A Two-Step Approach for Physically Competitive Sports: A Case Study on Fencing

Chun Kai, Yang
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan 10617
b10901027@ntu.edu.tw

Yu Ju, Cheng
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan 10617
b10901009@ntu.edu.tw

Yu Hua, Chen
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan 10617
b10901100@ntu.edu.tw

Chu Rong, Chen
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan 10617
b10901002@ntu.edu.tw

Abstract

In the realm of two-player competitive sports, athletes exhibit a blend of efficiency and tactical acumen, particularly evident during the heat of competition. This paper delves into the creation of a learning framework specifically tailored for fencing, a sport that demands not just efficiency but also precision in movement. Our approach in developing this framework is unique in that we have engineered both the reward system and the training methodology to facilitate direct learning from the competitive environment, thereby eliminating the need for a preliminary pretraining phase. To evaluate the effectiveness of our model, we have undertaken an extensive series of experiments. These involve scenarios where the model competes against itself, allowing us to observe and analyze its performance in a controlled yet competitive setting. In render mode, it can be observed that the model can effectively attack opponent in valid area and dodge opponent's sword. The results of our experiments shows that our model can adapt and respond effectively within the dynamic and complex environment of fencing. This research not only contributes to the field of competitive sports modeling but also opens new avenues for exploring the application of learning frameworks in other sports demanding similar levels of precision and tactical skill.

1 Introduction

Sports pose one of the most intricate challenges in the field of machine learning, necessitating precise control over both joint movements and body coordination. We have chosen fencing as our target sport because it not only demands models to learn generalized movements like advancing and retreating but also requires accuracy in joint control.

Some efforts have been made to address this problem using reinforcement learning approaches. In most of these studies, there are two possible approaches, *imitation learning* [1] or *skill-based learning* [2], both of them employ pre-training phase before moving on to the targeted problem. However, these methods often demand larger dataset and greater computational resources for successful training. Whereas imitation learning utilize human video to extract knowledge for specific action and state, skill-base learning require expensive computation resource for pre-training. Our objective is to design a way to train from scratch without pre-training and additional dataset.

To address this task, we use a two-step approach. First, we use a *curriculum* [3] way for our training. At initial, the fencer attacks a stationary opponent for an interval. In the second stage, the opponent loads the checkpoint of the attacker, which leads to a self-play condition. We refresh the version of model periodically, which can be tuned as hyper-parameter. After conducting curriculum training, we apply the model in the second step, non-curriculum setting. In *non-curriculum* setting, we also train with 2-stage. First, we fix the opponent to single version, which is the one we get from the curriculum training. Then, update the version of model periodically with the checkpoint of the attacker, the same as the second-stage in curriculum setting. *Non-curriculum* learning result in not only more stable performance but also higher training efficiency. But the performance of non-curriculum method rely largely on the quality of opponent which comes from curriculum training model.

Since training from scratch, the design of reward function is of paramount importance. We categorize skills that model need to learn into two part, *attack* and *dodge*. For attacking, we calculate the distance from the sword tips to the target point we set in opponent, which are sampled from the core part of body. For dodging, we use the distance from opponent's sword tip to the agent's target point.

Our contribution is as followed:

1. Construct a complete and handy environment for competitive fencing.
2. Design state and reward for training model from scratch.
3. Our model is shown to be a great standard for non-curriculum training, which can be apply to other works.

2 Related Work

2.1 Curriculum RL

Curriculum Reinforcement Learning [3] is akin to a structured educational approach, where the learning process is divided into progressively challenging stages or tasks. Curriculum RL systematically increases the complexity of the learning environment. This method often starts with simpler tasks, gradually moving towards more complex ones as the agent's performance improves. The primary advantage of Curriculum RL lies in its efficiency and effectiveness in teaching agents complex behaviors by breaking down the learning process into more manageable segments. However, designing an effective curriculum can be challenging, as it requires predefined knowledge about the task hierarchy and optimal progression. In our work, we use curriculum for our baseline model.

2.2 Competitive RL

Competitive Reinforcement Learning [4] involves scenarios where multiple agents are trained simultaneously in an adversarial setup. Each agent's goal is typically to maximize its own reward in the face of competition from other agents. This framework is often represented in games and simulations where agents compete against each other, such as in two-player board games, strategic games, or economic simulations. The essence of Competitive RL lies in its ability to simulate real-world competitive scenarios, making it a valuable tool for studying strategic behavior, negotiation, and conflict resolution. In our work, we train only one model at one time, which is not similar to conventional competitive RL.

2.3 Model-Based RL

Model-Based Reinforcement Learning [5]fundamentally revolves around the concept of understanding and predicting the dynamics of the environment. In this framework, an agent aims to learn a model that captures the environment's behavior, including the transitions between states and the rewards associated with these transitions. This model is then used to plan and make decisions, often enabling the agent to learn effective policies with fewer interactions with the actual environment compared to model-free approaches.

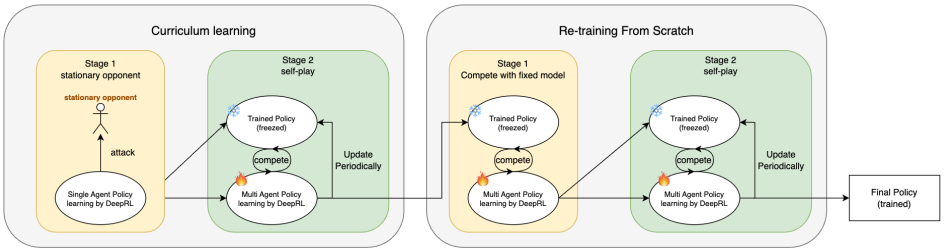


Figure 1: Overview of Training Framework.

The core advantage of Model-Based RL lies in its efficiency in sample usage, leading to faster learning from fewer real-world interactions. This characteristic is particularly valuable in scenarios where real-world interactions are costly, risky, or time-consuming.

2.4 Imitation RL

In a similar work from Meta research [6], they use imitation learning to train a generalized model for boxing and fencing task. They divide the training process to multi-stage. In each stage, the humanoid learn specific skills, such as offence and defence. In humanoid environment, the observation state is large, thereby demand high computation resource. For high dimension observation feature, they use a *task encoder* and *motor decoder* to distill the observation, which make training process more efficient.

3 Problem Formulation

Our goal is to achieve a well-performed control policy without collecting human data in a two-player physically competitive game. Our environment employs two policy models to control agents and make them compete. Each model takes relative observational states with respect to the controlled agent as input and generates actions to control a physically simulated player. Fig. 1 illustrates the overview of our training framework.

3.1 Environment

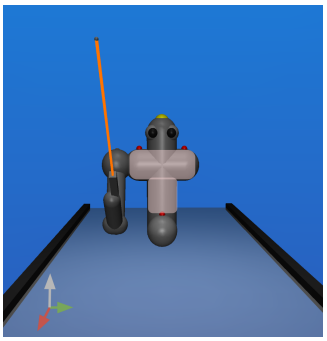
We created a *fencing* environment in MuJoCo[7] as our example, where two agents compete in an attempt to win the match. Our environment consists of two identical agents in a rectangular arena (Fig. 2(b)). We use the rule of *foil* in fencing, where the target area only covers the torso instead of the whole upper body. Contact with the side of the blade does not result in a valid attack[8]. Similar to the previous work [6], we attached a blade to the right fist of the agent to mimic a firmly held sword. At the end of the blade, a small spherical tip is placed to help us detect contact with the target area using the collision event of MuJoCo[9].

The agent is shown in Fig. 2(a). For simplicity, the agent only has the right arm since our agent does not need to maintain balance in posture, and only the right arm will be used in fencing. The agent can translate in 2D slides and has 8 hinge joints, each with one rotational dimension of freedom, to control its body. Thus, it has a total of 10 dimensions of freedom. We control the agent by applying torque in rotational joints and force in translational slides. There are three red dots on the shoulders and stomach acting as reference points to indicate the target area.

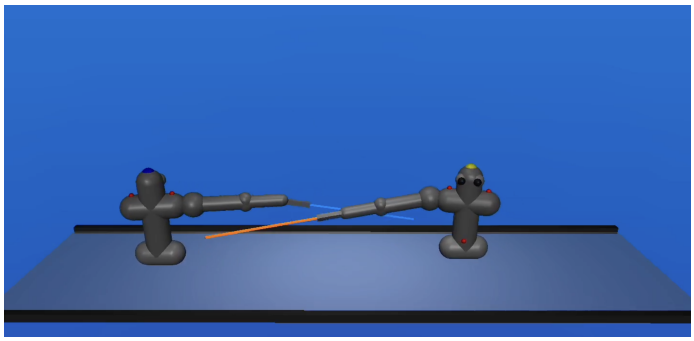
The observation states of the agent consist of both body states and match-related states along with game time. The body states include the position and linear velocity of every joint, while the match-related states encompass both the relative position and velocity of tips to the other's reference points, as shown in Fig. 3. The dimensions of observation states for one agent are 60.

3.2 Rules

We treat the status of a match as a finite state machine (FSM), including idle, win, lose, draw, foul, and timeout states. The agent who touches its opponent at the target area with the tip of the sword



(a) Fencing Agent. The target area are marked in pink.



(b) Illustration of Competition Arena

Figure 2: Fencing Environment

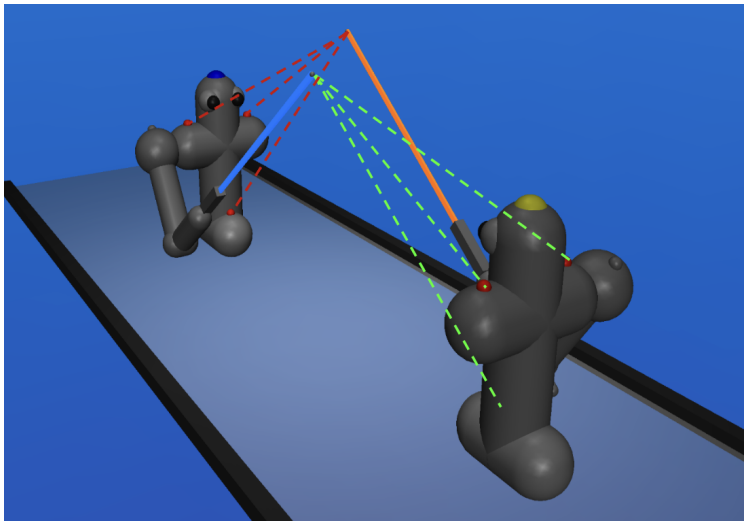


Figure 3: Illustration of relative position of tips to the other's reference points.

first wins the match; however, the match becomes a draw if the opponent counterattacks within 30 steps after the first touch occurs. When the agent is out of the arena, i.e., touches the borders (black rails on the sides of the arena, as shown in Fig. 2(b)), it is considered a foul. If the match exceeds 600 steps, it will enter the timeout state. After entering these terminate states, i.e., win, lose, draw, foul, and timeout, the match will last for an extra 30 steps, which is the same as the time waiting for a counterattack.

3.3 Match rewards

When the match ends, we give a match reward r_{match} based on the final status.

$$r_{match} = \begin{cases} 10, & \text{if win.} \\ -10, & \text{if lose.} \\ 0, & \text{if draw.} \\ -5, & \text{if foul.} \\ -5, & \text{if timeout.} \end{cases} \tag{1}$$

4 Method

Our framework involves two steps (Fig. 1). In the first step, we perform a two-stage curriculum learning to enable the agent to gradually learn the game rules and the policy of competition. In the second step, we initially use the competitive model trained in the first step as an opponent to train a new agent from scratch, then engage in self-play to update the policy in the second stage, as in the first step.

4.1 Step 1: Curriculum Learning

In the first stage, the agent will interact with a stationary opponent to learn the basic rules of the game and simple motion control strategies. At the beginning of the second stage, we transplant the model learned in the first stage as the opponent to compete with the agent. Then, for the rest of this stage, we periodically update the opponent with the newly stored model from the agent, i.e., the agent will engage in self-play to learn the competitive rules of the game. This two-stage method not only reduces the complexity and the state dimensions of training the policy model for two agents but also avoids the situation where two beginners randomly swing their swords, hindering the exploration process to find the correct winning strategy.

4.2 Step 2: Re-training from Scratch

In this step, we use the competitive model trained in the first step to train a new policy model from scratch. Our experimental result shows that the Step 2 model has better generalization capability than the Step 1 model. Although the Step 1 model shows more precise attack accuracy, it loses its direction, randomly swinging and running after missing the initial attack. In contrast, the Step 2 model can sustain its momentum even after a miss.

4.3 Reward Shaping

Since the match reward is sparse, we design several auxiliary rewards. The total reward function is as follows:

$$r = r_{match} + w_{agent\_near} r_{agent\_near} - w_{opponent\_near} r_{opponent\_near} - w_{energy} r_{energy} \tag{2}$$

where r_{match} is discussed in equation (1) and is included only at the end of the match. r_{agent_near} is calculated based on the temporal difference of the mean distance d between the agent's sword tip and the opponent's three reference points (red dots in Fig.2(a)), indicating the target area. r_{agent_near} is a linear function with a clip at certain maximum and minimum values:

$$r_{agent\_near} = -max(0, 0.5 * (d - 0.1)) + 0.2 \tag{3}$$

The equation signifies that r_{agent_near} becomes positive when the sword tip is very near to the opponent, encouraging the agent to stretch its arm to bring the tip as close as possible to the target area.

We use $r_{opponent\_near}$ to measure the threat of the opponent. Considering it only becomes a threat when the opponent's tip is too close to the agent, we use an exponential function to assign a significant penalty when the opponent is too close, but a penalty near 0 when the opponent is far.

$$r_{opponent\_near} = 1.5 * e^{-5*(d-0.05)} \tag{4}$$

where d here is the same as discussed above, representing the difference in mean distance between the opponent's sword tip and the agent's three reference points. We also add an energy penalty to stabilize the physical movement.

$$r_{energy} = \sum ||a_i||^2 \tag{5}$$

where a_i is either angular acceleration or linear acceleration of the i^{th} joint.

Table 1: Network setting of different algorithms

Algorithm	Network	Hidden layer setting
PPO	policy	(60, 128, 64)
	value	(60, 128, 64)
SAC	actor	(60, 256)
	critic	(70, 256, 256)
A2C	policy	(60, 128, 64)
	value	(60, 128, 64)
TD3	actor	(60, 400, 300)
	critic	(70, 400, 300)
DAC	actor	(64, 64)
	critic	(64, 64)
	option	(64, 64)

5 Experiment

In this section, we start by comparing various deep reinforcement learning methods alongside a hierarchical reinforcement learning approach known as Double Actor-Critic (DAC) [10]. Subsequently, we shift our attention to PPO, a prominent method from the comparison, and carry out an ablation study. This study analyzes the effects of several modifications, such as the exclusion of auxiliary rewards, alterations to match rewards, random initialization, and the difference between the two steps in our training method. All models are trained for 2 million steps, and all are Step 1 models, i.e., they attack a stationary opponent in the first stage, except for the ones in section 5.6, which are Step 2 models training with a mature opponent initially.

5.1 Implementation Setup

All of our experiments are conducted with the help of Gymnasium[11], which provides an interface to communicate with our environment built in MuJoCo. The algorithm implementations, except for DAC, are all from stable baselines 3[12].

5.2 Algorithms Comparison

We first compared 5 different algorithms: PPO, SAC, A2C, TD3, and DAC, to decide with algorithm to use in later comparison and analysis. The network setting and the training curves are shown in Table 1 and Fig. 4 respectively.

In the case of A2C, the agent consistently engages in fouls and rarely successfully hits its opponent. For SAC and DAC, the foul probability remains around 20% without reaching zero, which is not the desired outcome. As for TD3, although the win probability eventually converges to 100%, it lacks the stability observed in PPO during the mid-training phase, where PPO exhibits lower foul and lose probabilities. Consequently, we opt to use PPO as the chosen algorithm for the subsequent experiments.

5.3 Analysis: Effect of Auxiliary Rewards

In the experiment, we are examining the impact of auxiliary rewards through three different reward systems. The first system includes only the goal reward (r_{match}). The second one encompasses all rewards except for the opponent tip near penalty ($r_{opponent\_near}$). The third system includes all rewards. This experiment is conducted using the PPO algorithm. The result is shown as Fig.5 and in Table.2.

We observed that in the environment with only goal rewards, the agent tends to push its opponent towards the border line instead of executing proper stabbing actions, deviating from the realistic behavior expected in a fencing game. In scenarios where all rewards are present except for the opponent tip near penalty, the agent consistently achieves victories, especially in the later stages of training. However, when all rewards are included, the win probability initially increases rapidly, only

Figure 4: Training curves of different RL algorithms.

Figure 5: Match results under different auxiliary reward settings.

Table 2: Competition of w/o (1) and w/ (2) opponent penalty

	Game result statistics						
				Foul by			
View of	Win	Lose	Draw	(1)	(2)	both	Timeout
w/o Opponent penalty	45.26%	17.08%	10.71%	2.17%	20.86%	2.65%	1.27%

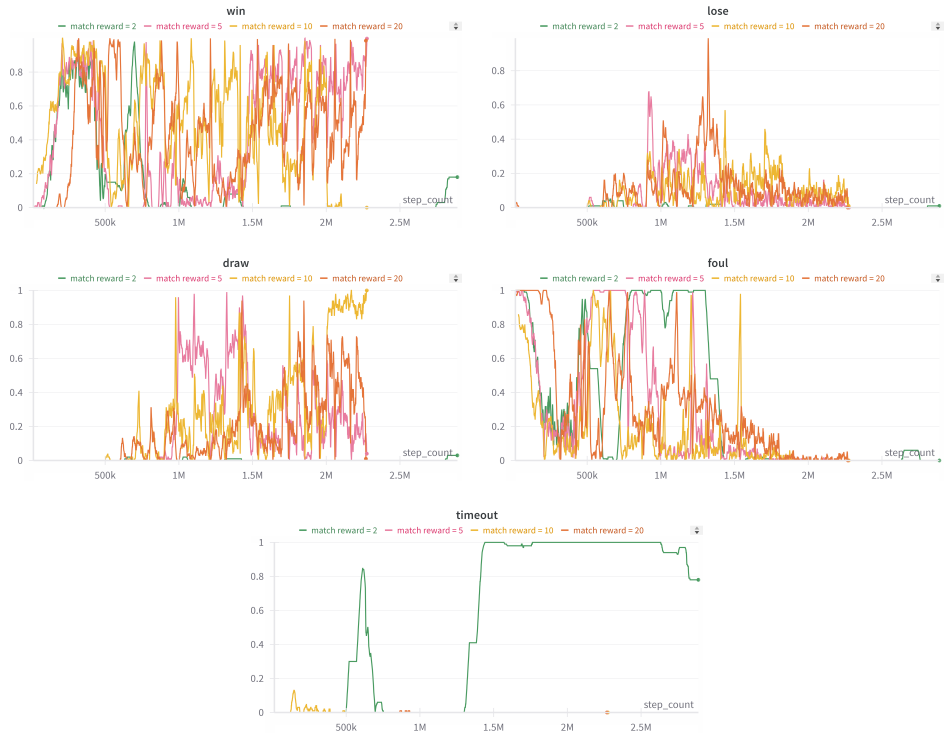


Figure 6: Match results under different value of goal reward.

to decline later as the draw probability approaches 100%. This phenomenon occurs because both the agent and its opponent (previous version of the agent) learn effective defensive strategies against attacks.

The comparison between models trained with and without opponent penalty is shown in Table 2. It is observed that the model trained with opponent penalty performs worse than the other one. After examining several records of the competition, we found a possible reason: the all-reward model is too conservative when facing the aggressive model trained without opponent penalty since it is trained with the consideration of dodging attacks.

5.4 Analysis: Value of Match Rewards

In the experiment, we are investigating the impact of adjusting the value of match rewards if the agent wins or loses. We are testing variations, specifically $\pm 2, \pm 5, \pm 10$ (default), and ± 20. This experimentation is conducted using the PPO algorithm. The result is shown as Fig. 6.

For goal rewards set to ± 5 and ± 20, the win probability increases at a slower rate compared to the default environment with goal rewards set to ± 10. However, both variations can eventually achieve a high winning rate. On the other hand, when goal rewards are set to ± 2, the win probability starts very high in the initial stage (first 500k steps) but rapidly declines to zero as the foul probability approaches 100%. Notably, none of these three variations result in the agent and its opponent having draws consistently, unlike the original environment.

Figure 7: Comparison between random and fixed initialization of positions and velocities.

5.5 Analysis: Random Initialization

In the experiment, we investigate whether random initialization of velocities and positions of joints can enhance the agent's learning performance. The changes in positions and the range of velocity changes follow a random distribution of $\mathcal{U}(-0.1, 0.1)$. The result is shown as Fig. 7

We observe that despite the agent with random initialization exhibiting instability in the middle of training, as the training advances, the agent attains a high win probability similar to the original one.

5.6 Analysis: Effect of Re-training

In the experiment, we contrast a Step 1 model with a Step 2 model by letting them have a match with each other. The game result is shown in Table 3, with respect to the Step 1 model.

It may seem that the two models' performance has no significant difference at first glance. However, if we count a foul as a loss for the one who caused it, i.e., if agent A fouls and agent B does not, agent A is considered to have lost, it shows a large gap between the Step 1 model and the Step 2 model's performance.

The Curriculum learning (Step 1) model exhibits a higher probability of ending the match with a foul. When the initial attack misses, the Step 2 model can consistently attempt rapid stabbing actions, while the other continually retreats and tends to commit fouls. The discrepancy could arise from the fact that the Step 1 model retains knowledge acquired in the initial stages that is not related to the actual game dynamics. It can hit the opponent quickly, but it fails to maintain the momentum if the first attempt misses and often commits fouls after that. In contrast, the Step 2 model is learned by playing the game right from the start, eliminating the incorrect experience of playing with a stationary opponent.

Additionally, our reward shaping method provides valuable information for the blank model to find a way to get close to the target, enabling the latter agent to learn how to attack instead of only dodging when faced with an aggressive opponent from the start.

Table 3: Step 1 v.s. Step 2

	Game result statistics				
Scoring rule	Win	Lose	Draw	Foul	Timeout
Normal	38.31%	36.15%	3.69%	21.85%	0.00%
Include foul in lose	38.35%	57.96%	3.69%	N/A	0.00%

6 Conclusion

We have presented a novel two-step method for training agents in a complicated competition environment, like fencing, without collecting human data. Furthermore, we empirically analyzed the impact of reward shaping, reward scaling, random initialization and re-training.

Our main contribution is proposing a training procedure that, unlike existing imitation-based methods or other algorithms requiring pre-training, can train high-performance and generalized agent from scratch in competition setting. Since our method doesn't restrict the underneath RL optimization algorithm, it can work with arbitrary algorithm for different consideration, providing modular capability.

As part of future work, a more in-depth exploration of match results among various RL training algorithms can be conducted. Additionally, there is potential for applying the same methodology to more complex environments.

References

[1] Kamil Ciosek. Imitation learning by reinforcement learning. 2021.

[2] Lucy Xiaoyang Shi, Joseph J. Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *Conference on Robot Learning*, 2022.

[3] Matteo Leonetti Jivko Sinapov Matthew E. Taylor Peter Stone Sanmit Narvekar, Bei Peng. Curriculum learning for reinforcement learning domains: A framework and survey. 2020.

[4] Yu Bai and Chi Jin. Provable self-play algorithms for competitive reinforcement learning, 2020.

[5] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey, 2022.

[6] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Trans. Graph.*, 40(4), jul 2021.

[7] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[8] *USA Fencing Rulebook*. Colorado Springs, CO, USA, jan 2023.

[9] Mujoco documentation. `https://mujoco.readthedocs.io/en/stable/overview.html`. Accessed: 2023-12-23.

[10] Shangtong Zhang and Shimon Whiteson. Dac: The double actor-critic architecture for learning options. *Advances in Neural Information Processing Systems*, 32, 2019.

[11] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.

[12] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019.