Creating a Dynamic Quadrupedal Robotic Goalkeeper with Reinforcement Learning

Xiaoyu Huang[2*], Zhongyu Li[1*], Yanzhen Xiang[1], Yiming Ni[1], Yufeng Chi[1], Yunhao Li[1], Lizhi Yang[1], Xue Bin Peng[3], and Koushil Sreenath[1]

***Abstract*—We present a reinforcement learning (RL) framework that enables quadrupedal robots to perform soccer goalkeeping tasks in the real world. Soccer goalkeeping using quadrupeds is a challenging problem, that combines highly dynamic locomotion with precise and fast non-prehensile object (ball) manipulation. The robot needs to react to and intercept a potentially flying ball using dynamic locomotion maneuvers in a very short amount of time, usually less than one second. In this paper, we propose to address this problem using a hierarchical model-free RL framework. The first component of the framework contains multiple control policies for distinct locomotion skills, which can be used to cover different regions of the goal. Each control policy enables the robot to track random parametric end-effector trajectories while performing one specific locomotion skill, such as jump, dive, and sidestep. These skills are then utilized by the second part of the framework which is a high-level planner to determine a desired skill and end-effector trajectory in order to intercept a ball flying to different regions of the goal. We deploy the proposed framework on a Mini Cheetah quadrupedal robot and demonstrate the effectiveness of our framework for various agile interceptions of a fast-moving ball in the real world.**

I. Introduction

Developing a robotic goalkeeper is an appealing but challenging problem. This task requires the robot to perform highly agile maneuvers such as jumps and dives in order to accurately intercept a fast moving ball in a short amount of time. Solving this problem is attractive because it can offer us solutions to combine dynamic legged locomotion with fast and precise non-prehensile arm manipulation. Recent developments in quadrupedal robots, which allow for more agile and versatile maneuvers, provides a suitable hardware platform for tackling this task. Furthermore, recent advances in model-free reinforcement learning (RL) has shown promising results on developing controllers for dynamic motor skills on quadrupedal robots [1]–[3]. However, previous efforts on applying RL on quadrupedal robots mainly focus on low-level locomotion control, such as tracking a desired walking velocity [3] or mimicking a reference motion [1], without extending the learned locomotion skills to a higher level task, such as precisely intercepting a fast-moving soccer ball using agile maneuvers. This is challenging because it is a combination of highly dynamic locomotion control and accurate non-prehensile manipulation of a fast moving object, each of which is already a difficult task on its own. Therefore, there have been few prior attempts on developing goalkeeping controllers with agile maneuvers using quadrupeds in the real world.

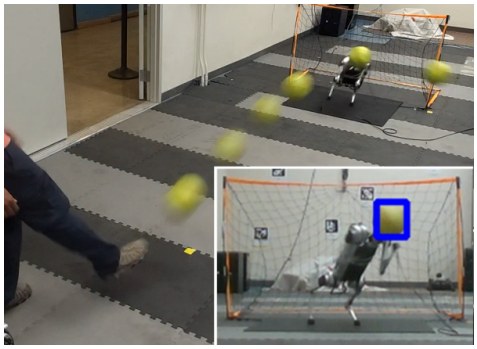


Figure 1: A quadrupedal robot goalkeeper, Mini Cheetah, saves a flying soccer ball towards the goal using the proposed hierarchical RL framework with multiple locomotion control policies and a motion planning policy. The ball flying time is only around 0.5 second. Video is at https://youtu.be/iX6OgG67-ZQ.

In this work, we propose to address the goalkeeping task using a hierarchical model-free RL framework. This framework decomposes the goalkeeping task into two sub-problems: 1) low-level locomotion control to enable the robot to perform various agile and highly-dynamic locomotion skills, and 2) high-level planning to decide an optimal skill and motion to perform in order to intercept the ball.

A. Related Work

The soccer goalkeeping problem using quadrupedal robots can be viewed as a combination of three domains of robotics research: robotic manipulation to intercept a fast moving object, locomotion control to enable a quadruped to perform highly dynamic maneuvers, and the robot soccer.

1) Robotic Catching/Hitting of Fast Moving Objects: Enabling robots to catch or hit fast moving objects, such as a ball, has been studied extensively in the robotic manipulation field. Typically, robotic arms, with a fixed base [4] or a mobile base [5], and quadrocopter [6] are used for these tasks. A common approach to tackling catching tasks is to separate it into two sub-tasks: prediction of the ball's trajectory based on the estimated ball position and velocity using models of the ball's dynamics [4], [7], [8], and generation of a trajectory for the robot's end-effector based on robot's dynamics model [6], [7], [9] or model-free RL [10], [11] to catch the ball at the predicted interception point. An alternative approach [12] is to learn an end-to-end policy in simulation that directly takes the camera's RGB image as input, followed by fine-tuning in the

This work was in part supported by NSF Grant CMMI-1944722.

* Authors contributed equally

[1] University of California, Berkeley, [2] Georgia Institute of Technology, [3] Simon Fraser University. zhongyu_li@berkeley.edu.

real world [13], [14]. However, for quadrupeds, the previous model-based methods which require accurate modeling of the ball and the robot will be hard to utilize due to the complexity of the dynamics models, while previous model-free RL methods have not been applied to control such dynamic legged robot for manipulation tasks.

2) Dynamic Locomotion Control for Quadrupeds: In recent years, there have been considerable advances in legged robot hardware and control algorithms that enable quadrupedal robots to preform highly dynamic locomotion maneuvers, such as jumping [1], [15]–[19] or running [2], [3], [20], in the real world. One approach is to utilize an optimal control framework with the robot's dynamics models, which can be the robot's full-order models and optimized offline [15], [16], [18], or simplified models and deployed online [17], [20]. Another approach is to leverage model-free deep RL to train the quadrupedal robots through trail-and-error in simulation first and then transfer to the real robot [1]–[3], [19]. However, most previous work only focuses on a specific dynamic locomotion skill without attaining a more diverse repertoire of maneuvers based on learned skills to achieve a longer horizon task, such as jumping while tracking different swing leg trajectories to intercept a ball.

3) Legged Robot Soccer: Developing robots that can one day compete with humans in soccer games has been an enduring goal in the robotics community, and a notable soccer robot game is RoboCup [21]. Related to the goalkeeping problem of this work, there are some efforts to develop an intelligent goalkeeper using holonomic wheeled robots [22]–[24]. However, most previous work only consider the robot moving in 2D plane to intercept a ball rolling on the ground at low speeds [22], [23]. Intercepting balls in a 3D and at high speeds, like a flying ball with a speed up to 8 m/s, as in this work, has not been studied in robot soccer. Legged robots, such as humanoid robots and quadrupedal robots, are also used in RoboCup, but most presented soccer skills by legged robots, such as shooting [25], kicking [26], and goalkeeping [27], are based on rule-based motion primitives due to their challenging dynamics. Most recently, by leveraging deep RL, a quadrupedal robot demonstrates the capacity to dribble a soccer ball to a target at a low walking speed [28], and a quadruped is also trained to precisely shoot a soccer ball to a random given target while the robot is standing with a single shooting skill [29]. However, enabling legged robots to play soccer while performing multiple highly dynamic locomotion skills, such as using jump and dive skills, and precise ball manipulation has not yet been demonstrated.

B. Contributions

The core contribution of this work is the creation of an agile and dynamic quadrupedal goalkeeper for robot soccer. This work presents one of the first solutions that combines both highly dynamic locomotion and precise object interception (manipulation) on real quadrupedal robots by using a hierarchical reinforcement learning framework. The proposed method allows quadrupeds to track parametric trajectories with its end-effector(s) while engaging in dynamic locomotion maneuvers. The hierarchical framework is used to learn and compose a diverse set of low-level locomotion skills, and to select the most appropriate skill and motion for the robot to intercept a flying ball. We show that our system can be used to directly transfer dynamic maneuvers and goalkeeping skills learned in simulation to a real quadrupedal robot, with an 87.5% successful interception rate of random shots in the real world. We note that human soccer goalkeepers average around a 69% save rate, [30]. Although, this is against professional players shooting towards regulation sized goals, we hope this paper takes us one step closer to enabling robotic soccer players to compete with humans in the near future.

II. Hierarchical RL Framework for Goalkeeping Task with Multi-Skills

In this section, we introduce the Mini Cheetah robot which is the experimental platform for this work. We also provide a brief overview of the framework for developing goalkeeping skills as illustrated in Fig. 2.

A. The Mini Cheetah Quadrupedal Robot

As shown in Fig. 1, Mini Cheetah [20] is a quadrupedal robot having a weight of 9 kg and height of 0.4 m when it is fully standing. It has 12 actuated motors $q_m \in \mathbb{R}^{12}$ and a 6 degree-of-freedoms (DoFs) floating base, representing its translational $q_{x,y,z}$ (sagittal, lateral, and vertical) positions and orientation $q_{\psi,\theta,\phi}$ (roll, pitch, yaw), respectively.

B. Locomotion Skills for Goalkeeping

Inspired by human goalkeepers, we propose a collection of skills for intercepting a ball flying to different regions of the goal, as illustrated in Fig. 3. The main concern underlying the design of goalkeeping locomotion skills is that the robot needs to react very quickly, since the total timespan of a ball's ballistic trajectory is typically under 1 sec. Therefore, from an initial standing pose in the middle of the goal, the robot needs to perform very dynamic maneuvers to intercept the ball. To accomplish this, our system uses three locomotion skills: *sidestep*, *dive*, and *jump* to cover different goal regions.

1) Sidestep: During a sidestep, the robot takes a quick step in the lateral direction to intercept the ball when it is rolling on the ground or flying toward the goal at a low attitude. Depending on the size of the step, the robot may only need to swing up one of its front leg while the rest can remain in the stance phase. But for larger steps, the stance legs may also need to leave the ground, resulting in a small sideways hop. However, the sidestep skill may not be able to cover regions that are farther away from the robot, such as the lower corners of the goal or the upper regions.

2) Dive: The dive skill is based on quadrupedal jumping behaviors [16], which allows the robot to cover a larger area of the goal. Using the dive skill, the robot should first pitch its body up onto the rear legs, then turn to the lateral side towards the direction that the ball is traveling, extend its two swing legs to reach the ball, and finally land back on its feet. This skills enables the robot to quickly block the lower corners of the goal. During the dive, the rear legs may or may not leave the ground, depending on how far the robot needs to travel.

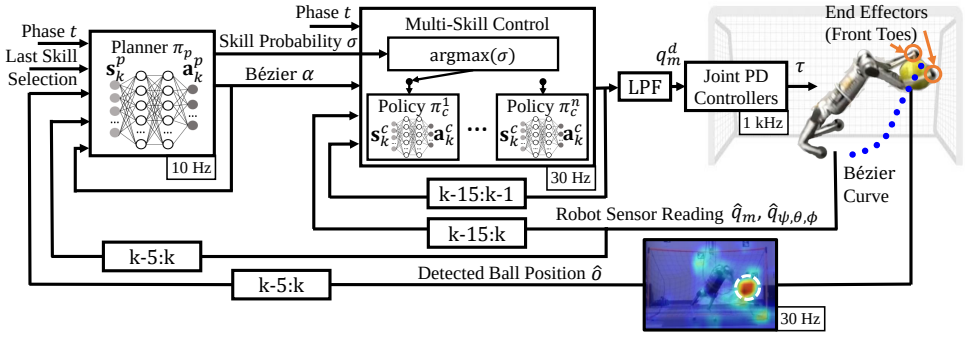


Figure 2: Proposed hierarchical reinforcement learning framework for creating a quadrupedal robotic goalkeeper. We firstly develop a set of locomotion control policies for different skills, such as sidestep, dive, and jump. The locomotion control policies are designed to follow random parametric Bézier curve using the robot end-effectors (swing front toes). The controller outputs desired motor position at 30 Hz for the joint-level PD controller to generate motor torques, after passing through a Low Pass Filter (LPF) [1], [31]. A motion planner running at 10 Hz is developed on top of the multiple skill-specific controllers to select the specific skill to perform as well as the desired end-effector trajectory for the controller to track. The goal of the planner is to enable the robot to intercept the ball via its body before the goal. The controllers and planners are trained by RL and the ball position is detected by a deep neural network using a RGB-Depth camera (30 Hz).

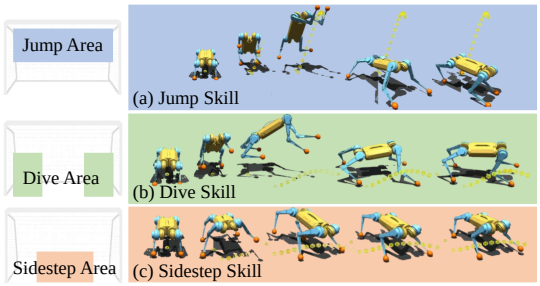


Figure 3: Three different locomotion skills for goalkeeping. The robot can use different skills to cover different regions of the goal.

3) Jump: Similar to the dive skill, the jump skill also requires the robot to pitch up its body and swing its front legs upwards as fast as possible. But for the jump, the robot also needs to extend its swing legs higher to intercept the ball, when it is traveling towards the upper regions of the goal. To perform this dynamic jump, the robot needs to use its rear legs to push off the ground in order to extend its front legs to reach to higher regions. After the ball has been intercepted, the robot needs to reconfigure itself in the air to a more stable landing pose.

For each of these three skills, a nominal reference motion for the Mini Cheetah is manually authored using a $3D$ animation tool [32]. Each reference motion starts in the same default standing pose. RL is then used to train controllers to perform each skill by imitating the respective reference motion.

C. Parameterize Multiple Skills using Bézier Curves

Our robot leverages the above-mentioned three locomotion skills to reach different commanded locations in front of the goal in order to intercept the ball. Inspired by previous work [29], [31], we use Bézier curves to parameterize the desired robot motion. A Bézier curve parameterized by Bézier coefficients α and is defined as a function of phase t, $B_\alpha(t)$ [29, Sec.II-C] where $t \in [0, 1]$ is the normalized time w.r.t. a total duration T of the entire trajectory. Similar to [29], the Bézier curve represents the desired trajectory for the robot's end-effectors, which is designated to be the toes. In the *sidestep* skill, the end-effector is designated to be the toe of the robot's swing leg, either the front right or left toe. However, for the skills that need to use two swing legs to catch the ball, like *jump* and *dive*, the Bézier curve specifies a trajectory for the center of the two end-effectors (two front swing toes). In this work, we choose to use 5 control points in $3D$ space, therefore, $\alpha \in \mathbb{R}^{5\times 3}$. The duration T of each skill is set to be 0.5 seconds. In this way, we can parameterize and represent different robot's end effector trajectory to reach different locations by using different Bézier coefficients.

D. Hierarchical Reinforcement Learning Framework

We now present the proposed hierarchical RL framework for controlling a quadrupedal robotic goalkeeper shown in Fig. 2.

For each goalkeeping locomotion skill, we choose to train a control policy to have joint-level commands for the robot using model-free RL. This enables the robot to mimic the nominal reference motion while tracking a large range of randomized end-effector trajectories represented by parametric Bézier curves. This process produces multiple control policies $(\pi_c^1 \dots \pi_c^n)$ for different goalkeeping skills. In this work, we trained three control policies for the *sidestep*, *dive*, and *jump* skills, and each controller runs at 30 Hz. Since each skill performs a distinct behavior, training individual skill-specific control policies avoids the challenge of training a single multi-task policy for all skills. To select the appropriate skill for different scenarios representing the ball traveling towards different regions of the goal, we train a high-level planning policy π_p using model-free RL to select a desired skill depending on the detected ball position and the robot's current states. The planning policy operates at 10 Hz. It also specifies the desired Bézier curve for the chosen controller to track in order to intercept the ball. The ball position is obtained by an external camera and a deep supervised learning algorithm YOLO [33], without building a ball state estimator.

We first train each control policy in simulation, and test them extensively on the real robot with zero-shot transfer. After obtaining reliable controllers in the real world, we then use the same control policies to train the planning policy in simulation and then directly deploy the entire pipeline in the real world. This approach decouples the complex goalkeeping

task into the locomotion control and manipulation planning problems, and solves each problem separately.

III. Dynamic Locomotion Control using Deep RL

In this section, we detail our framework for training control policies for each goalkeeping skill.

A. Training Environment

The control policies are trained using a simulation of the Mini Cheetah in Isaac Gym [34], a GPU-accelerated dynamics simulator. For a skill-specific controller c, at each timestep k of an episode, the robot takes an action $\mathbf{a}_k^c$ based on its current observations $\mathbf{s}_k^c$, and the environment transits to the next state and provides a reward $r_{c,k}$. The RL is used to maximize the expected accumulated reward over the course of an episode.

1) Action Space: The skill-specific control policy outputs a 12-dimensional action that specifies target motor positions for each joint q_m^d and is used by joint-level PD controllers to compute motor torques τ.

2) Observation Space: As shown in Fig. 2, the policy's observations consist of three components. The first component is the desired end-effector trajectory for the robot to track, represented by a set of Bézier coefficients and normalized phase t w.r.t. the 0.5-sec-timespan of the motion. The second component is the robot's raw sensor reading, which consists of the measured motor positions $\hat{q}_m$ and base rotation $\hat{q}_{\psi,\theta,\phi}$. We also include a history of the sensor readings in the past 15 timesteps, which corresponds to a time window of around 0.5 seconds. The last component is the robot's actions (q_m^d) in the past 15 timesteps. Incorporating a history of the robot's actions and sensor readings provides the policy with some information necessary for inferring the dynamics of the system, which can be vital for sim-to-real transfer [31].

B. Reward Formulation

The reward $r_{c,k}$ at timestep k is designed to encourage the control policy to track the given Bézier curve for the robot's end-effector, while smoothly following the skill-specific reference motion and maintaining gait stability. The reward function contains three main components:

$$r_{c,k} = 0.5r_{c,k}^E + 0.3r_{c,k}^I + 0.2r_{c,k}^S, \tag{1}$$

where $r_{c,k}^E$ represents the robot's end-effector tracking term, $r_{c,k}^I$ is the imitation term, and $r_{c,k}^S$ is a smoothing term. The end-effector tracking term has the highest weight while the smoothing term has the lowest, which is to put higher priority on tracking the desired Bézier curve. Next, we define an abstract reward function

$$r(u, v) = \exp[\rho||u - v||_2^2], \tag{2}$$

which calculates the normalized distance between vector u and v, with $\rho > 0$ being a hyperparameter. With this, the $r_{c,k}^E$ term can be decomposed into:

$$r_{c,k}^E = 0.8r(B_\alpha(t), x_{e,k}) + 0.2r(\dot{B}_\alpha(t), \dot{x}_{e,k}), \tag{3}$$

where $B_\alpha(t)$ is the desired end-effector position evaluated at phase t at current timestep k and $\dot{B}_\alpha(t)$ is the derivative of the curve at phase t which represents the desired end-effector velocity. The $x_{e,k}$ is robot's end-effector position in Cartesian space while $\dot{x}_{e,k}$ is its velocity. We empirically found that adding the end-effector velocity tracking term can improve the smoothness of the end-effector trajectory.

Similarly, the imitation reward $r_{c,t}^I$ encourages the robot to mimic the skill-specific reference motion, and consists of 3 terms: the motor tracking reward $r(q_m, q_m^r)$, robot base height following reward $r(q_z, q_z^r)$, and base orientation tracing reward $r(q_{\psi,\theta,\phi}, q_{\psi,\theta,\phi}^r)$, by calculating the distance between robot's current values and the one from reference motion ($q_{m,z,\psi,\theta,\phi}^r$) at each timestep. Similarly, $r_{c,k}^S$ is the smoothing reward and consists of $r(\dot{q}_{\psi,\theta,\phi}, 0)$, $r(\ddot{q}_{\psi,\theta,\phi}, 0)$, and $r(\tau, 0)$ to minimize the robot base rotational velocity, rotational acceleration, and torque consumption, respectively. By incorporating the smoothing reward, we can effectively minimize nonessential angular movement that hampers robot stability, and energy consumption throughout the motion.

C. Motion and Dynamics Randomization

With the skill-specific nominal reference motion unchanged, the desired end-effector trajectory is randomized by adding a random change of Bézier coefficients $\tilde{\alpha}$ to a nominal set of Bézier coefficients $\bar{\alpha}$, *i.e.*, $\alpha = \bar{\alpha} + \tilde{\alpha}$. The coefficients change $\tilde{\alpha}$ of each control point is uniformly sampled from a large range, especially in the lateral direction to cover farther sides of the goal. This then allows each policy to track a large variety of end-effector trajectories defined by the randomized α.

During training in simulation, dynamics parameters are randomized to facilitate transfer from simulation to the real world. Similar to previous work [29], we randomize the robot's link mass, inertia, and center of mass in a given range to mitigate modeling error. We also randomize the PD gains of the motors to accommodate modeling errors in the motor dynamics. We also simulate sensor noise and delay using a similar randomization range as [29]. Ground friction plays a critical role during sim-to-real transfer for the jump and dive skills where the robot needs to use its rear legs to push off the ground. Therefore, the friction coefficient is randomized within a large range of $[0.5, 4.5]$. The ground restitution is also randomized from 0 to 0.5 for better adaptation to soft padded ground. Additionally, a random 6 DoFs wrench (force varying from -1 to 1 N and torque is from -3 to 3 Nm) is applied on the robot base to perturb the robot and improve the robustness of the policy. Specifically, we found that a large perturbation in roll direction, ranging from -12.5 Nm to 12.5 Nm, helps to prevent the robot from rolling over when jumping sideways.

D. Episode Design and Training

Each episode lasts for 300 timesteps, corresponding to 10 seconds and consists of three stages. In the first stage, the robot starts in a nominal standing pose. After a random span of time has elapsed, a set of desired Bézier coefficients is randomized and given to the policy. The robot is then supposed to follow the desired end-effector trajectory, while also imitating the reference motion of the specific skill. After the desired goalkeeping maneuver has been completed, the

reference motion switches to a nominal standing motion to encourage the robot to recover back to the nominal pose.

Two early termination conditions are applied during training. First, the episode is terminated whenever there is an unsafe behavior, like the robot falling over or one leg colliding with another part of the body, resulting in zero reward for all future timesteps. Secondly, a large deviation from the commanded curve for the end-effectors also results in termination of the episode. This criteria further encourages the robot's end-effector to stay as close as possible to the desired trajectory. We found that early termination leads to faster training and more precise tracking of different end-effector trajectories.

All policies are trained with Proximal Policy Optimization (PPO) [35]. The actor and critic networks are modeled by separate MLPs with hidden layers consisting of 512, 256, 128 units, with ELU activation functions.

IV. Multi-Skill Motion Planning using Deep RL

Each obtained skill-specific locomotion control policy is first tested thoroughly on the robot hardware. After the sim-to-real test, these control policies can then be used to train a planning policy that enables the robot to intercept the ball by performing dynamic locomotion maneuvers after examining the detected ball position and the current robot states.

A. Training Environment

The planning policy is also trained in Isaac Gym with the Mini Cheetah driven by its controller and a rigid ball. The goal in the simulation is sized 1.5 m wide and 0.9 m high, and the robot is placed 0.2 m in front of the goal line.

1) Action Space: As shown in Fig. 2, the planning policy outputs the desired skill type to perform, and the desired end-effector trajectory for the selected controller to track. Therefore, the action of the planner $\mathbf{a}_p$ contains two parts: the skill type and desired Bézier coefficients α^d. For n skills, the planner outputs skill selection probabilities $\sigma \in \mathbb{R}^{n+1}$, and the desired skill type can be determined by finding the argmax of σ. The extra skill utilizes a time-invariant standing pose with a PD controller. In this work, $n = 3$. Although there are 5 control points to construct the Bézier curve we used in the control policy, the planner outputs only 4 of them, excluding the first control point, *i.e.*, $\alpha^d \in \mathbb{R}^{4\times 3}$. The first control point is always overwritten by a nominal initial position of the robot's end-effector to reduce the learning complexity.

2) Observation Space: The observation of the planning policy $\mathbf{s}_k^p$ consists of four components. The current (at timestep k) detected ball position in global frame $\hat{o}_k$. Instead of developing ball position estimation and prediction system, a history of the ball positions in the last 6 timesteps is given to the planning policy. This is helpful for the planning policy to filter out the noisy readings from the camera in the real world, to estimate its velocity and high order information, and to infer the ball's future trajectory, implicitly. The second component is the current robot sensor reading, including motor angles $\hat{q}_m$ and base orientation $\hat{q}_{\psi,\theta,\phi}$, to enable the policy to be aware of robot's current states. Similar to the control policy, a history (6 timesteps) of the robot feedback and actions is provided to the planning policy to implicitly learn to estimate dynamics. Please note that, although the planner runs at 10 Hz, all the feedback history is sampled at 30 Hz which is the higher frequency that the controller uses. Moreover, the last planner action, which is the previous skill selection and desired Bézier coefficients are also input to the policy. By informing the planning policy the previous skill and motion selection, together with the robot current states, planner can learn to avoid making sudden but infeasible changes of the skills, such as commanding the robot to perform sidestep skill while the robot is in the air with a previous selected jump skill. Finally, the phase t of the performed skill is also included.

B. Reward Formulation

The reward of the planning policy is designed to perform a "save" by intercepting the flying ball with the robot's body, robot's end-effectors or trunk. To facilitate this, the reward is formulated as:

$$r_{p,k} = b_k(r^{\dot{o}}_{p,k} + 0.6r^{x_e}_{p,k} + 0.2r^{x^d_e}_{p,k} + 0.2r^{\alpha^d}_{p,k}), \tag{4}$$

where

$$b_k = \begin{cases} 1, & \text{if } ||o_k - x_{e,k}|| \leq 0.3 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

is a binary variable indicating if the current ball position o_k is close enough to robot's current end-effector position $x_{e,k}$. In this way, we only calculate the reward when the ball is close to the robot. Since the robot is only allowed to perform one save in each episode, by such a sparse reward design, the robot is encouraged to just stand still without performing other goalkeeping skills until the ball is nearby for a better return over the entire episode.

The dominating reward term $r^{\dot{o}}_{p,k}$ represents the reward for the ball velocity. It is set to 1 if the ball speed $||\dot{o}||_2$ is zero and to 0 otherwise. By this term, we encourage the robot to stop the ball. Furthermore, the reward $r^{x_e}_{p,k}$ stimulates the robot to minimize the distance between the robot's end-effector position x_e to the ball by $r(x_{e,k}, o_k)$ where $r(\cdot,\cdot)$ is defined via (2). Similarly, $r^{x^d_e}_{p,k}$ incentivizes the planning policy to enable the desired end-effector trajectory to track the ball by $r(B_{\alpha^d}(t), o_k)$. Please note that, since the dominating term is to stop the ball ($r^{\dot{o}}_{p,k}$), the robot can also receive reasonably good reward if it uses other parts of the body other than the end-effectors, such as the trunk, to save the ball. Finally, the smoothing term $r^{\alpha^d}_{p,k} = r(\alpha^d, 0)$ is introduced to encourage the planner to regularize the desired Bézier coefficients α^d to prevent having fluctuating curves.

Please note that we did not directly include a reward to minimize the energy consumption of the robot. The reason is that the motor torques have a much larger variance between skills than within skills, and thus, the robot may easily learn to keep using a skill with low energy consumption, such as sidestep, but overlook the ball saving task.

C. Early Termination Condition

Besides the reward design, the termination conditions to end the episode earlier are critical to enable the robot to save the

ball in front of the soccer goal. We terminate the episode to prevent the agent from having a future return if the ball flies into the goal area. This can stimulate the robot to try its best to save the ball instead of adopting conservative behaviors like just standing where the robot can still receive some rewards such as the smoothing reward. Further, the episode will be terminated if the robot falls over. This can prevent the planning policy from outputting infeasible end-effector trajectories or commanding wrong skills, like selecting a sidestep skill while the robot is in the air, because these will cause the locomotion controllers to fail.

D. Episode Design, Domain Randomization, and Training

Each episode lasts for 90 timesteps (3 seconds) in total. Upon reset, the ball's initial $3D$ position and velocity in the transverse plane is randomly sampled. The target position for the ball is sampled within the goal area, and the initial vertical speed is obtained accordingly.

We again leverage domain randomization to improve the robustness of the policy. In addition to those during training the control policy, we also applied Gaussian noise with zero mean and 0.05 m standard deviation and constant delay randomized from 80 to 100 ms for the perceived ball positions. The constant delay is key to succeed in the sim-to-real transfer because there is substantial delay mainly from the camera.

The planning policy has separated and identical actor and critic networks that consist of a 2-layer MLP with 256 and 128 hidden dimensions with ELU as the activation function. The last layer of the actor network is followed by two action heads, one for continuous Bezier coefficients, and one for categorical skill selection action. A $tanh$ and a $softmax$ activation are appended to the continuous and discrete heads respectively. The planning policy is also optimized by PPO. We found that the planning policy trained in simulation can be directly deployed in the real world without finetuning with a soft ball unlike what was done in [29].

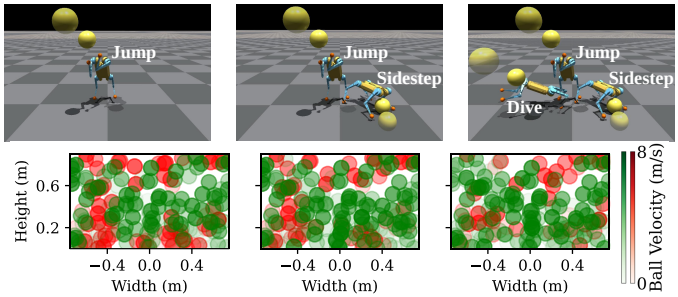
V. Results

After having developed all components of the proposed framework (Fig. 2), we now validate the proposed framework in simulation and experiments. Experiments are also recorded in the accompanying video (https://youtu.be/iX6OgG67-ZQ).

A. Simulation Validation

We firstly evaluate the performance of the proposed multi-skill framework in simulation. Specifically, we compare the ball saving rate using different number of locomotion skills. They are: *jump-only*, *jump-sidestep*, *jump-dive-sidestep* (ours). We trained three planners using the same method introduced in Sec. IV for these different combination of skills, respectively. We denote these planners as *1-skill*, *2-skill*, and *3-skill* (ours) planner accordingly. Each method is repeated 200 times with randomized scenarios as specified in Sec. IV-D.

As shown in Fig. 4, comparing with the planners with more than one skill, the *1-skill* planner achieves a comparable saving rate with flying balls, but misses almost all of the ground rolling balls, resulting in 65.09% saving rate. Such a result demonstrates that using a single skill is not sufficient for the goalkeeping task. While the *2-skill* planner can catch most of the balls (72.46%), however, there are two notable corner cases on the lower left and right that hampers the performance. We discovered that in these cases a majority of the balls travel underneath the robot when the jumping skill is activated, and can not be reached by the robot when it steps to the side. This problem highlights the necessity of dive skill to save the ball flying to these regions. Considering the dive skill, the *3-skill* planner shows the best saving rate of 78.11%.



(a) 1-Skill Planner (b) 2-Skill Planner (c) 3-Skill Planner

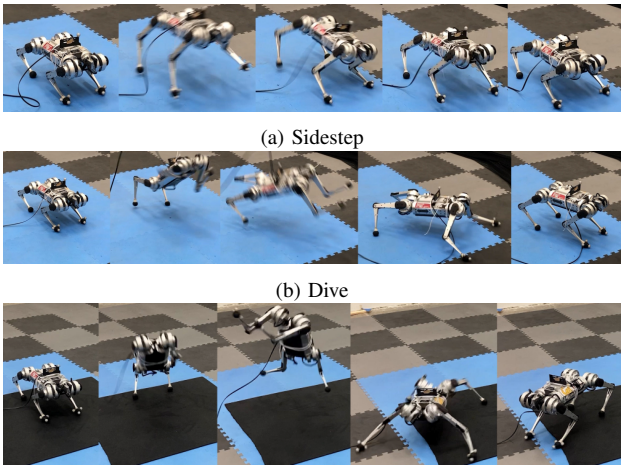
Figure 4: Snapshots and shot interception map in simulation with more skills added from left to right. The map represents the goal region. Green records a goal save while red is a goal (miss). Darker colors indicates faster ball speeds. The snapshots visualize how the planner leverages the new skills, and the shot interception map quantitatively illustrate the benefits of adding each skill. Note that the failing corner cases are noticeably reduced by adding the second sidestep skill in 4b, and further reduced by the third dive skill in 4c. The goal saving rates are 65.09%, 72.46%, and 78.11%, respectively.

B. Experiments

We now deploy the proposed framework on the Mini Cheetah robot to save soccer goals in the real world.

1) Experiment Setup: We set up a mini penalty field to conduct the experiments, as shown in Fig 1, with a $1.5m \times 0.9m$ goal. The robot is placed at the center with its rear feet 0.1 m in front of the goal line. The soccer ball, which is size 3, is either kicked or thrown roughly 4 m in front of the robot with a random initial speed towards a random target in the goal. We set an external RGB-Depth camera (Intel RealSense D435i) placed 6 m away from the goal line. We also setup a Motion Caption System (MoCap) with markers on the robot's trunk, front toes, and soccer ball, to evaluate the tracking performance of the locomotion controllers. Please note that our system does not require accurate measurement from the external MoCap.

2) Performance of Skill-specific Locomotion Controllers: The performance of the low-level controllers on the robot hardware is firstly validated, as demonstrated in Fig. 5. The control policies are able to produce similar maneuvers in real world (Fig. 5) as in the simulation (Fig. 3) without finetuning. Furthermore, given a random set of Bézier coefficients, all three policies are able to track the desired trajectories for the robot's end-effector to the best of its physical limitations, as shown in Fig. 6, with an average tracking error of 0.11 m (measured by MoCap) over all trials.



(a) Sidestep

(b) Dive

(c) Jump

Figure 5: Experiments with control policies for different skills. The policy is able to directly transfer to the hardware. As designed in Sec. II-B, we can observe that the dive skill 5b is able to reach a significantly larger range horizontally than sidestep 5a, while the jump skill 5c can produce a notable period of flight time, swing the front legs to cover more upper-altitude area, and land safely.

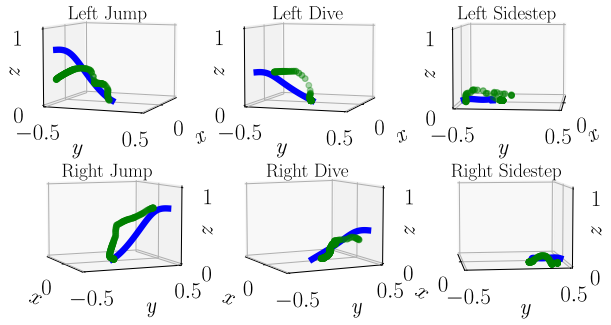
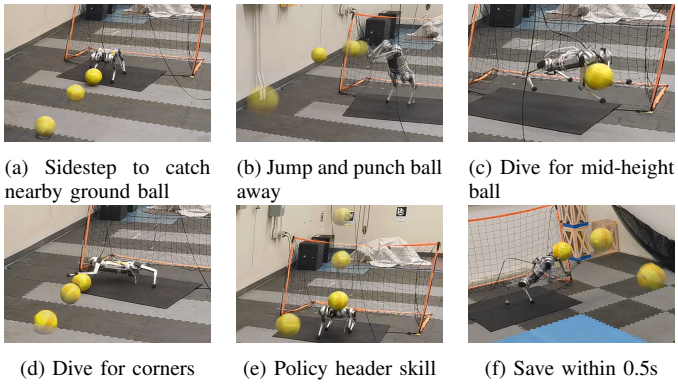


Figure 6: Comparision between robot's actual end-effector trajectory (green) and desired one (blue). The actual trajectory of robot's end-effector is obtained by MoCap while the desired one is randomly specified. The average tracking error for left jumping, left diving, and left sidestepping are 0.10, 0.15, 0.08 m, and those for right sides are 0.12, 0.13, 0.05 m, respectively.

3) Goalkeeping Performance: As demonstrated in Fig. 7, the proposed framework that utilizes three different goalkeeping skills is able to enable Mini Cheetah to save goals in different scenarios. For easier ones (Fig 7a), the most energy-efficient way (taking a sidestep) is leveraged, while in harder cases such as in Fig 7b,7f, the robot takes a large jump and punches out the ball in the air intentionally. In most shots, the soccer ball interception time is within 0.9 second and the robot is able to quickly react to it. Note that another advantage of our planner is that it may leverage the existing skills to infer other skills, such as the header in Fig. 7e, which prevents the ball from slipping through its feet.

To further evaluate the performance in the goalkeeping task using the proposed framework, we conducted extensive ablation study on three methods: 1) a *model-based* planner, 2) the *2-skill* planner with jump and sidestep skills, and the proposed planner which utilizes *3-skill*. The *model-based* planner runs an optimization online to determine the desired Bézier coefficient. Like most prior work using model-based methods [4], we also develop a Kalman Filter to estimate ball



(a) Sidestep to catch nearby ground ball

(b) Jump and punch ball away

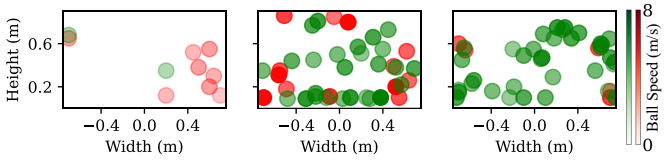
(c) Dive for mid-height ball

(d) Dive for corners

(e) Policy header skill

(f) Save within 0.5s

Figure 7: Snapshots of the real-world experiments showing the robot goalkeeper Mini Cheetah handling various scenarios. In 7a, the robot chooses sidestep when the ball is nearby, whereas in 7d a dive save is selected as the ball is rolling towards the corner. When the ball comes high as in 7b, the robot jumps and intentionally pushes the ball away, while dive is chosen for balls in the lower half, as in 7c. Shown in 7e, the planner generalizes to leverage other parts of the body to complete the task. These experiments are conducted with a RGB-Depth camera, while the robot responded to a fast ball in less than 0.5 second supported by MoCap in 7f.



(a) Model-based (b) 2-Skill (c) 3-Skill

Figure 8: Shot interception maps in real-world experiments. Model-based method yields a very low saving rate due to imprecise ball prediction. The lower-corner failures in 8c are significantly reduced compared to 8b, coherent with the result in simulation. The saving rates are 20.0%, 66.7%, and 87.5%, respectively. These experiments are conducted using a RGB-Depth camera.

velocity from measurements of the ball position and dynamical model of the ball (assuming there is only gravitational force acting on the ball). The planner is designed to find a Bézier curve that can intercept the ball along the ball's predicted path and the locomotion skill is selected based on the ball height at the predicted interception point. The shot interception map using the *model-based* planner, the *2-skill* planner, and the *3-skill* planner (proposed) are recorded in Fig. 8.

The *model-based* planner results in the worst performance with only a $2/10$ saving rate. This is because the state estimation of the ball velocity is not reliable while the model-based planner relies on an accurate predicted interception point. On the other hand, the learning-based planners, which are *2-skill* and *3-skill* planners, do not require knowing the velocity of the ball, and as a result, they demonstrate big jump in saving rates. Both *2-skill* and *3-skill* planners are tested consecutively for 40 trials, and the shot interception maps are shown in Figs. 8b,8c. The most frequent failure spots for the *2-skill* planner occurs noticeably on the lower corners, which is innately difficult for the robot without learning the dive skill. In contrast, the proposed *3-skill* planner (87.5% saving rate) with all three skills noticeably alleviate these corner cases and outperform the *2-skill* planner by 20.9% in the experiment.

However, the limitation of the proposed framework is that

the robot usually fails to save the ball whose flying time is less than 0.5 s, considering that the entire robot motion's timespan is 0.5 s and ball detection is delayed from the camera.

4) Penalty Kicks with Humans and a Quadrupedal Robot: We further showcase the capacity of the proposed goalkeeping framework by inviting human soccer players to conduct penalty shots with the robot goalkeeper. Penalty kicks between a quadrupedal robot soccer ball shooter developed in [29] and our goalkeeper are also demonstrated. These experiments are recorded in the video.

VI. Conclusion and Future Work

In conclusion, we proposed a multi-skill reinforcement learning framework that enables quadrupedal robots to function as soccer goalkeepers with precise and highly dynamic maneuvers. We developed a RL-based framework in simulation and demonstrated its performance with zero-shot transfer to the real world. The framework consists of multiple locomotion controllers specialized in specific skills (sidestep, dive, and jump) and a multi-skill manipulation planner to find the optimal skill and desired trajectory for robot's end-effector to intercept the incoming ball. We showcase that the multi-skill RL framework significantly outperformed a model-based planner, and was able to adequately leverage the speciality of each skill. In this work, we focused solely on the goalkeeping task, but the proposed framework can be extended to other scenarios, such as multi-skill soccer ball kicking.

Acknowledgements

We thank Prof. S. Kim, the MIT Biomimetic Robotics Lab, and NAVER LABS for lending the Mini Cheetah. We also thank Prof. M. Mueller for use of the motion capture space and P. Kotaru and J. Chen for the help with the experiments.

References

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

[2] G. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," in *Robotics: Science and Systems*, 2022.

[3] 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," *IEEE Robot. Automat. Lett.*, pp. 4630–4637, 2022.

[4] Y. Gao, J. Tebbe, and A. Zell, "Optimal stroke learning with policy gradient approach for robotic table tennis," *arXiv preprint arXiv:2109.03100*, 2021.

[5] K. Dong, K. Pereida, F. Shkurti, and A. P. Schoellig, "Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning," in *Proc. Int. Conf. Intell. Robots Syst.*, 2020.

[6] R. Ritz, M. W. Müller, M. Hehn, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *Proc. Int. Conf. Intell. Robots and Syst.*, 2012, pp. 4972–4978.

[7] O. Koç, G. Maeda, and J. Peters, "Online optimal trajectory generation for robot table tennis," *Robot. Auton. Syst.*, vol. 105, pp. 121–137, 2018.

[8] Y. Huang, D. Büchler, O. Koç, B. Schölkopf, and J. Peters, "Jointly learning trajectory generation and hitting point prediction in robot table tennis," in *Proc. Int. Conf. Human. Robots*, 2016, pp. 650–655.

[9] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *Proc. Int. Conf. Robot. Automat.*, 2011, pp. 3719–3726.

[10] J. Tebbe, L. Krauch, Y. Gao, and A. Zell, "Sample-efficient reinforcement learning in robotic table tennis," in *Proc. Int. Conf. Robot. Automat.*, 2021, pp. 4171–4178.

[11] L. Yang, H. Zhang, X. Zhu, and X. Sheng, "Ball motion control in the table tennis robot system using time-series deep reinforcement learning," *IEEE Access*, vol. 9, pp. 99 816–99 827, 2021.

[12] W. Gao, L. Graesser, K. Choromanski, X. Song, N. Lazic, P. Sanketi, V. Sindhwani, and N. Jaitly, "Robotic table tennis with model-free reinforcement learning," in *Proc. Int. Conf. Intell. Robots Syst.*, 2020.

[13] S. Abeyruwan, L. Graesser, D. B. D'Ambrosio, A. Singh, A. Shankar, A. Bewley, and P. R. Sanketi, "i-sim2real: Reinforcement learning of robotic policies in tight human-robot interaction loops," *arXiv preprint arXiv:2207.06572*, 2022.

[14] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters, "Learning to play table tennis from scratch using muscular robots," *IEEE Trans. Robot.*, 2022.

[15] Q. Nguyen, M. J. Powell, B. Katz, J. Di Carlo, and S. Kim, "Optimized jumping on the mit cheetah 3 robot," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 7448–7454.

[16] 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, pp. 2132–2139.

[17] H.-W. Park, P. M. Wensing, and S. Kim, "Jumping over obstacles with mit cheetah 2," *Robot. Auton. Syst.*, vol. 136, p. 103703, 2021.

[18] C. Nguyen and Q. Nguyen, "Contact-timing and trajectory optimization for 3d jumping on quadruped robots," in *Proc. Int. Conf. Intell. Robots and Syst.*, 2022.

[19] M. Bogdanovic, M. Khadiv, and L. Righetti, "Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization," *Frontiers in Robotics and AI*, vol. 9, 2022.

[20] 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.

[21] M. Veloso and P. Stone, "Video: Robocup robot soccer history 1997–2011," in *Proc. Int. Conf. Intell. Robots Syst.*, 2012, pp. 5452–5453.

[22] H. Lausen, J. Nielsen, M. Nielsen, and P. Lima, "Model and behavior-based robotic goalkeeper," in *Robot Soccer World Cup*, 2003.

[23] J. Cunha, N. Lau, and J. Rodrigues, "Ball interception behaviour in robotic soccer," in *Robot Soccer World Cup*, 2011, pp. 114–125.

[24] P. Cooksey, J. P. Mendoza, and M. Veloso, "Opponent-aware ball-manipulation skills for an autonomous soccer robot," in *Robot World Cup*, 2016, pp. 84–96.

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

[26] 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, pp. 34–39.

[27] J. G. Masterjohn, M. Polceanu, J. Jarrett, A. Seekircher, C. Buche, and U. Visser, "Regression and mental models for decision making on robotic biped goalkeepers," in *Robot Soccer World Cup*, 2015.

[28] S. Bohez, S. Tunyasuvunakool, P. Brakel, F. Sadeghi, L. Hasenclever, Y. Tassa, E. Parisotto, J. Humplik, T. Haarnoja, R. Hafner *et al.*, "Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors," *arXiv preprint arXiv:2203.17138*, 2022.

[29] Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, and K. Sreenath, "Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot," in *Proc. Int. Conf. Intell. Robots Syst.*, 2022.

[30] "A deep dive into the goalkeepers' stats," https://soccerment.com/the-vitruvian-goalie-a-deep-dive-into-the-goalkeepers-stats/, accessed: 2022-10-07.

[31] 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.

[32] Z. Li, C. Cummings, and K. Sreenath, "Animated cassie: A dynamic relatable robotic character," in *Proc. Int. Conf. Intell. Robots Syst.*, 2020.

[33] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[34] 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.

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