# Bipedal Locomotion for Football

Simon Giegerich

***Abstract*— Playing football with humanoid robots is a highly complex task. Among other challenges, it involves maintaining balance on two legs, interacting with the ball, cooperating with other robots, and quickly changing direction. In this work, we study three recent works that use Reinforcement Learning to learn a hierarchical policy that enables humanoid robots to play football. They all rely on hierarchical policies consisting of low-, mid-, and high-level skills. The development of low-level skills involves learning basic movements that are essential for football, such as walking or standing up. These basic movements are applied to learn football-specific skills, such as dribbling and shooting. Finally, these skills are further reused to enable strategic and goal-oriented football play as high-level skills. The policies can subsequently be transferred to a real robot zero-shot. Our analysis examines the similarities and differences in the methods used across the three studied papers for each skill level within the hierarchical policy.**

## I. INTRODUCTION

Humanoid robots can perform a wide range of tasks. Thanks to their structure, they are perfectly suited to assist humans in their daily work. However, maintaining balance on two legs, interacting with a range of objects, or cooperating with other robots and humans are just some of the many challenges that must be overcome. Playing football combines these challenges, among others, such as interacting with objects while maintaining balance, and quickly changing direction, into a single task.

In recent years, several studies have investigated the use of Reinforcement Learning (RL) to enable humanoid robots to play football [1][2][3]. However, when using RL for playing football, it is important to distinguish between the short-term goals, such as the movement of the robot's joints to perform a kick, and the long-term goals, such as scoring goals to win matches. Therefore, Hierarchical Reinforcement Learning (HRL) can be used to abstract these individual subtasks from one another. In this report, we will analyze three papers [1][2][3] that use HRL to teach humanoid robots to play football, as shown in Figure 1.

In the following section, we will introduce the concepts behind RL and hierarchical policies. We will then compare the works on how they each use HRL to teach humanoid robots to play football. The methods for transferring their approaches to the real robot without further training are shown in Section IV. In Section V we will discuss the limits of each work and point out possible future work for achieving more general humanoid robot players. Finally, this work will be concluded in Section VI.

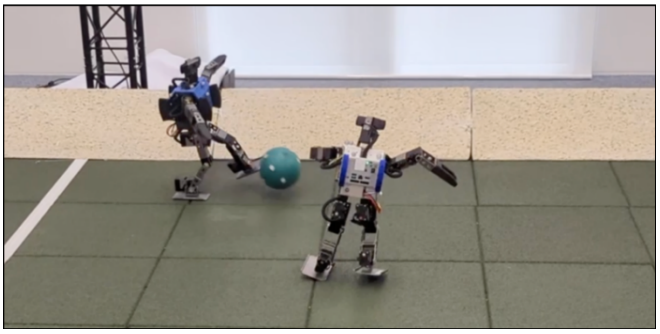


Fig. 1. Two humanoid robots playing football against each other. Their hierarchal policy is trained in simulation using RL and transferred to the robot zero-shot [3].

## II. PRELIMINARIES

### A. Reinforcement Learning

RL is a learning approach in which an agent interacts with an environment to achieve a goal. A central concept of RL is the Markov Decision Process (MDP), which consists of the following components:

- A set of states $\mathcal{S}$
- A set of actions $\mathcal{A}$
- A transition function $\mathcal{P}(s_{t+1}|s_t, a_t)$ that defines the probability of transitioning to state $s_{t+1}$ given that the agent takes action $a_t$ in state $s_t$
- A reward function $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$ that defines the reward $r_t$ that the agent receives when transitioning from state $s_t$ to state $s_{t+1}$ by taking action $a_t$.
- A discount factor $\gamma$ that impacts the importance of future rewards compared to immediate rewards.

The agent's goal in RL is to find a policy $\pi$ that maximizes the expected return

$$\mathcal{J}(\pi) = \mathbb{E}_{\tau\sim\pi}[R(\tau)] \tag{1}$$

where $R(\tau)$ is the discounted return

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{2}$$

of trajectory $\tau = (s_0, a_0, s_1, a_1, ...)$ following policy $\pi$.

### B. Hierarchical Policies

One of the most fundamental problems of RL is Bellman's "curse of dimensionality" [4], which describes the exponential growth of a problem with an increasing size of the state space. Vanilla RL methods work on a single level of abstraction to solve problems as a whole. However, they are unsuitable for very complex problems that consist of several

| | Liu et al. [1] | Bohez et al. [2] | Haarnoja et al. [3] |
|---|---|---|---|
| Target Task | Team Play | Dribbling Skills | Real-World Compatible Football Play |
| Methods for Learning Low-Level Skills | Imitation of MoCap Data | Imitation of MoCap Data | - |
| Achieved Low-Level Skills | Natural Movements | Natural Movements | - |
| Methods for Learning Mid-Level Skills | RL reusing Low-Level Skills | RL reusing Low-Level Skills | RL reusing Low-Level Skills |
| Achieved Mid-Level Skills | Follow, Dribble, Shoot, and Kick-To-Target | Walk and Dribble | Score Goals, Prevent Conceding Goals, Ball Interaction, and Get-Up |
| Methods for Learning High-Level Skills | Self-Play | - | Self-Play |
| Achieved High-Level Skills | 2v2 Team Play | - | 1v1 Play |
| Transfer to Real Environment | - | System Identification + Domain Randomization | System Identification + Domain Randomization + Shaping Rewards |

TABLE I
OVERVIEW OF THE STUDIED APPROACHES

sub-tasks as well as for long-horizon tasks that require the identification of these sub-tasks beforehand.

HRL is an alternative approach that uses a divide-and-conquer technique to abstract the subtasks into different levels of smaller problems, which in turn can be solved using RL. The hierarchy consists of multi-level policies that range from choosing between different options or sub-goals to taking primitive actions to achieve those sub-goals. Using the hierarchical approach therefore makes it possible to learn low-level skills independently of high-level skills. An example of low-level skills is learning to move different joints to perform a simple behavior in robotics. The strategies learned at the lower level can be reused at higher levels. The high-level skills consist of choosing between the learned lower-level skills to achieve a more complex goal. For example, using the learned behaviors of the robot to score a goal in football.

Overall, HRL reduces the complexity of the whole problem, as the individual sub-goals are much easier to solve than the entire global goal.

## III. HIERARCHICAL POLICY FOR FOOTBALL

Learning football using RL is a very complex task that could be divided into several subtasks. Rather than learning to play football in an end-to-end manner, most current approaches divide the learning process into several phases. In the first phase, basic, natural-looking movement low-level skills required for football, such as walking or standing up, are learned. In the subsequent phase, these low-level skills are reused, to learn football-related mid-level skills, such as dribbling and kicking the ball. In the final phase, these mid-level skills are again retargeted to learn high-level skills that enable goal-oriented football play.

It is important to point out that although current methods deal with the use of RL for humanoid robot football, each focuses on a specialized goal. In [3], the main objective is to learn football skills in 1v1 scenarios in simulation which are also suitable for real robots and can therefore be transferred to the real platform zero-shot. In [1], on the other hand, the focus is on 2v2 matches to promote team play and division of labor. However, the strategies learned are only used in simulation. Finally, the objective of [2] differs the most from the other two papers. The authors' goal is not to master as much of the football task as possible, instead, they focus on achieving natural-looking running and dribbling skills. They also transfer their results to real robots, and in addition to the other works, they use a quadruped robot alongside the humanoid one.

Despite these different specializations, the papers share a common focus on learning hierarchically, using the results of the previous phases in the subsequent ones. In the following subsections, we compare the different ways and phases in which these skills are learned and reused. Table I provides an overview of the similarities and differences between the individual approaches.

### A. Learning Low-Level Skills

Humanoid robots are particularly well-suited for performing human activities, such as playing football, due to their human-like structure, as they can perform human movements. This is why in [1] and [2] human Motion Capture (MoCap) data is used to achieve natural-looking behaviors. This MoCap data allows for the use of *imitation learning*. The training of the low-level controller based on MoCap data is divided into two steps. First, the MoCap clips are imitated using a low-level policy created using RL. Both approaches follow the procedure described in [5]. Secondly, an encoder $q$ and decoder $\pi_d$ network are used to distill these movements into a low-level controller. Thus, the encoder $q$ generates latent variables $z \in \mathcal{Z}$ that represent the desired future trajectories based on the current state. $\mathcal{Z}$ refers to the corresponding latent space. The decoder $\pi_d$ takes these latent variables $z$ and the current state $s_t$ into account to generate the action $a_t$ that leads to joint commands for the robot. The networks are trained end-to-end, and the decoder network can then be reused as the low-level controller. The supervised

objective to train the low-level controller $\pi$ is represented as:

$$\mathbb{E}_q\Bigg[\sum_{t=1}^{T} \overbrace{\log \pi_d(a_t|s_t, z_t)}^{\text{decoder / low-level controller}} + \beta\Big(\underbrace{\log p_z(z_t|z_{t-1})}_{\text{latent prior}} - \underbrace{\log q\big(z_t|z_{t-1}, s_{t+1:t+k}\big)}_{\text{encoder}}\Big)\Bigg] \tag{3}$$

where $s_t$ are the states and $a_t$ are the actions from the tracking policies, and $\beta$ is a hyperparameter. Using the encoder-decoder technique enables the use of RL in the latent space rather than directly in the action space, resulting in more realistic humanoid movements, as random exploration in the latent space makes this more feasible. A visualization of this method in [1] can be seen in Figure 3.

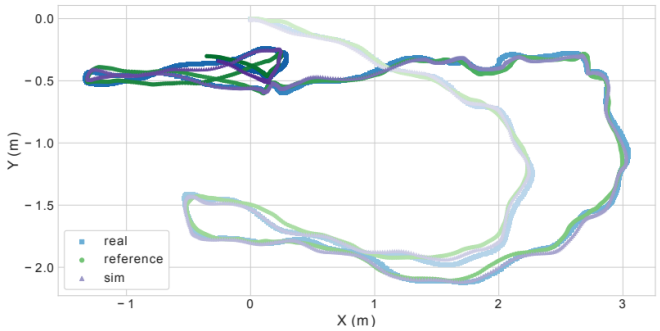



Fig. 2. Trajectories in the X-Y-plane of the imitation on ANYmal of one MoCap reference clip [2].

The evaluation of [2] shows that the robot can follow the imitated movements reliably while moving naturally. Figure 2 shows the xy-position error of the base following a trajectory. In [1], the performance of imitation learning is not evaluated in isolation.

### B. Learning Mid-Level Skills

Although the actions learned by the low-level controller lead to natural-looking movement behaviors, they do not encourage any football-related behaviors, such as ball interaction and goal-directed play. In [1], the phase for learning this type of skill is called "Acquiring Transferable Mid Level Skills".

In [1], four drill tasks are defined to enable football-related movements:

- Follow: The agent must follow a moving target
- Dribble: The agent must follow a moving target while keeping the ball close to it
- Shoot: The agent must score a goal with a maximum of three ball contacts
- Kick-to-target: The agent must kick the ball towards a target within a certain time

This step is also divided into two steps visualized in Figure 3.

In the first step, the $k$-th drill is represented as a reward function $r^k$, resulting in an objective function

$$\mathcal{J}^k(\pi_1) = \mathbb{E}\left[\sum_{t=1}^{T} r^k(s_t)\right] \tag{4}$$

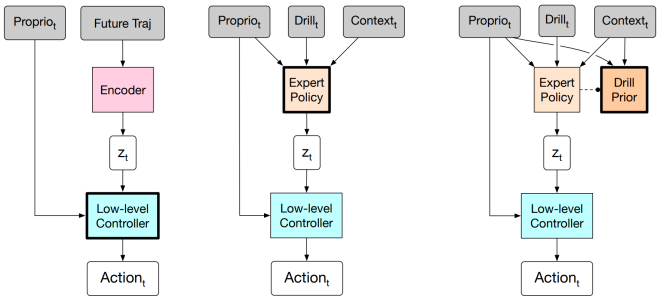



Fig. 3. Overview of the phases for learning the low- and mid-level skills in [1]. The bold components are reused in later phases: **(Left)** Encoder-Decoder Setup of the MoCap Phase. **(Middle)** Creation of the Expert Drill Policies. **(Right)** Creation of the Expert Drill Priors.

for a policy $\pi_1$. This objective is then maximized for each of the drill policies, resulting in an expert policy $\bar{\pi}^k$ per drill task. The policies act within the latent space of the previous step so that the low-level controller can be reused. Additionally, several shaping rewards are used to optimize the policies for the drill tasks. For example, to maximize the ball-to-target velocity or to encourage ball interaction.

In the second step, the drills are distilled into transferable drill priors $\mu^k$. This is achieved by minimizing the KL-divergence between each expert policy and drill prior in order to imitate the expert policy:

$$\mathbb{E}\left[\sum_{t=1}^{T} D_{KL}\left(\bar{\pi}^k(\cdot|h_t)||\mu^k(\cdot|\tilde{h}_t)\right)\right] \tag{5}$$

with the observation-action history $h_t$ at time $t$ regarding the drill expert, and the observation-action history $\tilde{h}_t$ regarding the prior's observation set.

In [2], the encoder gets replaced by a task-specific controller as well. The new controller is also trained in the latent space and also acts in it. Figure 4 shows an overview of the approach of [2]. It is visible that it is very similar to [1], as the low-level controller of the imitation learning part has been adopted to the phase where the mid-level skills are learned. The task policy, which is similar to the drills, also replaces the encoder.

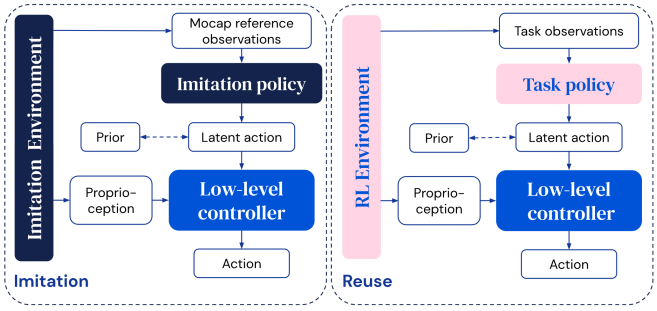



Fig. 4. Overview of the phases for learning the low- and mid-level skills in [2]: **(Left)** Imitation learning using the MoCap data to train the low-level controller. **(Right)** Reuse of the low-level controller in combination with a trained task policy that outputs the latent actions to enable dribbling.

In [2], however, the task policy focuses mainly on walking and dribbling rather than scoring goals, This is achieved by

the following rewards which depend on a scale factor $\phi$ for the resolution:

$$r_t^{\boldsymbol{v}} = \exp\left(-\frac{||\boldsymbol{v}_t - \hat{\boldsymbol{v}}_t||_2^2}{\phi}\right) \tag{6}$$

for minimizing the difference between the target velocity $\hat{\boldsymbol{v}}_t$ and current velocity $\boldsymbol{v}_t$, and

$$r_t^{\boldsymbol{p}} = \exp\left(-\frac{||\boldsymbol{p}_t^{\text{ball}} - \hat{\boldsymbol{p}}_t||_2^2}{\phi}\right) \tag{7}$$

for minimizing the distance between the current ball position $\boldsymbol{p}_t^{\text{ball}}$ and target position $\hat{\boldsymbol{p}}_t$. Another difference to [1] is that there is also no distillation into drill priors.

The approach described in [3] differs slightly from that of the other two papers, as it combines the learning of low- and mid-level skills. In this phase two teacher policies are trained: one for scoring goals and one for getting up from the ground. Whereas in the other two approaches, the training is performed for an isolated agent, in [3] the training takes place in an environment in which an untrained opponent is already present.

The main goal of the soccer teacher $\pi_f$ is to score goals. This is achieved through a variety of shaping rewards:

- Scoring: positive reward for scoring a goal
- Conceding: negative reward for conceding a goal
- Velocity to ball: agent's velocity toward the ball to encourage ball interaction, as in [1]
- Velocity: agent's forward velocity, similar to the reward in [2]

The purpose of the get-up teacher $\pi_g$, on the other hand, is to teach the robot to stand up from the ground. This is achieved in a similar way to the imitation learning of the other two approaches by guiding the policy to several target poses of a trajectory. However, the trajectory does not come from MoCap data, but from a pre-programmed get-up skill, which is shown in Figure 5.

(a) Get up from front

(b) Get up from back

Fig. 5. Key poses used to train the get-up teacher [3].

Given the current joint positions $\boldsymbol{p}_t$, and the current gravity direction $\boldsymbol{g}_t$ an error is computed for both of these variables regarding the target values $\boldsymbol{p}_{\text{target}}$ and $\boldsymbol{g}_{\text{target}}$, respectively:

- Scaled error in joint positions:

$$\tilde{p}_t = \left(\pi - ||\boldsymbol{p}_{\text{target}} - \boldsymbol{p}_t||_2\right)/\pi \tag{8}$$

- Scaled angle between gravity:

$$\tilde{g}_t = \left(\pi - \arccos\left(\boldsymbol{g}_t^T \boldsymbol{g}_{\text{target}}\right)\right)/\pi \tag{9}$$

The objective of the get-up teacher is to maximize the following reward:

$$\hat{r}_{\text{pose}} = -\tilde{p}_t \tilde{g}_t \tag{10}$$

In [2], the policy successfully led to the robot walking toward the ball and even positioning itself to shoot it in the correct direction. The skill module thus enabled precise, goal-oriented movement, even though the MoCap data differed significantly from the dribbling task.

To evaluate the effect of the drill priors in [1], the authors use differently trained agents to let them play multiple matches against evaluation agents to compute an ELO score for each agent, as shown in Figure 6. The following agents are used:

- Sparse rewards with the drill priors
- Shaping rewards without the drill priors
- Shaping rewards with the drill priors
- Shaping rewards with the drill priors and additional rewards to encourage team coordination

The agent using the sparse rewards and drill priors performs consistently poorly. The agent, using only the shaping rewards, starts crawling across the ground to reach the ball, trying to score with his arms as well. Interestingly, using additional shaping rewards to encourage team coordination leads to poorer results than not using these rewards. This could be a consequence of the agent exploiting these rewards and starting to focus more on team play than on scoring goals.

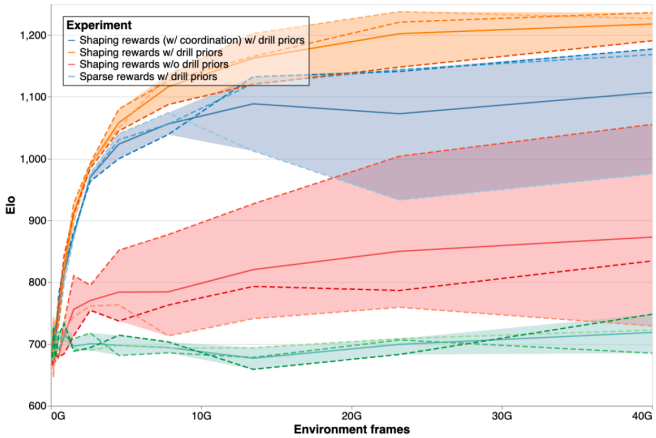



Fig. 6. ELO scores for the different training methods across all games in [1]: **(Green)** Sparse rewards with the drill priors **(Red)** Shaping rewards without the drill priors **(Orange)** Shaping rewards with the drill priors **(Blue)** Shaping rewards with the drill priors and additional rewards to encourage team coordination.

Similarly to [1], when learning without regularization towards the teachers in [3], but instead with sparse rewards, the

agent learned an alternative technique to score by crawling to the ball and scores while lying down, as shown in Figure 7.

(a) Crawling to the ball

(b) Scoring while lying down

Fig. 7. No regulation towards the teachers leads to crawling to the ball and scoring a goal while lying down [3].

### C. Learning High-Level Skills

The drill priors learned in the previous phase in [1] enabled football-related movements but are not sufficient to solve the entire football task. Therefore, in this phase, these high-level football skills are learned. Both, [1] and [3] use self-play to train the agents. For this, the results of the previous phases are used to train a population of agents. In [1], teams of two players are uniformly sampled with replacement from the population, with the players of each team being two separate instantiations of the same agent. During play, however, they act independently of each other. This way, high-performing policies are propagated through the population. In [3], on the other hand, the opponents are sampled uniformly from previous policies, as this led to better performance.

To enable transitions between the drill priors in [1], the behavior of the agents is biased towards the drill priors during self-play. By using KL regularization in the latent space to penalize the difference between the football policy and a mixture distribution of the $n = 4$ drill priors, they receive the following loss regarding policy parameter $\theta$ and hyper-parameters $\theta^h$:

$$\mathcal{L}_{\text{priors}}(\theta; \beta_{1:n}) := \mathbb{E}\left[\sum_{t=1}^{T} D_{KL}\left(\pi_\theta(\cdot|h_t) \middle\| \sum_{i=1}^{n} \beta_i \mu^i\left(\cdot|\tilde{h}_t^i\right)\right)\right] \tag{11}$$

with $h_t$ being the observation-action history at time $t$ from the football policy $\pi_\theta$ and $\tilde{h}_t^i$ the one from the prior $\mu^i$. The weights $\beta_i$ control the impact of each prior. Using priors instead of shaping rewards for these tasks has the advantage that they can adapt their influence to the current context. However, additional shaping rewards are introduced for higher goals, such as scoring and conceding goals. This results in a surrogate RL objective for football $\mathcal{J}^0(\theta; \theta^h)$. The authors did not include shaping rewards to encourage team play, as the results show that performance is worse than without these types of rewards. Finally, the surrogate objective got regularized towards the priors:

$$\bar{\mathcal{J}}^0(\theta; \theta^h) := \mathcal{J}^0(\theta; \theta^h) - \lambda \mathcal{L}_{\text{priors}}(\theta; \beta_{1:n}) \tag{12}$$

with step size $\lambda$, to learn the football task.

Similar to [1], in [3], policy distillation is used to enable smooth transitions between the soccer and get-up skills. Compared to [1], however, the abilities are mutually exclusive, i.e. they are never executed in parallel but rather depend on whether the robot is standing or lying on the ground. This can also be seen in the objective:

$$\mathbb{E}_\xi\left[\mathbb{1}[\boldsymbol{s} \in \mathcal{U}]\mathcal{J}_f(\pi_\varphi) + \mathbb{1}[\boldsymbol{s} \notin \mathcal{U}]\mathcal{J}_g(\pi_\theta)\right] \tag{13}$$

where $\mathcal{U}$ are all the states where the agent is standing. $\mathcal{J}_f(\pi_\varphi)$ and $\mathcal{J}_g(\pi_\varphi)$ are the training objectives that again use KL regularization to adapt the football training objective $\mathcal{J}(\pi_\varphi)$ towards the individual teachers:

$$\mathcal{J}_f(\pi_\varphi) = (1-\lambda_f)\mathcal{J}(\pi_\varphi) - \lambda_f \mathbb{E}_\xi\left[KL\left(\pi_\varphi(\cdot|\boldsymbol{s})||\pi_f(\cdot|\boldsymbol{s})\right)\right] \tag{14}$$

$$\mathcal{J}_g(\pi_\varphi) = (1-\lambda_g)\mathcal{J}(\pi_\varphi) - \lambda_g \mathbb{E}_\xi\left[KL\left(\pi_\varphi(\cdot|\boldsymbol{s})||\pi_g(\cdot|\boldsymbol{s})\right)\right] \tag{15}$$

The weights $\lambda_f$ and $\lambda_g$ are continuously updated using stochastic gradient descent to minimize the costs:

$$c(\lambda_f) = \lambda_f(\mathbb{E}_\xi\left[Q^{\pi_\varphi}(\boldsymbol{s}, \boldsymbol{a})\right] - Q_f) \tag{16}$$

$$c(\lambda_g) = \lambda_g(\mathbb{E}_\xi\left[Q^{\pi_\varphi}(\boldsymbol{s}, \boldsymbol{a})\right] - Q_g) \tag{17}$$

This allows the agent to outperform the teacher, as the weight decreases to 0 when the agent's predicted return exceeds the corresponding threshold $Q_f$ or $Q_g$, so the agent switches from behavioral cloning of the teacher to pure RL on the objective.

To evaluate the effects of self-play in [3], the authors set up several matches of a self-play that plays against opponents that are trained against different combinations of agents:

- An untrained agent
- An untrained agent and a soccer teacher (without a get-up teacher)
- An untrained agent, a soccer teacher, and a 1v1 policy trained with self-play

Looking at the results in Figure 8, it is clear that the self-play agent performs better over time than the agents trained against an untrained agent and a soccer teacher. But even when the self-playing agent is included as an opponent during training, this agent does not always beat him.

The authors of [1] did not perform experiments on the effects of self-play.

## IV. TRANSFER TO REAL ENVIRONMENT

As a final step, two of the approaches are deployed, and evaluated on a real robot. In [2] and [3], the learned policies are transferred to a real robot zero-shot. Both approaches use system identification to reduce the sim-to-real gap. Additionally, they use domain randomization to further increase robustness on the real platform. In the following, we will discuss in more detail how this is realized. While in [2] the

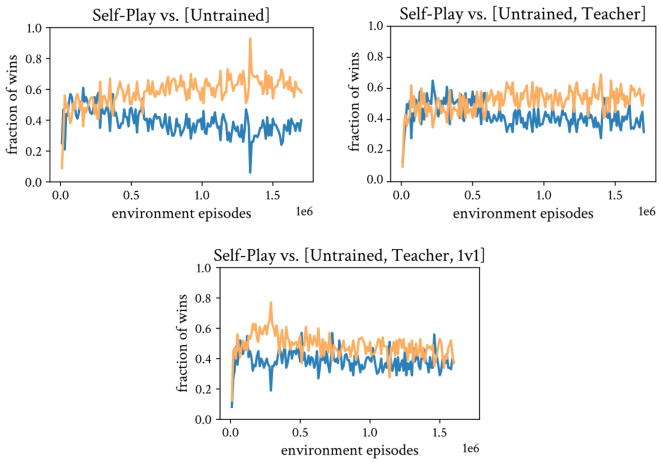
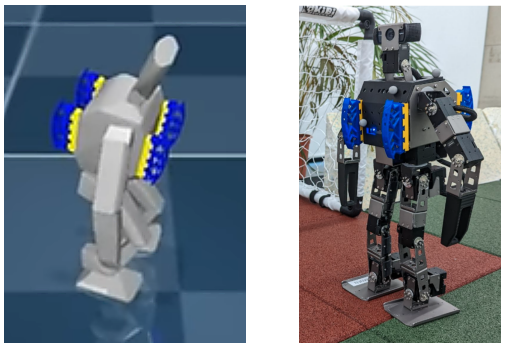



Fig. 8. Comparison of the self-play agent to the following opponents in [3]: **(top-left)** untrained agent, **(top-right)** untrained agent and soccer teacher, **(bottom)** untrained agent, soccer teacher, and 1v1 agent. Win fractions across 100 matches during training for self-play agent **(orange)** vs. opponent **(blue)**.

focus is mainly on the quadruped robot ANYmal and only slightly on the humanoid robot OP3 (see Figure 9), [3] uses OP3 exclusively. Both approaches use an external MoCap system to track the position of the robot(s) and the ball.

Compared to the other two papers, the approach described in [1] is not transferred to a real environment.



(a) OP3 in simulation (b) OP3 in reality

Fig. 9. The humanoid football robot OP3 (Images from [3]).

### A. System Identification

The first step in minimizing the discrepancy between simulation and reality is to create a simulation that is as close to the real world as possible.

In [2], they use the built-in actuators from MuJoCo to model OP3. In [3], the actuator parameters are determined by applying varying frequency sinusoidal control signals to a motor with a known load. The parameters of the actuator model are then optimized in the simulation to achieve the resulting joint angle trajectories.

### B. Domain Randomization and Perturbations

In addition to system identification, both of the approaches use domain randomization and additional perturbations to further reduce the sim-to-real gap. Domain Randomization involves introducing different variations during simulation training by changing certain parameters to improve the adaptability of robotic systems to real-world conditions and reduce performance differences caused by wear and tear, environmental factors, or hardware discrepancies.

Some examples of the used perturbations are randomized floor friction, joint angular offsets, variation of the orientation and position of the IMU, attachment of randomly heavy masses at randomly selected points on the robot torso, random noise, and time delays in the simulated sensor readings, and the application of impulses, random in strength, duration, and position.

### C. Regularization for Safe Behaviors

In [3], the trade-off between a sufficient range of motion for playing football and moving the joints too rapidly, which may damage them, is addressed. Therefore, additional rewards have been added to reduce these robot breakages:

- Joint torque: negative reward, proportional to the measured torques in the knee joint
- Upright: positive reward for smaller tilt angles to prevent the robot from falling over

## V. LIMITATIONS AND FUTURE WORK

Each of the three approaches achieves its individual goal. However, several limitations are not covered by the works.

First of all, all of the approaches use an external system to track the current game state. This is very unrealistic compared to real football, as the robots are omnipresent and know where every game object is on the pitch. A human player, on the other hand, would have to look around and sometimes not be able to see the ball if it is obscured by a player, for example. However, using the on-board robot sensors for perception is a hard problem, as it would not only require tracking all game objects, but it would also further increase the sim-to-real gap, as moving sensors have much more noise in reality.

Another limitation of the approaches is that they do not encourage dribbling with the ball to pass to an opponent. In the works, the ability to dribble is learned as a mid-level skill, but scoring goals is the main reward of the high-level skill. This leads to the players running to the ball and trying to score a goal without moving with the ball to get a better shot position towards the goal.

Several factors could be adjusted to increase the realism of the football scenarios with regard to real football in future works. First, combining the approaches of [1] and [3] to enable team play in a real-world environment. The team sizes could also be increased. In [1], decision-making is very binary: should the robot shoot toward the goal, or should it pass the ball to its teammate? By increasing the size of the team, each player would have to decide which of his teammates has the most promising position on the pitch, allowing for many more tactical options during play. Currently, the robots are only trained to score goals without taking the score into account. In a real football match, tactics

change depending on the score, e.g. if the lead is high, the team will defend more. This adaptation to the current state of play could also be included in future work. Finally, the robots used could be more and more adjusted to the physical form of humans. OP3 has a height of $51\,\mathrm{cm}$ and has $20$ degrees of freedom. As larger robots with more degrees of freedom are harder to control in reality due to the heavier limbs, it would be advantageous to adapt the robot to a human to be able to compete against a human team at some point.

## VI. CONCLUSION

In this work, we compare three papers that use HRL to teach humanoid robots to play football. The papers all have a different global goal that they want to achieve. In [1] the focus is on learning 2v2 team play in simulation, in [2] on learning dribbling skills, and in [3] on learning goal-oriented 1v1 play in real scenarios.

[1] and [2] use MoCap data in combination with imitation learning to achieve natural-looking behaviors. [3] combines the learning of low-level and mid-level skills by training two teacher policies, one for scoring goals and one for getting up from the ground. In [1], the mid-level skills are learned in the form of drill policies while reusing the low-level controller, which are then distilled into transferable drill priors. Similarly, in [2], task policies are learned, also reusing the low-level controller. In [1] and [3], high-level skills are learned to enable goal-oriented football play, by using self-play between agents trained on the mid-level skills. Finally, the results of [2] and [3] are transferred to a real robot zero-shot, by using system identification, domain randomization, and perturbations.

Overall, all three papers show promising results for achieving realistic football play using humanoid robots.

## REFERENCES


[1] S. Liu, G. Lever, Z. Wang, J. Merel, S. Eslami, D. Hennes, W. M. Czarnecki, Y. Tassa, S. Omidshafiei, A. Abdolmaleki, *et al.*, "From motor control to team play in simulated humanoid football," *arXiv preprint arXiv:2105.12196*, 2021.

[2] 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.

[3] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, M. Wulfmeier, J. Humplik, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, *et al.*, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *arXiv preprint arXiv:2304.13653*, 2023.

[4] R. Bellman and R. Kalaba, "On adaptive control processes," *IRE Transactions on Automatic Control*, vol. 4, no. 2, pp. 1–9, 1959.

[5] L. Hasenclever, F. Pardo, R. Hadsell, N. Heess, and J. Merel, "Comic: Complementary task learning & mimicry for reusable skills," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4105–4115.