Hierarchical Reinforcement Learning for Precise Soccer Shooting Skills using a Quadrupedal Robot

Yandong Ji*,1, Zhongyu Li*,1, Yinan Sun1, Xue Bin Peng1, Sergey Levine1, Glen Berseth2, Koushil Sreenath1

***Abstract*— We address the problem of enabling quadrupedal robots to perform precise shooting skills in the real world using reinforcement learning. Developing algorithms to enable a legged robot to shoot a soccer ball to a given target is a challenging problem that combines robot motion control and planning into one task. To solve this problem, we need to consider the dynamics limitation and motion stability during the control of a dynamic legged robot. Moreover, we need to consider motion planning to shoot the hard-to-model deformable ball rolling on the ground with uncertain friction to a desired location. In this paper, we propose a hierarchical framework that leverages deep reinforcement learning to train (a) a robust motion control policy that can track arbitrary motions and (b) a planning policy to decide the desired kicking motion to shoot a soccer ball to a target. We deploy the proposed framework on an A1 quadrupedal robot and enable it to accurately shoot the ball to random targets in the real world.**

I. Introduction

Quadrupedal robots have attracted a great deal of interest in the robotics community, and classical model-based methods [1]–[3] have been effective paradigms for designing controllers for these robots. However, model-based methods need careful system modeling, with intricate models being less practical for online deployment due to computational limits. Furthermore, it can be difficult to apply model-based methods in settings where the dynamics are difficult to model, such as a deformable soccer ball. To shoot a ball, a controller is needed to enable the robot to swing its leg quickly to gain enough momentum to kick the ball to roll on the ground while also maintaining balance of its body during the fast kicking motion. Besides the motion control of the robot, motion planning is also challenging and requires the robot to consider the real-time ball and goal positions to generate a reasonable body motion while respecting the robot's physical limits. Moreover, in order to precisely shoot the ball to the goal, the motion planner needs to also deal with the hard-to-model contact between the robot and the deformable soccer ball, as well as uncertainties associated with the rolling friction between the ball and the ground. Although this task can be challenging for model-based methods, recent deep reinforcement learning (RL) methods have presented potential techniques for tackling this without the need for explicit models of systems. In this paper, we develop a hierarchical model-free RL framework that includes a motion control policy and a planning policy to perform precise shooting skills on quadrupedal robots in the real world, as shown in Fig. 1.

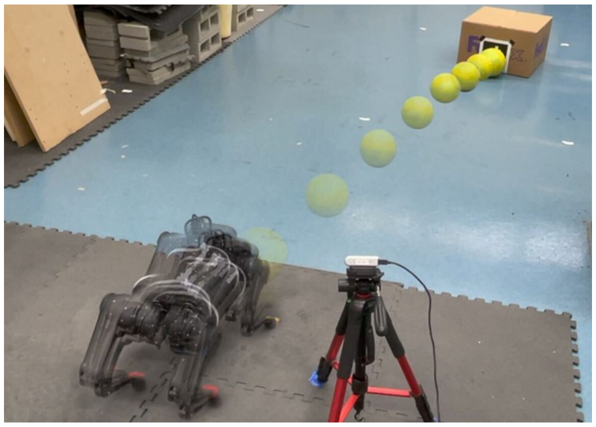


Fig. 1: A quadrupedal robot precisely shoots a soccer ball to a randomly given target using the proposed hierarchical RL framework which includes a motion control policy and a motion planning policy. The results are best seen at the video: `https://youtu.be/bteipHcJ8BM`.

A. Related Work

There has been exciting recent progress on applying deep RL on nonprehensile manipulation tasks using robotic arms and locomotion tasks with legged robots, respectively. In this section, we review the most relevant work in these domains.

1) Nonprehensile Manipulation using Robotic Arms: Deep RL has been used to train robotic arms to hit, throw, or toss objects in both simulated and real-world environments. Controllers have been trained to precisely hit a moving object in the air [4]–[6], as well as to throw an object to a target [7]. For example, in [8], motor primitives are learned by imitation learning and RL for a dart throwing task and for hitting flying table tennis balls. For tossing objects to a target, both model-based RL and model-free RL techniques have been utilized in [9]. However, most of these prior systems focus on tasks that primarily involve ballistic motion of the objects. For more contact-rich tasks, such as kicking a soccer ball rolling on the ground, the movement of the object involves complex rolling friction and contact dynamics that are substantially harder to control. There are recent endeavors using guided policy search to train robots to hit a hockey puck on the ground [10] and using Bayesian system identification to play mini golf [11]. However, the objects used in these work are primarily rigid, with simpler contact dynamics compared to

* Authors contributed equally.
1 University of California, Berkeley. {yandong0204, zhongyu_li, syn1122, xbpeng, koushils}@berkeley.edu, svlevine@eecs.berkeley.edu
2 Université de Montréal, Mila, glen.berseth@mila.quebec

deformable objects, such as a soft soccer ball. Furthermore, all of the aforementioned systems use fully actuated robotic arms with static bases. The problem becomes more difficult when controlling a legged robot, since the controller now must also maintain stability of a robot with a floating base, which is not considered by previous work. For example, a quadrupedal robot learns to manipulate a soft ball while laying the robot base on the ground to avoid considering the balancing problem in [12].

2) Locomotion on Quadrupedal Robots with RL: There has a been a large body of recent work on applying model-free RL to control quadrupedal robots for agile locomotion skills in the real world [13]–[17]. Many of these systems use some form of sim-to-real transfer, which allows policies trained in simulation to be deployed on a real robot. These sim-to-real techniques can be broadly classified into two categories: zero-shot transfer by training a policy for direct deployment on the hardware [13], [15], [17]–[19], and methods that continue to incorporate real world data [14], [16] after pretraining in simulation. However, all of above-mentioned RL approaches focus on developing low-level locomotive skills, such as walking. Using quadrupeds to accomplish a more complex and higher level tasks, such as combining the robot's body motion control and motion planning to shoot a deformable object, is not addressed.

3) Legged Soccer Robots: Developing a legged robot that can kick a ball like a human has attracted lot of interest in the robotics community, and notably, in the RoboCup Leagues [20]. Most previous approaches tackling this problem focus more on rule-based motion primitives, without considering shooting to a desired location, such as the work using quadrupedal robots [20]–[23] and humanoid robots [24]–[26] in the RoboCup. There have been some attempts to improve shooting skills, such as [27] where trajectory optimization is utilized to improve shooting distance, but this is only validated in simulation. Recently, model-free RL has demonstrated promising results for training a single policy to dribble, kick, and shoot a soccer ball with simulated humanoid robots [28]–[30]. These end-to-end methods have the advantage of being able to produce more agile behaviors by directly leveraging the robot's full-order dynamics and contacts with the ball. However, transferring policies learned in simulation to real legged robots is challenging due to differences between the dynamics of the simulator and the physical system. Therefore, effectively leveraging RL to develop soccer shooting skills for legged robots in the real world remains an open problem and has yet to be demonstrated on a real-world legged robot.

B. Contributions

The central contribution of this work is the design and development of a hierarchical reinforcement learning frameworks for precise soccer shooting skills using quadrupedal robots. This framework leverages model-free RL to enable a standing quadrupedal robot to precisely shoot a soccer ball by coupling robot motion control and motion planning. Our motion control policy learns various full-body motions in order to track random parametric end-effector (toe) trajectories while maintaining balance during standing. The motion planning policy is responsible for shooting the deformable soccer ball to a desired target. This uses a multi-stage training approach, wherein a planning policy is first trained with a rigid ball in simulation, then fine-tuned with a soft ball in the real world. We validate the proposed methodology in real-world experiments using a quadrupedal robot, and demonstrate the feasibility of attaining a robust control policy for dynamic soccer shooting motions and precise planning policy to shoot a deformable soccer ball to a random specified target in a relatively large range. This paper serves as a step towards the development of RL-based quadrupedal robotic soccer players in the real world that could one day compete with humans.

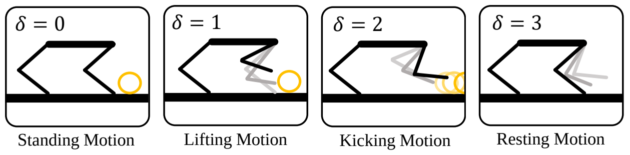


Fig. 2: Motions that constitutes the shooting maneuver of a soccer ball (marked by the yellow circle). The quadrupedal robot needs to start with a standing motion, then lift its leg up and hold it in the air to prepare for the kicking motion once commanded. After kicking, the robot should put the leg down and get back to standing.

II. Soccer Shooting Skill using a Quadruped

In this section, we provide a brief overview of the A1 quadrupedal robot, which serves as the experimental platform for this work. We then present our proposed approach for developing soccer shooting skills using the A1 robot.

A. A1 Robot

The A1 quadrupedal robot, shown in Fig. 1, has a total of 18 Degree-of-Freedoms (DoFs). There are 6 DoFs for its base that include the saggital q_x, lateral q_y, vertical q_z translational positions and roll q_ψ, pitch q_ϕ, and yaw q_θ rotational positions. Each of the four legs has three actuated motors and we denote these joint coordinates as $q_m \in \mathbb{R}^{12}$.

B. Soccer Shooting Skill

Our goal is to develop controllers that allow a quadrupedal robot to use one leg to shoot a ball to a specified target while standing. As illustrated in Fig. 2, we breakdown the entire shooting maneuver into four motions denoted by an indicator $\delta \in \mathbb{Z}$. The robot starts from a standing motion $\delta = 0$ where all of the four legs are on the ground. Then, the robot needs to execute a lifting motion, denoted by $\delta = 1$, where it lifts up one of its legs and holds it in position to prepare for the kick. We termed the leg that is lifted and used for kicking as the *kicking leg*, while other legs are *stance legs*. When the robot is commanded to kick, the robot executes a kicking motion, denoted by $\delta = 2$, where the kicking leg should accelerate in order to gain momentum and then transfer the momentum to the ball on contact with the ball. The robot should also adjust the contact direction and contact force with the ball in order to shoot the ball to the specified target location. This kicking motion is difficult to perform on a quadrupedal robot, as the robot is not only required to transit from a stationary motion to a fast motion and slow down again in a very short time span, but also needs to prevent itself from falling over while executing fast kicks. When the kicking is completed, the robot needs to put down the kicking leg and transit back to the nominal standing motion and get ready for the next round of kicks. We refer to this last motion as the resting motion ($\delta = 3$). Each of these motions has its own time-span which is denoted as T_δ.

C. *Motion Representation with Bézier Curves*

In this paper, we will use Bézier curves [32] to represent each motion ($\delta = 0, 1, 2, 3$) of the soccer shooting maneuver for the control policy to perform, and will have a planning policy to compute the parameters of the desired Bézier curve. We do this to take into account various features of parametric Bézier curves, such as smoothness of the curve and physical meaning of the parameters.

We use the Front Right (FR) leg of the robot as the kicking leg. If we consider the robot's kicking leg as a robot arm attached to the robot's floating base, then the toe of the kicking leg can be considered as the *end-effector*. We define the toe position of the kicking leg as $x_e \in \mathbb{R}^3$. The trajectory of the end-effector in the $3D$ space can be parameterized by a Bézier curve [32],

$$B_\alpha(t) = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!}(1-t)^{n-i} t^i \alpha_i \tag{1}$$

where $\alpha_i \in \mathbb{R}^{3\times 1}$ are the Bézier parameters, with $n+1$ being the number of the parameters. We use $n = 4$ in this paper. The variable $t \in [0, 1]$ is the phase time that is scaled by the timespan of the trajectory T_δ. Therefore, the end-effector trajectory can be defined by a discrete set $\alpha = \{\alpha_0, \alpha_1, \ldots, \alpha_4\}$. Please note that, the resulting Bézeir curve is infinitely differentiable with respect to phase t and we can not only obtain the reference position information from the curve but also higher order terms such as reference velocity and acceleration. Furthermore, the four motions discussed in Fig. 2 can have different Bézier parameters α. Thus, enabling the robot to perform different motions in the shooting maneuver can now be generalized to tracking Bézier curves with different parameters.

Given the above end-effector trajectory for the swing leg with the constraint that the stance legs have their toes on the ground, one can typically use Inverse Kinematics (IK) to solve for all the leg joints and the body pose. However, as we will see in the following sections, instead of solving IK online, we can train a policy to directly move the robot so as to directly follow the reference swing leg end-effector trajectory specified by the Bézier parameters.

III. Hierarchical Learning Framework

We now propose a framework that leverages hierarchical reinforcement learning to enable a quadrupedal robot to learn soccer shooting skills. As illustrated in Fig. 3, we break down the soccer shooting problem into two parts: to plan for the robot's shooting motion in order to kick the ball to a given target, and to control the robot to perform the planned motion while staying balanced.

A control policy π_c is first developed to enable the robot to track arbitrary end-effector (FR toe) trajectories while preventing falling by adjusting the robot pose via stance legs. This is realized by training a policy to output desired robot motor positions q_m^d in order to track a desired end-effector trajectory represented by Bézier parameters α while respecting the robot dynamics. This control policy runs at 30 Hz and its output q_m^d is passed through a Low Pass Filter (LPF) [14], [19] and joint-level PD controllers to obtain desired motor torques τ on the robot at 1 kHz.

The soccer ball position $\hat{o} \in \mathbb{R}^3$ in the robot body frame is detected by a RGB-Depth camera, and in order to shoot the soccer to a given goal location $g \in \mathbb{R}^2$, we develop a planning policy π_p on top of the control policy. This planning policy is trained to examine the current ball position, robot states, and the shooting goal, and based on these, to generate optimal end-effector Bézier parameters for the controller to track. This planning policy replans at 1 Hz to deal with online disturbance and tracking errors due to the controller. The planning policy developed in this work only considers shooting the ball to a target rather than to enable the ball to follow a trajectory while reaching the target.

There is a rule-based motion selector to indicate to the robot what motion presented in Fig. 2 should be performed at the current time. The appropriate motion indicator δ will be selected and passed to both the planning and control policies.

The advantage of using this hierarchical learning framework is that we can separate the high-level shooting skills into two subproblems: control and planning. We can first focus on training a robust control policy in simulation that can be transferred from simulation to the real world to allow the robot to track arbitrary end-effector trajectories without causing the robot to fall over. Afterwards, we can reuse this control policy for developing the planning policy and focus only on how to precisely shoot the soccer to the goal.

IV. Learning the Soccer Shooting Control

We now present the development of the control policy, which is first trained in simulation by RL and then directly transferred to a real robot, allowing the robot to track arbitrary end-effector trajectories while balancing.

A. *Training Environment*

The environment for training the control policy for the A1 quadrupedal robot agent is developed in MuJoCo [33]. The details of the training environment are introduced below.

1) Action Space: The action $\mathbf{a}_k^c$ of control policy at time step k is the desired joint position $q_m^d \in \mathbb{R}^{12}$. These are passed through a Low Pass Filter (LPF) and input to joint-level PD controllers to obtain the motor torques $\tau \in \mathbb{R}^{12}$, as shown in Fig. 3.

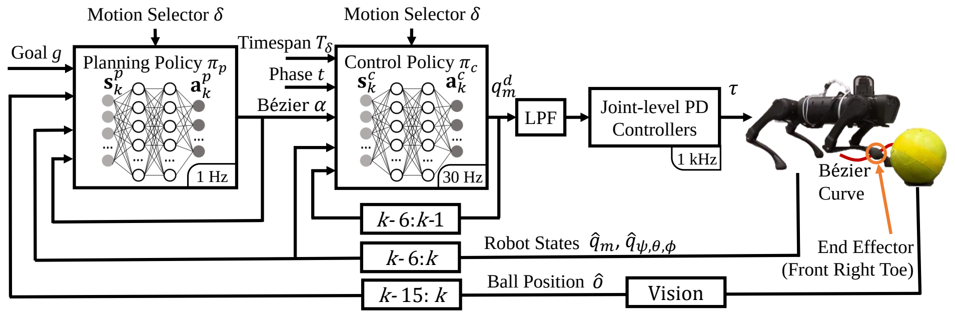


Fig. 3: Proposed hierarchical reinforcement learning framework for a quadrupedal robot to perform a precise shooting maneuver. A control policy π_c is firstly trained to track arbitrary end-effector (front right toe) trajectories represented by Bézier parameters α and motion time span T_δ. The control policy replans at 30 Hz. After obtaining a control policy that can reliably work on the real robot, we use it to train a planning policy π_p to output a desired shooting motion to the controller in order to shoot the soccer ball to the goal g. The goal can be randomly placed and identified by an AprilTag [31]. The planning policy updates its observation and action at 1 Hz and is synchronized with the controller, *e.g.*, the time step k spans $1/30$ second. There is a rule-based motion selector to output an indicator δ to inform the planner and controller about the current motion type: standing, lifting, kicking, or resting.

2) State Space: As illustrated in Fig. 3, the observation $\mathbf{s}_k^c$ of the control policy at time step k contains 6 parts. The first part is the current motion indicator $\delta \in \mathbb{Z}$ from the motion selector. As introduced in Sec. II-B, δ is selected from $\{0, 1, 2, 3\}$ that represents standing, lifting, kicking, and resting motions, respectively. This indicator can help the robot to understand and distinguish the different motions that it is required to perform. The second and third parts are the desired Bézier parameters $\alpha \in \mathbb{R}^{3\times5}$ and corresponding trajectory timespan T_δ, respectively. The entire desired trajectory can then be well defined by these two variables. The fourth input is the current motion phase $t \in [0, 1]$ scaled by timespan T_δ, which implicitly informs the robot about the desired end-effector position. The fifth part of the observation consists of the robot's current and past states. The feedback of the robot states at time step k includes measured robot joint position $\hat{q}_m \in \mathbb{R}^{12}$ and robot orientation $\hat{q}_{\psi,\theta,\phi} \in \mathbb{R}^3$. Please note that this feedback is raw sensor reading and we do not require a state estimator or other filters. Instead, we include a history of the past 6 time steps (around 0.2 seconds) of robot state feedback into the observation and encourage the policy to learn the filtering and state estimation by itself. Finally, the sixth part of the observation is a history of the past 6 time steps of previous action $\mathbf{a}_{k-1:k-6}^c$. We hypothesize that the history of past robot states and actions enables the control policy to infer the closed-loop dynamics of the robot using the input/output data.

3) Policy Representation: The control policy is represented by a deep neural network that consists of two hidden layers. Each one is a fully-connected layer with 512 units and $tanh$ activation.

B. Reward

At each time step, the robot takes an observation, executes one action obtained by the policy, and receives a reward. The reward function for the control policy is designed to encourage the robot to follow the desired end-effector trajectory while maintaining balance and improving the smoothness of the robot motions. The reward $r_{c,k} \in [0, 1]$ at time step k is formulated as:

$$r_{c,k} = w_c^T [r_{c,k}^e, r_{c,k}^m, r_{c,k}^{\dot{m}}, r_{c,k}^b, r_{c,k}^{\dot{b}}, r_{c,k}^\tau, r_{c,k}^{\Delta\mathbf{a}}]^T. \quad (2)$$

Here, $r_{c,k}^e$ stands for the reward in terms of the end-effector position tracking error and is formulated as

$$r_{c,k}^e = \exp(-\rho^e ||x_{e,k} - B_\alpha(t)||_2^2), \quad (3)$$

where $x_{e,k}$ is the current ground truth end-effector (front right toe) position at time step k, $B_\alpha(t)$ is the reference end-effector position calculated via (1), and $\rho^e > 0$ is a scaling variable. This term is designed to encourage the robot's front right toe (end effector of kicking leg) to follow the reference curve obtained by the Bézier parameters α, motion time span T_δ, and current motion phase t. The rest of the reward terms have a similar formulation as (3) and they serve different purposes. The term $r_{c,k}^m$ and $r_{c,k}^b$ are designed to encourage the other stance legs and the robot base to imitate a nominal standing motion, respectively. In order to make the robot motion smoother, we include $r_{c,k}^{\dot{m}}$ and $r_{c,k}^{\dot{b}}$ to stimulate the robot to damp out the motor and base velocities. $r_{c,k}^{\Delta\mathbf{a}}$ is also introduced to minimize the change of the control policy output $||\mathbf{a}_k - \mathbf{a}_{k-1}||_2$ between two adjacent time step. Moreover, we add $r_{c,k}^\tau$ to minimize the current torque consumption for energy efficiency. Additionally, w_c is a normalization weight vector with a dominating weight on the end-effector position tracking term $r_{c,k}^e$.

C. Domain and Motion Randomization

In order to generalize the control policy in the environments with uncertain dynamics properties, such as in the real world, the dynamics parameters of the robot and the environment are randomized during training in simulation and the range of the uniform randomization is presented in Table I. For example, in order to encourage the robot to stay robust to the modeling error between the simulation and real world, in each simulation episode, the mass, inertia, and mass center position of each robot link, as well as joint damping ratio, are changed and sampled within recorded range in Table I. Ground frictions are randomized in order to encourage the robot to be robust to the friction changes

TABLE I: Randomization range during training.

Parameter	Range	Unit
Control and Planning		
Robot Link Mass	$[0.5, 1.5] \times$ default	kg
Robot Link Inertia	$[0.7, 1.3] \times$ default	kgm^2
Robot Base Mass Center	$[-0.1, 0.1]$	m
Robot Link Mass Center	$[-0.05, 0.05]$	m
Robot Joint Damping	$[0.7, 4.0]$	Nms/rad
Ground Frictions	$[0.5, 3.0]$	1
Motor Encoder Noise Mean	$[-0.01, 0.01]$	rad
Gyro Rotation Noise Mean	$[-0.01, 0.01]$	rad
Communication Delay	$[0, 0.025]$	sec
Only for Control		
Standing Time Span $T_{\delta=0}$	$[1.0, 4.0]$	sec
Lifting Time Span $T_{\delta=1}$	$[3.0, 4.0]$	sec
Kicking Time Span $T_{\delta=2}$	$[0.2, 0.4]$	sec
Resting Time Span $T_{\delta=3}$	$[1.0, 3.0]$	sec
Bézier Parameters $\alpha_{0,1,4}$	$[-0.1, 0.1]$ + nominal	m
Bézier Parameters $\alpha_{2,3}$	$[-0.1, 0.3]$ + nominal	m
Perturbation Force and Torque	$[-20, 20]$, $[-5, 5]$	N, Nm
Only for Planning		
Ball Stiffness	$[0.7, 2.0]$	N/m
Ball Mass	$[0.5, 1.5] \times$ default	kg
Ball Inertia and Radius	$[0.7, 1.3] \times$ default	kgm^2, m
Ball Detection Noise	$[-0.05, 0.05]$	m
Ball Detection Delay	$[0, 0.3]$	sec

in the real world. Moreover, the sensor noise is simulated as Gaussian distribution with the mean sampled from the presented range, and communication delay between the computer running the RL policy and the low-level computer is also introduced. Additionally, we also randomly apply a 6 DoF random perturbation force to the robot base during training in order to increase the robustness of the policy.

Furthermore, as introduced in Sec. III, we want to develop a control policy that is able to track arbitrary end-effector (toe) trajectories for quadrupedal robots. Therefore, as shown in Table I, the desired Bézier parameters α and time span T_δ are also randomized during training. Such a motion randomization is based on nominal hand-crafted lifting, kicking, and resting motions, respectively. Based on the Bézier parameters of the end-effector trajectory of each nominal motion, we add a large range of randomization of the desired α for the control policy to learn. Furthermore, the time spans T_δ for each motion during one soccer shooting maneuver are also randomized. In this way, we can encourage the robot to learn a large repertoire of shooting motion.

D. Training Setup

During training, each episode has a horizon of $N = 2500$ timesteps, which is approximately 80 seconds in length. In each episode, the robot is repeatedly required to track random shooting motions sampled according to the parameters from Table I, and the dynamics parameters are also randomized according to Table I and kept fixed over the course of an episode. After one motion is completed, the robot needs to learn to keep standing until the next new motion begins. When the robot falls over, the episode will be terminated and the policy receives 0 return for all remaining timesteps. In this way, together with the end-effector tracking reward formulated in (2) by using the control policy, we can encourage the robot to stay as close as possible to the reference end-effector trajectory while respecting the robot dynamics limitation and motion stability (not falling over). The episode will also be terminated if end-effector deviates from the reference by more than a relatively large threshold. This can prevent the robot adopting a very conservative behavior, such as just holding the toe in the air. The parameters of the control policy is optimized by Proximal Policy Optimization [34] to maximize the total expected discounted reward in one episode.

V. Learning the Soccer Shooting Plan

After obtaining a control policy that is able to track arbitrary shooting motions, we now develop a planning policy using RL to decide an optimal motion that can shoot the soccer ball to the goal location in both simulation and the real world.

A. Training Environment

1) Action Space: As illustrated in Fig. 3, the action $\mathbf{a}_k^p$ of the planning policy at time step k is the Bézier parameters $\alpha \in \mathbb{R}^{3\times 5}$ representing the desired robot motion at current time. This is sent to the control policy to track.

2) State Space: The observation of the planning policy $\mathbf{s}_k^p$ consists of five components. It includes a goal position $g_k \in \mathbb{R}^2$ which is the $2D$ shooting target on the ground. The second part is the detected ball position $\hat{o}_k \in \mathbb{R}^3$ relative to the robot base and a history of its positions at last 15 time steps (lasting about half a second). By including this information, the policy can not only know the current ball position but learn to estimate its velocity and high order information. Moreover, the last planner output and past 6 time step observed robot states $\hat{q}_m \in \mathbb{R}^{12}$, $\hat{q}_{\psi,\theta,\phi} \in \mathbb{R}^3$ are also contained in the observation. Just like the control policy, we also need to inform the planning policy about the current motion by inputting the motion indicator $\delta \in \mathbb{Z}$.

3) Policy Representation: The planning policy is also represented by a multilayer perceptron which consists of one hidden layer with 256 units followed by one hidden layer with a size of 128, and both of them use $tanh$ activation.

B. Reward

The reward for the planning policy only has one term which is the distance between the ball position o_k and the goal g_k, formulated as

$$r_{p,k} = \begin{cases} 1.0, & \text{if goal is reached} \\ \exp\left(-\rho^g \|o_k - g_k\|_2\right), & \text{otherwise} \end{cases}, \quad (4)$$

where $\rho^g > 0$ is a scaling variable. If the ball has reached the goal (within a 0.2-meter range of the goal), the reward $r_{p,k}$ is always 1. In this way, we can encourage the robot to try its best to shoot to the target.

C. Training Setup

Real life soccer balls are deformable bodies in the shape of a truncated icosahedron, making it challenging to simulate the contact with the ground and the robot leg. In order to tackle the gap between the simulation and real world,

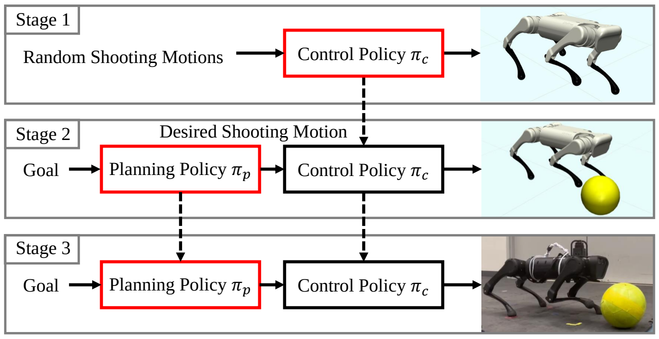


Fig. 4: Multiple stage training scheme. Red bounding box represents the policy being optimized at current stage. Stages 1 and 2 are performed in simulation and Stage 3 is in the real world. In Stage 1, a control policy π_c is firstly trained to track random shooting motions. It also learns to stay robust to the sim2real transfer by domain randomization. After the control policy is ready, a planning policy π_p is pretrained to plan for desired shooting motion for the controller in order to shoot the ball to the goal. Such a planning policy is firstly trained in simulation with a rigid ball in Stage 2 then fine-tuned in the real world to shoot a soft ball in Stage 3.

we adopt a multiple-stage training strategy that includes training in both simulation and the real world, as done in [16]. In the training stage for the planner, as presented in Fig. 4, the policy is firstly pretrained with a simulated rigid spherical ball, and then based on the optimal policy parameters obtained in simulation, this policy keeps training in the real world by collecting data from the interaction with the real deformable and icosahedron-shaped ball.

1) Pretrain in Simulation: We utilize MuJoCo to simulate both the A1 robot and a rigid spherical ball to pretrain the planning policy. During the pretraining stage, each episode lasts around 80 seconds which is the same as the training stage for the control policy. In each episode, the robot is trained to shoot the ball to a random target, and if the ball has not reached the goal after it has stopped or after a given time span, the episode will be terminated to prevent the agent having future return for this shooting failure. If the ball has reached the goal, the ball position will be reset to a random place next to the robot, and the robot will be required to perform the next round of shooting.

Similar to training the control policy for the sim2real transfer in Sec. IV-C, dynamics parameters of the training environment for the planning policy is also randomized using the range demonstrated in Table I in each episode. Besides considering the robot modeling errors and environment changes, we also include the randomness of the simulated ball, such as stiffness, mass, and size. Moreover, to consider uncertainty of the ball detection algorithm through vision in the real world, we further include a simulated Gaussian noise to the detected ball position and delay into the observation.

To optimize the parameters of the planning policy, we use Randomized Ensembled Double Q-Learning (REDQ) [35] due to its sample efficiency, which is important for fine-tuning in the real world.

2) Fine-tuning in Real World: After the training in simulation has converged, the planning policy is then deployed on the quadrupedal robot to learn the shooting skill in the real world. As shown in Fig. 4, the quadruped runs the control policy obtained in Sec. IV which is also the same policy used during pretraining in the simulation. The planning policy is warm started with the parameters obtained in simulation and keeps training using the same REDQ algorithm but with the samples collected in the real world. As shown in Fig. 1, the robot hardware is the A1 robot from Unitree Robotics. The ball position is detected by a RGB-Depth camera alongside the robot, Intel RealSense D435i camera, by color segmentation, and the goal location is marked by an AprilTag [31] which is also detected by this camera.

The training samples are collected when the robot hardware interacts with the soccer ball in the real life environment. The planner's observation introduced in Sec. V-A.2 can be obtained by the robot's onboard sensors and the RGB-Depth camera. The reward used in this stage is the same as the one in pretraining (4) to stimulate the agent to shoot the ball to the detected goal location. During the reset at each episode in the real world, we randomly place the goal location (AprilTag), reset the robot' pose and manually place the ball next to the robot.

VI. Experiments

The performance of the control policy on the A1 robot and shooting accuracy of the planning policy before/after fine-tuning in the real world are demonstrated in the accompanying video (`https://youtu.be/bteipHcJ8BM`) and analyzed below.

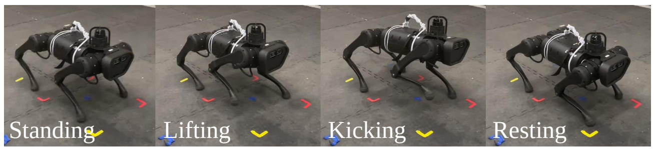
A. Control Performance

We first validate the performance of the control policy on the real robot. During the test, the robot is required to track a random end-effector trajectory using the proposed control policy. As shown in Fig. 5a, our control policy can be transferred and deployed on the real robot without any tuning. Furthermore, using the same control policy, by just changing the desired Bézier parameters and phase time span T_δ, the robot is able to perform different fast kicking motions while maintaining balance. The control policy also shows considerable robustness under random force. As demonstrated in Fig. 5b, the control policy is able to prevent the robot falling over by adjusting its stance legs when we perturb the robot randomly.

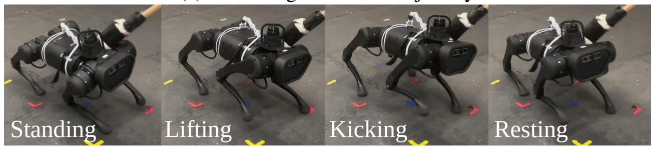
Fig. 5c shows examples of the robot tracking desired end-effector trajectories obtained by the planning policy to shoot a ball to different places. Please note that the given trajectory (drawn as solid line) is not necessary to be dynamically feasible for the quadrupedal robot. Looking at the robot actual end-effector position marked as red points, the control policy shows the capacity to stay close to the given trajectory while respecting to the dynamics limitation of the robot.

B. Planning Performance

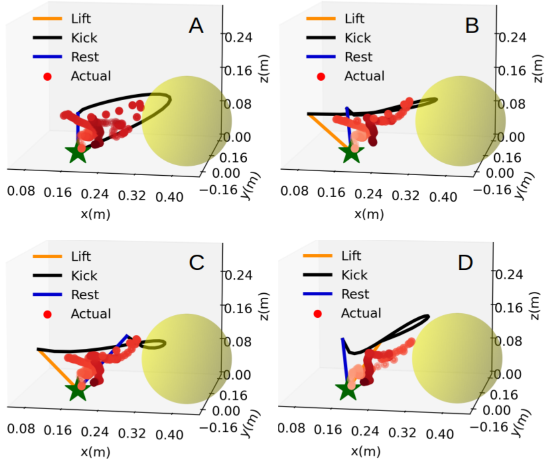
After validating the learned control policy on the robot, as illustrated in Fig. 4, we can reuse this well-tested control policy for training the planning policy. We next conduct experiments with the planning policy, as recorded in Fig. 6.



(a) Following a random trajectory



(b) Following a trajectory with random perturbation

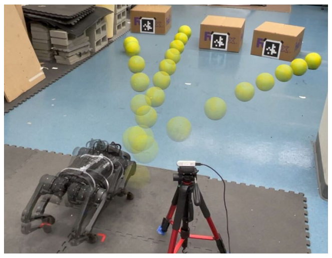


(c) Robot's end-effector (front right toe) tracking a desired trajectory

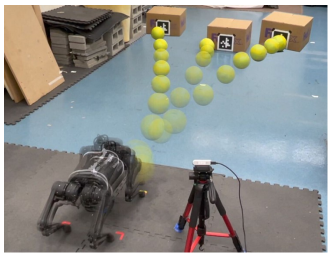
Fig. 5: Performance of the control policy deployed on the robot. (a) The control policy enables the robot to track random end-effector (front right toe) trajectories while maintaining balance, even in the scenario (b) where a random perturbation is applied to robot. (c) shows the desired end-effector trajectory (solid lines) obtained from the planning policy in different motions (marked by different colors) in order to shoot the detected ball (marked as yellow) to A) a goal to robot's left, B,C) a goal in front of the robot, and D) a goal to robot's right. The red dashed line depicts actual robot's end-effector position and the darkness of its point represents elapsed time. The green star is the start and the end of the reference trajectory.

If we directly use the planning policy pretrained in simulation to shoot the soccer ball to a random target in the real world, the ball can barely hit the target, especially for targets that are over 3 meters away, as demonstrated in Fig. 6a. This illustrates the huge sim2real gap due to the hard-to-model soft soccer ball, uncertain rolling friction, and contacts with the deformable ball, even though we randomized the simulated ball dynamics parameters in Table I. After training the planning policy with 1386 samples (23-minute data) in 32 iterations in the real world, the precision of the robot shooting skills increase significantly and the robot is able to shoot the ball to the target which can not be reached before fine-tuning, as shown in Fig. 6b. Please note that for the three locations demonstrated in Figs. 6a,6b, we repeat the shooting experiments to the same locations for three consecutive times, and the ball failed to reach the goal in all of the trials before fine-tuning while the fine-tuned policy enables the soccer ball to reach the goal in 8 out of 9 trials.

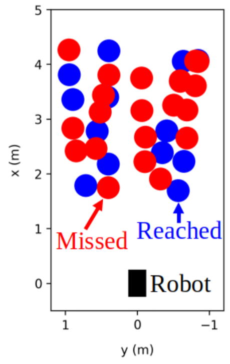
In order to quantitatively analyze the shooting accuracy



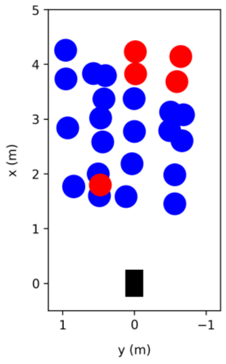
(a) Snapshot before fine-tuning



(b) Snapshot after fine-tuning



(c) Accuracy map before fine-tuning



(d) Accuracy map after fine-tuning

Fig. 6: Performance of the planning policy before (left) and after (right) fine-tuning in the real world. (a,c) The robot cannot precisely shoot the yellow soccer ball to a random target (the AprilTag on the box) if it uses the planning policy right after pretraining in simulation. Such failure is because of the sim2real gap. (b,d) After we keep training the robot in the real world to kick a real soccer ball, the shooting accuracy improves to a large extent and the robot is able to shoot the soccer ball to most of the region on the map.

before and after fine-tuning, we extensively test the planning policy on the robot, and record each trial in Figs. 6c, 6d. From the recorded data, the shooting accuracy before fine-tuning is 40.6% (13 reach in 32 trials), and is boosted after fine-tuning to 80.8% (21 reach in 26 trials) and the robot demonstrates the capacity to shoot the soccer ball to most of region in the 2×4.5 m^2 map. Such improvement showcases the importance of the fine-tuning stage in Fig. 4. However, we note that the resulting planning policy can still not shoot to all the targets, especially for the targets that are far way from the robot as shown in Fig. 6d. Such failures may be due to the limitation of maximum torque of the A1 robot. Interestingly, an emergent behavior shows up in some scenarios, typically when the goal is near to the wall, the robot tends to kick the ball towards the wall and then the ball can bounce off to the goal.

C. Locomotion and Shooting

We also combine both locomotive skills and shooting maneuver into one task where the robot walks to approach the ball away from it's initial position, switches to standing, and shoots it to a random desired location afterwards. Experimental results are demonstrated in the video where we use walking controller developed in [36]. Such experiments showcase the advantages of using quadrupedal robot, which can uses its leg to not only walk but to manipulate the ball, over the robotic arm which has a fixed base.

VII. Conclusion and Future Works

In conclusion, we demonstrate a hierarchical reinforcement learning framework to enable precise soccer shooting skills on quadrupedal robots. In this work, we decompose the soccer shooting problem into two sub-problems: motion control to perform arbitrary shooting motions using one leg while balancing on the others and motion planning to find an optimal motion to shoot the soccer ball to the target. By separating the high-level soccer shooting problem, we are able to firstly focus on developing a robust control policy that enables the robot to track arbitrary shooting motions in the real world and use it for training the motion planning policy. As a real soccer ball is deformable and its contact is hard to simulate, we leverage multi-stage learning to firstly train the planning policy in simulation with a rigid ball and keep training the policy on the real robot shooting a real soccer ball. In experiments, we demonstrate the capacity of the control policy to enable a quadrupedal robot to track a random but fast shooting motion while staying robust to sim2real transfer and random perturbations. After fine-tuning the planning policy in the real world, the robot is able to reliably shoot the soccer ball to random targets with the proposed framework. Note that this work only focuses on the soccer shooting maneuver when the quadrupedal robot is standing. In the future, it will be interesting to extend such a method to combine quadrupedal locomotion and ball manipulation skills to perform more complex soccer skills.

Acknowledgements

This work was supported in part by Hong Kong Centre for Logistics Robotics and in part by NSF Grant CMMI-1944722. The authors would also like to thank Laura Smith and Hongbo Zhang for their gracious help.

References

[1] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, “Gait and trajectory optimization for legged systems through phase-based end-effector parameterization,” *Robot. Automat. Lett.*, 2018.

[2] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, “Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control,” *arXiv preprint arXiv:1909.06586*, 2019.

[3] S. Gilroy, D. Lau, L. Yang, E. Izaguirre, K. Biermayer, A. Xiao, M. Sun, A. Agrawal, J. Zeng, Z. Li *et al.*, “Autonomous navigation for quadrupedal robots with optimized jumping through constrained obstacles,” in *Proc. Int. Conf. Automat. Sci. Eng.*, 2021.

[4] J. Peters and S. Schaal, “Reinforcement learning of motor skills with policy gradients,” *Neural Networks*, 2008.

[5] J. Peters, J. Kober, K. Mülling, O. Krämer, and G. Neumann, “Towards robot skill learning: From simple skills to table tennis,” in *Machine Learning and Knowledge Discovery in Databases*, 2013.

[6] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox, “Multi-task policy search for robotics,” in *Proc. Int. Conf. Robot. Automat.*, 2014.

[7] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman, “Deep predictive policy training using reinforcement learning,” in *Proc. Int. Conf. Intell. Robots Syst.*, 2017.

[8] J. Kober, E. Oztop, and J. Peters, “Reinforcement learning to adjust robot movements to new situations,” in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[9] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, “Tossingbot: Learning to throw arbitrary objects with residual physics,” *Robotics: Science and Systems*, 2019.

[10] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, “Combining model-based and model-free updates for trajectory-centric reinforcement learning,” in *Proc. Int. Conf. Machine Learn.*, 2017.

[11] F. Muratore, T. Gruner, F. Wiese, B. Belousov, M. Gienger, and J. Peters, “Neural posterior domain randomization,” in *Proc. Conf. Robot Learn.*, 2022.

[12] F. Shi, T. Homberger, J. Lee, T. Miki, M. Zhao, F. Farshidian, K. Okada, M. Inaba, and M. Hutter, “Circus anymal: A quadruped learning dexterous manipulation with its limbs,” in *Proc. Int. Conf. Robot. Automat.*, 2021.

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

[14] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, “Learning agile robotic locomotion skills by imitating animals,” *Robotics: Science and Systems*, 2020.

[15] A. Kumar, Z. Fu, D. Pathak, and J. Malik, “Rma: Rapid motor adaptation for legged robots,” *Robotics: Science and Systems*, 2021.

[16] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine, “Legged robots that keep on learning: Fine-tuning locomotion policies in the real world,” in *Proc. Int. Conf. Robot. Automat.*, 2022.

[17] G. Ji, J. Mun, H. Kim, and J. Hwangbo, “Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion,” *Robot. Automat. Lett.*, 2022.

[18] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *Proc. Int. Conf. Robot. Automat.*, 2018.

[19] 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,” in *Proc. Int. Conf. Robot. Automat.*, 2021.

[20] P. Stone, “Intelligent autonomous robotics: A robot soccer case study,” *Synth. Lect. Artif. Intell. Machine Learn.*, 2007.

[21] M. Veloso, W. Uther, M. Fijita, M. Asada, and H. Kitano, “Playing soccer with legged robots,” in *Proc. Int. Conf. Intell. Robots Syst.*, 1998.

[22] S. K. Chalup, C. L. Murch, and M. J. Quinlan, “Machine learning with aibo robots in the four-legged league of robocup,” *Trans. Syst., Man, and Cybern., Part C*, 2007.

[23] A. Cherubini, F. Giannone, L. Iocchi, D. Nardi, and P. F. Palamara, “Policy gradient learning for quadruped soccer robots,” *Robot. Auton. Syst.*, 2010.

[24] M. Friedmann, J. Kiener, S. Petters, D. Thomas, O. Von Stryk, and H. Sakamoto, “Versatile, high-quality motions and behavior control of a humanoid soccer robot,” *Int. J. Human. Robot.*, 2008.

[25] S. Behnke and J. Stückler, “Hierarchical reactive control for humanoid soccer robots,” *Int. J. Human. Robot.*, 2008.

[26] C. A. Acosta-Calderon, R. E. Mohan, C. Zhou, L. Hu, P. K. Yue, and H. Hu, “A modular architecture for humanoid soccer robots with distributed behavior control,” *Int. J. Human. Robot.*, 2008.

[27] N. Jouandeau and V. Hugel, “Optimization of parametrised kicking motion for humanoid soccer player,” in *Proc. Int. Conf. Auton. Robot Syst. Compet.*, 2014.

[28] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, “Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning,” *ACM Transactions on Graphics*, 2017.

[29] H. Teixeira, T. Silva, M. Abreu, and L. P. Reis, “Humanoid robot kick in motion ability for playing robotic soccer,” in *Proc. Int. Conf. Auton. Robot Syst. Compet.*, 2020.

[30] I. J. da Silva, D. H. Perico, T. P. D. Homem, and R. A. da Costa Bianchi, “Deep reinforcement learning for a humanoid robot soccer player,” *J. Intell. Robot. Syst.*, 2021.

[31] E. Olson, “Apriltag: A robust and flexible visual fiducial system,” in *Proc. Int. Conf. Robot. Automat.*, 2011.

[32] J.-w. Choi, R. Curry, and G. Elkaim, “Path planning based on bézier curve for autonomous ground vehicles,” in *World Congress on Engineering and Computer Science*, 2008.

[33] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *Proc. Int. Conf. Intell. Robots Syst.*, 2012.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[35] X. Chen, C. Wang, Z. Zhou, and K. W. Ross, “Randomized ensembled double q-learning: Learning fast without a model,” in *Proc. Int. Conf. Learn. Repres.*, 2021.

[36] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, “Fast and efficient locomotion via learned gait transitions,” in *Proc. Conf. Robot Learn.*, 2021.