人形机器人平滑行走姿势控制复现

摘要

在模拟器上使用强化学习训练有腿机器人的行走姿势控制，然后再迁移到现实环境中，已经成为了开发有腿机器人运动控制器的通用框架。为了避免这种迁移导致的 bang-bang 控制问题，前人提出了许多平滑策略，如低通滤波器和平滑奖励。然而，由于这些技术是不可微的，并且通常需要对大量超参数进行繁琐的调优，因此它们往往需要对每个机器人平台进行大量的手动调优。本文所复现的文章 [2] 为了解决这一问题，提出了一种简单而有效的方法来施加一个学习策略上的 Lipschitz 约束，称之为 Lipschitz 约束策略（LCP）。文章证明了 Lipschitz 约束可以以梯度惩罚的形式实现，它提供了一个可微的目标，可以很容易地与自动微分框架结合。并且证明 LCP 有效地取代了对平滑奖励或低通滤波器的需求，可以很容易地集成到许多不同的类人机器人的训练框架中。

关键词：双足机器人；步态规划；控制策略；平滑性

1 引言

类人研究的目标是开发能够在日常环境中自主操作的智能类人机器。该领域最基本的挑战之一是实现可靠的移动能力。开发鲁棒的运动控制器并使其适应真实的机器人将大大提高机器人的移动能力。传统的基于模型的方法，如模型预测控制（MPC），需要精确的系统结构和动力学建模，这是一个劳动密集型且具有挑战性的设计。相比之下，无模型强化学习提供了一种直接的端到端方法来开发鲁棒控制器，大大减轻了细致的动力学建模和系统设计的必要性。然而，由于无模型强化学习在训练过程中需要大量的试错样本，这在现实世界中是无法实现的，因此利用模拟器到真实环境的迁移技术来实现在现实环境中成功部署控制器。结合模拟到真实技术，基于无模型强化学习的方法在控制四足机器人和人形机器人方面取得了巨大成功。

然而，由于仿真中使用的简化动力学和驱动模型，所得到的模型往往接近理想化，这意味着电机可以在任何状态下产生所需的转矩。因此，在模拟中训练的基于强化学习的策略很容易产生类似于 bang-bang 控制的紧张行为。这导致连续时间步长的动作之间存在显着差异，导致真正的执行器无法产生过高的输出扭矩。因此，这些行为往往无法转移到真正的机器人身上。综上，执行平滑行为对于成功实现模拟到真实的迁移至关重要。

前人的工作主要使用平滑方法从学习策略中强制平滑行为，例如平滑奖励或低通滤波器。在训练中加入平滑性奖励是激发平滑行为的有效方法。在机器人运动控制中，研究人员通常会对关节速度、关节加速度和能量消耗进行惩罚。其他方法试图通过应用低通过滤器来平滑

策略行为。然而，平滑性奖励需要仔细调整权重，以平衡平滑行为和任务完成，低通滤波器通常会抑制或限制探索，在训练新机器人的控制器时造成额外的努力。

本文所复现的工作引入了 Lipschitz 约束策略（LCP），这是一种通用且可微的方法，用于鼓励强化学习策略形成平滑的行为。LCP 通过可微的梯度惩罚对策略输出动作相对于输入观测值施加 Lipschitz 约束。LCP 只需几行代码即可实现，并且可以轻松集成到现有的强化学习框架中。该文证明了这种方法可以直接应用于训练多种人形机器人的控制策略。本文实验表明，LCP 可以替代诸如平滑奖励和低通滤波器之类的不可微平滑技术。

2 相关工作

由于腿式机器人系统的高维性和不稳定性，腿式机器人的运动一直是机器人技术中一个关键而又具有挑战性的问题。经典的基于模型的控制方法在腿式机器人上取得了令人印象深刻的表现。近年来，基于学习的方法在自动化控制器开发过程中显示出巨大的潜力，为四足运动、两足运动以及类人运动提供了构建鲁棒控制器的通用方法。

2.1 模拟到真实迁移

基于强化学习的方法的主要挑战之一是模拟到真实迁移，其中策略首先在模拟中训练，然后在现实环境中部署。通常需要大量的努力来弥合模拟和现实世界之间的差距，例如开发高保真模拟器 [8]，并在训练期间结合领域随机化技术 [9]。另一种被广泛采用的方法是师生框架，其中首先训练具有完整状态信息访问权限的特权教师策略，然后通过蒸馏，训练基于观察的学生策略。这些方法已经成功地用于四足机器人 [3] 和人形机器人 [6] 的控制器转移。一些工作还探索了在现实世界中使用单一策略来控制具有不同形态的机器人。然而，该策略在真正的人形机器人上的表现尚未得到验证，并且不容易插入任何现有的训练框架。

2.2 学习平滑行为

由于模拟器的动力学简化，在模拟中训练的策略经常表现出无法转移到现实世界的抖动行为。因此，平滑性对于成功从模拟到真实的迁移至关重要。常用的平滑技术包括使用平滑奖励，例如惩罚动作的突然变化、自由度（DoF）速度、DoF 加速度 [10] 和能量消耗 [5]。除了平滑奖励，低通滤波器也被应用于策略的输出动作，以确保更平滑的行为 [4]。然而，平滑奖励通常需要仔细的手动设计和调优，而低通过滤器通常会抑制策略探索，导致次优策略。这些技术通常也不是直接可微的，需要基于样本的梯度估计器来优化这样的策略梯度。

2.3 梯度惩罚

梯度惩罚是生成对抗网络（GAN）稳定训练的一种常用技术，该网络容易受到梯度消失或爆炸的影响。Arjovsky 等人 [1] 提出了使用权值裁剪来稳定训练的 Wasserstein GAN（WGAN）。然而，权值裁剪仍然经常导致模型性能差和收敛问题。Gulrajani 等人 [7] 引入了梯度惩罚（WGAN-GP）作为权值裁剪的替代方法，该方法对判别器的梯度范数进行惩罚。自引入以来，梯度惩罚已成为 gan 中广泛使用的正则化技术。在运动控制中，梯度惩罚是提高对抗性模仿学习稳定性的有效方法。

3 本文方法

3.1 本文方法概述

本文利用 Lipschitz 连续性的思想来训练强化学习策略以产生平滑行为。本节将介绍 Lipschitz 约束策略（LCP），这是一种在训练过程中结合 Lipschitz 约束来训练策略以产生平滑行为的方法。接下来是 LCP 方法的详细描述。

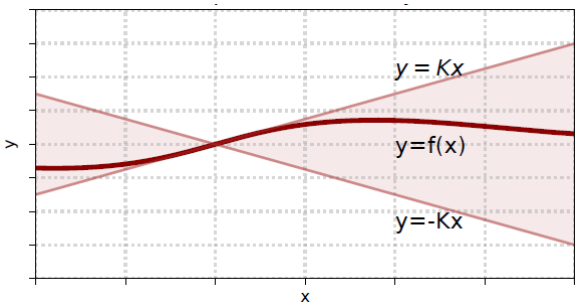

图 1. Lipschitz 连续性

3.2 Lipschitz 连续性

直观地说，Lipschitz 连续性是一个限制函数变化速度的性质。这个性质是表征函数平滑性的好方法。直观的可视化如图 1所示。形式上，我们给出 Lipschitz 连续性的定义如下：

$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2) \tag{1}$$

任何这样的 K 都被称为函数 f(x) 的 Lipschitz 常数。由 Lipschitz 连续性得到的一个推论是，如果函数的梯度有界，则这个函数是连续的:

$$\|\nabla_x f(x)\| \leq K \tag{2}$$

3.3 强化学习目标函数

在这项工作中，运动控制器通过强化学习进行训练，其中智能体根据策略 π 与环境交互，以最大化目标函数。在每个时间步 t，智能体观察环境的状态 s_t，并根据策略 $\pi(a_t|s_t)$ 采取行动 a_t。根据环境的动态 $p(s_{t+1}|s_t, a_t)$，这个动作会导致一个新的状态。agent 在每一步获得奖励 $r_t = r(s_{t+1}, s_t, a_t)$。agent 的目标是最大化其预期收益：

$$J(\pi) = \mathbb{E}_{p(\tau|\pi)}[\sum_{t=0}^{T-1} \gamma^t r_t] \tag{3}$$

式中 $p(\tau|\pi)$ 表示轨迹 τ 的概率，T 表示时间范围，γ 表示折现因子。

3.4 可微的 Lipschitz 约束目标

虽然平滑性奖励可以缓解紧张行为，但这些奖励函数的设计可能很复杂，需要调整大量的超参数。此外，这些平滑奖励是不可微分的，因为它们是作为底层环境的一部分实现的。因此，

它们通常需要通过基于抽样的方法进行优化，例如策略梯度。本文提出了一个基于 Lipschitz 连续性的策略优化的简单可微平滑目标。方程 2规定任何梯度有界的函数都是 Lipschitz 连续的。因此，我们可以制定一个通过梯度约束强制 Lipschitz 连续性的约束策略优化问题：

$$\begin{aligned} \max_{\pi} \quad & J(\pi) \\ s.t. \quad & \max_{s,a}[\|\nabla_s \log \pi(a|s)\|^2] \leq K^2 \end{aligned} \tag{4}$$

其中 K 是一个常数，$J(\pi)$ 是公式 3 中定义的 RL 目标。

4 复现细节

4.1 与已有开源代码对比

本文复现使用了原文的大部分代码，并在原有代码上增加了一个新的带手机器人类以验证该文所提出的 LCP 策略的泛化能力。新增代码如下：

```
class G1whWalkPhaseCfg(HumanoidCfg):
    class env(HumanoidCfg.env):
        num_envs = 4096
        num_actions = 52
        n_priv = 0
        n_proprio = 2 + 3 + 3 + 2 + 3*num_actions
        n_priv_latent = 4 + 1 + 2*num_actions + 3
        history_len = 10

        num_observations = n_proprio + n_priv_latent
        + history_len*n_proprio + n_priv

        num_privileged_obs = None

        env_spacing = 3.
        send_timeouts = True
        episode_length_s = 10

        randomize_start_pos = True
        randomize_start_yaw = True

        history_encoding = True
        contact_buf_len = 10

        normalize_obs = True

```

```
    class terrain(HumanoidCfg.terrain):
        mesh_type = 'trimesh'
        height = [0, 0.04]
        horizontal_scale = 0.1

    class init_state(HumanoidCfg.init_state):
        pos = [0, 0, 0.8]
        default_joint_angles = {
            'left_hip_pitch_joint': -0.4,
            'left_hip_roll_joint': 0.0,
            'left_hip_yaw_joint': 0.0,
            'left_knee_joint': 0.8,
            'left_ankle_pitch_joint': -0.35,
            'left_ankle_roll_joint': 0.0,

            'right_hip_pitch_joint': -0.4,
            'right_hip_roll_joint': 0.0,
            'right_hip_yaw_joint': 0.0,
            'right_knee_joint': 0.8,
            'right_ankle_pitch_joint': -0.35,
            'right_ankle_roll_joint': 0.0,

            'torso_joint': 0.0,

            'left_shoulder_pitch_joint': 0.0,
            'left_shoulder_roll_joint': 0.0,
            'left_shoulder_yaw_joint': 0.0,
            'left_elbow_pitch_joint': 0.0,
'Left_plam_joint': 0.0,
            'Da_Joint1': 0.0,
            'Da_Joint2': 0.0,
            'Da_Joint3': 0.0,
            'Shi_Joint1': 0.0,
            'Shi_Joint2': 0.0,
            'Shi_Joint3': 0.0,
            'Shi_Joint4': 0.0,
            'Wuming_Joint1': 0.0,
            'Wuming_Joint2': 0.0,
            'Wuming_Joint3': 0.0,
            'Wuming_Joint4': 0.0,
```

```
            'Xiao_Joint1': 0.0,
            'Xiao_Joint2': 0.0,
            'Xiao_Joint3': 0.0,
            'Xiao_Joint4': 0.0,
            'Zhong_Joint1': 0.0,
            'Zhong_Joint2': 0.0,
            'Zhong_Joint3': 0.0,
            'Zhong_Joint4': 0.0,
            'damuzhi_F1': 0.0,
            'damuzhi_F2': 0.0,
            'damuzhi_F3': 0.0,

            'right_plam_base_joint': 0.0,
            'shizhi_F1': 0.0,
            'shizhi_F2': 0.0,
            'shizhi_F3': 0.0,
            'shizhi_F4': 0.0,
            'wumingzhi_F1': 0.0,
            'wumingzhi_F2': 0.0,
            'wumingzhi_F3': 0.0,
            'wumingzhi_F4': 0.0,
            'xiaozhi_F1': 0.0,
            'xiaozhi_F2': 0.0,
            'xiaozhi_F3': 0.0,
            'xiaozhi_F4': 0.0,
            'zhongzhi_F1': 0.0,
            'zhongzhi_F2': 0.0,
            'zhongzhi_F3': 0.0,
            'zhongzhi_F4': 0.0,


        }
class control(HumanoidCfg.control):
        stiffness = {'hip_yaw': 150,
                     'hip_roll': 150,
                     'hip_pitch': 200,
                     'knee': 200,
                     'ankle': 20,
                     'torso': 200,
                     'shoulder': 40,
```

```
                        'elbow': 40,
                        }  # [N*m/rad]
        damping = {  'hip_yaw': 5,
                        'hip_roll': 5,
                        'hip_pitch': 5,
                        'knee': 5,
                        'ankle': 4,
                        'torso': 5,
                        'shoulder': 10,
                        'elbow': 10,
                        }  # [N*m/rad]  # [N*m*s/rad]

        action_scale = 0.5
        decimation = 20

class sim(HumanoidCfg.sim):
        dt = 0.001

    class normalization(HumanoidCfg.normalization):
        clip_actions = 5

    class asset(HumanoidCfg.asset):

        file =
        '{LEGGED_GYM_ROOT_DIR}/resources/robots/g1_with_hands/g1.urdf'
        # for both joint and link name
        torso_name: str = 'torso_link'  # humanoid pelvis part
        chest_name: str = 'torso_link'  # humanoid chest part
        forehead_name: str = 'head_link'  # humanoid head part

        waist_name: str = 'torso_joint'

        # for link name
        thigh_name: str = 'hip_roll_link'
        shank_name: str = 'knee_link'
        foot_name: str = 'ankle_roll_link'
        upper_arm_name: str = 'shoulder_roll_link'
        lower_arm_name: str = 'elbow_pitch_link'
        hand_name: str = 'hand'
       # for joint name
```

```
        hip_name: str = 'hip'
        hip_roll_name: str = 'hip_roll_joint'
        hip_yaw_name: str = 'hip_yaw_joint'
        hip_pitch_name: str = 'hip_pitch_joint'
        knee_name: str = 'knee_joint'
        ankle_name: str = 'ankle'
        ankle_pitch_name: str = 'ankle_pitch_joint'
        shoulder_name: str = 'shoulder'
        shoulder_pitch_name: str = 'shoulder_pitch_joint'
        shoulder_roll_name: str = 'shoulder_roll_joint'
        shoulder_yaw_name: str = 'shoulder_yaw_joint'
        elbow_name: str = 'elbow_pitch_joint'

        feet_bodies = ['left_ankle_roll_link', 'right_ankle_roll_link']
        n_lower_body_dofs: int = 12

        penalize_contacts_on = ["shoulder", "elbow", "hip"]
        terminate_after_contacts_on = ['torso_link']

    class rewards(HumanoidCfg.rewards):
        regularization_names = [
                            "dof_error",
                            "dof_error_upper",
                            "feet_stumble",
                            "feet_contact_forces",
                            "lin_vel_z",
                            "ang_vel_xy",
                            "orientation",
                            "dof_pos_limits",
                            "dof_torque_limits",
                            "collision",
                            "torque_penalty",
                            ]
        regularization_scale = 1.0
        regularization_scale_range = [0.8,2.0]
        regularization_scale_curriculum = True
        regularization_scale_gamma = 0.0001
        class scales:
            joint_pos = 1.6
            feet_clearance = 1.
```

```
            feet_contact_number = 1.2

            feet_air_time = 1.
            foot_slip = -0.1
            feet_distance = 0.2
            knee_distance = 0.2
        regularization_scale_curriculum = True
        regularization_scale_gamma = 0.0001
        class scales:
            joint_pos = 1.6
            feet_clearance = 1.
            feet_contact_number = 1.2

            feet_air_time = 1.
            foot_slip = -0.1
            feet_distance = 0.2
            knee_distance = 0.2

            tracking_lin_vel_exp = 1.875
            tracking_ang_vel = 2.0

            alive = 2.0
            dof_error = -0.15
            dof_error_upper = -0.2
            feet_stumble = -1.25
            feet_contact_forces = -5e-4

            lin_vel_z = -1.0
            ang_vel_xy = -0.1
            orientation = -1.0

            collision = -10.0

            dof_pos_limits = -10
            dof_torque_limits = -1.0
            torque_penalty = -6e-7

        min_dist = 0.2
        max_dist = 0.5
        max_knee_dist = 0.5
```

```
        target_joint_pos_scale = 0.17
        target_feet_height = 0.1
        cycle_time = 0.64
        double_support_threshold = 0.1
        only_positive_rewards = False
        tracking_sigma = 0.2
        tracking_sigma_ang = 0.125
        max_contact_force = 350
        termination_height = 0.3

    class domain_rand:
        domain_rand_general = True

        randomize_gravity = (True and domain_rand_general)
        gravity_rand_interval_s = 4
        gravity_range = (-0.1, 0.1)

        randomize_friction = (True and domain_rand_general)
        friction_range = [0.6, 2.]

        randomize_base_mass = (True and domain_rand_general)
        added_mass_range = [-3., 3]

        randomize_base_com = (True and domain_rand_general)
        added_com_range = [-0.05, 0.05]

        push_robots = (True and domain_rand_general)
        push_interval_s = 4
        max_push_vel_xy = 1.0

        randomize_motor = (True and domain_rand_general)
        motor_strength_range = [0.8, 1.2]

        action_delay = (True and domain_rand_general)
        action_buf_len = 8

    class noise(HumanoidCfg.noise):
        add_noise = True
        noise_increasing_steps = 5000
        class noise_scales:
```

```
            dof_pos = 0.01
            dof_vel = 0.1
            lin_vel = 0.1
            ang_vel = 0.05
            gravity = 0.05
            imu = 0.05

    class commands:
        curriculum = False
        num_commands = 3
        resampling_time = 3.

        ang_vel_clip = 0.1
        lin_vel_clip = 0.1

        class ranges:
            lin_vel_x = [0., 0.8] # min max [m/s]
            lin_vel_y = [-0.3, 0.3]
            ang_vel_yaw = [-0.6, 0.6]     # min max [rad/s]

class G1whWalkPhaseCfgPPO(HumanoidCfgPPO):
    seed = 1
    class runner(HumanoidCfgPPO.runner):
        policy_class_name = 'ActorCriticRMA'
        algorithm_class_name = 'PPORMA'
        runner_class_name = 'OnPolicyRunner'
        max_iterations = 20001 # number of policy updates

        # logging
        save_interval = 100
        experiment_name = 'test'
        run_name = ''
        # load and resume
        resume = False
        load_run = -1 # -1 = last run
        checkpoint = -1 # -1 = last saved model
        resume_path = None # updated from load_run and chkpt

    class policy(HumanoidCfgPPO.policy):
        action_std=[0.3,0.3,0.3,0.4,0.2,0.2]*2+[0.1]+[0.2]*8
```

```
    class algorithm(HumanoidCfgPPO.algorithm):
        grad_penalty_coef_schedule =[0.002,0.002,700,1000]
```

4.2 实验环境搭建

实验所用模拟器为 issacgym，需要前往 nvidia 官网下载对应的模拟器包并安装所需依赖库，具体环境流程代码里有。

5 实验结果分析

本次复现结果如下：

Method Action	Jitter	Dof Pos Jitter	Dof Velocity	Energy	Base Acc	Task Return
LCP	3.27	0.17	11.03	23.92	0.06	26.71

原文结果如下：

Method	Action Jitter ↓	DoF Pos Jitter ↓	DoF Velocity ↓	Energy ↓	Base Acc ↓	Task Return ↑
(a) Ablation on Smooth Methods						
LCP (ours)	$\mathbf{3.21 \pm 0.11}$	$\mathbf{0.17 \pm 0.01}$	$\mathbf{10.65 \pm 0.37}$	$\mathbf{24.57 \pm 1.17}$	$\mathbf{0.06 \pm 0.002}$	26.03 ± 1.51
Smoothness Reward	5.74 ± 0.08	0.19 ± 0.002	11.35 ± 0.51	25.92 ± 0.84	$\mathbf{0.06 \pm 0.002}$	26.56 ± 0.26
Low-pass Filter	7.86 ± 3.00	0.23 ± 0.04	11.72 ± 0.14	32.83 ± 5.50	$\mathbf{0.06 \pm 0.002}$	24.98 ± 1.29
No Smoothness	42.19 ± 4.72	0.41 ± 0.08	12.92 ± 0.99	42.68 ± 10.27	0.09 ± 0.01	$\mathbf{28.87 \pm 0.85}$
(b) Ablation on GP Weights (λ_{gp})						
LCP w. $\lambda_{gp} = 0.0$	42.19 ± 4.72	0.41 ± 0.08	12.92 ± 0.99	42.68 ± 10.27	0.09 ± 0.01	$\mathbf{28.87 \pm 0.85}$
LCP w. $\lambda_{gp} = 0.001$	3.69 ± 0.31	0.21 ± 0.05	11.44 ± 1.18	27.09 ± 4.44	0.06 ± 0.01	26.32 ± 1.20
LCP w. $\lambda_{gp} = 0.002$ (ours)	3.21 ± 0.11	0.17 ± 0.01	10.65 ± 0.37	24.57 ± 1.17	0.06 ± 0.002	26.03 ± 1.51
LCP w. $\lambda_{gp} = 0.005$	2.10 ± 0.05	0.15 ± 0.01	10.44 ± 0.70	26.24 ± 3.50	0.05 ± 0.002	23.92 ± 2.05
LCP w. $\lambda_{gp} = 0.01$	$\mathbf{0.17 \pm 0.01}$	$\mathbf{0.07 \pm 0.00}$	$\mathbf{2.75 \pm 0.12}$	$\mathbf{5.89 \pm 0.28}$	$\mathbf{0.007 \pm 0.00}$	16.11 ± 2.76
(c) Ablation on GP Inputs						
LCP w. GP on whole obs (ours)	$\mathbf{3.21 \pm 0.11}$	$\mathbf{0.17 \pm 0.01}$	$\mathbf{10.65 \pm 0.37}$	$\mathbf{24.57 \pm 1.17}$	$\mathbf{0.06 \pm 0.002}$	$\mathbf{26.03 \pm 1.51}$
LCP w. GP on current obs	7.16 ± 0.60	0.35 ± 0.03	13.70 ± 1.50	35.18 ± 4.84	0.09 ± 0.005	25.44 ± 3.73

由以上结果对比可知，LCP 策略在双足机器人的步伐控制平滑性上具备较好的效果，在本次复现中达到了原文的预期结果。

6 总结与展望

此次复现验证了 LCP 对双足机器人平滑步行的有效性，但是并未能完成从模拟器到现实环境的迁移，该文工作对平滑性的改善最终目的是使双足机器人的运动控制器在 SimToReal 中更好的部署，本文只是验证了 LCP 策略在双足机器人上实现平滑行走的可行性，但是未能继续验证 SimToReal 迁移的有效性，这是本文的不足。

另一方面，LCP 策略在平滑步行上的良好结果，是否能体现在更复杂的奔跑跳跃等复杂动作，还需要更深入的研究和评估。

参考文献

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv e-prints*, page arXiv:1701.07875, January 2017.

[2] Zixuan Chen, Xialin He, Yen-Jen Wang, Qiayuan Liao, Yanjie Ze, Zhongyu Li, S. Shankar Sastry, Jiajun Wu, Koushil Sreenath, Saurabh Gupta, and Xue Bin Peng. Learning Smooth Humanoid Locomotion through Lipschitz-Constrained Policies. *arXiv e-prints*, page arXiv:2410.11825, October 2024.

[3] Xuxin Cheng, Ashish Kumar, and Deepak Pathak. Legs as Manipulator: Pushing Quadrupedal Agility Beyond Locomotion. *arXiv e-prints*, page arXiv:2303.11330, March 2023.

[4] Gilbert Feng, Hongbo Zhang, Zhongyu Li, Xue Bin Peng, Bhuvan Basireddy, Linzhu Yue, Zhitao Song, Lizhi Yang, Yunhui Liu, Koushil Sreenath, and Sergey Levine. GenLoco: Generalized Locomotion Controllers for Quadrupedal Robots. *arXiv e-prints*, page arXiv:2209.05309, September 2022.

[5] Zipeng Fu, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Minimizing Energy Consumption Leads to the Emergence of Gaits in Legged Robots. *arXiv e-prints*, page arXiv:2111.01674, October 2021.

[6] Xinyang Gu, Yen-Jen Wang, Xiang Zhu, Chengming Shi, Yanjiang Guo, Yichen Liu, and Jianyu Chen. Advancing Humanoid Locomotion: Mastering Challenging Terrains with Denoising World Model Learning. *arXiv e-prints*, page arXiv:2408.14472, August 2024.

[7] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. *arXiv e-prints*, page arXiv:1704.00028, March 2017.

[8] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. *arXiv e-prints*, page arXiv:2108.10470, August 2021.

[9] Ilija Radosavovic, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik. Humanoid Locomotion as Next Token Prediction. *arXiv e-prints*, page arXiv:2402.19469, February 2024.

[10] Chong Zhang, Wenli Xiao, Tairan He, and Guanya Shi. WoCoCo: Learning Whole-Body Humanoid Control with Sequential Contacts. *arXiv e-prints*, page arXiv:2406.06005, June 2024.