last state of the trajectory before it is sent as the next observation.

IV. Optimization-based Motion Controller

Once the training of BG-policy is complete, we integrate the policy into the motion controller, which consists of MPC and WBIC [6] (Fig. 1 (b)). In real-time control, we first compile the observations for the BG-policy, including the humanoid robot's CoM state, body pitch angle/velocity, and the terrain height map. Based on this information, the policy outputs the desired forward velocity, the contact locations of the left and right feet on the x-axis, and the gait type, and then sends them to the MPC.

A. Model Predictive Control

Three gait types (i.e., walking, leaping, and jumping) are pre-specified based on appropriate swing/stance times. Each gait will lead to a different contact sequence for the next 2 steps.(Fig. 3) Based on the high level decisions from

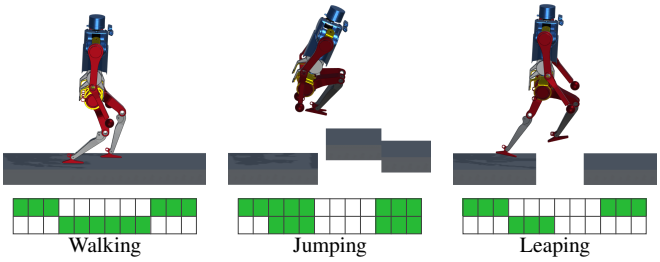


Fig. 3. **Three different gaits** All gaits have the same length with different contact sequence.

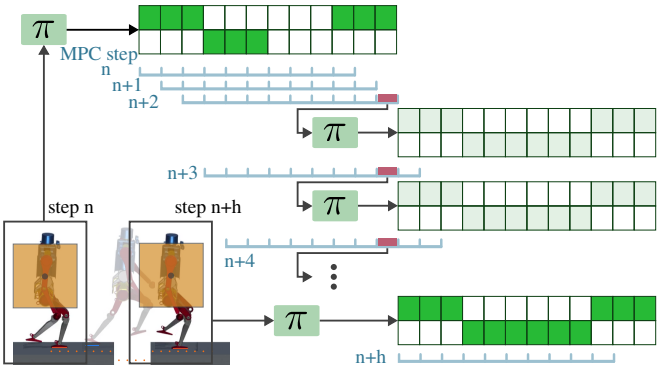


Fig. 4. **Illustration of how to maintain constant prediction horizon for MPC.** Between two actions, the BG-policy uses prediction from the MPC to compute the output, which ensure the MPC to have sufficiently long contact sequence to keep its constant prediction horizon.

the BG-policy, and state of a single rigid body is synchronized with the robot's CoM state and body orientation and angular velocity, then MPC calculates the reaction force, CoM/orientation trajectory, and reaction force profiles. The state of the MPC is given by

$$\mathbf{x} = [\mathbf{p}^\top \quad \mathbf{R_{vec}} \quad \mathbf{v}^\top \quad \boldsymbol{\omega}^\top]^\top, \tag{6}$$

where $\mathbf{p} \in \mathbb{R}^3$ and $\mathbf{v} \in \mathbb{R}^3$ are the position and the velocity of the robot's center of mass, $\mathbf{R_{vec}} \in \mathbb{R}^9$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ are the vectorized orientation matrix and angular velocity of the body frame. The cost function and constraint are formulated in similar fashion as the 2D trajectory optimization.

Each action from the policy include contact sequence for 0.6 seconds, while the MPC's prediction horizon is set at 0.5 seconds. The MPC operates in a sliding window fashion, and to maintain a constant 0.5-second window size, the policy needs to output a new sequence when MPC's window hit the end of contact sequence. If the series of contact sequences from the policy do not cover the MPC's prediction horizon (h), the policy will generate a new action based on MPC's forecast (refer to Fig. 4). For instance, at step n, the policy has just issued an action that satisfies the MPC's prediction horizon. However, by the time it reaches step $n+3$, the policy needs to produce another sequence based on the MPC's latest state prediction, so that the MPC can maintain its horizon length. From step $n+3$ to $n+h$, the policy outputs a new action each time the MPC produces a new prediction, ensuring the MPC's predictions align closely with the actual

state. At step $n+h$, the policy can plan the contact sequence based on the actual robot state.

B. Lateral Directional Step Location Selection

While the foot step position on the sagittal plane (x axis) is given by the policy, the position on the lateral direction (y axis) is calculated through the following formula:

$$p_{f,y} = p_y + p_{f,y_0} + k_d v_y, \tag{7}$$

where p_y is current CoM position in y, p_{c,y_0} is the nominal relative position offset on y axis. v_y is the current lateral directional CoM velocity. We select k_d based on velocity reversal planner from [43],

$$k_d = \sqrt{\frac{h_{\mathrm{CoM}}}{g}} \coth\left(\frac{t_{\mathrm{swing}}}{2}\sqrt{\frac{g}{h_{\mathrm{CoM}}}}\right), \tag{8}$$

which guarantee the asymptotic stability of linear inverted pendulum motion. The formula needs swing time (t_{swing}) and CoM height (h_{CoM}), and we use nominal gait parameter (swing time: 0.3 s, CoM height: 0.6 m). The selected number works across different gait type (walking, leaping, jumping) thanks to the planner's strong convergence. Once the x and y coordinates for the stepping position are determined, the corresponding landing height is obtained from the height map. Following this, the desired position, velocity, and acceleration of the swing foot are calculated by using a Bezier curve.

C. Whole Body Impulse Control (WBIC)

For the final step, WBIC [6] is used to calculates the joint torque. WBIC utilizes a null space projection technique to build a task hierarchy which allows lower priority tasks to be executed without interfering the higher priority tasks.

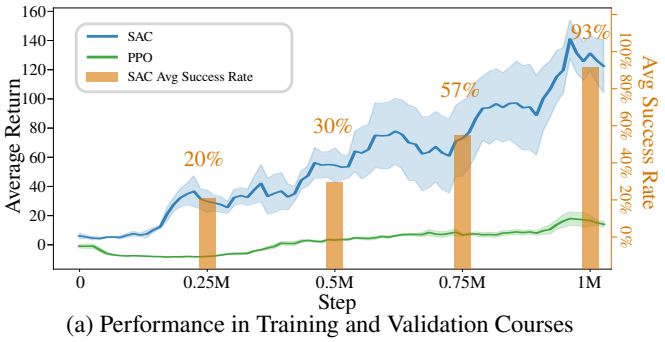
Our task hierarchy from high to low is as follow: contact constraint $>$ body orientation task $>$ CoM position task $>$ swing foot position task $>$ joint position task. The lower priority tasks are always executed in the null-space of the higher priority tasks.

V. Results

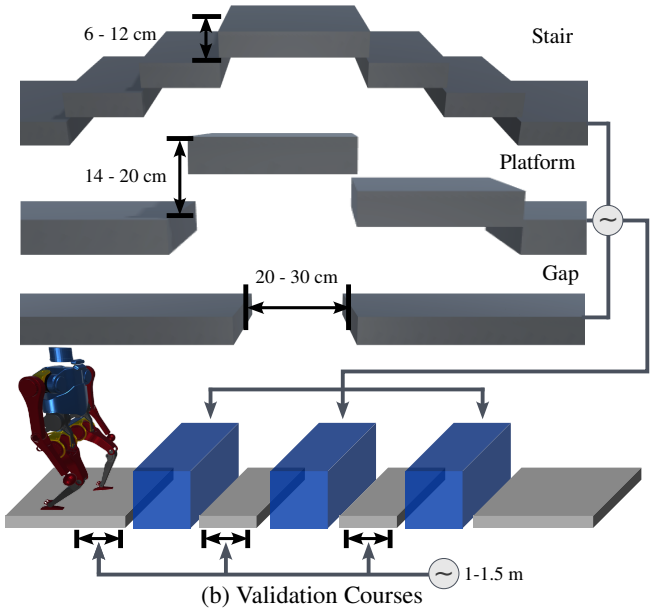
A. Experiment and Evaluation

We use Tello robot [44] for the validation of our learning framework and control architecture. Firstly, we train an RL agent in the 2D environment, where the planar single rigid-body's weight, inertia and box constraint for the kinematics are extracted from the Tello robot. The algorithm we use for the BG-policy training is Soft Actor-Critic (SAC), both the actor and the critic have two hidden layers of size 256. The input layer has a size of 26, 6 for state in the sagittal plane and 20 for the height map. The resolution of the height map is 0.05 m, enabling to perceive terrain features up to 1 m ahead. The output layer has 4 dimensions: desired velocity, left/right foot step, and gait selection as described in Fig. 2 (b).

In addition to the flat ground, the terrain features a variety of challenges, including gaps, high platforms, and stairs.



(a) Performance in Training and Validation Courses



(b) Validation Courses

Fig. 5. **Training performance and Validation Courses.** (a) Both SAC and PPO are trained for 1 million steps in 5 different seeds. The average return plot shows outstanding performance of SAC algorithm. The average success rate of the policy trained by SAC is also shown as orange bar. Once the iteration reaches to 1 million steps, the policy gets converged and shows robust locomotion performance. (b) To evaluate the actual performance of the trained policy, we made 30 validation courses with randomly generated obstacles in the full dynamic simulator.

At the start, the environment is dynamically configured by randomly selecting 10 obstacles across these categories. The distance between adjacent obstacles is also randomly determined, ranging from 1 to 1.5 m. Remarkably, the policy achieves convergence using just 1 million samples in less than an hour, running on a desktop outfitted with an 8-core AMD Ryzen 7 CPU and an RTX 3080 GPU.

The trained policy is deployed in full humanoid robot control by combining with optimization-based motion controller. In the dynamic simulation tests, we randomly generated 30 validation courses, each featuring three randomly selected obstacles with varying height/width as described in Fig. 5(b). The 93% success rate of SAC policy trained with 1M samples (refer Fig.5(a)) shows that the policy can manage the dynamic navigation over discrete terrains after an hour training.

TABLE I

NUMBER OF STEP REQUIRED TO ACHIEVE 90% SUCCESS RATE

BG humanoid	E2E biped (Cassie)	E2E humanoid
1M	40M	400M

B. Benchmark

To evaluate the training efficiency depending on RL algorithms, we also tried Proximal Policy Optimization (PPO), finding that SAC significantly outperformed PPO, as shown in Fig. 5(a). The result is also consistent with previous studies indicating that SAC, an off-policy algorithm, is more sample-efficient than on-policy algorithms like PPO, particularly in our framework that run trajectory optimization in each interaction with the environment.

Additionally, we have also compared our algorithm with the end-to-end method by recreating the same environment in the widely adopted open-source legged gym environment [45]. To verify the correctness of the implementation, the policy is firstly tested on an existing biped system (Cassie) included in [45]. Subsequently, the same end-to-end training setup is tested on the MIT Humanoid robot to evaluate the difference between a biped and a humanoid. The number of interaction steps required to achieve a 90 percent success rate in the validation courses was recorded. (Table I). Our policy proved to be 1-2 orders of magnitude more sample-efficient in terms of the number of interaction steps needed for training. While the end-to-end method with Cassie achieved a 90 percent success rate after approximately 40 million steps, the policy initially only learned to jump forward across the discrete terrain. Learning how to select the appropriate gait type for different terrains required 150 million steps. However, with the complete humanoid system, due to its high-dimensional action space, it converged to an unrealistic locomotion strategy that involved excessive torso twisting along with irregular gait sequence, and further training did not resolve this issue.

While our algorithm demonstrates greater efficiency in terms of the number of interaction steps required with the environment, it is important to acknowledge that current GPU-based physics simulators can collect millions of samples within minutes, rendering the total time cost for the end-to-end methods comparable to ours. The primary limitation of our approach lies in the use of a CPU-based optimization solver, which restricts our ability to generate thousands of environments in parallel. However, this training time could be significantly reduced with the availability of GPU-based optimization solvers, offering a clear pathway for further improvements in efficiency.

C. Algorithm's Robustness

Notably, the trained policy shows simultaneous adjustment of foot step location, gait and forward velocity to adapt to different terrain configuration. For instance, in Fig. 6, we present two distinct scenarios faced by the policy. In the scenario described in the bottom of Fig. 6, the terrain can be

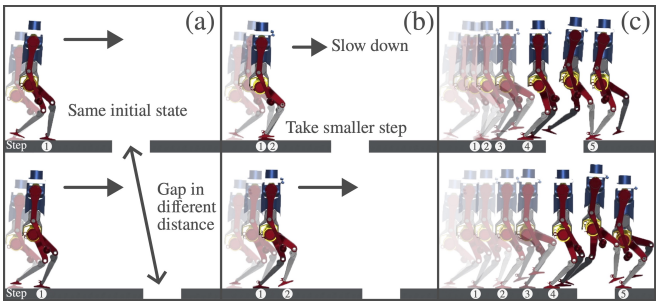


Fig. 6. **Demonstrating the BG-Policy's Adaptability.** (a) The robot was initialized at the exact same state except a different distance to the gap. (b) In the top row, the policy found it need to adjust its foot step to leap over the gap, so it choose to slow down and take smaller step, while at the bottom row, the policy choose to move forward with regular step size. (c) At the end, the robot is able to leap over the gap in both situation by dynamically coordinating its forward velocity and foot step location.

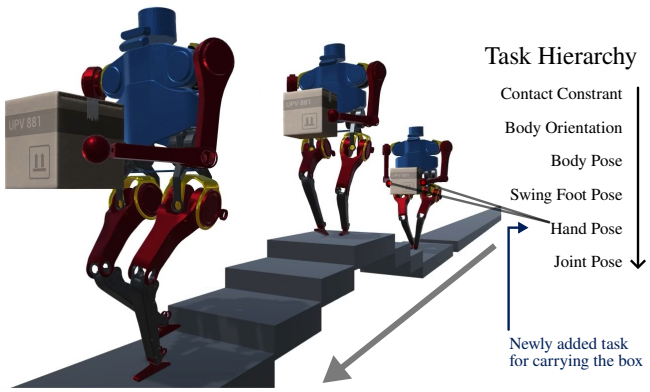


Fig. 7. **Tello carrying a box while traversing the irregular terrain.** By utilizing null-space projection based task prioritization, we can easily add additional task such as commanding the arm to carry a box without interfering the locomotion task, which will be very useful for object manipulation.

navigated by selecting footsteps that maintain a consistent distance from each other. However, in the scenario (shown at the top), where we deliberately reduced the gap distance from the initial position, the policy chooses to slow down, take several short steps, and then make a leap over the gap. This ability to adjust its locomotion strategy in real-time underscores the policy's effectiveness in handling diverse and challenging terrains.

Moreover, by using the WBIC, its null-space projection based task prioritization allows us to add additional task such as carrying a box (Fig. 7) without interfering the existing locomotion task nor it requires additional training typically necessary in end-to-end methods.

Another important benefit of our framework is its flexibility, unlike the end-to-end scheme, where the trained policy is specifically tailored to the robot it was trained on. Our policy can be directly applied to other robots without requiring any additional training. We demonstrate this by applying the same policy on MIT humanoid and Digit robots, simply by switching the robot model in the low-level motion controller (Fig. 8). Despite the significant difference in kinematics and dynamics between these robots and the Tello robot, for

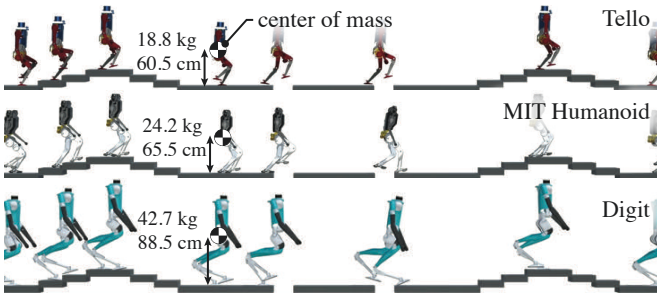


Fig. 8. **Locomotion of different robots using the same policy.** Since the policy is trained using an abstracted model, it can be deployed to various robots by simply switching the robot model in the motion controller. The results show that we can enable MIT Humanoid and Digit walk and jump over discrete terrain without additional training.

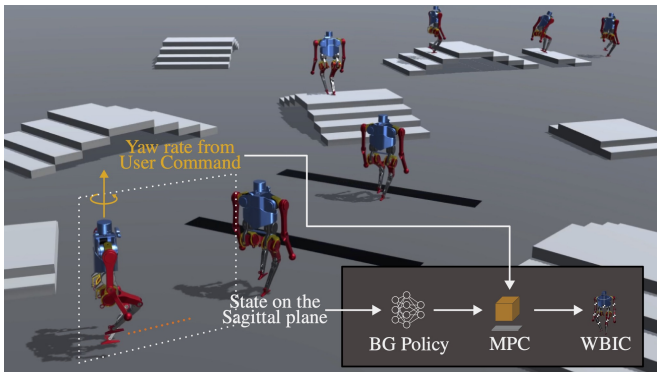


Fig. 9. **Demonstration of omni-directional walking.** By incorporating the user's yaw rate command, MPC+WBIC controller can manage the walking direction behavior. This rotation does not make a difference in BG-policy's perspective because it only focuses on the robot's local sagittal plane.

which the policy was originally trained, the same policy successfully guides different robots to navigate irregular terrain. It is noteworthy that our locomotion framework is robust to the difference between the target system being controlled and the robot on which the policy was trained. For example, although the Digit robot is significantly taller than the Tello robot, for which the policy was trained, the policy still functions effectively with a simple height adjustment of -250 mm applied to the observations before they are passed into the policy.

Last but not least, our algorithm also enables omni-directional walking on discrete terrain by seamlessly integrating an additional yaw rate command (Fig. 9). While taking high-level action from the BG policy, the MPC incorporates a desired yaw rate from the user command to change the walking direction. The BG policy processes state and 1D terrain information from the robot's sagittal plane – the robot's local xy plane. Then the MPC takes in the high-level actions along with the additional yaw rate command to calculate optimal trajectory and reaction force, and the WBIC calculates the joint torque commands. Because both MPC and WBIC take the robot's state in the global frame and have the complete 3D information, the algorithm does not need additional modifications to extend forward walking to omni-directional walking.

VI. Concluding Remarks

In this work, we introduced a novel training framework and control architecture that synergizes RL-trained policy with optimal control, tailored for dynamic humanoid locomotion. Our methodology involves a streamlined 2D training environment that encapsulates critical locomotion determinants – forward velocity, contact location, and gait selection – leveraging terrain and state abstractions. This strategic simplification has led to a drastic reduction in the requisite training samples, surpassing traditional end-to-end approaches by orders of magnitude. Employing this policy in tandem with low-level motion controller, our system adeptly navigates complex terrains, demonstrating a versatile locomotion repertoire that includes walking, leaping, and jumping over discrete terrains. Our control architecture facilitates the incorporation of auxiliary tasks, such as object manipulation, without compromising locomotion efficacy. Furthermore, the demonstrated zero-shot transferability underscores the policy's applicability across diverse robotic platforms. Another expected benefit is effortless sim-to-real transfer because the low-level motion controller have been successfully implemented in robot hardware.

We also identified a limitation of the framework: when the robot's motion becomes highly dynamic, the deviation of the actual motion from the predicted trajectory becomes noticeable. Consequently, the BG-policy does not effectively accomplish what it has learned during training. Future work could explore fine-tuning the BG-policy with full dynamics simulations or integrating an additional neural network policy to bridge this gap, potentially enhancing the fidelity of motion execution Further algorithmic enhancements could stem from leveraging GPU-accelerated optimization environments, promising substantial training speed improvements through parallelization. While our current RL setup, optimized for CPU execution, achieves significant efficiency, transitioning to GPU-based computation could further expedite the training process.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 2220924.

References

[1] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[2] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2436–2442.

[3] S.-H. Lee and A. Goswami, "Reaction mass pendulum (rmp): An explicit model for centroidal angular momentum of humanoid robots," in *Proceedings 2007 IEEE international conference on robotics and automation*. IEEE, 2007, pp. 4667–4672.

[4] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4102–4109.

[5] E. Dantec, R. Budhiraja, A. Roig, T. Lembono, G. Saurel, O. Stasse, P. Fernbach, S. Tonneau, S. Vijayakumar, S. Calinon *et al.*, "Whole body model predictive control with a memory of motion: Experiments on a torque-controlled talos," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8202–8208.

[6] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," 2019.

[7] G. García, R. Griffin, and J. Pratt, "Mpc-based locomotion control of bipedal robots with line-feet contact using centroidal dynamics," in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2021, pp. 276–282.

[8] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and feature-complete differentiable physics for articulated rigid bodies with contact," *arXiv preprint arXiv:2103.16021*, 2021.

[9] S. Le Cleac'h, T. A. Howell, S. Yang, C.-Y. Lee, J. Zhang, A. Bishop, M. Schwager, and Z. Manchester, "Fast contact-implicit model predictive control," *IEEE Transactions on Robotics*, 2024.

[10] S. H. Jeon, S. Kim, and D. Kim, "Online optimal landing control of the mit mini cheetah," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 178–184.

[11] G. Kim, D. Kang, J.-H. Kim, S. Hong, and H.-W. Park, "Contact-implicit mpc: Controlling diverse quadruped motions without pre-planned contact modes or trajectories," *arXiv preprint arXiv:2312.08961*, 2023.

[12] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[13] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," 2020.

[14] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," 2021.

[15] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2023, pp. 1–8.

[16] D. Kim, G. Berseth, M. Schwartz, and J. Park, "Torque-based deep reinforcement learning for task-and-robot agnostic learning on bipedal robots using sim-to-real transfer," *arXiv preprint arXiv:2304.09434*, 2023.

[17] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," 2018.

[18] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," 2021.

[19] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," 2023.

[20] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *5th Annual Conference on Robot Learning*, 2021. [Online]. Available: https://openreview.net/forum?id=wK2fDDJ5VcF

[21] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," *arXiv preprint arXiv:2309.14341*, 2023.

[22] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *Conference on Robot Learning (CoRL)*, 2023.

[23] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," 2023.

[24] G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. bae Kim, and P. Agrawal, "Learning to jump from pixels," in *5th Annual Conference on Robot Learning*, 2021. [Online]. Available: https://openreview.net/forum?id=R4E8wTUtxdl

[25] W. Yu, D. Jain, A. Escontrela, A. Iscen, P. Xu, E. Coumans, S. Ha, J. Tan, and T. Zhang, "Visual-locomotion: Learning to walk on complex terrains with vision," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 1291–1302. [Online]. Available: https://proceedings.mlr.press/v164/yu22a.html

[26] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2908–2927, 2022.

[27] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg, "Learning a contact-adaptive controller for robust, efficient legged locomotion," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 155. PMLR, 16–18 Nov 2021, pp. 883–894. [Online]. Available: https://proceedings.mlr.press/v155/da21a.html

[28] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," 2020.

[29] Z. Xie, X. Da, B. Babich, A. Garg, and M. v. de Panne, "Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2022, pp. 523–539.

[30] Y. Yang, G. Shi, X. Meng, W. Yu, T. Zhang, J. Tan, and B. Boots, "CAJun: Continuous adaptive jumping using a learned centroidal controller," in *7th Annual Conference on Robot Learning*, 2023. [Online]. Available: https://openreview.net/forum?id=MnANx01rV2w

[31] J. L. Lanciego, N. Luquin, and J. A. Obeso, "Functional neuroanatomy of the basal ganglia," *Cold Spring Harbor perspectives in medicine*, vol. 2, no. 12, 2012.

[32] M. Chignoli, D. Kim, E. Stanger-Jones, and S. Kim, "The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors," in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2021, pp. 1–8.

[33] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "Dtc: Deep tracking control - a unifying approach to model-based planning and reinforcement-learning for versatile and robust locomotion," *ArXiv*, vol. abs/2309.15462, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:263152143

[34] Y. Ding, C. Li, and H.-W. Park, "Single leg dynamic motion planning with mixed-integer convex optimization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–6.

[35] K. Yunt and C. Glocker, "Trajectory optimization of mechanical hybrid systems using sumt," in *9th IEEE International Workshop on Advanced Motion Control, 2006.*, 2006, pp. 665–671.

[36] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[37] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[38] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[39] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[40] A. Tang, T. Hiraoka, N. Hiraoka, F. Shi, K. Kawaharazuka, K. Kojima, K. Okada, and M. Inaba, "Humanmimic: Learning natural locomotion and transitions for humanoid robot via wasserstein adversarial imitation," 2023.

[41] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[42] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[43] D. Kim, S. J. Jorgensen, J. Lee, J. Ahn, J. Luo, and L. Sentis, "Dynamic locomotion for passive-ankle biped robots and humanoids using whole-body locomotion control," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 936–956, 2020. [Online]. Available: https://doi.org/10.1177/0278364920918014

[44] Y. Sim and J. Ramos, "Tello leg: The study of design principles and metrics for dynamic humanoid robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9318–9325, 2022.

[45] N. Rudin, "Isaac gym environments for legged robots," https://github.com/leggedrobotics/legged_gym, 2021.