

https://dspace.org/

Formation Strategy Optimization Using Multi-Agent Reinforcement Learning

Njupoun, Abdel Mfougouon

2024-08
University of the Witwatersrand, Johannesburg
https://hdl.handle.net/10539/46675
Njupoun, Abdel Mfougouon. (2024). Formation Strategy Optimization Using Multi-Agent Reinforcement Learning. [Master's dissertation, University of the Witwatersrand, Johannesburg]. WIReDSpace. https://hdl.handle.net/10539/46675

Downloaded from a DSpace repository. https://dspace.org/

University of the Witwatersrand, Johannesburg
School of Computer Science and Applied Mathematics

Formation Strategy Optimization Using Multi-Agent Reinforcement Learning

Abdel Mfougouon Njupoun
2631227



Supervised by:
Branden Ingram
Benjamin Rosman
Geraud Nangue Tasse

A thesis submitted to the Faculty of Science, University of Witwatersrand,
in partial fulfilment of the requirements for the degree of Master of Science.

Abstract

A long-lasting goal of artificial intelligence is to design agents capable of cooperative problem-solving. The RoboCup soccer competition provides a challenging environment for investigating the design of such intelligent and autonomous agents using machine learning techniques, such as Multi-Agent Reinforcement Learning (MARL). Cooperation is inherently difficult due to the need for agents to align their strategies, adapt to each other's actions, and make decisions that benefit the collective goal over individual success. In this context, developing effective defensive strategies is particularly challenging. It requires agents to not only understand and anticipate the actions of opponents but also to coordinate with teammates in a dynamic environment where split-second decisions can determine the outcome of a game. This complexity is compounded by the unpredictable nature of the opponent's strategies and the continuous adaptation required to counter them effectively.

This research aims to investigate the application of Multi-Agent Reinforcement Learning in learning an effective defensive strategy in the RoboCup soccer competition. We use reward shaping to positively influence the behaviour of our simulated soccer players such that they can effectively defend against attacking soccer strategies. This reward function is then utilized to train agents and learn the optimal policy in centralized settings, namely Central Proximal Policy Optimization (CPPO). The training process involves exposing our agents to different fixed policies such as the Keepaway approach, where a team keeps the ball away from opponents, the Half-field offense strategy where the objective of the offense team is to strategically outplay the defense team to score goals, and the random direction changes aiming to mimic the unpredictability of human soccer, where players often change direction suddenly to evade defenders or create attacking opportunities. To evaluate our proposed strategy, we conduct a comparative analysis against established baselines like the NeuroHassle approach, which emphasizes early disruption of the opponent's attack, and the Stable Marriage approach, focusing on optimal defender-attacker pairings. These methods, one based on reinforcement learning and the other on preference-based pairing, serve as benchmarks to gauge the effectiveness of our strategy in improving defensive gameplay. Evaluation metrics, including goals conceded and average distance between opponent players and our goal are used to analyze and identify the strengths and weaknesses of each approach. We evaluated our approach against the NeuroHassle and Stable Marriage methods by observing agent performance in keepaway, Half-field, and random direction changes offense scenarios. Utilizing those key metrics, we identify that our model demonstrated better defense strategies, offering insights for enhancing multi-agent systems in competitive environments like RoboCup soccer.

Acknowledgments

I am profoundly grateful for the opportunity to pursue my Master of Science in Robotics at the University of the Witwatersrand, Johannesburg, a journey made possible by the generous support of the Google DeepMind Scholarship. This esteemed scholarship not only financed my academic endeavors but also significantly contributed to my professional and personal growth.

My deepest gratitude goes to Branden Ingram, Geraud Nangue Tasse and Benjamin Rosman my supervisors, for their invaluable guidance, patience, and immense knowledge. Their mentorship was pivotal in shaping the research direction and execution of this thesis. I would also like to extend my heartfelt thanks to all the Lab's member for their insightful feedback, encouragement, and expert advice throughout my research.

I am especially thankful for the opportunity to work in the RAIL Lab thanks to Benjamin Rosman (Director of RAIL Lab) and Pravesh Ranchod. The access to the lab's state-of-the-art resources, both software and hardware, was instrumental in the success of my research. The stimulating discussions with fellow researchers and lab members have been immensely rewarding and have significantly contributed to my learning experience.

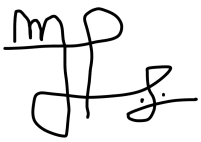
Furthermore, I wish to express my gratitude to my family and friends for their unwavering support and belief in me. Their encouragement and understanding during the highs and lows of this academic journey have been a constant source of strength.

Lastly, I thank all those who were involved in any capacity in my research or provided support at the University of the Witwatersrand. This journey would not have been possible without the collaborative effort and shared wisdom of many.

Declaration

I, Abdel Mfougouon Njupoun, hereby declare that the thesis titled **Formation Strategy Optimization Using Multi-Agent Reinforcement Learning**, submitted for the partial fulfillment of the requirements for the degree of Master of Science at the University of Witwatersrand, Johannesburg, is my original work and has not been submitted to any other university or institution for any degree, diploma, or other qualification.

I further affirm that the work reported in this thesis has been carried out by me under the guidance of my supervisors. Any references to work done by any other person or institution or any material obtained from other sources have been duly cited and acknowledged.



Signature

16 August 2024

Date

Dedication

In Memory of Professor Marco Garuti,

This thesis is dedicated to the cherished memory of Professor Marco Garuti, whose guidance and wisdom profoundly shaped my journey through the world of mathematics. Professor Garuti was not only a beacon of scientific brilliance but also a remarkable human being whose kindness, dedication, and passion for knowledge left an indelible mark on all who had the privilege of knowing him.

His relentless pursuit of understanding, combined with his genuine care for students, fostered an environment where curiosity flourished and learning knew no bounds. His legacy, embodied in the lessons he imparted and the lives he touched, continues to inspire and guide us.

With profound gratitude and respect.

Contents

1 Introduction **6**
- 1.1 Overview . . . 6
- 1.2 Research Problem . . . 7
- 1.3 Contributions . . . 8
- 1.4 Research Structure . . . 9

2 Theoretical Background **10**
- 2.1 Introduction . . . 10
- 2.2 Reinforcement Learning . . . 10
  - 2.2.1 Markov Decision Process . . . 10
  - 2.2.2 Optimal Policy . . . 12
- 2.3 Multi-Agent Reinforcement Learning . . . 12
  - 2.3.1 Overview of Stochastic Games . . . 12
  - 2.3.2 Nash Equilibrium . . . 13
  - 2.3.3 Adversarial, Cooperative, and Mixed Environments . . . 13
  - 2.3.4 Centralized and Decentralized Approaches in MARL . . . 14
- 2.4 Overview of RL and MARL Algorithms . . . 15
- 2.5 Proximal Policy Optimization (PPO) . . . 16
- 2.6 Conclusion . . . 19

3 Literature Review **20**
- 3.1 Introduction . . . 20
- 3.2 Overview of the RoboCup Competition . . . 20
  - 3.2.1 3D RoboCup simulation league . . . 20
  - 3.2.2 2D RoboCup simulation league . . . 21
- 3.3 Multi-Agent Reinforcement Learning in Centralized and Decentralized Approaches 22
- 3.4 Multi-Agent Reinforcement Learning in RoboCup . . . 23
  - 3.4.1 Keepawawy strategy in the 2D simulation . . . 23
  - 3.4.2 Half-Field Offense . . . 23
  - 3.4.3 NeuroHassle in learning a defensive strategy . . . 24
  - 3.4.4 Multi-Agent PPO . . . 25
- 3.5 Conclusion . . . 25

4 Methodology **26**
- 4.1 Introduction . . . 26
- 4.2 Defensive Behaviour Learning Framework . . . 26
  - 4.2.1 Multi-Agent Defensive Strategy . . . 27
  - 4.2.2 Scalability . . . 27
- 4.3 Implementation of the opponent strategy . . . 27
- 4.4 Enhancing Decision-Making through Accumulated Step Training in Soccer Simulation . . . 29

4.5 Conclusion . . . 29

5 Experiments **30**
5.1 Introduction . . . 30
5.2 Environment . . . 30
5.2.1 Environment in the 1vs1 scenario . . . 31
5.2.2 Environment in the 2vs2 scenario . . . 31
5.2.3 Environment in the 3vs2 scenario . . . 31
5.2.4 Ball dynamic and strategies used by the opponent team for both 1vs1 and 2vs2 scenarios . . . 31
5.3 Training . . . 32
5.3.1 Marking in the 1vs1 scenario . . . 33
5.3.2 Marking in the 2vs2 scenario . . . 35
5.3.3 General defensive strategy in the 1vs1 scenario with ball interaction . . . 37
5.3.4 General defensive behavior in the 2vs2 scenario with Ball interaction . . . 39
5.3.5 Adapting Defensive Strategy in the 3vs2 Scenario with Ball Dynamics . . . 41
5.4 Baselines . . . 41
5.4.1 NeuroHassle Approach . . . 41
5.4.2 Stable Marriage . . . 42
5.5 Comparative Metrics . . . 43
5.5.1 Success Rate . . . 43
5.5.2 Average Distance to Goal . . . 44
5.6 Experimental Setup . . . 44
5.7 Conclusion . . . 46

6 Results **47**
6.1 Introduction . . . 47
6.2 Marking in the 1vs1 Scenario . . . 47
6.3 Marking in the 2vs2 Scenario . . . 49
6.4 General defensive strategy in the 1vs1 Scenario . . . 52
6.5 General defensive strategy in the 2vs2 scenario . . . 53
6.5.1 General defensive strategy in the 2vs2 scenario (Fundamental Learning Approach) . . . 53
6.5.2 Enhanced Training Approach for Defensive Strategy Refinement . . . 54
6.6 Marking and preventing from scoring goals in the 3vs2 scenario . . . 55
6.6.1 Outcome Analysis . . . 55
6.7 Comparative Study of Multi-Agent Reinforcement Learning Strategies in 2D Soccer Simulation . . . 56
6.7.1 Quantitative Comparative Analysis . . . 56
6.7.2 Assessing Defensive Effectiveness through Average Opponent-to-Goal Distance . . . 57
6.7.3 Qualitative Comparative Analysis . . . 58
6.8 Conclusion . . . 60

7 Conclusion and Future Work **62**
7.1 Conclusion . . . 62
7.2 Future Work . . . 62

Chapter 1

Introduction

1.1 Overview

Learning collaborative strategies in dynamic and unpredictable settings presents a significant challenge in the field of robotics and artificial intelligence. The RoboCup Competition[1] exemplifies this challenge, offering a platform where advancements in multi-agent systems and robotics research are put to the test. Particularly, the 2D RoboCup Soccer simulation, see Figure 1.1, stands out by creating an environment where autonomous agents must develop and execute sophisticated strategies to outmanoeuvre their opponents. This competition not only advances AI research but also serves as an invaluable arena for testing and enhancing teamwork and adversarial tactics among robotic agents.

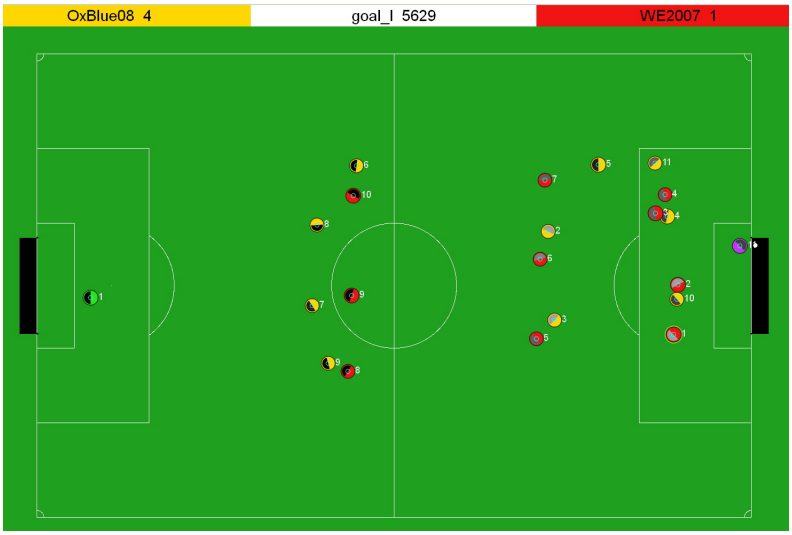


Figure 1.1: 2d soccer simulation environment

Within this context, one critical aspect of gameplay is an effective defensive strategy. Traditional handcrafted approaches to marking in these dynamic environments often lack adaptability and optimization, leading to sub-optimal defensive performance as explained by Smit *et al.* [2023]. Recognizing these limitations, there is a growing emphasis on leveraging Multi-Agent Reinforcement Learning (MARL) techniques as discussed by Ning and Xie [2024]. These techniques offer a promising avenue to develop more efficient and adaptive marking strategies that can dynamically respond to the fluid nature of the game, ensuring that the agents can effectively counter diverse offensive tactics.

[1] https://www.robocup.org/

To address the complexities of simulating collaborative and adversarial strategies in multi-agent systems, we developed our own custom environment, shown in Figure 1.2, tailored for in-depth analysis and strategy development. This bespoke environment was crafted to encapsulate the essential dynamics and challenges characteristic of multi-agent interactions, providing a robust platform for our research. Through detailed visualizations, we illustrate the intricacies of agent interactions and the strategic depth required to navigate these challenges, laying the groundwork for our exploration of defensive strategies in dynamic team-based scenarios.

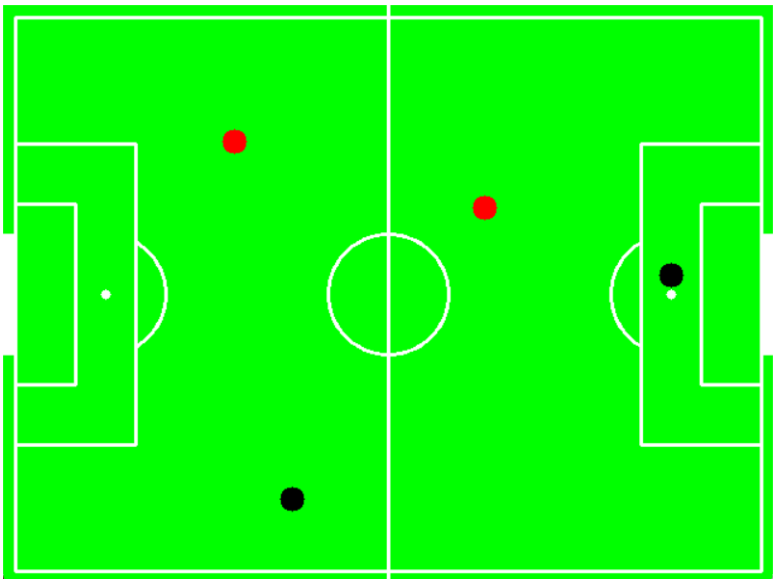


Figure 1.2: 2D Soccer Simulator

1.2 Research Problem

The complexity of cooperative problem-solving in dynamic and unpredictable environments underscores the importance of this research. The challenge lies in developing effective defensive strategies where agents must synchronize their actions, adapt collectively in real-time, and prioritize team success over individual goals. In the RoboCup soccer simulation, this challenge is intensified by the need for agents to interpret and predict opponent behaviours while coordinating with teammates under rapid and fluctuating conditions. The unpredictable tactics of adversaries necessitate continual strategic adjustments, making the development of a robust defensive strategy through Multi-Agent Reinforcement Learning (MARL) both critical and challenging.

Despite the progress in Multi-Agent Reinforcement Learning (MARL), existing solutions exhibit limitations when applied to dynamic and complex environments like RoboCup soccer. For instance, traditional MARL approaches may struggle with the non-stationarity of the environment, where the agents' policies continually change, affecting the learning stability as explained by Papoudakis *et al.* [2019]. Additionally, many existing models face challenges in effective communication and coordination among agents, as highlighted by Smith and Nau [2005] who pointed out that without proper communication protocols, agents could act based on outdated or incomplete information. Another limitation is the scalability of these solutions. As the number of agents increases, the complexity of the state and action spaces exponentially grows, making the learning process less efficient and more computationally demanding. For example, Yang [2018] discussed how increasing the number of agents in a cooperative task significantly impacts the learning convergence rate. Also, existing solutions often do not account for the adaptability required in real-time changing environments. For instance, Lanctot [2017] demonstrated that

many MARL approaches are not equipped to handle scenarios where opponents' strategies evolve during the course of interaction, leading to suboptimal performance in adversarial settings.

Many techniques employed in RoboCup soccer simulations have traditionally concentrated on offensive strategies, highlighting a gap in the development of comprehensive defensive tactics. Kyrylov and Hou [2007] focused primarily on the positioning of defenders, which, while crucial, represents only one facet of an effective defense. A robust defensive strategy should encompass both optimal positioning and proactive actions to disrupt or intercept the opponent player with the ball. While Gabel *et al.* [2009] offers an approach to address the latter, it largely concentrates on individual agent behavior. However, the inherently cooperative nature of RoboCup soccer necessitates a strategy that promotes synchronization and collaboration among multiple defenders, ensuring a unified and effective defensive front.

Our research presents an innovative MARL framework tailored for the RoboCup environment, dedicated to developing a sophisticated defensive strategy. This framework helps players not only position themselves and act against opponents but also introduces a new method that greatly improves how quickly and effectively they can adapt and work together. Crucially, our strategy extends beyond a single-agent focus, embracing multi-agent collaboration to optimize team performance comprehensively. This approach ensures a dynamic, well-rounded defensive strategy, enhancing both individual and collective efficacy in the complex, multi-agent context of RoboCup soccer. Our approach is specifically designed for the fast-paced and unpredictable nature of the RoboCup soccer simulation. It tackles the need for agents to make quick decisions, adapt to changing strategies from opponents, and work together as a cohesive unit while also demonstrating scalability and generalisation. By honing in on these key areas, our method directly improves how agents react and collaborate under pressure, enhancing their in-game performance.

1.3 Contributions

We describe the contributions of this research.

- Learning a Comprehensive Defensive Strategy: Our core achievement is the development of a defensive strategy that is both adaptive and robust. We have a method that trains agents to not only anticipate and counter opponents' moves but also to position themselves strategically on the field, creating a multi-faceted defensive approach that responds effectively to the dynamic nature of RoboCup soccer.

- Scaling to Multiple Agents: A significant aspect of our contribution is the successful scaling of our defensive strategy to involve multiple agents, namely two agents in our team. Unlike strategies focused on individual performance [Gabel *et al.* 2009], our approach ensures that agents learn to cooperate, coordinate, and make collective decisions, which is crucial for maintaining a strong defense in scenarios with more than one player on each team. This scalability demonstrates the potential of our strategy to enhance team performance in complex, multi-agent environments.

In evaluating the effectiveness of the proposed MARL strategy against established methods like NeuroHassle [Gabel *et al.* 2009] and the Stable Marriage [Irving *et al.* 1987] approach, we employed two key metrics: success rate and the average distance between opponent players and our goal. The success rate metric provides a direct measure of defensive effectiveness, quantifying the frequency at which our agents prevent opponents from scoring. Meanwhile, the average distance metric offers insights into the spatial effectiveness of our defensive strategy, illustrating how well our agents maintain positioning to reduce scoring opportunities for the opposing team.

1.4 Research Structure

Our research is structured in different chapters to provide a thorough investigation into the development and evaluation of defensive strategies within the 2D RoboCup Soccer simulation. It is organized into several key sections:

- Theoretical Background (Chapter 2): This section lays the foundational concepts and principles underlying our research, providing context and understanding of the key elements involved in multi-agent reinforcement learning and soccer simulation.
- Related Works (Chapter 3): Here, we first present how MARL has been used in other areas other than the RoboCup domain. We then delve into existing research, examining previous approaches to defensive strategies in similar contexts, and identifying gaps and opportunities that our research aims to address.
- Methodology (Chapter 4): This segment details our approach to developing and refining defensive strategies, from initial marking strategies to more advanced interactions involving ball handling and goal prevention, culminating in a challenging 3vs2 scenario.
- Conclusion (Chapter 7): The conclusion synthesizes our findings, reflecting on the implications of our research, the effectiveness of the strategies developed, and potential avenues for future exploration in the field of AI-driven soccer simulation.

Chapter 2

Theoretical Background

2.1 Introduction

In this chapter, we delve into the theoretical foundations and essential mathematical concepts underpinning Reinforcement Learning (RL) and Multi-Agent Reinforcement Learning (MARL). We explore various algorithms that drive advancements in both fields, focusing on their distinctive mechanisms and applications. A part of our discussion is dedicated to the Proximal Policy Optimization (PPO) algorithm, which we employ to cultivate the desired behavior in our agents.

2.2 Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning focused on the development of agents that learn decision-making through interactions with an environment. The interaction dynamic is illustrated in Figure 2.1, where an agent in state S_t performs an action A_t. The Environment responds by transitioning the Agent to the next state S_{t+1} and delivering a reward R_{t+1}. The Agent's objective is to develop an optimal policy—a strategy for action selection in various states—that maximizes cumulative rewards over time. This involves exploration of the environment, with the Agent adjusting its policy based on the outcomes of its actions to enhance future decisions.

Distinct from other machine learning paradigms that depend on pre-labeled data, RL algorithms adopt a trial-and-error learning method, obtaining evaluative feedback through rewards or penalties associated with their actions.

The cornerstone of RL is the utilization of a reward signal to direct the Agent's learning trajectory. Positive rewards prompt the Agent towards actions yielding favorable results, while negative rewards or penalties serve to deter actions leading to undesirable outcomes. This process of repeated interaction and incremental learning allows the Agent to progressively identify and adopt actions that optimize expected cumulative rewards.

RL has been successfully applied across various fields, including in the RoboCup competition [Stone *et al.* 2005], robotics [Kober *et al.* 2013], and gaming [Mnih *et al.* 2015]. It has achieved remarkable feats in complex tasks such as mastering the games of Go and Poker [Silver *et al.* 2016], navigating autonomous vehicles [Shalev-Shwartz *et al.* 2016], and optimizing energy use.

In essence, RL equips agents with the capacity to make intelligent choices in dynamic, uncertain environments, enabling them to gain experience and progressively enhance their performance.

2.2.1 Markov Decision Process

A Markov Decision Process (MDP) [Han 2018] is a mathematical framework used to describe the environment in RL. It provides a formalization where decisions are made, considering that

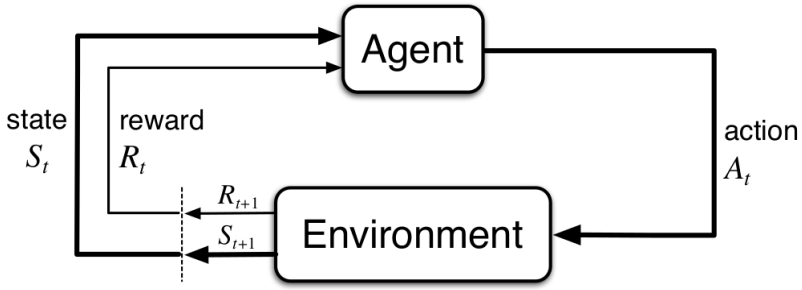


Figure 2.1: Illustration of agent-environment interaction [Sutton *et al.* 1998]

the outcome is partly random and partly under the control of a decision-maker. More formally, an MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where

1. $\mathcal{S}$ is the states space,
2. $\mathcal{A}$ is the actions space,
3. $\mathcal{P}$ is the state transition probability
$$\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1],$$
4. $\mathcal{R}$ is the reward function where
$$\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow \mathbb{R},$$
5. γ is the discount factor which is a trade-off between immediate and future rewards.

The policy $\pi : \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$ is the policy of the agent. The reward function $\mathcal{R}$ of the agent depends on the action $\mathcal{A}_t$, the state $\mathcal{S}_t$, then the return depends on the policy π. The return is the total accumulated reward the agent receives in the future, starting from the current state. It's the sum of rewards the agent expects to gather over time, often discounted by the factor γ to prioritize immediate rewards over distant ones. The return is formally defined as

$$G_t = \sum_{k=0}^{+\infty} \gamma^k \mathcal{R}_{t+k+1},$$

and we have the recursive relationship

$$G_t = \mathcal{R}_{t+1} + \gamma G_{t+1}.$$

In RL, understanding the potential future rewards for an agent's decisions is crucial. This is where the concepts of state value and action value functions come into play, providing insights into the expected rewards from different perspectives within the environment. The state value function indicates the expected total reward from a particular state, showing its overall desirability. Conversely, the action value function assesses the expected reward for taking a specific action in a given state, guiding the agent on which action is most beneficial. We define the state-value function and the action-value function respectively as follows

$$V_\pi(s) = \mathbb{E}[G_t | S_t = s, A_{t:\infty} \sim \pi] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_{t:\infty} \sim \pi\right]$$

and

$$Q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a, A_{t+1:\infty} \sim \pi] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a, A_{t+1:\infty} \sim \pi\right].$$

2.2.2 Optimal Policy

An optimal policy of an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ is a policy π^* such that for any $s \in \mathcal{S}$,

$$V_{\pi^*}(s) \geq V_\pi(s) \text{ for any } \pi,$$

meaning that the state value function under the optimal policy π^* is greater than or equal to the state value function under any other policy π . In other words, the optimal policy π^* yields the highest expected rewards from any state compared to all other policies. π^* is called the optimal policy.

The optimal policy can be derived from the optimal action-value function $Q^*_\pi(s, a)$ for any $(s, a) \in \mathcal{S} \times \mathcal{A}$ by

$$\pi^*(s) = \arg\max_a Q^*_\pi(s, a).$$

2.3 Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is an extension of reinforcement learning, involving the concurrent learning and interaction of multiple agents within a common environment [Buşoniu *et al.* 2010]. In MARL, each agent operates as an independent decision-maker, capable of perceiving its environment, executing actions, and receiving rewards based on those actions. The central objective of MARL is to equip these agents with the ability to develop effective strategies in sophisticated, interactive settings, where the actions of one agent influence the behaviors and outcomes of others.

MARL extends the concept of a Markov Decision Process (MDP) to scenarios involving multiple agents, as depicted in Figure 2.2. In this multi-agent context, each agent i engages in the Agent-Environment interaction similar to that in a single-agent RL setting. These agents may operate independently or with varying degrees of information sharing, influencing their learning and decision-making processes.

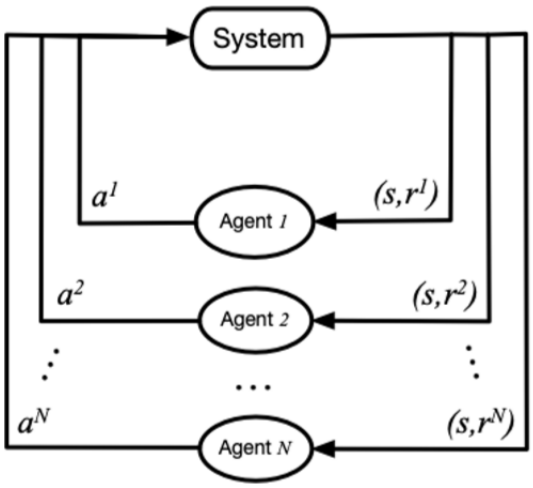


Figure 2.2: Illustration of multi-agents environment interaction [Zhang *et al.* 2021]

2.3.1 Overview of Stochastic Games

We often model MARL settings as stochastic games, which involve multiple decision makers interacting in an environment. More formally, stochastic games [Buşoniu *et al.* 2010] are defined as a tuple $(n, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where:

1. n represents the number of agents involved,
2. $\mathcal{S}$ denotes the state space,
3. $\mathcal{A} = (\mathcal{A}^i, i = 1, ..., n)$ is the combined action space for all agents, where $\mathcal{A}^i$ is the set of actions available to agent i,
4. $\mathcal{P}$ is the probability of state transitions, defined as $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$,
5. $\mathcal{R} = (\mathcal{R}^i, i = 1, ..., n)$ is the joint reward function, where $\mathcal{R}^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ specifies the reward function for agent i,
6. γ is the discount factor, balancing immediate and future rewards.

In MARL, state transitions result from the collective actions of all agents, denoted as $\mathcal{A}_t = (A_t^1, A_t^2, .., A_t^n)$, where $\mathcal{A}_t$ is part of $\mathcal{A}$ and A_t^i is the action taken by agent i at time t. The joint policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ governs the agents' actions, with each agent i having its policy $\pi^i : \mathcal{S} \times \mathcal{A}^i \to [0, 1]$. The reward for agent i depends on the collective action $\mathcal{A}_t$, the state $\mathcal{S}_t$, thus the return is influenced by the joint policy π. The return for agent i which is the total reward an agent is expected to collect in the future, beginning from a given state, is given by

$$G_t^i = \sum_{k=0}^{+\infty} \gamma^k \mathcal{R}_{t+k+1}^i,$$

with the recurrent relation

$$G_t^i = \mathcal{R}_{t+1}^i + \gamma G_{t+1}^i.$$

The state-value and action-value functions for agent i are defined respectively as

$$V_\pi^i(s) = \mathbb{E}[G_t^i | S_t = s, A_{t:\infty}^i \sim \pi] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}^i | S_t = s, A_{t:\infty}^i \sim \pi\right],$$

and

$$Q_\pi^i(s, a) = \mathbb{E}[G_t^i | S_t = s, A_t^i = a, A_{t+1:\infty}^i \sim \pi] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}^i | S_t = s, A_t^i = a, A_{t+1:\infty}^i \sim \pi\right].$$

2.3.2 Nash Equilibrium

In a stochastic game defined as $(\mathcal{S}, \mathcal{A} = (\mathcal{A}^i, i = 1, ..., n), \mathcal{P}, \mathcal{R} = (\mathcal{R}^i, i = 1, ..., n), \gamma)$, a Nash equilibrium, as described by Zhang *et al.* [2021], is a joint policy $\pi^* = (\pi^{1,*}, \pi^{2,*}, ..., \pi^{n,*})$ where, for any state $s \in \mathcal{S}$ and for any agent i,

$$V_{\pi^*}^i \geq V_\pi^i \text{ for any } \pi.$$

Here, π^* is considered the optimal joint policy.

2.3.3 Adversarial, Cooperative, and Mixed Environments

Stochastic games can be categorized based on the nature of interactions among agents [Zhang *et al.* 2021]:

Fully Cooperative Games

In a fully cooperative stochastic game, all agents share the same reward function, denoted as $\mathcal{R}^1 = \mathcal{R}^2 = ... = \mathcal{R}^n$. Consequently, all agents have identical returns, $G_t^1 = G_t^2 = ... = G_t^n$, with the shared goal of maximizing the common return.

Fully Adversarial Games

Conversely, in a fully adversarial stochastic game, each agent aims to maximize its benefit, often at the expense of others. In a two-agent scenario ($n = 2$), if $\mathcal{R}^1 = -\mathcal{R}^2$, agents have opposing objectives. In a more general adversarial game with n agents, the sum of all agents' rewards equals zero, $\sum_{k=1}^{n} \mathcal{R}^k = 0$.

Mixed Games

Mixed games represent neither fully cooperative nor fully adversarial settings. Such games involve multiple teams of agents, each team working cooperatively internally to maximize their collective benefit, represented as $\mathcal{R}_i^1 = \mathcal{R}_i^2 = ... = \mathcal{R}_i^n = \mathcal{R}_i$ for team i among m teams. Simultaneously, these teams compete against each other such that the sum of the teams' rewards is zero, $\sum_{i=1}^{m} \mathcal{R}_i = 0$.

2.3.4 Centralized and Decentralized Approaches in MARL

MARL employs two primary approaches to learning and decision-making [Zhang *et al.* 2021]: centralized and decentralized.

Centralized Approach

In the centralized approach, as depicted in Figure 2.3, agents have access to the global state and can coordinate their actions. This approach allows agents to reason about the entire state of the environment, leading to more informed and collaborative decision-making. Despite this global coordination, the execution of actions remains decentralized, with each agent acting independently.

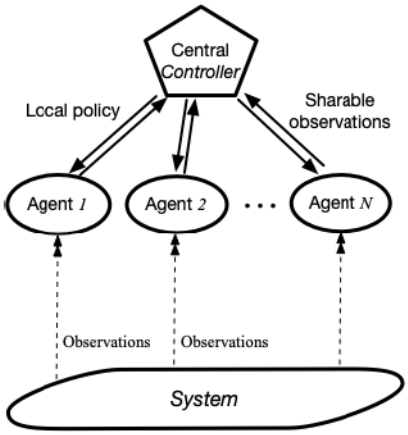


Figure 2.3: Centralized approach in MARL [Zhang *et al.* 2021]

Centralization often involves a central controller or algorithm that collates information from all agents, making joint action recommendations. This method is particularly effective in scenarios requiring high levels of cooperation and coordination.

Decentralized Approach

The decentralized approach, illustrated in Figure 2.4, involves each agent making decisions based on their local perspective, without comprehensive communication or coordination with others. Agents act autonomously, optimizing their objectives based on their individual observations.

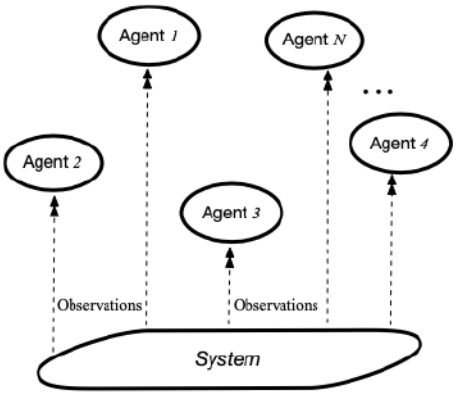


Figure 2.4: Decentralized approach in MARL [Zhang *et al.* 2021]

This approach is suitable for situations where communication is constrained or agents have conflicting goals. Decentralization can lead to emergent behaviors that arise from the collective but independent actions of agents, although it may sometimes result in suboptimal outcomes due to the lack of coordination.

2.4 Overview of RL and MARL Algorithms

Within Reinforcement Learning (RL), various methodologies have been developed to optimize agents' decision-making processes. These can broadly be categorized into value-based methods, policy-based methods, and a hybrid of both known as mixed methods. Each approach offers unique strategies for learning and decision-making in RL environments.

Value-Based Methods

Value-based methods are centered around the estimation and optimization of value functions, which quantify the expected return from specific states or state-action pairs. The core goal is to ascertain an optimal value function that can effectively maximize cumulative rewards. This function subsequently aids in deriving the optimal policy for the agent.

For instance, algorithms such as Q-learning [Watkins and Dayan 1992] and SARSA [Zhao *et al.* 2016] are quintessential examples of value-based methods. Q-learning, a fundamental algorithm in this category, operates by maintaining a Q-value table to approximate the value of state-action pairs. It utilizes the Bellman equation for updating Q-values and iteratively refines the policy. The Deep Q-Network (DQN) [Mnih *et al.* 2015] extends Q-learning to handle complex, high-dimensional state spaces by employing deep neural networks to approximate the Q-values.

In the realm of Multi-Agent Reinforcement Learning (MARL), techniques like Independent Q-learning [Abed-Alguni *et al.* 2016] and Independent SARSA have been prominent. These methods facilitate each agent to learn independently, without considering the actions of other agents, thus simplifying the learning process in multi-agent contexts.

Policy-Based Methods

Policy-based methods, in contrast, directly parameterize the policy – a function mapping states to actions. These methods aim to find the optimal policy that maximizes expected rewards by adjusting the policy's parameters. They are particularly adept at managing continuous and high-dimensional action spaces, a limitation often encountered in value-based methods.

Prominent examples of policy-based algorithms include REINFORCE [Sutton and Barto 2018], Proximal Policy Optimization (PPO), in 2.5, [Schulman *et al.* 2017], and Trust Region Policy Optimization (TRPO) [Schulman *et al.* 2015]. REINFORCE employs gradient ascent to update policy parameters, calculating the gradient of the expected return for policy optimization. PPO and TRPO are more advanced, offering improved stability and sample efficiency. These algorithms incorporate additional constraints or modifications in the policy update process, enhancing the overall effectiveness of the learning.

Hybrid of Policy and Value-Based Methods

Hybrid, or mixed methods, combine the strengths of both value-based and policy-based approaches. By integrating value functions to guide policy updates and utilizing policy gradients to refine value functions, these methods seek to capitalize on the advantages of each individual approach.

A notable hybrid approach is the Actor-Critic architecture, comprising two components: an actor (policy) and a critic (value function). The actor directs the search for optimal policies, while the critic evaluates the potential of states or state-action pairs, offering a balanced approach to learning.

In the scope of Multi-Agent Deep RL (MADRL), advancements have been focused on adapting deep RL methods to multi-agent scenarios. Strategies such as Centralized Training with Decentralized Execution (CTDE) [Li *et al.* 2020] have emerged, where agents are trained using centralized information but act independently during deployment. Actor-Critic models, such as MADDPG [Lowe *et al.* 2017] and COMA [Foerster *et al.* 2018], have been developed for multi-agent systems. These models aim to optimize joint policies that not only maximize collective rewards but also effectively manage the intricacies of multi-agent interactions.

2.5 Proximal Policy Optimization (PPO)

This section delves deeper into Proximal Policy Optimization (PPO), which is integral to our study. We chose PPO for our research because PPO offers stable and reliable learning, is well-suited for environments with high-dimensional input spaces, and handles continuous action spaces effectively, making it ideal for the complex and dynamic nature of soccer simulations. PPO represents a significant advancement in the field of reinforcement learning (RL), particularly in policy gradient methods. Introduced by Schulman *et al.* [2017], PPO has become popular for its simplicity, efficiency, and general applicability to a wide range of problems, from video game AI to robotics. At its core, PPO aims to address the critical challenges of data efficiency and training stability in policy optimization tasks.

Background

To appreciate the novelty of PPO, it is essential to understand the context in which it was developed. Policy gradient methods work by adjusting the parameters of a policy in the direction that maximizes the expected return. Early approaches, like REINFORCE [Sutton and Barto 2018], demonstrated the potential of policy gradients but suffered from high variance and inefficiency. Techniques such as Trust Region Policy Optimization (TRPO) [Schulman *et al.* 2015]

sought to improve stability by limiting policy updates to a trust region, ensuring that new policies would not deviate too far from old ones. However, TRPO's complexity and computational requirements limited its accessibility and applicability.

Proximal Policy Optimization: Core Principles

PPO inherits the motivation behind TRPO—maintaining training stability by controlling the size of policy updates—but achieves this goal more efficiently and simply. PPO introduces a novel objective function that penalizes changes to the policy that move the probability ratio away from 1, effectively keeping the updates within a predefined range. This mechanism allows PPO to take larger, more stable steps during optimization.

The Objective Function

The PPO objective function incorporates the concept of probability ratio, defined as $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, where π_θ denotes the policy parameterized by θ, a_t is the action taken at time t, and s_t is the state at time t. The function penalizes the ratio $r_t(\theta)$ when it moves outside of the interval $[1-\epsilon, 1+\epsilon]$, with ϵ being a small positive number, typically around 0.1 to 0.2. The clipped objective function can be written as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

where:

- θ represents the policy parameters.
- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio.
- $\hat{A}_t$ is an estimator of the advantage function at time t, which measures the benefit of taking action a_t in state s_t relative to the baseline expectation.
- ϵ is a hyperparameter, typically set to a small value like 0.1 or 0.2.

Advantages of PPO

1. Simplicity and Efficiency: Unlike TRPO, which requires complex second-order optimization, PPO achieves similar benefits with first-order optimization methods, making it easier to implement and computationally less intensive.
2. Robust Performance: PPO has demonstrated robust performance across a variety of domains, including continuous and discrete action spaces.
3. Balance Between Exploration and Exploitation: By limiting the extent of policy updates, PPO naturally balances exploration and exploitation, reducing the likelihood of catastrophic policy collapses.

Implementation Details

Implementing PPO, as highlighted in the pseudocode 1, involves iterating over a sequence of steps where the agent interacts with the environment, collects experiences, and updates the policy based on the collected data. Key implementation aspects include:

- Rollouts: Collecting a set of trajectories by executing the current policy in the environment. Each trajectory consists of states, actions, rewards.

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
    Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
    Compute rewards-to-go $\hat{R}_t$.
    Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
    Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}^{\pi_{\theta_k}}(a_t, s_t), g(\epsilon, \hat{A}^{\pi_{\theta_k}}(a_t, s_t)) \right),$$

    typically via stochastic gradient ascent with Adam.
    Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\max_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_{\phi_k} - \hat{R}_t \right)^2,$$

    typically via some gradient descent algorithm.
end for

- Advantage Estimation: Calculating the advantage $\hat{A}_t$ for each time step, typically using Generalized Advantage Estimation (GAE) for a balance between bias and variance. The advantage function $\hat{A}_t$ estimates how much better an action is compared to the average action at state s_t. It can be computed using the Temporal Difference (TD) error:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1}$$

  where $\delta_t = R_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD error, and λ is the trace-decay parameter.

- The core of the PPO-Clip algorithm is updating the policy. It involves adjusting the policy parameters to favor actions with positive advantage. The update is constrained (clipped) to prevent drastic policy changes that could lead to unstable learning.

- Value Function Update: In parallel, the value function, which estimates the expected return from states, is refined by minimizing the difference between its predictions and the observed rewards-to-go.

Challenges and Considerations

While PPO is powerful, it's not without challenges. Hyperparameter selection, particularly the clipping parameter ϵ and the coefficients for entropy regularization, can significantly impact performance. Additionally, PPO's effectiveness in very high-dimensional action spaces or extremely complex environments may require further innovations in policy representation and optimization techniques.

Proximal Policy Optimization stands as a landmark in reinforcement learning, offering a blend of efficiency, simplicity, and robust performance. Its development reflects a broader trend in RL towards algorithms that are not only theoretically sound but also practical for a wide range of applications.

2.6 Conclusion

In this work, we explored the fundamentals of Reinforcement Learning (RL), focusing on its mathematical underpinnings and the principles that guide decision-making processes. We explained key concepts like state and action value functions, which help predict the rewards of actions and states, and discussed how an optimal policy is determined in RL. Extending these concepts, we ventured into Multi-Agent Reinforcement Learning (MARL), where the dynamics become more complex due to the interactions between multiple agents. We discussed various algorithms that operate within RL and MARL frameworks, highlighting their distinctive features. Our exploration culminated in a detailed examination of the Proximal Policy Optimization (PPO) algorithm, which stands at the heart of our research, providing a deeper understanding of its mechanism and effectiveness in learning behaviors within these environments.

Chapter 3

Literature Review

3.1 Introduction

In this chapter, we first start by presenting the RoboCup Soccer competition and the different categories embedded into it. We then delve into the Multi-Agent Reinforcement Learning (MARL) research, exploring its application within diverse cooperative and competitive settings. This exploration is pivotal, as it lays the groundwork for understanding how MARL strategies can be effectively deployed to enhance team dynamics and individual agent performance in complex, dynamic environments. We examine both centralized and decentralized MARL approaches, illustrating how these methodologies are tailored to meet the demands of various interaction paradigms within multi-agent ecosystems.

Special attention is given to notable strategies within the RoboCup domain, such as the half-field offense and the Keepaway approach, which offer insights into how MARL facilitates strategic positioning, ball control, and cooperative team behaviors. Furthermore, we scrutinize the NeuroHassle approach, a method that exemplifies the application of MARL for proactive defensive maneuvers, where agents learn to disrupt and intercept opponents to gain a tactical advantage. We also discuss the effectiveness of PPO in games.

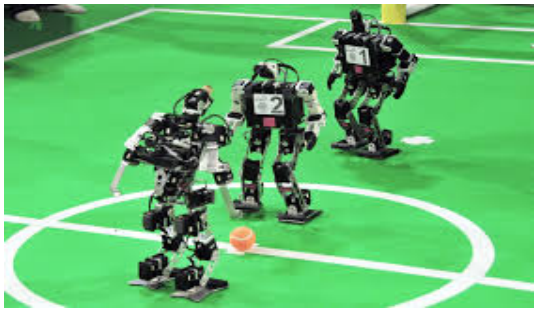
By synthesizing these diverse strands of MARL research, this chapter aims to provide a comprehensive backdrop for our subsequent investigation into leveraging MARL to refine and enhance marking strategies in the 2D RoboCup Soccer competition, contributing to the broader discourse on intelligent agent design and cooperative problem-solving in artificial intelligence.

3.2 Overview of the RoboCup Competition

The RoboCup Competition is an international robotics event that aims to advance the field of artificial intelligence and robotics. It serves as a platform for researchers and developers to showcase their innovative solutions in a competitive environment. The ultimate goal of RoboCup is to develop robot soccer players capable of beating the human FIFA World Cup champions by 2050. The RoboCup soccer competition is divided into several leagues, each with its unique set of challenges, such as the humanoid, small size, and simulation leagues, illustrated in Figure 3.1. The humanoid league aims to advance robots to the point where they can compete against humans, aligning with RoboCup's ultimate goal.

3.2.1 3D RoboCup simulation league

Figure 3.2 illustrates the utilization of a 3D RoboCup simulator, which facilitates soccer matches with 3D virtual humanoid robots. In this simulation, each team comprises 11 players who, at each timestep of the game, acquire selective, player-centric data on visible objects and stationary markers on the field. Instead of raw visual data, players are furnished with processed information



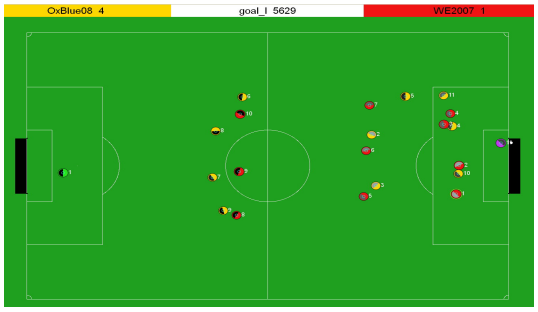
(a) Humanoid league



(b) Small size league



(c) 3D simulation league



(d) 2D simulation league

Figure 3.1: Different leagues in the RoboCup Soccer Competition

detailing the position and movement of each observed object, encapsulating distances, angles, and velocity changes relative to the player. Additionally, players obtain self-related data, including fatigue levels, joint angles, acceleration metrics, and auditory communications from teammates. This information enables players to adjust their actuators accordingly. The 3D simulated environment demands players to master basic maneuvers like walking and dribbling while developing team tactics, presenting a multifaceted challenge.

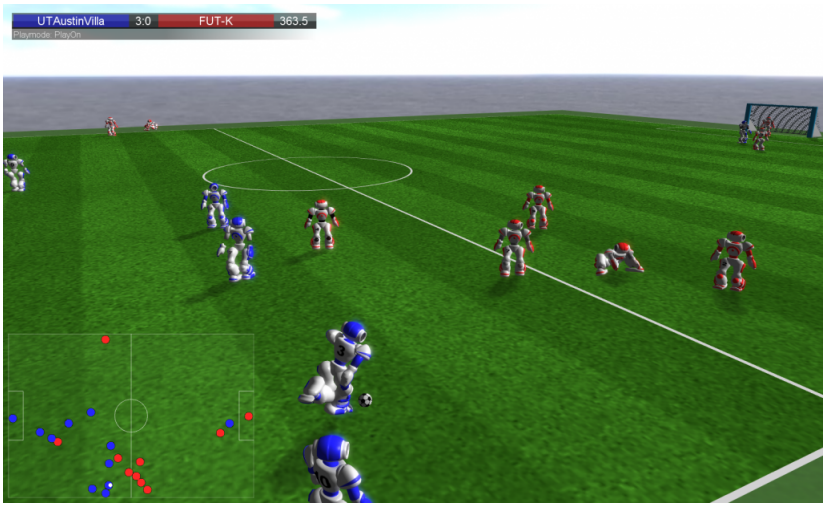


Figure 3.2: 3D soccer simulation environment

3.2.2 2D RoboCup simulation league

The RoboCup 2D Simulation League, as depicted in Figure 3.3, operates on a virtual two-dimensional pitch, featuring two teams with 11 autonomous players each, capable of intermittent communication. The field hosts 22 players, including one goalkeeper per team, complemented

by a coach who offers enhanced game insights and guidance. A digital referee oversees the game, making calls like the ball exiting the play area.

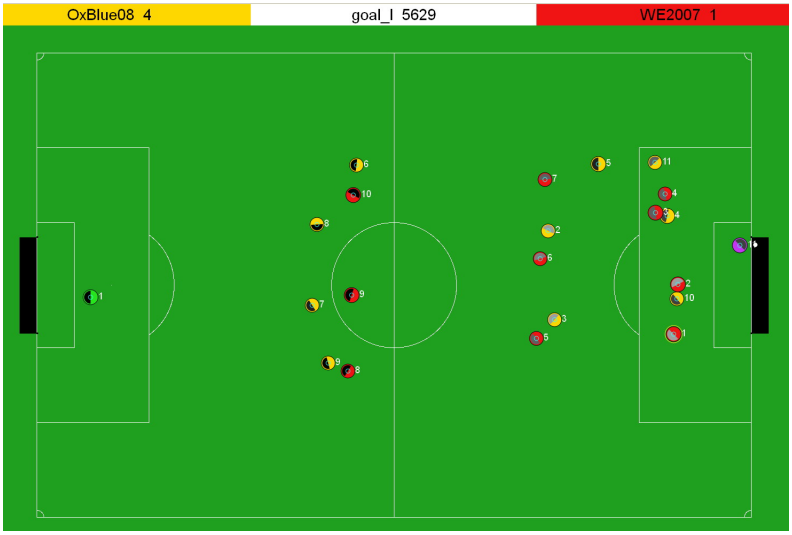


Figure 3.3: 2D soccer simulation environment

In the visual representation, players appear as circles split into two colors: red for one team yellow for the other team. Goalkeepers are distinctly colored for easy identification. Players interact with their virtual environment via auditory, visual, and physical sensors. The auditory sensor facilitates communication with teammates, coaches, the referee, and even opponents, simulating real-world verbal interaction. Messages may be lost with distance, except those from the coach and referee. Players receive auditory data every second simulation step, equating to 100 milliseconds. Players get details on object positions and velocities, with options to tweak vision quality and angle, affecting information update frequency. Visual data includes intentional noise to mimic real-world inaccuracies. Players influence their environment by moving forwards or backward, rotating, and adjusting neck orientation, which allows for nuanced control, especially when kicking the ball, aligning the ball's trajectory with the player's neck direction for strategic play.

3.3 Multi-Agent Reinforcement Learning in Centralized and Decentralized Approaches

In various real-world scenarios, numerous tasks necessitate the collaboration of multiple agents. These agents often operate under conditions of limited communication and have only a partial view of their environment. Such settings pose significant challenges for learning algorithms, primarily due to the dynamic nature of the environment caused by the simultaneous actions of multiple agents. This dynamism often renders the environment non-stationary from the perspective of any single agent, complicating the learning process [Panait and Luke 2005].

In many practical situations, agents are required to execute specialized policies tailored to specific tasks. However, the applicability of these specialized policies in real-world settings is frequently constrained. One key challenge is that agents are expected to learn and retain distinct policies for each specific task, which increases the complexity of the learning process. Furthermore, agents often face difficulties in real-world scenarios where the identities of the tasks they are performing may not be readily observable or distinguishable [Taylor and Stone 2009].

The paradigm of Centralized Training with Decentralized Execution (CTDE) is a cornerstone in the field of MARL. In CTDE, agents are trained collectively using central information but are designed to operate independently in execution. A quintessential illustration of this concept is

seen in actor-critic methods, where a central critic evaluates the global state of the environment, while independent actors make decisions based on their individual perceptions [Foerster *et al.* 2018].

Despite the widespread adoption of CTDE, particularly in scenarios necessitating coordinated multi-agent actions, the implications of employing a central critic in these methods have not been thoroughly investigated. Lyu *et al.* [2021] address this lacuna by conducting a formal analysis that contrasts central and decentralized critic methods in MARL [Lyu *et al.* 2021]. Their research provides critical insights into the dynamics of these approaches, elucidating the nuanced strengths and weaknesses inherent in each. By empirically evaluating these methods in a spectrum of settings, Lyu *et al.* [2021] not only dispel prevalent misconceptions in the existing literature but also illuminate the conditions under which one approach might be favored over the other. They conclude that central critic designs do not universally guarantee superior performance; rather, the decision to employ a central or decentralized critic should be informed by the specific requirements and constraints of the task at hand.

These studies collectively underscore the evolving landscape of MARL, highlighting the importance of carefully considering the design of learning algorithms, especially in relation to how agents acquire and apply knowledge in shared environments [Zhang and others 2019]. The ongoing exploration and refinement in this field are instrumental in advancing our understanding of complex multi-agent systems and their applications in real-world scenarios.

3.4 Multi-Agent Reinforcement Learning in RoboCup

3.4.1 Keepawawy strategy in the 2D simulation

The Keepaway task, as initially introduced by Stone *et al.* [2005], constitutes a distinctive and strategic component of the RoboCup soccer framework. It stands as an invaluable experimental platform for testing and refining reinforcement learning techniques within a simulated soccer environment. In the Keepaway setting, the primary objective revolves around a team of "keepers" who endeavor to maintain control of the ball while contending against a team of "takers". The gameplay unfolds within a specifically delineated area, and each episode of the game concludes either when the takers successfully seize the ball or when the ball exits the designated play zone. The overarching goal for the keepers is to prolong the duration of each episode, thereby demonstrating effective ball retention skills. To quantitatively measure performance, the keepers receive rewards that correlate with the time elapsed following each specific action they undertake. This reward structure incentivizes strategies focused on sustained ball control. Stone *et al.* [2005] employ a Sarsa-based reinforcement learning approach, which enables the keeper agents to assimilate and execute complex strategies that enhance their efficacy in maintaining ball possession. In our work, we employ the Keepaway strategy as an offensive tactic for the opponent players. This approach facilitates the learning and enhancement of our team's defensive strategies during training sessions.

3.4.2 Half-Field Offense

Expanding upon the foundation laid by the Keepaway task, the Half Field Offense task, as developed by Kalyanakrishnan *et al.* [2007], introduces a more intricate and challenging dimension to the 2D RoboCup soccer simulation. This task transpires over a significantly larger portion of the soccer field and incorporates a greater number of agents on each team. The primary aim for the offensive team in this scenario is to orchestrate goal-scoring opportunities. This involves not only maintaining possession of the ball but also strategically maneuvering it towards and successfully attacking the opponent's goal.

In the half-field offense strategy within the RoboCup soccer domain, teams scale up to three agents, each endowed with the ability to choose from a set of parameterized actions, specifically

dribble, pass_k, shoot. The "pass_k" action denotes a direct kick to the teammate who is the k-th closest to the ball, avoiding reliance on fixed jersey numbers and instead utilizing a dynamic, distance-based indexing system.

The "dribble" action is designed to motivate the ball-carrying offensive player to advance toward the goal. This is facilitated through a strategy where the player kicks the ball within a goal-directed cone, the angle of which adjusts based on proximity to the goal and defensive players, promoting forward movement yet allowing evasion of close defenders. The "shoot" action is executed by kicking the ball towards the goal, aiming through the widest available angle not obstructed by the goalie, defenders, or goalposts. Offensive players without ball possession engage in distinct behaviors: the nearest player to the ball will attempt to gain possession, while others maintain formation, positioning themselves optimally to support the attack.

Additionally, this approach incorporates inter-agent communication, enabling agents to share information and synchronize their actions effectively, drawing on protocols akin to those employed in artificial predator-prey learning environments. Such communication enhances the team's strategic coherence and responsiveness, crucial for executing coordinated offensive maneuvers in the dynamic and competitive setting of RoboCup soccer. Moreover, similar to the Keepaway approach [Stone *et al.* 2005], this offensive strategy employs a SARSA-based reinforcement learning framework, enabling agents to learn and refine their actions based on ongoing experiences within the simulation.

3.4.3 NeuroHassle in learning a defensive strategy

A pertinent investigation into defensive scenarios in this context is presented by Gabel *et al.* [2009], who underscore the criticality of defensive maneuvers, particularly in situations that necessitate active disruption of an opponent's control over the ball. Their study zeroes in on scenarios where an agent is required to intervene and disrupt an opponent who has the ball, with the intent to thwart offensive plays and potentially regain control for initiating a counter-attack.

The approach adopted by Gabel *et al.* [2009] employs reinforcement learning techniques to facilitate agents in independently acquiring assertive behaviors suited for one-on-one duels. The NeuroHassle approach, while innovative, concentrates its training on four distinct field regions as highlighted in Figure 5.9: top left, bottom left, quarter left, and midfield, within a 1vs1 scenario involving one defender and one attacker. This segmentation offers targeted insights but may limit generalizability across the entire soccer field. Furthermore, the focus on individual duels may not fully encapsulate the complexities of team-based interactions and strategies. Gabel *et al.* [2009] uses TD(0) and function approximation to handle the continuous nature of the state space. The learning agent is allowed to use dash(x) and turn(y) commands where the domains of bots commands' parameters ($x \in [-100, 100], y \in [-180, 180]$) are discretized such that in total 76 actions are available to the agent at each time step.

The approach described by Gabel *et al.* [2009], NeuroHassle, employs a form of TD-learning, namely TD(0), to approximate the value function $V(s)$. This method involves training a neural network to estimate the value of each state, which is then used to determine the optimal actions via a one-step lookahead strategy.

The neural network used in NeuroHassle comprises an input layer corresponding to the state space, a hidden layer, and an output layer that predicts the value of a given state. For a state space dimension of 9 [Gabel *et al.* 2009], the network architecture is defined as follows:

- Input layer: 9 neurons (state dimension)
- Hidden layer: 18 neurons
- Output layer: 1 neuron (value of the state).

In our project, we build upon the foundations laid by NeuroHassle, expanding its scope to encompass more dynamic and variable team interactions. By adapting and extending the core

principles of NeuroHassle, we develop strategies that are not confined to specific field regions or 1vs1 encounters. Our enhanced approach maintains the adaptability and strategic depth of NeuroHassle while introducing scalability to accommodate larger team sizes and varied game scenarios.

3.4.4 Multi-Agent PPO

Recent research by Yu *et al.* [2022] demonstrated the effectiveness of the proximal policy optimization algorithm in cooperative multi-agent environments, incorporating modifications like enhanced inputs for the value function via agent-specific global states, optimized utilization of training data, and a reduced PPO clipping value. In our research, we implement these suggested modifications to optimize agent coordination and decision-making processes.

In our work, we adopt a variation of the proximal policy optimization (PPO) algorithm, which we refer to as Central PPO (CPPO). The "C" in CPPO signifies our use of a centralized setting, where PPO serves as the central controller or learning mechanism. This centralized approach allows us to manage and coordinate multiple agents within the environment, leveraging the strength of PPO [Yu *et al.* 2022] to optimize their collective actions.

3.5 Conclusion

In this chapter, we discussed the RoboCup Soccer competition by presenting the different categories. We focused more on the 2D league, which is a pivotal arena for showcasing and evaluating the advancements in MARL and robotic cooperation. This chapter also provided an examination of the landscape of Multi-Agent Reinforcement Learning (MARL) research, delving into both centralized and decentralized methodologies within various cooperative and competitive settings. The discussion highlighted how MARL is adeptly applied to enhance team dynamics and strategies in the context of RoboCup soccer, an environment that exemplifies the complexity and unpredictability inherent in real-world scenarios. By reviewing key strategies like the half-field offense and the Keepaway approach, along with their respective action frameworks and the incorporation of inter-agent communication, we gain a comprehensive understanding of current methodologies and their applications. These explorations not only showcase the adaptability and potential of MARL but also set the stage for our research. We aim to extend these foundations by applying MARL to develop advanced defensive strategies, addressing the unique challenges presented by the 2D RoboCup Soccer competition. This endeavor seeks not only to contribute to the field of artificial intelligence and robotics but also to advance our capability to orchestrate intricate team behaviors in complex, dynamic environments.

In addition to the strategies discussed, the NeuroHassle approach emerges as a noteworthy method within the RoboCup domain, emphasizing proactive disruption and ball interception to undermine opponents' offensive maneuvers. This approach, which pivots on the concept of individual agents learning to assertively confront and disrupt ball carriers, serves as a valuable point of comparison for our research. While NeuroHassle offers a robust strategy for individual agent behavior, our focus extends to harnessing MARL for coordinated team-based defensive tactics, aiming to cultivate a synergy among agents that transcends individual action. By integrating the insights garnered from NeuroHassle and other MARL applications, our research aspires to innovate within the defensive playbook of RoboCup soccer.

Chapter 4

Methodology

4.1 Introduction

The core objective of our research is to develop a mechanism to learn a defensive strategy using MARL, within the context of the 2D RoboCup Soccer simulation. This endeavour aims to equip the agents with the capabilities necessary to not only understand their spatial positioning in relation to opponents but also to effectively counter various offensive maneuvers. Our methodology is divided into three main phases.

The initial phase, Section 4.2, is dedicated to learning a defensive strategy using MARL in a simulated soccer environment. We present the defensive learning framework used throughout our dissertation requiring a predefined scenario under which CPPO can perform. Here CPPO represents a centralised version of Multi-agent RL whereby we have extended the standard PPO [Schulman *et al.* 2017] to output a single action for each of the agents in a particular scenario. In this way, we have a single model controlling all the agents involved for our team.

Section 4.3 discusses the implementation of the strategies used by the opponent team throughout the training of our agents in acquiring defensive behaviour.

Lastly, we discuss enhancing decision-making through Accumulated Step Training in Soccer Simulation useful to add more variability to provide enough information for the agents to make a strategic decision.

4.2 Defensive Behaviour Learning Framework

We look to learn Multi-Agent Defensive Strategies within a simulated soccer environment. These agents need to learn appropriate coordination such that they can learn to defend against opponents whose goal it is to score. We therefore, designed a framework as seen in Figure 4.1. This framework requires a defined scenario from which our Multi-agent CPPO process can operate. These Scenarios are defined in the Experiments, chapter 5, and represent both the nature of the goal we wish to learn as well as the characteristics of the environment. Based upon the provided scenario we utilise reward shaping and state space design in order to guide our agent to correctly learning a defensive strategy. It is important to note that the opponents' strategy is not a feature of the scenario but rather an unseen behaviour that our agents will need to learn to beat. After the reward and state design is completed, we use this information in a traditional MARL training pipeline. Here the CPPO agent interacts within an environment whose reward signal was defined in the previous step. This Environment component is partially dependent on the Scenario, once again discussed in the Experiments, chapter 5. Once training converges we utilise the learned policy in comparison to both learned and designed baselines in order to measure the quality of the multi-agent defensive strategy.

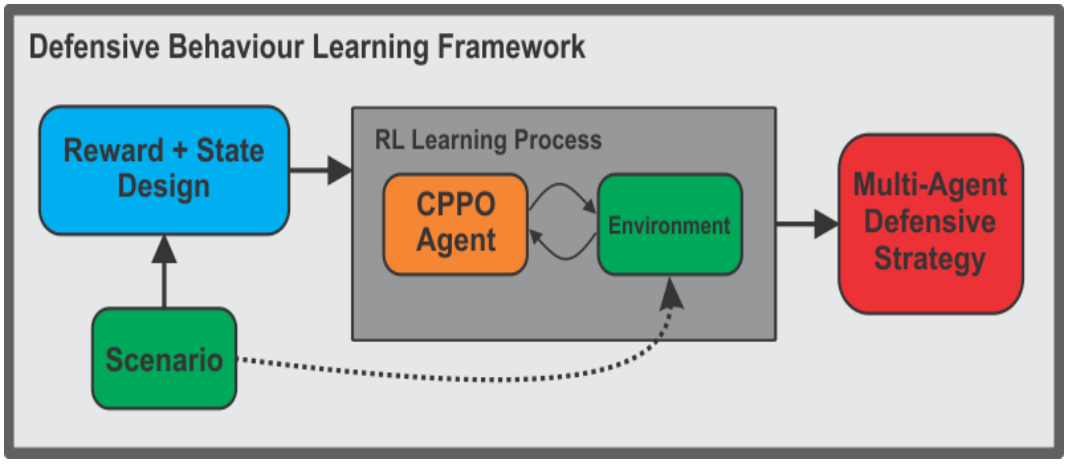


Figure 4.1: MDP problem definition process

4.2.1 Multi-Agent Defensive Strategy

State and action spaces and reward function

We formalize the problem as an MDP since we use CPPO for training.

- State space: The state space is constituted by the position of all the agents and their corresponding velocities. More formally,

$$S = \{x^i_{def}, y^i_{def}, x^j_{opp}, y^j_{opp}, V^i_{xdef}, V^i_{ydef}, V^j_{xopp}, V^j_{yopp}\},$$

  where $i \in [1, 2, \ldots, n]$ and $j \in [1, 2, \ldots, m]$ and n and m are respectively the number of defenders and offenders, whose values are either both 1 in the 1vs1 scenario or 2 in the 2vs2 scenario.

- Action space: The action space is defined as

$$A = \{v_x, v_y\},$$

  where v_x and v_y denote the velocities along the x-axis and y-axis, respectively.

- Reward: Depending on the scenario we deal with, we define the appropriate reward.

4.2.2 Scalability

In order to accommodate more agents, we access the adaptability of our Central Proximal Policy Optimization (CPPO) approach as the number of agents and the complexity of the environment increase. By adjusting the state space we are able to freely expand to larger settings, that is the the 3vs2 scenario where we have two defenders and three attackers.

4.3 Implementation of the opponent strategy

In the context of developing general defensive behaviors for autonomous agents in the RoboCup simulated soccer environment, our training methodology incorporates three strategies: Random Direction Changes, Keepaway, and Half Field Offense. These strategies are chosen not only for their relevance to the simulated soccer domain but also for their relationship, where Keepaway serves as a foundational subtask within the broader context of Half Field Offense.

Random Direction Changes

By training agents in these scenarios, we aim to achieve a balance between specific skill acquisition and the development of generalizable strategies that can adapt to the dynamic and unpredictable nature of full-scale soccer matches. To assess and improve our marking agent's adaptability to unpredictable movements, we implemented a Random Direction change scenario within their training regimen. This scenario aimed to mimic the unpredictability of human soccer, where players often change direction suddenly to evade defenders or create attacking opportunities. The opponent players can randomly pick one direction among eight available, namely north, south, west, east, northwest, southwest, northeast, and west-east.

- Mechanism: Opponents change their movement direction randomly, adhering to the new trajectory for 100 time steps. This interval was chosen to balance between providing enough time for the agent to recognize and react to the change and simulating the rapid shifts in direction typical of a soccer game.
- Objective: By integrating this unpredictability, the scenario tests the defensive agent's ability to quickly adapt their positioning and strategies in response to sudden changes in the offensive dynamics of the opposing team.

The impact on training is

- Enhanced Adaptability: This approach was pivotal in training our agents to maintain effective marking against dynamic and unpredictable offensive maneuvers, a common challenge in actual soccer matches.
- Strategic Flexibility: Agents developed a more flexible marking strategy, improving their capability to handle real-world scenarios where opponents exhibit non-linear and erratic movement patterns.

This focused intervention not only enriched our agent's tactical repertoire but also provided insights into optimizing marking formations and reactions in the face of unforeseen opponent strategies.

Keep Away: The Foundational Subtask

Keepaway is recognized as a fundamental subtask in RoboCup soccer, focusing on ball possession and control [Stone *et al.* 2005]. In this setup, one team (the possession team) aims to maintain control of the ball within a predefined small rectangular area, evading the opposing team's attempts to intercept. This task emphasizes spatial awareness, quick decision-making, and the ability to maneuver in tight spaces, laying the groundwork for more complex strategic behaviors.

Half Field Offense: Extending Keepaway

Building upon the skills and strategies developed through Keepaway, the Half Field Offense [Kalyanakrishnan *et al.* 2007] scenario introduces a more complex and dynamic challenge. In this task, an offensive team works to outmaneuver a defensive team with the ultimate goal of scoring. Unlike Keepaway, Half Field Offense encompasses the entire half of a soccer field, significantly expanding the operational area and introducing the objective of goal-scoring. This scenario demands a higher level of strategic planning, coordination among team members, and the integration of ball control with offensive maneuvers to penetrate the defense and create scoring opportunities.

4.4 Enhancing Decision-Making through Accumulated Step Training in Soccer Simulation

One pivotal aspect of our training methodology is the adoption of an "accumulated step" approach, where one step during the training process is composed of six ordinary steps. This technique is crucial for enhancing the decision-making process of the agents. In the context of our soccer simulation, an ordinary step represents a minimal movement or action taken by an agent, which on its own may not significantly contribute to a discernible change in the game's state or provide ample information for the agent to make a strategic decision.

By aggregating six ordinary steps to form one accumulated step, we ensure that the agents experience a more pronounced and informative change in their environment within a single training step. This accumulation allows the agents to observe the consequences of a sequence of actions rather than just a single, often inconsequential, action. As a result, the agents can better understand the dynamics of the game and the effects of their actions, leading to more informed and strategic decision-making. During evaluation, we use the ordinary time steps.

4.5 Conclusion

In conclusion, our methodology chapter delineates a structured, phased approach to developing defensive strategies in the 2D RoboCup Soccer simulation. In Phase 1, we present the defensive learning framework useful in acquiring the desired behaviour. We presented the different strategies employed by the opponent team during the training process of our agents. The goal here is for our agent, after being trained on these unseen strategies, will beat them. Lastly, we discuss a technique used throughout our training useful in adding more information into the state space.

This comprehensive methodology underscores our commitment to a holistic exploration of defensive strategies, setting a robust foundation for understanding and enhancing agent performance in soccer simulations.

Chapter 5

Experiments

5.1 Introduction

In this chapter, we delve into the core of our research, presenting the experimentation of the study aimed at learning a defensive strategy in the scenario of soccer, using Multi-Agent Reinforcement Learning in the 2D RoboCup soccer simulation. The primary focus is on evaluating the effectiveness of Central Proximal Policy Optimization (CPPO) in developing a robust marking strategy, specifically compared to traditional methods like NeuroHassle [Gabel *et al.* 2009] and Stable Marriage [Irving *et al.* 1987] in the 2D Soccer simulation. Following the assessment of these strategies, the research transition to exploring the implementation of CPPO in orchestrating a comprehensive general defensive strategy, aiming to further enhance the tactical acumen and overall performance of the agents within the simulated environment.

This chapter is structured to methodically walk through the experimental setup, detailing the implementation of the marking strategy and more generally the defensive strategy, the training process, and the evaluation metrics employed.

The findings are critically examined, not only in the light of our predefined success criteria but also in comparison with established baselines, namely the NeuroHassle strategy [Gabel *et al.* 2009] and the Stable Marriage approach [Irving *et al.* 1987].

In section 5.2, we discuss the environment setup in all the scenarios we discuss in this dissertation. Section 5.3 discusses the training process. Namely, we discuss how we define the action and state spaces in more detail and how the reward function is designed in each specific scenario to get the optimal behavior we are looking for. The implementation of baselines and assumptions we take into account are discussed in section 5.4. Section 5.5 dives into the metrics used to compare our method with the different baselines, namely the NeuroHassle and Stable Marriage approaches. Finally, we discuss the experimental setup in section 5.6.

All the hyperparameters used throughout the training can be found at Appendix 7.2.

5.2 Environment

The 2D RoboCup soccer simulation stands as a cornerstone in the field of artificial intelligence and robotics, offering a dynamic and challenging platform for exploring the intricacies of Multi-Agent Reinforcement Learning (MARL). This simulation environment serves not only as a competitive arena for robotic teams but also as a rich ground for academic research and technological advancements.

Given the complexity of the 2D soccer simulation, we decided to design a simplified environment from scratch to learn the marking and more generally defensive strategies. Once the model was trained, we utilized the MuJoCo physics engine [Todorov *et al.* 2012] through the mujoco-py Python wrapper for testing. Figure 5.2 represents the environment we built from scratch. The dimensions of the soccer field are 640 units in width and 480 units in height.

We defined the environment to imitate the way the 2D soccer simulation works. Indeed, the 2D soccer simulation provides us at any time step with the positions of all the agents including opponents as well as their corresponding velocities. It also provides directions where the agents face but in this context, it is irrelevant since we did not take it into account for simplification. The advantage of using our environment over the original 2D simulator is the processing time. Indeed, the data processing takes less time and is less noisy as is the case in the simulator. Moreover, the environment is simplified to make the training faster, and our focus on the desired behavior we are looking for.

5.2.1 Environment in the 1vs1 scenario

The environment we use in the 1vs1 scenario is seen in Figure 5.1. Our player is black and the opponent player is red. Each player has a radius of 15 units.

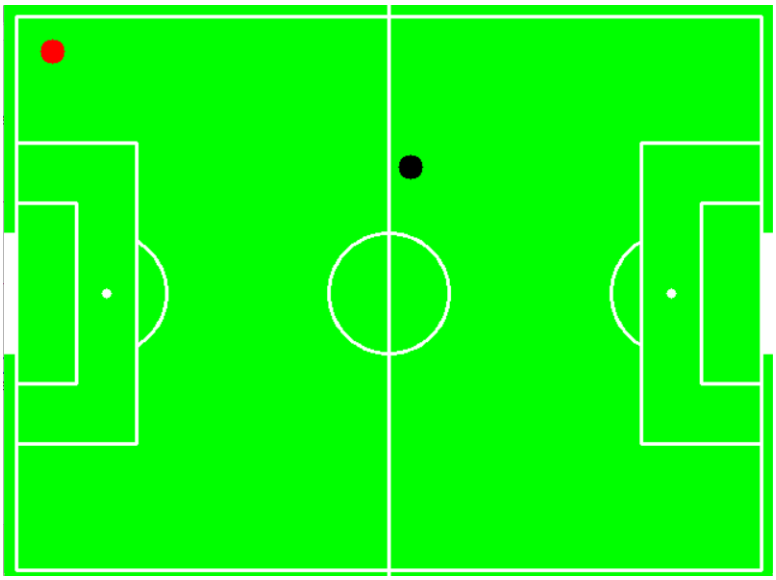


Figure 5.1: 1vs1 Environment

5.2.2 Environment in the 2vs2 scenario

The environment we use in the 2vs2 scenario is shown in Figure 5.2. The black players are ours and the red players are the opponents.

5.2.3 Environment in the 3vs2 scenario

The environment we use in the 3vs2 scenario is shown in figure 5.3. This environment particularly takes into account the ball dynamic. In this situation, we have two defenders, black players, and three attackers, red players. Attention has to be paid to the blue player. Indeed, as explained in section 5.2.4, the opponent player with the ball has a blue colour, and when it passes the ball to its teammate, it retrieves its red colour and the new ball carrier now has a blue colour.

5.2.4 Ball dynamic and strategies used by the opponent team for both 1vs1 and 2vs2 scenarios

Ball dynamic

In our study, we employed an approach to simulate ball dynamics within the RoboCup soccer framework, focusing on color-coded player representations to indicate possession of the ball.

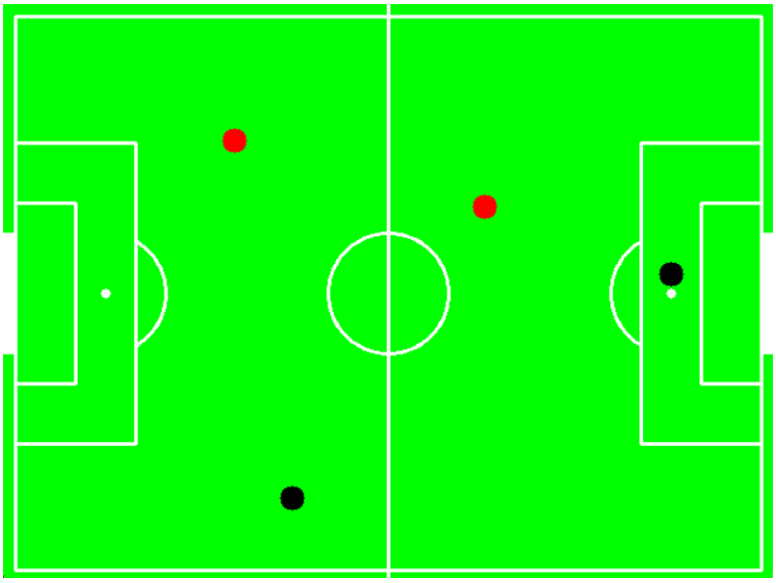


Figure 5.2: 2vs2 Environment

Players on the field are distinguished by their colours, with our team's players assigned a black colour and the opponent players a red colour. To indicate ball possession, we deviated from the traditional approach of including a separate ball entity. Instead, the player currently in possession of the ball is highlighted in blue, visually signaling that this player is the current ball carrier. This design choice simplifies the perception of ball possession for both the agents and the researchers, streamlining the learning and evaluation process.

Dynamic Ball Possession Representation

The dynamic nature of ball possession is captured through a straightforward colour transition mechanism. Initially, an opponent player holding the ball is marked in blue, while its teammates retain their red coloration, indicating they are not in possession of the ball. When the ball is passed to a teammate, the color transition occurs: the original player reverts to red, and the new ball carrier is now colored blue. The interception can happen but in a way that, if our player is to intercept the opponent player with the ball, it intercepts both the ball and the player.

The ball dynamic, characterized by color-coded possession, was implemented to focus our agent's learning on preventing the opponent team from scoring goals. A goal is considered scored if the opponent player carrying the ball (indicated by blue color) comes within a predefined distance threshold from a designated goal location in our territory. This setup simplifies the learning objectives, directing the agent's attention towards identifying and intercepting the blue-colored opponent to thwart scoring attempts, effectively teaching our agents defensive strategies centered around spatial awareness and goal protection.

5.3 Training

As mentioned in section 2.5, hyperparameter selection, particularly the clipping parameter ϵ and the coefficients for entropy regularization, can significantly impact performance. The entropy regularization does not impact the training much as we took it to be zero and the clipping $\epsilon = 0.2$ during the whole experiment. The other hyperparameters, depending on the setup used, are defined in appendix 7.2.

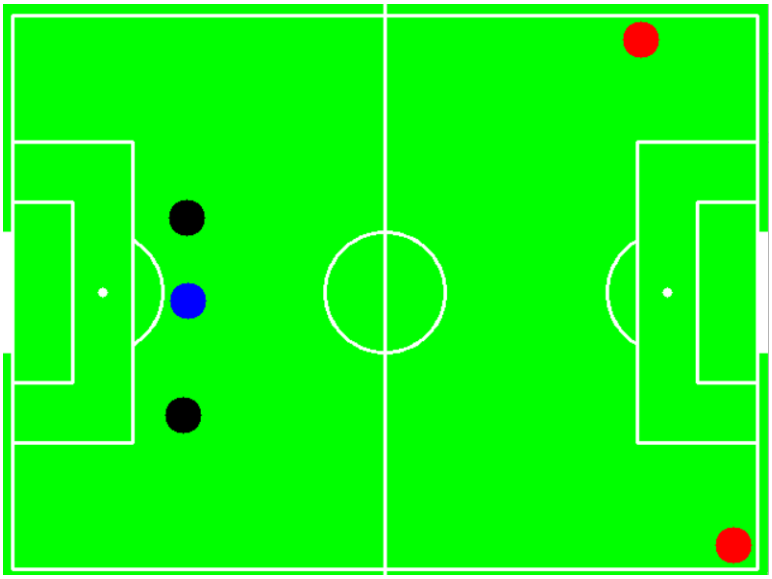


Figure 5.3: 3vs2 Environment

5.3.1 Marking in the 1vs1 scenario

This section delves into the details of training the Central Proximal Policy Optimization (PPO) model in a 1vs1 as seen in Figure 5.4, scenario within the 2D RoboCup soccer simulation environment. The focus lies on the definition of the state and action spaces. In this scenario, the opposing player randomly alters its direction, maintaining these new trajectories for 100 time steps.

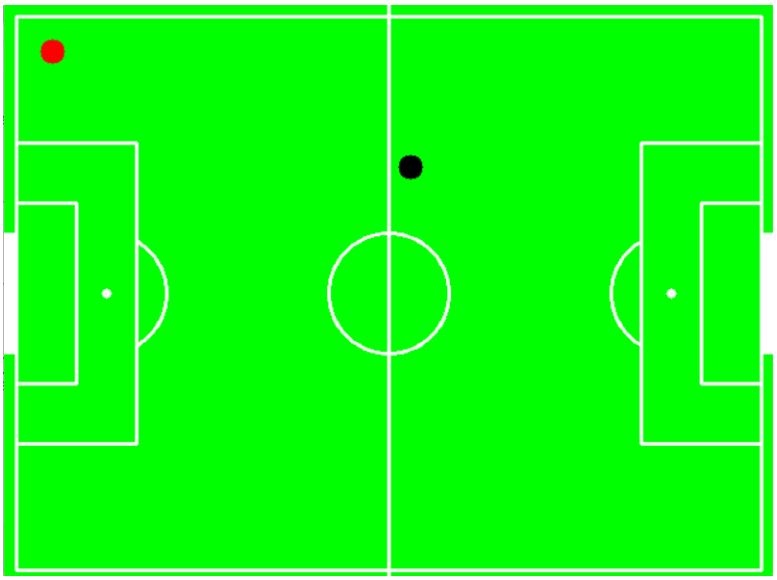


Figure 5.4: 1vs1 Environment

State Space

The state space in this scenario is defined by the positions and velocities of both the defending agent (our agent) and the opponent agent. Mathematically, the state space S is represented as:

$$S = \{x_{\text{def}}, y_{\text{def}}, x_{\text{opp}}, y_{\text{opp}}, v_{x_{\text{def}}}, v_{y_{\text{def}}}, v_{x_{\text{opp}}}, v_{y_{\text{opp}}}\}$$

where $(x_{\text{def}}, y_{\text{def}})$ and $(v_{\text{xdef}}, v_{\text{ydef}})$ denotes the coordinates and velocities of the defending agent and $(x_{\text{opp}}, y_{\text{opp}})$ and $(v_{\text{xopp}}, v_{\text{yopp}})$ represents the coordinates and velocities of the opponent agent. At each time step, the simulator provides us with the positions and velocities of all the agents.

Action Space

The action space in the 1vs1 scenario is defined with a focus on the movement dynamics of the defending agent. Specifically, the actions are represented in terms of velocity along the x and y axes of the simulation environment. The choice of velocity as the defining parameter for the action space allows for a more nuanced control over the agent's movement, facilitating precise and responsive maneuvers.

Formally, the action space A is represented as:

$$A = \{v_x, v_y\}$$

where v_x and v_y denote the velocities along the x-axis and y-axis, respectively. Each action taken by the agent, $a \in A$, is thus a vector:

$$a = (v_x, v_y)$$

This vector dictates the rate of change of the agent's position in the simulation environment, providing a direct and effective means of influencing its movement.

Given the physical constraints of the simulated environment and the agents, the velocities v_x and v_y are bounded. These bounds ensure realistic and feasible movements, aligning the simulation closer to real-world scenarios. The bounds are defined as:

$$v_{\text{min}} \leq v_x, v_y \leq v_{\text{max}}$$

where $v_{\text{min}} = -1$ and $v_{\text{max}} = 1$ are the minimum and maximum velocity limits, respectively.

Reward

The reward function is designed to encourage the defending agent to closely mark the opponent agent. It is defined as follows:

$$R(s,a) = \begin{cases} 1 & \text{if } d \leq 50 \\ -d & \text{otherwise} \end{cases} \tag{5.1}$$

where $R(s,a)$ is the reward given a state s and action a, and d is the Euclidean distance between the defending and opponent agents, calculated as:

$$d = \sqrt{(x_{def} - x_{opp})^2 + (y_{def} - y_{opp})^2}$$

The threshold value of 50 units is set to define the proximity required for successful marking. If the distance d is less than or equal to this threshold, a reward of 1 is given, indicating successful marking and the episode ends. Otherwise, the agent receives a negative reward proportional to the distance, incentivizing it to minimize this distance. This reward design is motivated by the work accomplished by Trott *et al.* [2019].

5.3.2 Marking in the 2vs2 scenario

This section provides an in-depth exploration of training the Central Proximal Policy Optimization (CPPO) model in a 2vs2 setting, as depicted in figure 5.2, within the 2D RoboCup soccer simulation. Emphasis is placed on defining the state space, action space, and reward function.

Building upon the Random Direction Changes scenario from the 1vs1 setup, we extended this framework to a 2vs2 case to further challenge our defensive agents by adding one more player to each team. In this expanded scenario, both opposing players now randomly alter their direction, maintaining these new trajectories for 100 time steps. This extension aims to simulate more complex and dynamic interactions, testing our agents' ability to adapt to multiple sources of unpredictability simultaneously. It emphasizes not only individual adaptability but also coordination between the two defensive agents, crucial for maintaining an effective defense against a pair of attackers exhibiting erratic movement patterns.

State Space

In the 2vs2 scenario, the state space is expanded to include the positions and velocities of both our defending agents and the opposing agents. Mathematically, the state space S is represented as:

$$S = \{x^i_{def}, y^i_{def}, x^j_{opp}, y^j_{opp}, v^i_{xdef}, v^i_{ydef}, v^j_{xopp}, v^j_{yopp}\}$$

where (x^i_{def}, y^i_{def}), and (v^i_{xdef}, v^i_{ydef}) denote the coordinates and velocities of the defending agents, and (x^j_{opp}, y^j_{opp}) and (v^j_{xopp}, v^j_{yopp}) represent the coordinates and velocities of the opponent agents, with $i, j \in \{1, 2\}$. The simulator provides the positions and velocities of all agents at each time step.

In brief, the Central Proximal Policy Optimization (PPO) agent in this context operates by taking as its input the joint states of all the agents involved in the scenario. This means that the central PPO agent receives a comprehensive set of data that encapsulates the current status (or states) of each individual agent. These states typically include information such as positions and velocities.

Action Space

In the 2vs2 scenario, the action space focuses on the movement dynamics of both defending agents. The actions are defined in terms of the velocities along the x and y axes for each defending agent. This choice allows for detailed control over each agent's movement, enabling precise and coordinated maneuvers.

Formally, the action space A for each defending agent is represented as:

$$A = \{v^1_x, v^1_y, v^2_x, v^2_y\}$$

where v^1_x, v^1_y and v^2_x, v^2_y denote the velocities along the x and y axes for the first and second defending agents, respectively. Each action taken by the agents, $a \in A$, is a vector:

$$a = (v^1_x, v^1_y, v^2_x, v^2_y)$$

This vector governs the rate of change of the agents' positions in the environment, offering a direct way to influence their movements.

Considering the physical limits of the simulation, the velocities $v^1_x, v^1_y, v^2_x, v^2_y$ are bounded, ensuring realistic agent movements. These bounds are defined as:

$$v_{\min} \leq v^1_x, v^1_y, v^2_x, v^2_y \leq v_{\max}$$

with $v_{\min} = -1$ and $v_{\max} = 1$ as the minimum and maximum velocity limits.

Based on this input, the central PPO agent processes the information and outputs the joint actions for all of our agents. Essentially, it determines the actions that each of our agents should take at any given moment. These actions are 'joint' in the sense that they are coordinated and decided upon collectively for all agents, ensuring that the actions of one agent are in harmony with and complementary to the actions of the others.

Reward

In our study, we experimented with various reward functions to optimize the performance of our agents in the 2vs2 marking scenario. After thorough testing and evaluation, we identified and selected the optimal reward function that consistently enhanced the effectiveness and efficiency of our agents' defensive behavior in this specific scenario.

Reward function 1

In our multi-agent reinforcement learning framework, the reward function is designed to motivate each agent in our team to mark the closest opponent. This is achieved by calculating the distances between each of our agents and the opponent agents and considering the minimum distance for each pairing. The reward function is then formulated based on the average of these minimum distances.

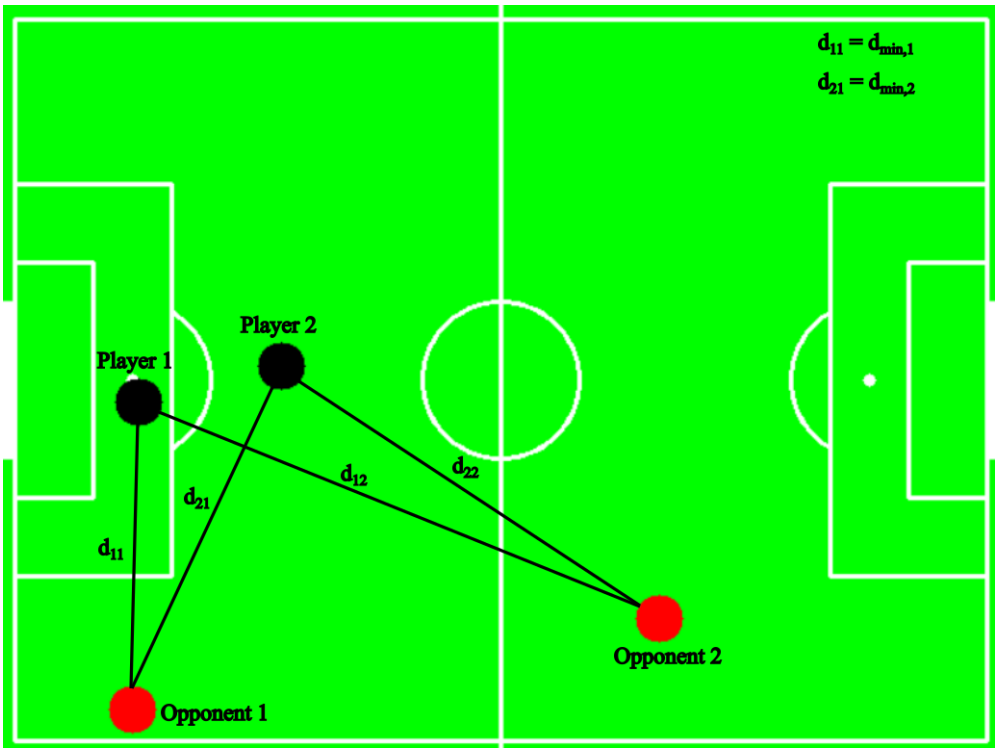


Figure 5.5: Reward design

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ denote our team of agents and $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$ denote the team of opponent agents with $n, m = 2$ in this case. The position of each agent a_i in our team is given by (x_{a_i}, y_{a_i}), and the position of each opponent agent o_j is given by (x_{o_j}, y_{o_j}).

The Euclidean distance d_{ij} between our agent a_i and an opponent agent o_j is calculated as:

$$d_{ij} = \sqrt{(x_{a_i} - x_{o_j})^2 + (y_{a_i} - y_{o_j})^2}$$

For each agent a_i in our team, the minimum distance $d_{\min,i}$ to any opponent agent is determined as:

$$d_{\min,i} = \min_{j \in \{1,2,\ldots,m\}} d_{ij}$$

The reward function R is then defined based on the average of these minimum distances $\bar{d}_{\min}$, calculated as follows:

$$\bar{d}_{\min} = \frac{1}{n} \sum_{i=1}^{n} d_{\min,i}$$

The reward R is computed as:

$$R = \begin{cases} 1 & \text{if } \bar{d}_{\min} < 50 \\ -\bar{d}_{\min} & \text{otherwise} \end{cases}$$

This reward structure ensures that each agent in our team is incentivized to stay within 50 units of the nearest opponent, with the goal of achieving an average minimum distance of less than 50 units. If this criterion is met, a reward of 1 is assigned, otherwise, the negative of the average minimum distance is assigned as the reward, encouraging the agents to reduce this distance in subsequent actions. Also to discourage the agents from marking the same opponent as is the case in Figure 5.5, we penalized the agents with a negative reward of -1000. Indeed, this happens when our players have the same closest opponent, meaning that $d_{\min,1} = d_{\min,2}$.

Reward function 2

In the context of the real soccer scenario, an effective marking strategy often involves assigning each player to a specific opponent. This approach ensures focused marking, where each player is responsible for tracking and neutralizing a particular opponent's influence in the game. Motivated by this strategic consideration in real-world soccer, we designed an alternative reward function that aligns with this marking strategy.

Our reward function is formulated based on the distances between each player in our team and a specifically assigned opponent in the opponent team. Let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ represent our team of players, and $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$ denote the opponent team, with $n, m = 2$ being the number of players in each team. Each player a_i in our team is assigned to mark a specific opponent o_i in the opponent team.

The position of player a_i is denoted by (x_{a_i}, y_{a_i}), and the position of their assigned opponent o_i is denoted by (x_{o_i}, y_{o_i}).

The reward function R is then defined based on the average of these distances $\bar{d}$, which is calculated as:

$$\bar{d} = \frac{1}{n} \sum_{i=1}^{n} d_i,$$

where d_i is the Euclidean distance between player a_i and their assigned opponent o_i. The reward R is computed as:

$$R = \begin{cases} 1 & \text{if } \bar{d} < 50 \\ -\bar{d} & \text{otherwise.} \end{cases}$$

5.3.3 General defensive strategy in the 1vs1 scenario with ball interaction

Defining the state and action spaces

The state and action spaces are identical as is the case in section 5.3.1.

Reward

In this advanced scenario, the reward function is meticulously designed to reflect the dual objectives of marking the opponent and preventing goals. We define a goal-scoring event in terms of the distance d_1 between the ball carrier, who is the opponent, and the goal location, which is denoted as (x_0, y_0). Specifically, a goal is considered to be scored if:

$$d_1 = \sqrt{(x_{\text{opponent}} - x_0)^2 + (y_{\text{opponent}} - y_0)^2} < 50 \text{ units}$$

Under this condition, our agent receives a substantial penalty of -1000, and the episode terminates, signifying the importance of preventing the opponent from scoring.

Calculation of the Intermediate Reward

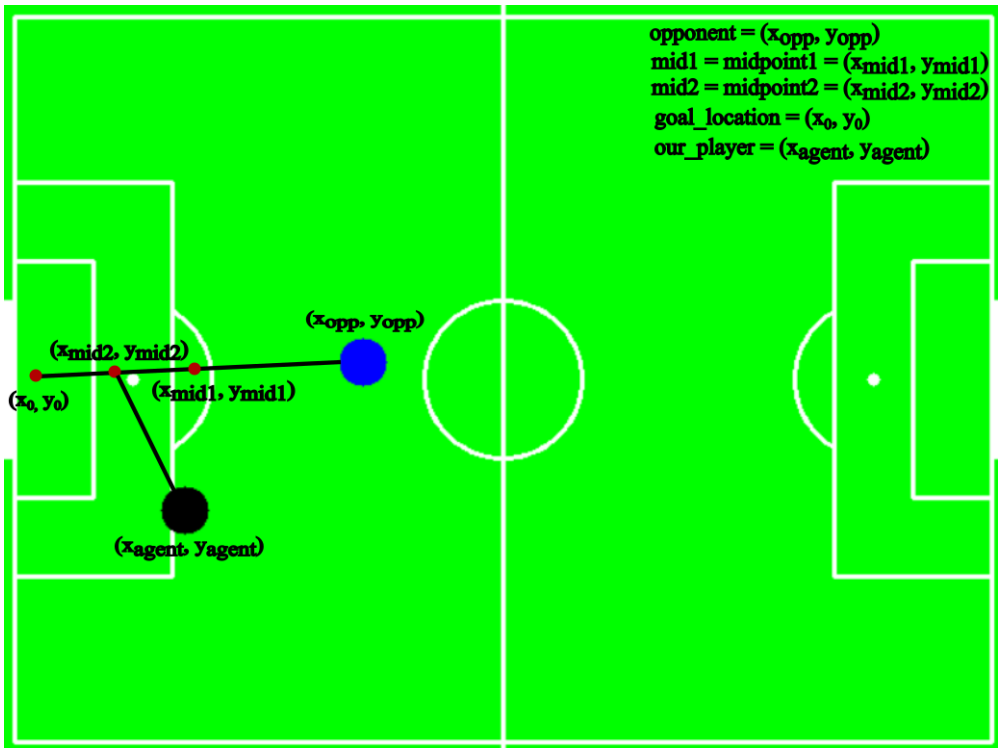


Figure 5.6: Reward Calculation

To encourage proactive defense beyond merely preventing goals as seen in Figure 5.6, we calculate an intermediate strategic position that our agent should aim to occupy. This position is determined by first computing the midpoint between the ball carrier and the goal location, and then calculating a second midpoint closer to the goal from the first midpoint. The reason for designing the reward in this way is that since we are learning a defensive strategy, we want our agent to position itself in a strategically defensive manner, that is, depending on the proximity of the opponent to the goal. Then if the opponent player is far from the goal, our defender will not go that far from our goal, and if the opponent player is closer, our player will position himself in the opponent agent's way to intercept him and the ball, therefore, preventing him from scoring.

The position of the second midpoint, which we denote as $(x_{\text{mid2}}, y_{\text{mid2}})$, is calculated as follows:

$$(x_{\text{mid1}}, y_{\text{mid1}}) = \left(\frac{x_{\text{opponent}} + x_0}{2}, \frac{y_{\text{opponent}} + y_0}{2} \right)$$

$$(x_{\text{mid2}}, y_{\text{mid2}}) = \left(\frac{x_{\text{mid1}} + x_0}{2}, \frac{y_{\text{mid1}} + y_0}{2} \right)$$

The reward at each timestep is then defined as the negative distance between our agent and this second midpoint:

$$R = -\sqrt{(x_{\text{agent}} - x_{\text{mid2}})^2 + (y_{\text{agent}} - y_{\text{mid2}})^2}$$

Time Step Penalty

Additionally, to expedite the learning process, we impose a time step penalty of -1 on our agent. This penalty serves as an incentive for the agent to learn the desired behavior more efficiently, encouraging quicker and more effective interventions to prevent the opponent from scoring.

$$R_{\text{time step}} = R - 1.$$

5.3.4 General defensive behavior in the 2vs2 scenario with Ball interaction

State action spaces

In the 2vs2 scenario, we define both the state and action spaces the same way as we did in section 5.3.2.

Reward

In this section, we explore various approaches to identify the most effective strategy for teaching our agents the desired defensive behavior. This involves experimenting with different training methodologies, reward structures, figure 5.8, and agent coordination mechanisms. Our goal is to develop a robust defensive strategy that enables agents to effectively mark opponents and prevent them from scoring, adapting to the dynamic challenges presented in a 2vs2 setup.

Reward design 1

In this scenario, the reward function was designed similarly as we did in 5.6. The opponent with the ball (ball carrier) is always indexed by 1.

$$\begin{aligned}
d_0 &= \textit{distance}(\text{goal location}, \text{ball carrier}) \\
d_1 &= \textit{distance}(\text{our player } 1, \text{mid}_{12}) \\
d_2 &= \textit{distance}(\text{our player } 1, \text{mid}_{22}) \\
d_3 &= \textit{distance}(\text{our player } 2, \text{mid}_{12}) \\
d_4 &= \textit{distance}(\text{our player } 2, \text{mid}_{22})
\end{aligned}$$

The reward R is computed as:

$$R = \begin{cases} -1000 & \text{if } d_0 < 50 \\ -D & \text{otherwise, where } D = \dfrac{\sum_{i=1}^{4} d_i}{4} \end{cases}$$

We devised a reward function to train our agents to adopt an effective defensive strategy in a soccer simulation. The reward function is structured as follows: if the distance between the opponent's ball carrier and our goal location is less than 50 units, indicating a high threat level, the agent receives a substantial negative reward of -1000. This penalization mechanism is

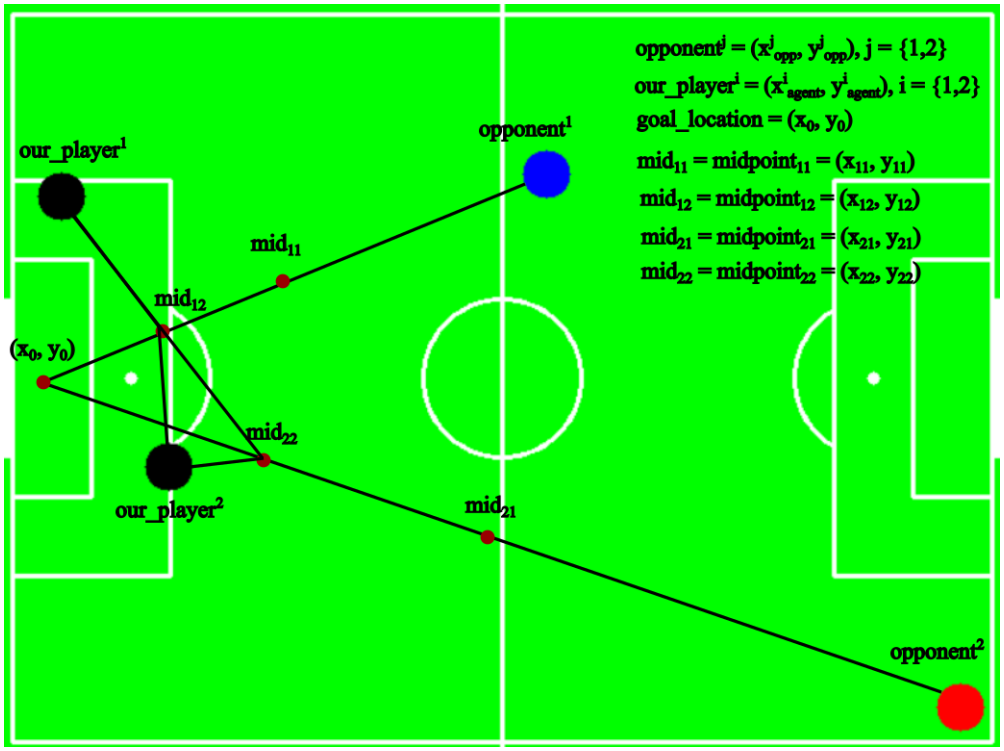


Figure 5.7: Reward Calculation

designed to instill a sense of urgency in maintaining a tight defense and preventing the opponents from nearing the goal area.

Conversely, when the ball carrier is not in immediate proximity to the goal, the reward is based on the average distance D between opponent players and our goal, which encompasses multiple factors: the agents are encouraged to monitor not only the ball carrier but also other opponent players and maintain a strategic position that balances defensive integrity and readiness to act. This component of the reward function ensures that our agents learn to position themselves optimally, staying close enough to intervene when necessary while also not straying too far from the goal, reflecting a comprehensive defensive strategy that accounts for various in-game scenarios.

Additionally, to expedite the learning process, we impose a time step penalty of -1 on our agent. This penalty serves as an incentive for the agent to learn the desired behavior more efficiently, encouraging quicker and more effective interventions to prevent the opponent from scoring.

$$R_{\text{time step}} = R - 1.$$

Reward design 2

In this case, the reward function was designed as follows:

$$\begin{aligned} d_0 &= \mathit{distance}(\text{goal location}, \text{ball carrier}) \\ d_1 &= \mathit{distance}(\text{our player } 1, \text{mid}_{12}) \\ d_2 &= \mathit{distance}(\text{our player } 2, \text{mid}_{22}) \end{aligned}$$

The reward R is computed as:

$$R = \begin{cases} -1000 & \text{if } d_0 < 50 \\ -D & \text{otherwise, where } D = \dfrac{\sum_{i=1}^{2} d_i}{2} \end{cases}$$

The difference between this reward function and the previous one [5.7] is that any agent in our team will pay specific attention to a specific opponent player.

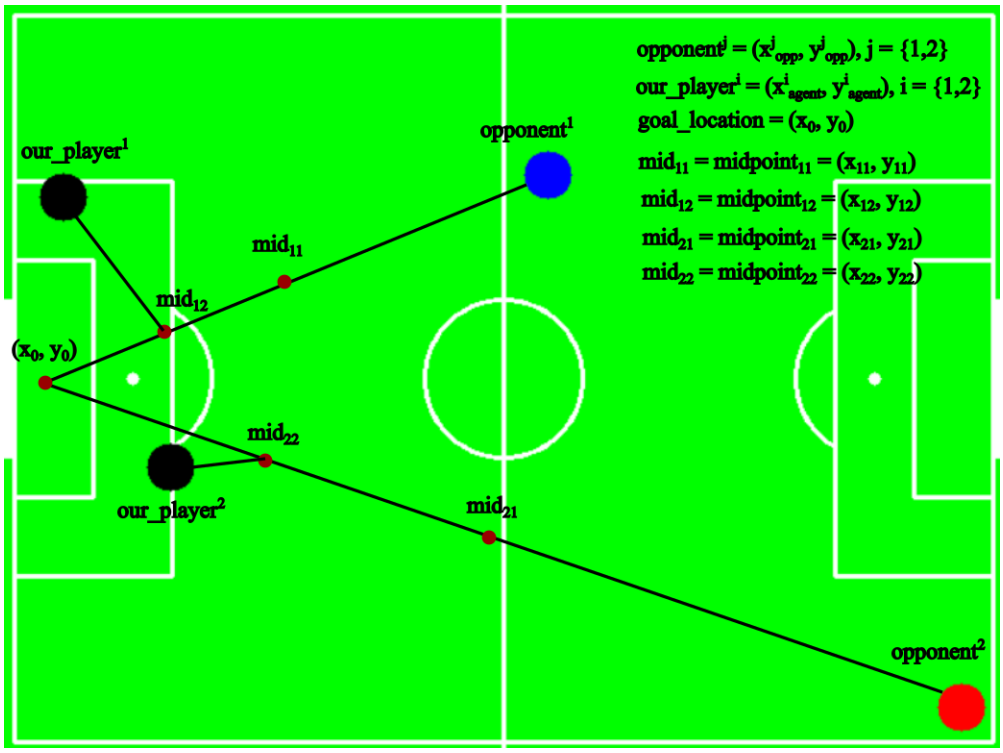


Figure 5.8: Reward Calculation

Still, to expedite the learning process, we impose a time step penalty of -1 on our agent. This penalty serves as an incentive for the agent to learn the desired behaviour more efficiently, encouraging quicker and more effective interventions to prevent the opponent from scoring.

$$R_{\text{time step}} = R - 1.$$

5.3.5 Adapting Defensive Strategy in the 3vs2 Scenario with Ball Dynamics

State and action spaces

The state and action spaces are defined similarly as it is in section 5.3.2.

Reward

We use the same reward function as it is in figure 5.8 just by adding one more player and applying the same mathematical formula.

5.4 Baselines

5.4.1 NeuroHassle Approach

In our research, while endeavoring to implement the NeuroHassle approach as a comparative benchmark, we encountered significant constraints due to the intrinsic differences in how our

simulation environment is structured compared to the original study. A critical aspect of the NeuroHassle approach is its dependency on player directions influencing the state space, a feature not accounted for in our environment's design. This discrepancy necessitates an adapted methodology rather than a direct implementation of NeuroHassle as described in their paper.

Moreover, the NeuroHassle method segments the field into four distinct regions, as shown in Figure 5.9, for training purposes: the left bottom region, the left top region, the left quarter, and the half field. While this regional focus might cater to specific situational learning, it inherently limits the strategy's generalizability across the entire field, potentially diminishing its effectiveness in untrained or novel field regions. The yellow circles depicted in Figure 5.9 represent the different positions the opponent player can occupy during the training.

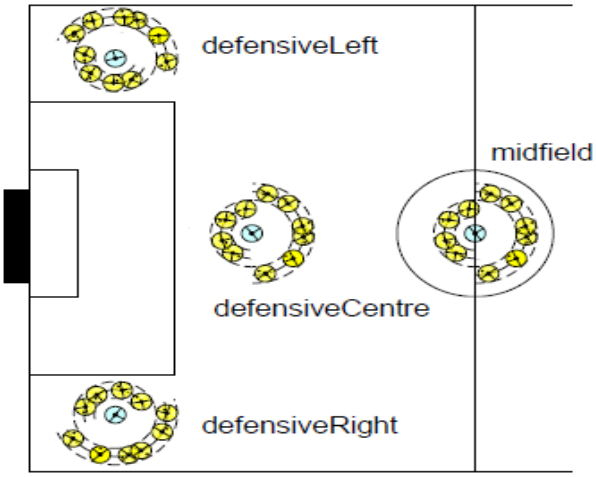


Figure 5.9: Considered Sets of Training and Test Situations [Gabel *et al.* 2009]

The original NeuroHassle approach also concentrates on training a singular agent against one opponent, subsequently replicating the learned behavior across similar agents. This methodology, while efficient, may not fully capture the complexity and variability of multi-agent interactions within our simulation framework.

Given the sake of maintaining consistency with our CPPO-based strategies, we opted to leverage PPO to learn a behavior analogous to the NeuroHassle approach. This decision allows us to harness PPO's advantages in our established environment while deriving a behavior that resonates with the essence of NeuroHassle.

In doing so, we aim to cultivate a representative NeuroHassle-like strategy within our CPPO framework, enabling a more aligned and coherent comparative analysis between the three approaches.

5.4.2 Stable Marriage

The Stable Marriage algorithm [Irving *et al.* 1987] is a classic algorithm that solves the Stable Marriage Problem, a problem that involves matching members of two sets (often referred to as "men" and "women") in a stable manner based on their preferences for each other. The pseudocode 2 outlines the basic steps of the Stable Marriage algorithm. It starts by initializing all men and women as free. Then, it enters a loop where each free man proposes to the highest-ranked woman on his list who he hasn't proposed to yet. If the woman is free, they become engaged. If she's not, she decides between her current engagement and the new proposal based on her preferences. The algorithm terminates when all individuals are engaged, resulting in stable pairs.

Algorithm 2 Stable Marriage Algorithm

Initialize all $m \in M$ and $w \in W$ to free
while there exists a free man m who still has a woman w to propose to **do**
    $w \leftarrow$ the highest-ranked woman in m's preference list to whom m has not yet proposed
    if w is free **then**
        (m, w) become engaged
    elsesome pair (m', w) already exists
        if w prefers m to m' **then**
            m' becomes free
            (m, w) become engaged
        else
            (m', w) remain engaged
        end if
    end if
end while
return the set of engaged pairs

In the context of our soccer simulation, the Stable Marriage approach is adapted to establish a defensive marking strategy by creating optimal pairings between defenders and attackers based on the midpoints between attackers and the goal location. This approach ensures that each defender is assigned an attacker in a manner that optimizes the overall defensive efficacy and stability of the team, even as the dynamics of the game evolve.

In the comparative analysis, it is crucial to underscore a distinctive aspect of the Stable Marriage approach: unlike the CPPO and NeuroHassle methods, which involve learning a defensive strategy, the Stable Marriage approach does not inherently learn to defend in the traditional sense. Instead, this method employs a deterministic algorithmic process where defenders are assigned to attackers based on predefined criteria, such as minimizing the distance to the midpoints between attackers and the goal.

In this approach, defenders are primarily guided to follow the midpoints between attackers and the goal, a tactic aimed at positioning them strategically. However, this method does not explicitly teach defenders how to intercept or block the opponents's shots. Therefore, while it ensures a structured assignment of defenders to attackers, the Stable Marriage approach does not actively equip the defenders with learned behaviors to prevent the opponent team from scoring. The defensive actions are limited to following and marking based on spatial positioning rather than engaging in proactive defensive maneuvers to thwart the attacking team's efforts.

5.5 Comparative Metrics

To assess the performance, we compared our method with existing ones, namely the NeuroHassle and stable baseline approaches. No training was performed for the stable marriage approach since it is a deterministic method. We used only the trained models for our method and the NeuroHassle approach in the comparison. No comparisons were made during the training phase.

5.5.1 Success Rate

In our research, we quantified the effectiveness of the defensive strategies through the metric of success rate. This metric was calculated by determining the proportion of episodes where our defensive strategy successfully prevented the opponent team from scoring a goal after a fixed number of time steps. After conducting a predetermined number of simulation episodes, we tallied the instances where the opponent failed to score and divided this count by the total number of episodes. The result was then multiplied by 100 to convert it into a percentage. This

success rate serves as a crucial benchmark, offering a quantitative evaluation of the defensive strategy's efficacy in thwarting scoring attempts by the opposition, thereby providing a comparative measure to assess the relative performance of different strategies implemented within the 2D RoboCup soccer simulation environment.

The success rate for each episode was calculated using the following steps:

1. Initialize a $n \times 100$ matrix with zeros elements for a set of n episodes. During a given episode, if a goal is conceded, replace 0 by 1. Compute the average across the columns of the $n \times 100$ matrix to obtain a 1×100 matrix, representing the average number of goals conceded per episode over the n sets.

2. Subtract this 1×100 matrix from a 1×100 matrix of ones to compute the success rate for each episode: $S = 1 -$ average number of goals conceded per episode.

3. The overall success rate is then obtained by averaging the elements of the S matrix.

Mathematically, this can be represented as:

$$S = 1 - \frac{1}{n} \sum_{i=1}^{n} M_i$$

where M_i is the i-th row of the $n \times 100$ matrix, and S is the 1×100 success rate matrix.

5.5.2 Average Distance to Goal

In our comparative analysis, apart from scenario-specific performances, we also employed the average distance $\bar{D}$ between the opponent players, and our goal was computed as follows:

$$D^t_{\text{mean}} = \frac{d_1 + d_2}{2} \text{ and } \bar{D} = \frac{\sum_{t=1}^{T} D^t_{\text{mean}}}{T}$$

and was seen as a key metric for comparison, where d_1 and d_2 are depicted in Figure 5.10. So after computing D^t_{mean} which represents the average distance during one timestep, and so we will average it over the total number of steps T. This metric is instrumental in assessing each method's effectiveness in maintaining a safe defensive buffer. The underlying strategy is to maximize the distance between opponent players and our goal, inherently reducing their likelihood of scoring. By evaluating the ability of each method to increase this average distance, we gain valuable insights into how well each defensive strategy can deter opponent advances and safeguard the goal area. This metric serves as a crucial indicator of each method's capacity to control space and influence the opponent's positional play, providing a clear measure of defensive prowess in maintaining goal security.

5.6 Experimental Setup

During the course of the simulation, TensorBoard, an open-source tool developed by TensorFlow, was utilized extensively for monitoring and analyzing the performance of the agents. TensorBoard provides a suite of visualization tools to track and visualize metrics like loss and accuracy, view the behavior of the model during training, and much more.

Interpretation of Graphs from TensorBoard

Several key graphs were generated using TensorBoard, each providing valuable insights:

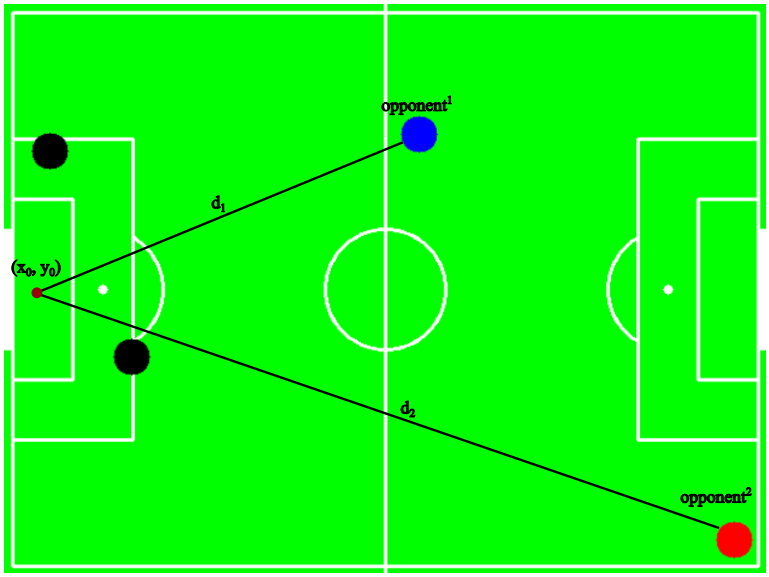


Figure 5.10: Distances between opponent players and our goal

1. **Reward Function Over Time:** This graph showed the change in the cumulative reward obtained by the agents over time. An upward trend in this graph indicated that the agents were learning effectively, improving their marking strategy as training progressed.

2. **Loss Function:** The loss graph was critical in understanding the learning efficiency. A steady decrease in loss suggested that the model's predictions were becoming more accurate over time.

3. **Accuracy Metrics:** These graphs provided insight into the model's performance in accurately predicting the outcomes of different strategies.

4. **Action Distribution:** This graph depicted the frequency of different actions taken by the agents. It was instrumental in analyzing the behavior patterns and strategic preferences of the agents.

5. **Learning Rate and Exploration Rate:** These graphs helped in understanding how the learning and exploration rates affected the agent's performance and decision-making process over time.

Each of these graphs played a pivotal role in comprehensively understanding the dynamics of the learning process. They not only offered a visual representation of the agents' performance but also provided deep insights into the effectiveness of the implemented strategies and algorithms.

For the implementation and training of the Central Proximal Policy Optimization (Central PPO) model in our marking strategy, we utilized the Stable Baselines library [Hill *et al.* 2018; Raffin *et al.* 2021]. Stable Baselines is a set of high-quality implementations of Reinforcement Learning algorithms in Python, offering a straightforward way to train and evaluate agents with state-of-the-art algorithms.

Integration with the 2D RoboCup Soccer Simulation

The integration of Stable Baselines with our 2D RoboCup soccer simulation involved several key steps:

- **Defining the Environment:** We configured our custom-built 2D RoboCup soccer environment to be compatible with Stable Baselines' expectations. This included defining

the state and action spaces, reward function, and episode termination conditions in a way that aligns with the library's interface.

- **Training the Model:** The Central PPO model was instantiated using Stable Baselines, with parameters and hyperparameters tuned to our specific scenario. The training process involved running multiple episodes, where the agent interacted with the environment, collected experiences, and gradually improved its marking strategy.
- **Monitoring and Evaluation:** Utilizing Stable Baselines' integration with TensorBoard, we continuously monitored various metrics such as rewards, losses, and agent performance, making adjustments as needed to optimize the learning process.

5.7 Conclusion

We discussed the experiments we performed in order to assess the defensive behaviour we were looking for. We first started by defining the different environments in which we performed our different experiments. We started by learning the marking strategy in the 1vs1 scenario and scaled it to the 2vs2 scenario to add more complexity. After the marking strategy was acquired, we moved on with a broader defensive strategy. The use of transfer learning did not help to learn the prevention from scoring behaviour after learning the marking behviour. So, we decided to learn the general defensive behaviour from scratch. Through reward shaping, we were able to assess the defensive behaviour.

Moreover, to assess the effectiveness of our method, a comparative analysis against the NeuroHassle and Stable Marriage methods needed to be made. To do that, we defined the different metrics used to assess the effectiveness of each of those methods and came out with more informed conclusions.

Chapter 6

Results

6.1 Introduction

In this chapter, we delve into the outcomes of our experimentation, assessing the performance of our defensive strategies across various scenarios within the 2D RoboCup soccer simulation. Section 6.2 presents the different results obtained in learning a marking strategy in the 1vs1 scenario. Moving forward, section 6.3 shows the results obtained in learning the marking strategy after scaling the 1vs1 to the 2vs2 scenario. We explain how through reward shaping, we get the desired behaviour we are looking for. We added the ball and the goal-scoring opportunity to learn a more general defensive strategy. So Section 6.4 describes the results we got in the 1vs1 scenario. Section 6.5 and section 6.6 show the results in the 2vs2 and 3vs2 scenarios after performing reward shaping to come out with the optimal reward function. All of this is performed through reward shaping and only during the training phase.

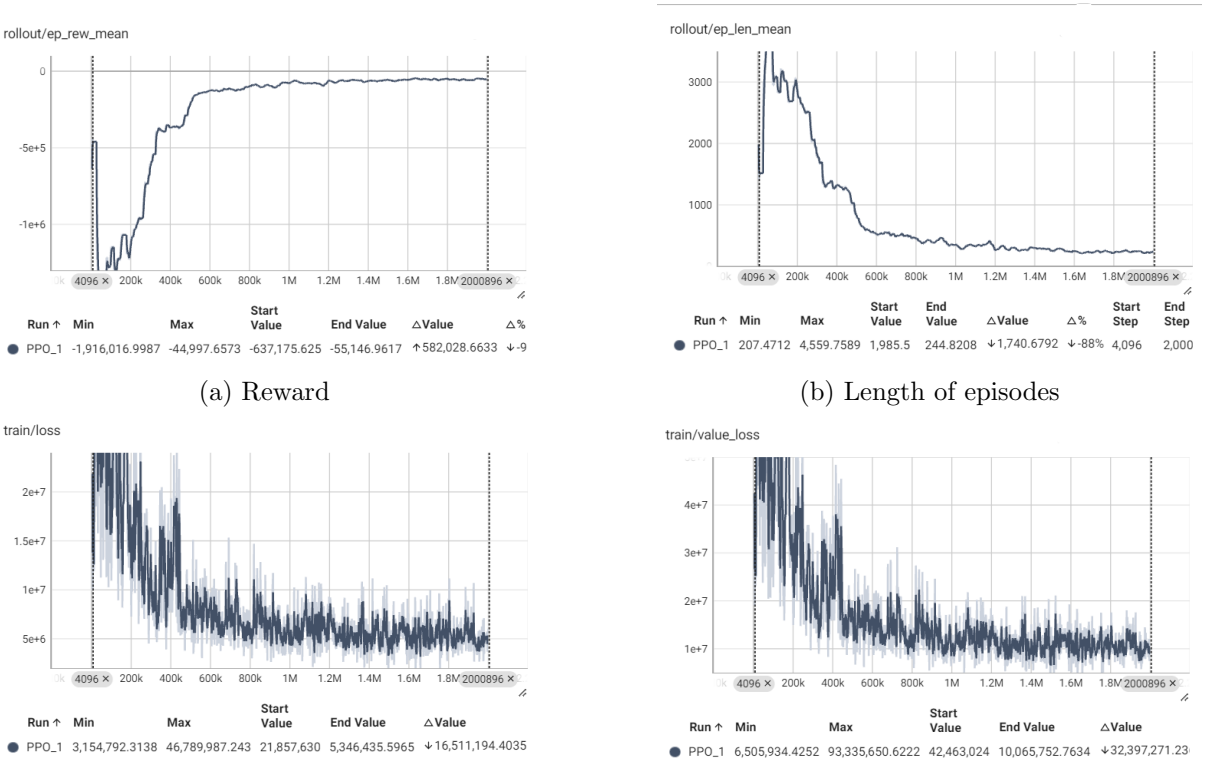
Finally, to assess the performance of our method, a comparative analysis is made in section 6.7 against the NeuroHassle and the Stable Marriage approaches, only after the model is trained. No comparison is made during the training phase.

6.2 Marking in the 1vs1 Scenario

The training progress was monitored using TensorBoard, focusing on the following key metrics:

- **Reward Function:** The initial instability observed in the reward function as shown in Figure 6.1a, characterized by fluctuations at the onset of training, can be attributed to the agent's exploration phase. During this phase, the agent is learning to navigate the environment and understand the consequences of its actions. As training progresses, the observed increase in the reward function indicates that the agent is effectively learning the marking strategy. This improvement is a result of the agent gradually refining its approach to closely and promptly mark the opponent, as guided by the reward function. The reward function's design, which incentivizes minimizing the distance to the opponent, aligns well with the strategic goal of effective marking. The increasing trend, despite the initial instability, suggests that the agent is not only learning to mark the opponent more effectively but also becoming more consistent in executing this strategy over time.

- **Average length of the episodes:** The observed decrease in this metric as highlighted in Figure 6.1b, over time is indicative of the agent's increasing efficiency in executing the marking strategy. As the agent learns to more quickly and effectively approach and mark the opponent, the episodes tend to end sooner. This trend aligns with the objective of the marking strategy, which is to minimize the time taken by the agent to close in on the opponent. The decreasing episode length suggests a successful learning process where the



(a) Reward

(b) Length of episodes

(c) Loss function

(d) Value Loss

Figure 6.1: Results obtained after training in the 1vs1 scenario

agent is achieving the desired objective of rapidly and efficiently reaching the proximity threshold set for effective marking.

- **Value Loss:** The reduction, see Figure 6.1d, observed during the training process is indicative of the agent's enhanced ability in accurately predicting the expected future rewards from different states. In the initial stages of training, higher value loss suggests a discrepancy between the predicted and the actual rewards, which is typical as the agent is still understanding the environment. As training advances, a decreasing value implies that the value function, used to estimate the potential rewards from various states, is becoming more accurate. This improvement is crucial for the agent's decision-making process, as it relies on these value predictions to choose actions that maximize the expected rewards. Consequently, the decreasing metric underscores the agent's increased predictive accuracy and its ability to evaluate the potential outcomes of its actions more effectively within the context of the marking strategy.

- **Policy Loss:** The observed decrease in this metric, as shown in Figure 6.1c throughout training signifies a positive trend in the learning process of the agent. Initially, the higher loss values reflect the agent's trial-and-error phase, where it is still learning the dynamics of the environment and the effectiveness of various actions within the marking strategy. As the training progresses, the diminishing train/loss indicates that the agent is making more accurate decisions and aligning its actions more closely with the optimal policy. This loss reduction is a clear indicator of the agent's improving proficiency in executing the marking strategy, and learning to make more optimal moves that lead to successful outcomes. The decreasing trend in train/loss is thus a strong sign of the agent's growing competence in the task at hand.

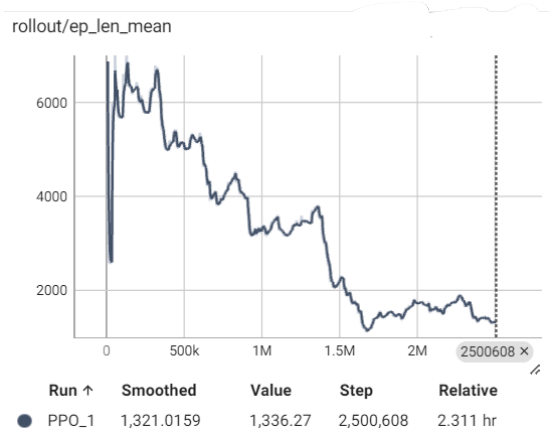
These metrics collectively indicated that the agent was learning effectively, as evidenced

by its increased capability to stay close to the opponent agent, as per the objectives set by the reward function. The observed improvement in the agent's performance, as indicated by the TensorBoard metrics, validated the efficacy of the PPO algorithm in learning the marking strategy. The increasing cumulative reward and decreasing losses signified that the agent was successfully minimizing the distance to the opponent, adhering to the set threshold, and thereby effectively learning the intended marking behaviour. In addition to the encouraging trends observed in the reward function, which suggested a successful learning process, the practical outcomes manifested in the simulation games were equally good. A visualization of a demo can be seen at the following link: Demo.

6.3 Marking in the 2vs2 Scenario

Reward function 1

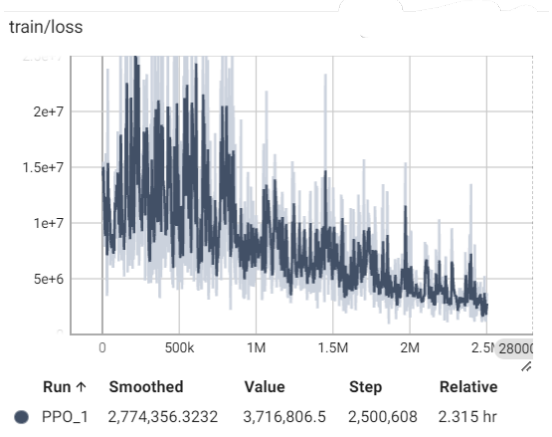
For the reward function 5.3.2 and this set of parameters table 2, we got these following results 6.2:



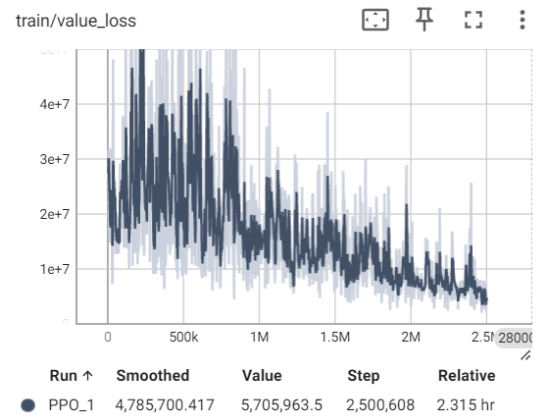
(a) Average length of episodes



(b) Average Reward



(c) Loss function



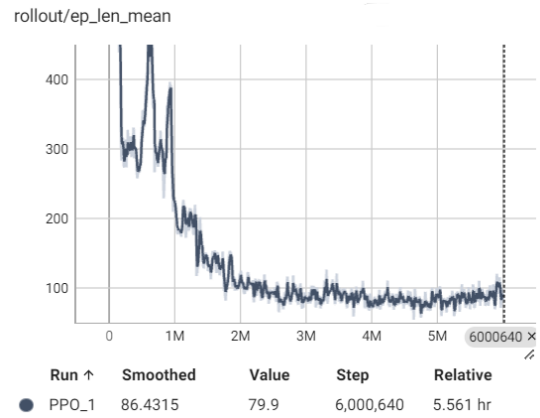
(d) Value Loss

Figure 6.2: Results obtained after training in the 2vs2 scenario using the reward defined in Figure 5.3.2

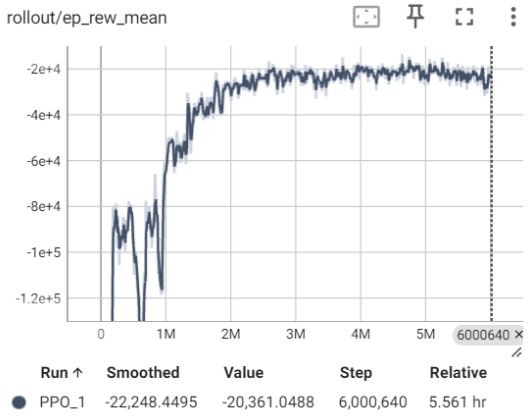
We observed a notable increase in the reward function 6.2b over time, accompanied by a decrease in the average length 6.2a of episodes meaning that the agent goes to the opponent quicker over time. However, a critical observation was that these two metrics did not converge or stabilize as time tended to become larger. We further trained the agents for a longer time but

the reward at some point started decreasing. This was also justified by the behavior observed during simulations which had nothing the do with the desired behavior. This outcome highlights further refinement in our training approach or the need to investigate alternative strategies to enhance the learning efficacy of the agents.

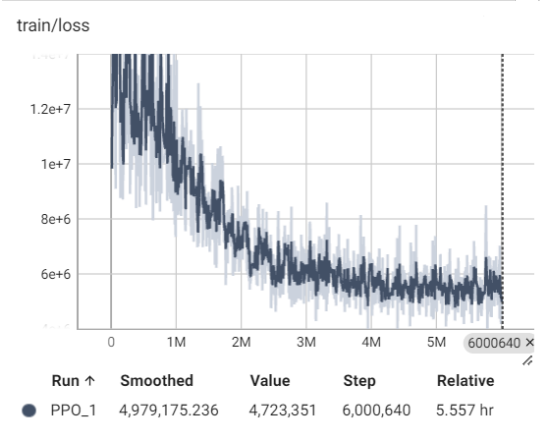
The initial threshold in our reward function was set to 50 units, which we later changed to 100 units. This modification aimed to provide a more lenient criterion for the agents to achieve a positive reward, thereby encouraging them to learn effective marking strategies over a larger area of the playing field. We used the same reward function as defined in Figure 5.3.2. we also fine-tuned several key hyperparameters 3 by trying different parameters.



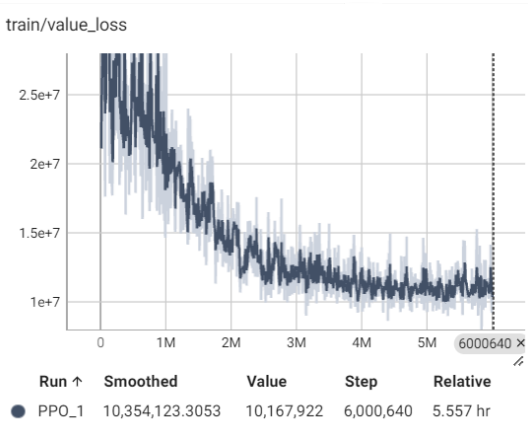
(a) Average length of episodes



(b) Average Reward



(c) Loss function



(d) Value Loss

Figure 6.3: Results obtained after fine-tuning in the 2vs2 scenario using the reward defined in Figure 5.3.2

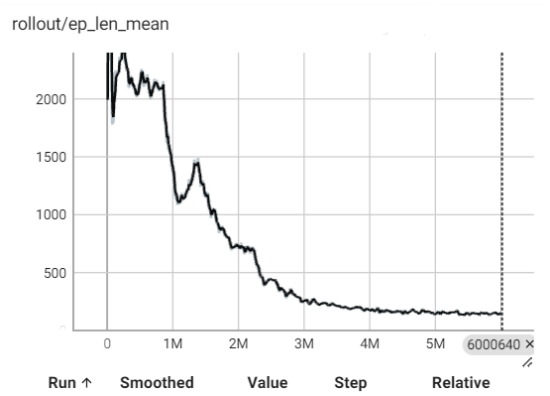
Despite achieving promising results in the reward function's 6.3b performance with a threshold of 100, a closer examination of the agents' behaviour within our 2D soccer simulation revealed some strategic deficiencies. Notably, while our agents demonstrated improved performance in terms of the reward metric, the practical application of their learned behavior in the simulation environment did not align entirely with our strategic objectives. An illustration of the agents' behaviour after training can be watched at the following link: demo.

An interesting, yet suboptimal, behaviour was observed during the simulation. Our two agents, while operating under the modified threshold and hyperparameters, tended to concentrate their efforts on marking a single opponent player most of the time, rather than effectively distributing their focus between multiple opponents. Additionally, they often maintained a distance from the opponent players that was strategically larger than desired. This behaviour,

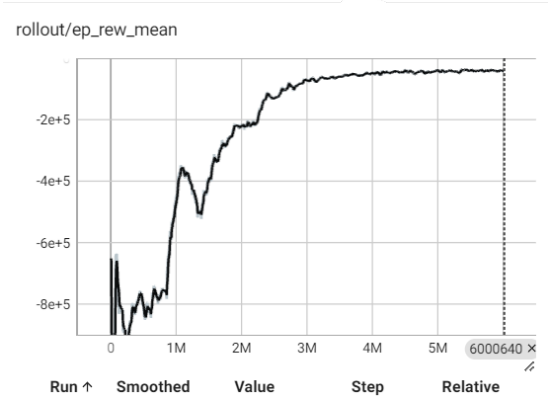
although falling within the parameters set by the 100-unit threshold, resulted in a less effective marking strategy, reducing the overall marking efficiency of our team.

Reward function 2

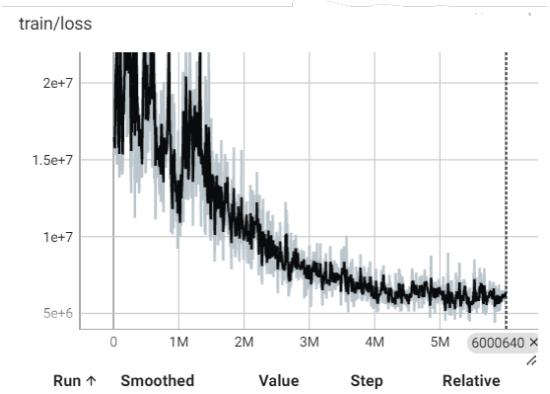
We used the reward function defined in Figure 5.3.2 and the hyperparameters 3 as previously [3]. We got the following results



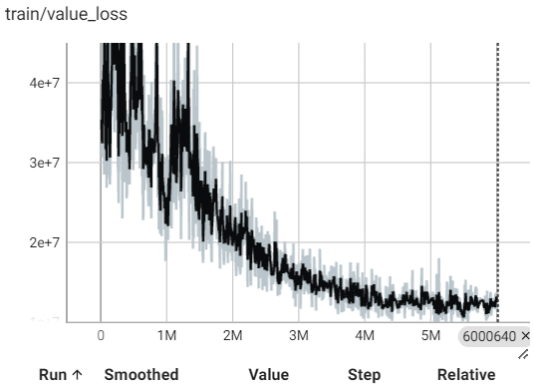
(a) Average length of episodes



(b) Average Reward



(c) Loss function



(d) Value Loss

Figure 6.4: Results obtained in the 2vs2 scenario using the reward defined in Figure 5.3.2

The implementation of this alternative reward function yielded significant improvements as highlighted in Figure 6.4, and the observable behavior within the simulation. Notably, we witnessed a consistent increase in the reward over time, indicative of the agents' successful adaptation and learning of the desired marking behavior. More importantly, we observed improved behaviour of our agents in the simulations after training. Find an illustration at the following link: Demo.

Conclusion

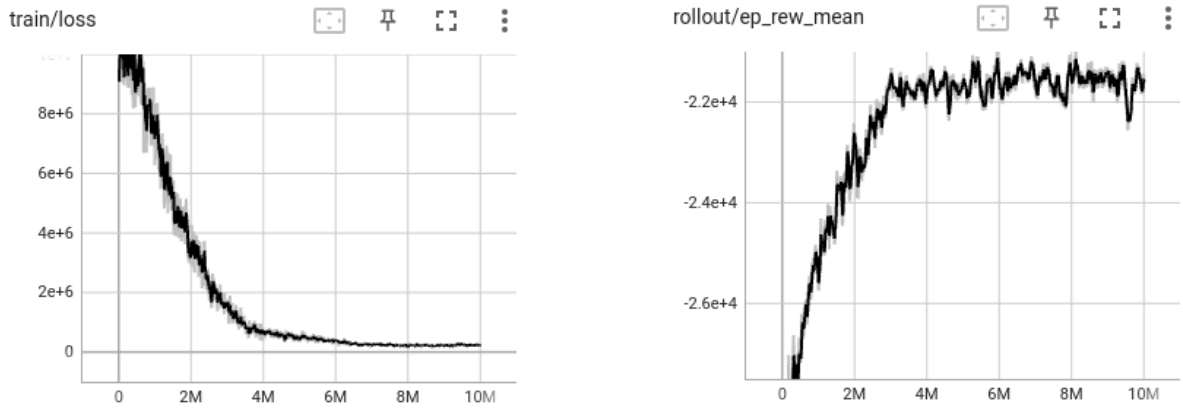
Moving from the initial focus on marking opponents without the ball to a more comprehensive defensive strategy that incorporates ball dynamics represents a significant advancement in our agents' learning process. Initially, agents were taught to closely monitor and mark opponents, a fundamental skill aimed at disrupting the opponents' formation and reducing their scoring opportunities. This foundational skill set laid the groundwork for understanding spatial positioning and opponent anticipation.

Incorporating the ball dynamics necessitated an evolution in the agents' defensive capabilities. Now, not only do they need to understand and predict opponent movements, but they

also must strategically react to the shifting focal point of play—the ball carrier. This added complexity introduces a richer set of scenarios where agents must balance between marking opponents and directly engaging the ball carrier to prevent scoring opportunities.

6.4 General defensive strategy in the 1vs1 Scenario

Upon the implementation of our refined reward function, see Figure 5.6, and strategic adjustments of the hyperparameters, Table 4, we observed a series of positive developments that collectively indicate the successful learning of the desired defensive behavior, as shown in Figure 6.5, by our agent in the 1vs1 scenario of the 2D RoboCup soccer simulation.



(a) Average length of episodes

(b) Average Reward

Figure 6.5: Results obtained after training in the 1vs1 scenario with the ball using the reward function defined in Figure 5.6

A key indicator of the efficacy of our training approach was the optimal increase in the reward, see Figure 6.5b, over time. This trend signifies that our agent progressively adapted to the simulation environment, effectively learning to balance the objectives of marking the opponent and preventing goals. The reward function, designed to encapsulate both these aspects, proved instrumental in guiding the agent towards achieving a higher level of strategic sophistication.

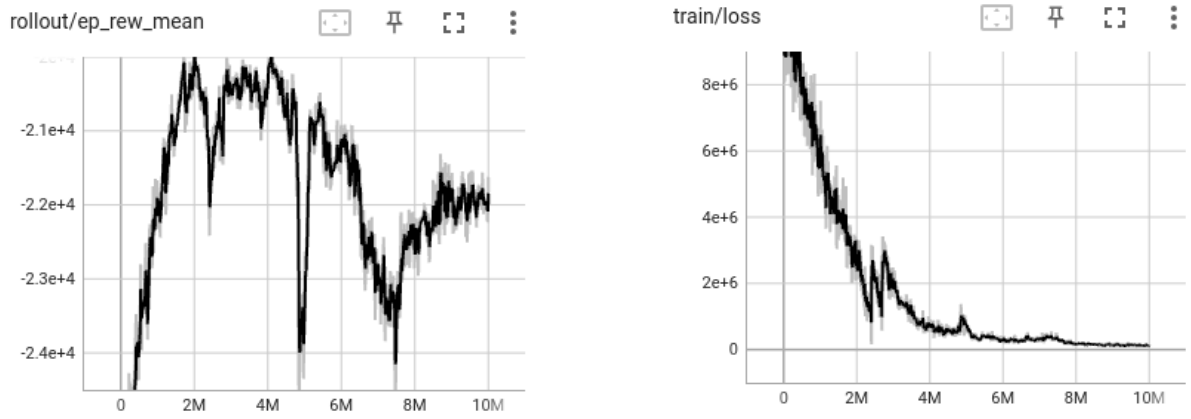
We noted a significant decrease in the loss function 6.5a throughout the training period. This reduction is indicative of the agent's improving proficiency in predicting the optimal actions within given states, thereby minimizing discrepancies between expected and actual outcomes. The diminishing loss function is a quantitative measure of the agent's learning progress, reflecting an enhancement in its decision-making capabilities.

Observations from the Simulation

Beyond these quantitative measures, qualitative assessments of the agent's behavior in the simulation further corroborated the success of our training methodology. The agent demonstrated a highly effective and realistic defensive strategy, closely mirroring the tactical nuances of real-world soccer defense. Notably, the agent's ability to anticipate the opponent's actions and strategically position itself to thwart goal-scoring opportunities was particularly commendable. This adept behavior, characterized by timely interventions and optimal positioning, underscored the agent's mastery of the learned behavior.

6.5 General defensive strategy in the 2vs2 scenario

In this phase of our work, after our agents acquired the marking skill as explained in section 6.3, we wanted to leverage the power of transfer learning to acquire the prevention from scoring behaviour. After the use of transfer learning, we got the following results figure 6.6.



(a) Reward function

(b) Loss function

Figure 6.6: Results obtained after the use of transfer learning

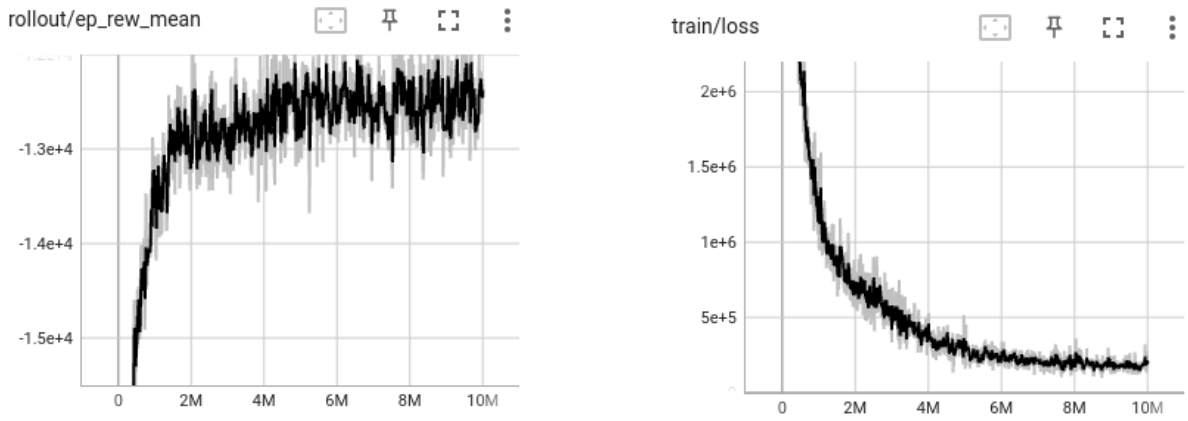
Challenges and Observations

Despite the theoretical advantages of applying transfer learning to extend the agents' capabilities, the practical application of this approach encountered challenges in this context. One of the most notable issues was the fluctuation in performance observed throughout the training process as seen in Figure 6.6. These fluctuations indicated inconsistencies in the agents' adaptation to the new tasks, possibly stemming from the complexity of integrating ball dynamics with the previously learned marking behavior. The modified reward function, while designed to encourage the desired defensive actions, may have introduced complexities or ambiguities in the learning objectives, contributing to the observed performance variability. We even trained longer but there was no improvement. Find an illustration at the following link under the transfer learning file: Demo.

6.5.1 General defensive strategy in the 2vs2 scenario (Fundamental Learning Approach)

After using the reward function, depicted in Figure 5.7, we got the following results:

The retraining initiative from scratch yielded a discernible improvement in the defensive behaviour of our agents compared to the results obtained through transfer learning. We observed a marked increase in the reward function 6.7a alongside a decrease in the loss 6.7b, signaling that the agents were indeed learning and improving their defensive strategies. These improvements were notably better compared to the outcomes from the transfer learning approach, illustrating the agents' improved ability to adapt and respond within the simulation environment. However, despite these positive trends, the agents' behaviour did not perform well enough in the simulations. An illustration can be found at the following link under the foundational skill file: Demo.



(a) Reward function

(b) Loss function

Figure 6.7: TensorBoard Metrics

6.5.2 Enhanced Training Approach for Defensive Strategy Refinement

While the learning approach outlined in the previous section marked a significant stride toward developing an effective defensive strategy in our soccer simulation, the performance of the agents, though improved, did not entirely align with our targeted defensive objectives. Indeed, our players will tend to pay attention most of the time to the ball carrier opponent meaning they learned how to prevent the ball carrier from scoring. But the consequence was that our players would leave the other opponent alone, and once the ball was passed to him in a strategic position, it scored most of the time. This observation led us to contemplate a more granular refinement of our strategy, specifically through a tailored adjustment of the reward function.

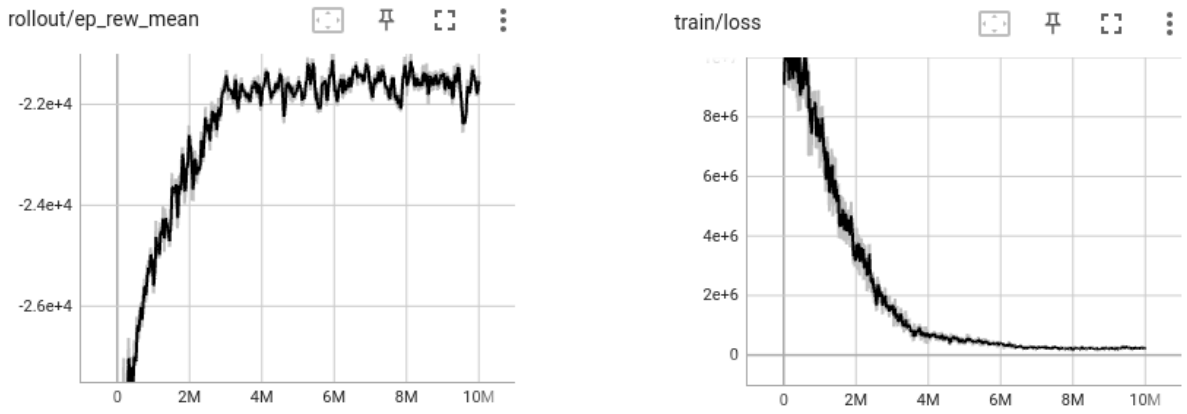
We hypothesized that a more personalized approach in the reward function could enhance the specificity and effectiveness of each agent's defensive behavior. The revised strategy involved assigning each agent a specific opponent player to focus on, as highlighted in Figure 5.8, thereby incentivizing not just team-wide defensive coherence but also individual accountability in marking and strategic positioning.

Outcome

The reward function curve exhibited an increasing trend as shown in Figure 6.8a, indicating progressive learning and improvement in the agents' defensive behavior. Additionally, an observed decrease in the loss function as depicted in Figure 6.8b, further corroborated the agents' learning progression. Find an illustration at the following link: Demo.

Having established the defensive strategies in the 2vs2 scenario, we successfully identified the desired behaviour our agents could utilize to effectively counteract opposing actions. The integration of ball dynamics allowed our agents to engage dynamically with the evolving game situation, showcasing a good level of team coordination and defensive skill.

Transitioning from the 2vs2 to the 3vs2 scenario, we aim to extend and adapt these acquired defensive behaviors to a more challenging context where our two defenders are outnumbered by three attackers.



(a) Reward function

(b) Loss function

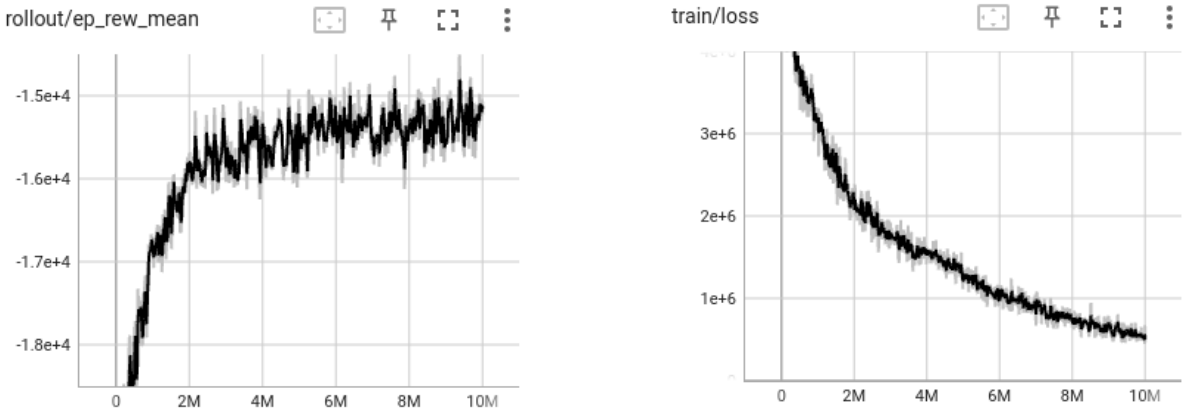
Figure 6.8: TensorBoard Metrics

6.6 Marking and preventing from scoring goals in the 3vs2 scenario

Reward function

The same reward design as shown in Figure 5.7 was used. This reward function component aimed to reinforce behaviours where agents not only maintain a general defensive formation but also adapt their positioning and movement in response to the actions of the opponent players.

6.6.1 Outcome Analysis



(a) Reward function

(b) Loss function

Figure 6.9: TensorBoard Metrics

Upon analyzing the outcomes of our training and simulations in the 3vs2 scenario, we observed a notable increase in the reward function 6.9a and a corresponding decrease in the loss function 6.9b. These trends are indicative of the agents' learning progression.

In practical simulation tests, the agents exhibited acceptable defensive behaviour, effectively implementing the learned strategies to navigate and respond within the game environment. However, despite these positive behavioral indicators, the performance outcome highlighted a substantial challenge. The opposing team's attackers managed to score goals most of the time,

underscoring the inherent difficulty faced by two defenders against three attackers. Find an illustration at the following link under the 3vs2 file: Demo.

6.7 Comparative Study of Multi-Agent Reinforcement Learning Strategies in 2D Soccer Simulation

This section delves into a comparative analysis of three distinct methodologies employed to develop defensive marking strategies within a 2D soccer simulation. The primary method under investigation leverages the Central Proximal Policy Optimization (CPPO) algorithm, which stands at the core of our research. The secondary method, neurohassle, is outlined in the work of [Gabel *et al.* 2009]. The third approach explored is the stable marriage approach. Throughout this comparative analysis, only the trained models are used to assess the performance of each approach, except for the stable marriage method, which is deterministic and does not require any training.

6.7.1 Quantitative Comparative Analysis

To objectively assess and compare the effectiveness of the three defensive strategies, we focus on a key performance metric on the success rate.

Comparative Analysis

To facilitate visual comparison between the three methods, we construct three boxplots based on the success rate matrices derived from each method. These boxplots provide insights into the variability and central tendency of the success rates across episodes, enabling a comprehensive comparative analysis of the defensive strategies under investigation.

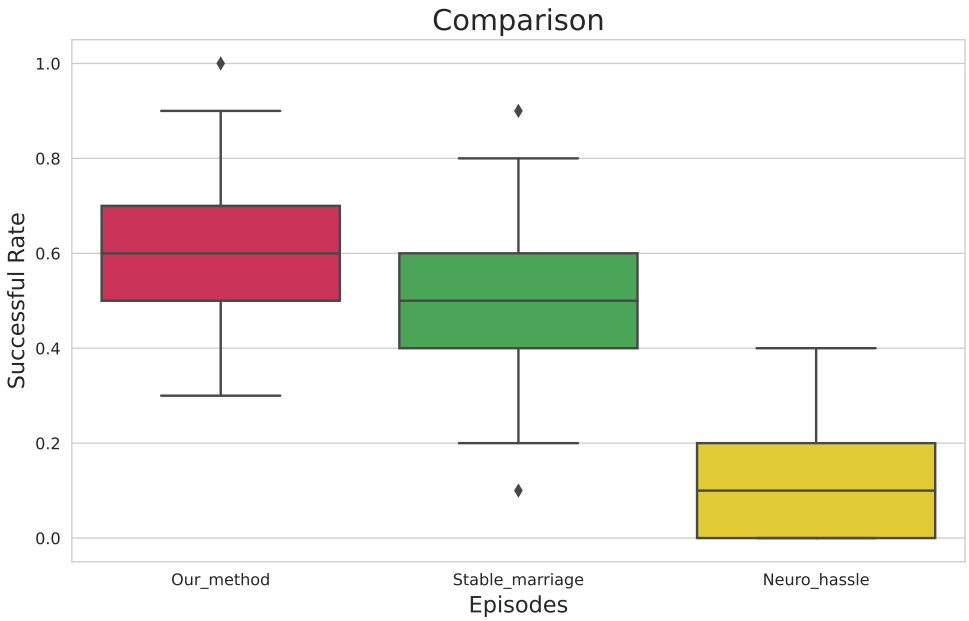


Figure 6.10: Comparison of the three methods

Interpretation

In the comparative analysis of defensive strategies within the context of the 2D RoboCup Soccer Competition, the boxplot representation of success rates provides insightful data on the performance of three distinct methods: our method, Stable Marriage, and NeuroHassle. The success rate is evaluated based on the number of goals conceded per episode, averaged over a set of 100 episodes, and serves as an inverse measure of defensive effectiveness.

Our method emerges as the most favorable approach, indicated by its highest median success rate, which suggests a robust central tendency in preventing goals. The interquartile range (IQR) denotes a moderate spread in the data, which points to a reasonable level of consistency in the strategy's performance, albeit with some episodes exhibiting significant variability. Notably, outliers below the lower whisker suggest occasional lapses in defensive effectiveness, highlighting instances where the strategy may be prone to uncharacteristic breakdowns.

Stable Marriage, while demonstrating a median success rate marginally lower than our method, displays the most extensive IQR. This suggests a higher degree of variability in defensive performance across episodes. The broader spread of data reflects a strategy that, while capable of near-optimal performance, is also subject to episodes of reduced effectiveness. The presence of lower outliers reinforces the potential for occasional significant defensive failures.

NeuroHassle, characterized by the lowest median success rate, indicates a tendency towards less effective defense compared to the other methods. However, it boasts the narrowest IQR, suggesting a high level of consistency in its performance. Despite this consistency, the range of success rates, marked by the whiskers, is skewed towards the lower end, and the presence of an extreme lower outlier indicates the potential for rare but notable defensive collapses.

In summary, our method stands out for its generally superior defensive performance in the RoboCup environment but is accompanied by a risk of inconsistency. Stable Marriage offers a variable performance that can match or exceed the effectiveness of our method but is less reliable. NeuroHassle provides a consistent defensive strategy, although it is, on average, less effective. The choice of defensive strategy might thus be influenced by the team's risk tolerance and the value placed on consistency versus peak performance within the stochastic environment of the 2D RoboCup Soccer Competition.

6.7.2 Assessing Defensive Effectiveness through Average Opponent-to-Goal Distance

In our comparative analysis, apart from scenario-specific performances, we also employed the average distance between the opponent players and our goal. The metric highlighted in section 5.5.2, serves as a crucial indicator of each method's capacity to control space and influence the opponent's positional play, providing a clear measure of defensive prowess in maintaining goal security.

In our analysis, we consistently employed the same mixture of initialization scenarios as previously mentioned, ensuring a comprehensive and uniform evaluation across all defensive methods. We simulated different values of the opponent players' velocities, namely when the opponent players have the velocities v, $1.1v$, and $1.2v$ where $v \in [-1, 1]$ is the normal velocities of all the agents. The idea is that we want our players to adapt their speed accordingly.

After many evaluations, our method demonstrated superior performance by effectively maximizing the distances between opponent players and our goal (table 6.1). The NeuroHassle approach, another strategy designed with a defensive orientation, ranked second in our comparisons. In contrast, the Stable Marriage approach was observed to be less effective in this context, placing last among the methods evaluated.

We think the underlying reason for this ranking is rooted in the inherent objectives of the defensive strategies. Both our method and the NeuroHassle approach are explicitly designed to push opponent players away from the goal, actively working to reduce the threat they pose. On

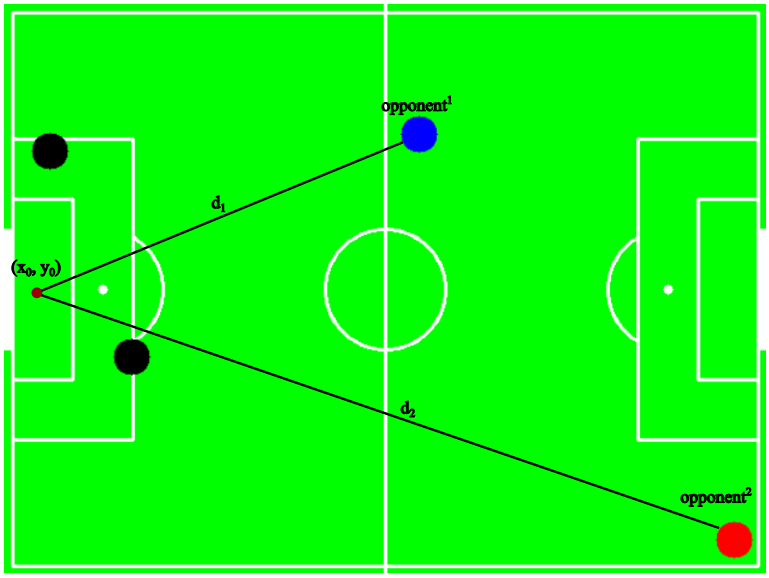


Figure 6.11: Distances between opponent players and our goal

Method	Trial 1: v	Trial 2: 1.1v	Trial 3: 1.2v
Our method	**341.06**	**416.12**	**275.28**
Stable Marriage	139.09	200.36	98.3
NeuroHassle	232.45	315.12	128

Table 6.1: Average distance opponent players-our goal

the other hand, the Stable Marriage approach primarily focuses on positioning our players at the midpoints between opponent players and our goal. While this ensures that our players are strategically placed, it does not necessarily prevent opponents from getting close to the goal. Unlike our method and NeuroHassle, Stable Marriage does not actively seek to increase the distance of opponents from the goal but rather maintains a positioning strategy that may not be as effective in maximizing the distance between opponent players and our goal.

6.7.3 Qualitative Comparative Analysis

A qualitative analysis was conducted to further understand the effectiveness of our method compared to the Stable Marriage approach, focusing on two distinct scenarios given their demonstrated effectiveness.

Defenders Behind Attacking Opponents

The goal of this experiment is to further understand the effectiveness of our method and the Stable Marriage method. Stable Marriage has proven to better place the players over the NeuroHassle approach to prevent opponent players from scoring, as seen in Figure 6.10. That is the motivation for us to consider only Stable Marriage against our method in this experiment. We place our players in disadvantageous positions, our players start from a position behind the opponent players, and want to check whether they will be able to catch up and position themselves in such a way as to prevent opposing players from scoring goals. Our players are black, the opponents are red and the opponent with the ball is blue. The trajectories occupied by all players are marked with their respective colours to track their movements. Particular attention is paid to the player with the ball. When he has the ball, his past trajectories are blue, meaning that he had the ball during that time interval, and when he passes the ball to his teammate, he

regains his red color, and consequently, subsequent trajectories will now be red, and so on. The black trajectories are the positions occupied by our players. We want to get an idea of whether our players are moving effectively in order to thwart opposing players and prevent them from scoring goals.

Our method showed a higher degree of effectiveness in this challenging situation, emulating real-life soccer dynamics where defenders chase an attacker heading toward the goal. Despite the inherent difficulty where an attacker with a lead is likely to approach the goal closely, our method 6.12 managed to mitigate the threat more effectively than the Stable Marriage approach 6.13, which primarily focuses on positioning without an active defensive maneuver to intercept or delay the attacker.

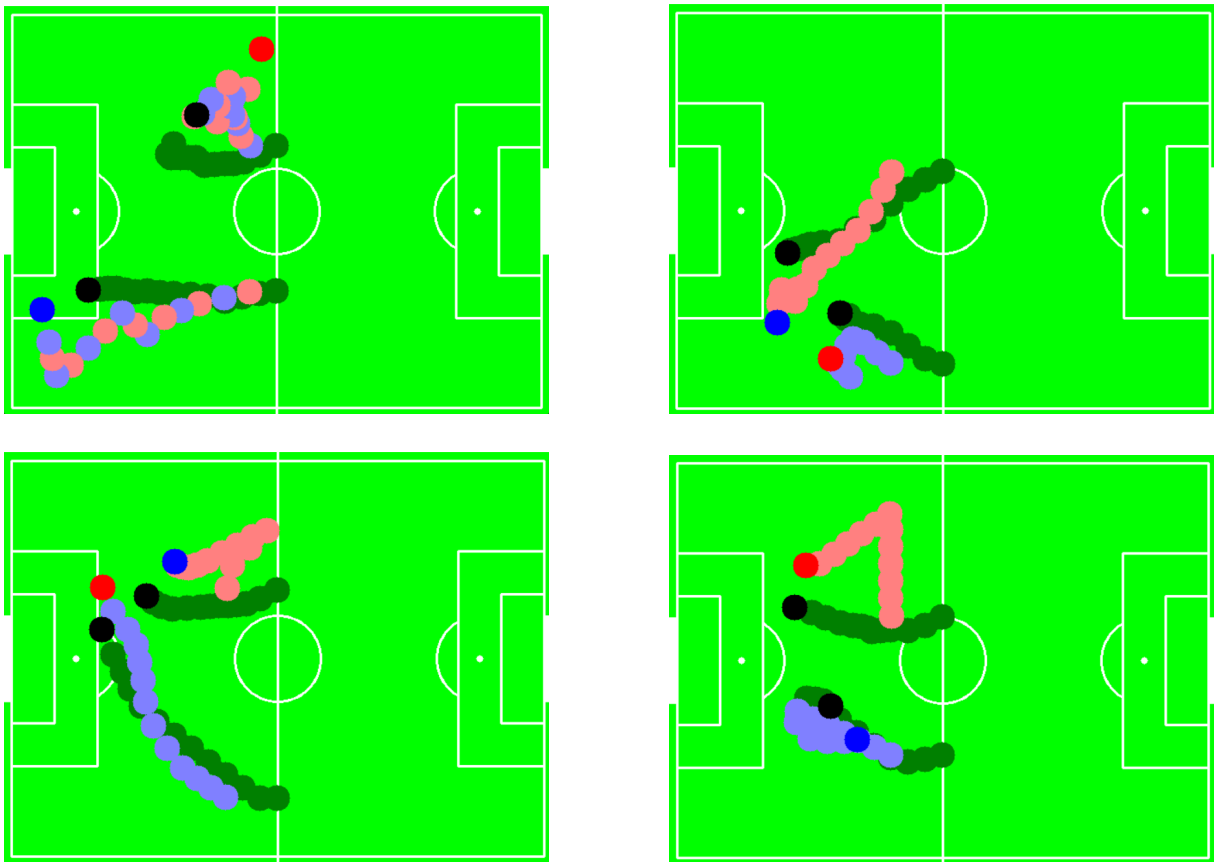
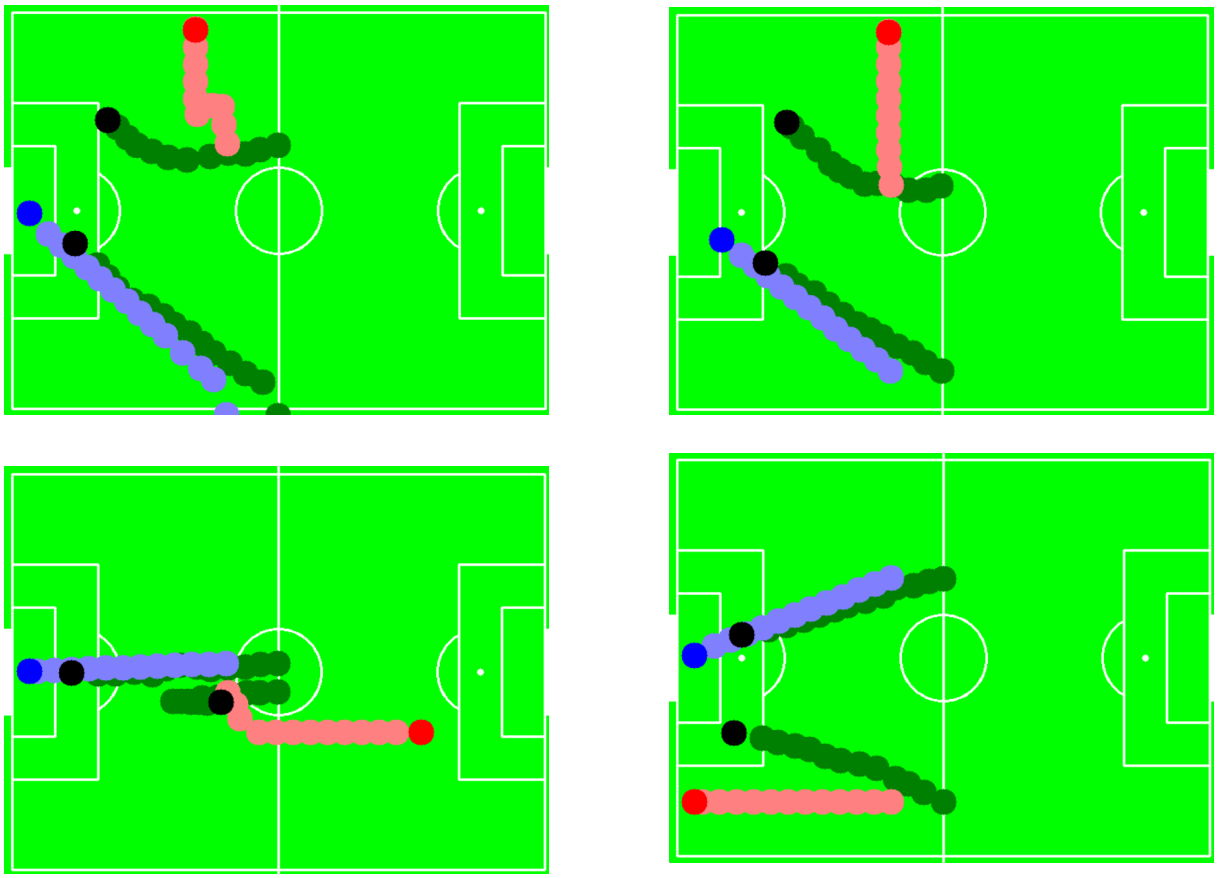


Figure 6.12: Trajectories followed by our agents under CPPO's policy

In this scenario, as shown in Figure 6.12, our defenders started behind the opponent players with the ball, and our method frequently demonstrated the ability to catch up with and effectively challenge the attacking players. This capability is indicative of our method's dynamic and proactive approach to defense, where the agents are not only positioned strategically but are also programmed to actively pursue and intercept opponents to thwart their advance toward the goal.

In contrast, the Stable Marriage approach, as highlighted in Figure 6.13, which primarily focuses on positioning defenders based on a predefined logic of pairing with opponents, namely position opponent at the midpoints between opponent players and our goal, often struggled in these situations. The approach tends to maintain positions that are optimal but lacks the dynamic responsiveness required to adapt to the fast-paced changes in the attackers' movements. As a result, in many instances where quick reaction and adaptation were necessary, the Stable Marriage defenders were unable to catch up to the attacking players, leading to a higher

likelihood of the opponents getting closer to the goal or even scoring.



(a) Value Loss

Figure 6.13: Trajectories followed by our agents under Stable Marriage policy

Defenders Between Goal and Opponents

In the second scenario, our defenders are initially positioned between the goal and the opponent players with the ball, a more traditional defensive setup. Here, the Stable Marriage method demonstrated its strength in positioning the players optimally; however, it fell short in terms of active defense. When the duration of the episodes was extended, it was observed that the opponent players, despite the initial optimal positioning of our players using the Stable Marriage approach, often managed to get close to the goal and score. Our method, while facing some challenges, exhibited a more robust defensive capability, actively engaging with opponents to prevent them from advancing and reducing their scoring opportunities.

6.8 Conclusion

Our comprehensive experimentation and analysis have shed light on the effectiveness of various defensive strategies within the 2D RoboCup soccer simulation we built. Through a series of carefully designed scenarios, we evaluated our method against the NeuroHassle and the Stable Marriage approaches, focusing on their performance in dynamically changing game situations.

More precisely, we presented the different results for different experiments we ran to get the optimal behavior in different scenarios, namely 1vs1, 2vs2, and 3vs2 scenarios. A particular focus was paid to the reward function that encapsulates how well the agents perform over time.

Interpretations and illustrations through some videos were done to check if theoretical results and applications corroborated which was the case.

After getting the optimal behaviour, we performed a comparative analysis, both quantitative and qualitative, to compare our method with the NeuroHassle and Stable Marriage methods. Using key metrics such as success rate and average distance between opponent players and our goal, our method turned out to outperform the Neurohassle and Stable Marriage approach most of the time. Our qualitative analysis reveals that our method consistently outperforms the Stable Marriage approach and NeuroHassle, particularly in scenarios requiring quick adaptation and proactive defense. While the Stable Marriage strategy provides a solid foundation for player positioning, it falls short in scenarios where active pursuit and interception are crucial. Our method's ability to dynamically respond to the opponent's actions, especially in situations where defenders start behind the attacking players, demonstrates a significant advantage in preventing goal-scoring opportunities. The NeuroHassle approach, while effective in certain contexts, generally ranked between our method and the Stable Marriage approach in terms of reducing the average distance between opponent players and our goal, showcasing decent defensive capabilities but not excelling to the same degree as our method in dynamic and challenging scenarios.

These results underscore the importance of agility and adaptability in defensive strategies within the RoboCup environment.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In conclusion, our research embarked on a comprehensive journey to develop and evaluate effective defensive strategies within the 2D RoboCup soccer simulation. We meticulously crafted a custom environment from scratch, tailored to replicate the complexities and dynamics of the 2D RoboCup soccer simulator, after encountering challenges with the Mujoco simulator. This foundational step was crucial in establishing a controlled and relevant testing ground for our defensive methodologies. Although this study used a custom simulator to train the defensive strategies, it is important to verify the results using the official RoboCup Soccer Simulation 2D league simulator. This would provide stronger evidence for the effectiveness of the proposed strategies. Due to time constraints, these tests were not performed in the current work.

Our work was methodically examined to access the desired behaviour we were looking for, namely the defensive strategy. Chapter 2 presented the theoretical background of RL and MARL as well as the different algorithms usually used for training. Special attention was given to the PPO algorithm because we used it during training and we refer to it as Central Proximal Policy Optimization (CPPO) because we used a centralized MARL setting that is one central controller and that controller was PPO. Moving toward our goal, we presented in Chapter 3 the related works in MARL and in the RoboCup domain. We presented how other researchers had used RL and related AI methods, to study some behaviour in the RoboCup soccer domain. Chapter 4 presented the methodology we used throughout our work to assess the behaviour we were looking for. We namely presented the defensive learning framework we used in the experiments we performed in Chapter 5. Finally, the results we got and the comparison we made with established baselines were discussed in Chapter 6.

The comparative analysis against established methods like the Stable Marriage and NeuroHassle approaches offered valuable insights into the strengths and areas for improvement of our strategies. By emphasizing spatial control and goal prevention, our research contributes to the ongoing discourse on developing intelligent, adaptive defensive strategies in simulated sports environments.

The successful application of our trained models in the MuJoCo simulator, despite initial setbacks, is a testament to their robustness and versatility. This adaptability underscores the potential of our strategies to be applied across different simulators, broadening the scope of their applicability in the field of autonomous agents and AI-driven simulations.

7.2 Future Work

The ultimate aim of the RoboCup initiative is to create a team of autonomous robots capable of playing soccer against a human team. This ambitious goal sets a high bar for the integration of AI and robotics, pushing the boundaries of what's possible in the realm of autonomous systems.

Our research, centered on developing defensive strategies within the 2D soccer simulation, lays a groundwork that could be instrumental in advancing toward this overarching objective.

While our current approach leverages CPPO within a centralized framework, future research could explore the potential benefits of a decentralized setting. Decentralization offers the promise of scalability and robustness, particularly in scenarios where individual agents must make quick, independent decisions based on local perceptions. This shift could enhance the responsiveness and adaptability of the agents, qualities that are crucial in the dynamic, fast-paced environment of a soccer game. Exploring decentralized reinforcement learning approaches would allow each robot to operate autonomously, fostering a team of agents that can collectively achieve complex objectives while responding adeptly to the unpredictable nature of soccer. This could include the development of sophisticated multi-agent communication protocols, enabling the robots to coordinate their strategies and actions on the fly.

In addition, we should be able to include conducting the same tests using the official RoboCup Soccer Simulation 2D league simulator to validate the results obtained from the custom simulator. This will ensure the robustness and applicability of the developed strategies in the official league environment. In case we were to use our trained model in the 2D Robocup simulator, because the outputs from the model are the velocities of all players along the x and y-axis, we will need this transformation to be able to use it in the 2D Robocup soccer simulator:

$$\text{direction} = \arctan\left(\frac{v_y}{v_x}\right) \times \frac{180}{\pi}, \tag{7.1}$$

$$\text{magnitude} = 100 \times \sqrt{v_x^2 + v_y^2}, \tag{7.2}$$

$$\text{power} = \text{sign}\left(\frac{v_y}{v_x}\right) \times \text{magnitude}, \tag{7.3}$$

where

$$\text{sign}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

All in all, our work provides a framework that could be used to train an effective defensive strategy in the concrete 2D Robocup soccer simulator.

Moreover, extending our simulation work to physical robots will introduce new challenges and opportunities. The translation from a simulated environment to real-world robotics will necessitate tackling issues related to sensor accuracy, actuator control, and the physical constraints of the robots. It will also open up avenues to incorporate advanced robotics technologies, such as computer vision and real-time decision-making algorithms, to enhance the robots' performance on the field.

Incorporating elements of adversarial training, where defensive strategies are continuously refined against increasingly sophisticated offensive tactics, could also be a valuable area of focus. This could lead to the development of more nuanced and resilient defensive behaviors, further propelling the capabilities of soccer-playing robots.

Ultimately, the insights gained from our research and future explorations in decentralized settings will contribute to the broader RoboCup goal, driving forward the integration of AI and robotics in creating teams capable of competing with human players, thereby advancing the frontier of collaborative, intelligent systems.

References

[Abed-Alguni *et al.* 2016] Bilal H Abed-Alguni, David J Paul, Stephan K Chalup, and Frans A Henskens. A comparison study of cooperative q-learning algorithms for independent learners. *Int. J. Artif. Intell*, 14(1):71–93, 2016.

[Buşoniu *et al.* 2010] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pages 183–221, 2010.

[Foerster *et al.* 2018] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Gabel *et al.* 2009] Thomas Gabel, Martin Riedmiller, and Florian Trost. A case study on improving defense behavior in soccer simulation 2d: The neurohassle approach. In *RoboCup 2008: Robot Soccer World Cup XII 12*, pages 61–72. Springer, 2009.

[Han 2018] Xintian Han. A mathematical introduction to reinforcement learning. *Semantic Scholar*, pages 1–4, 2018.

[Hill *et al.* 2018] Ashley Hill, Antonin Raffin, Maximilian Ernestus, →Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, →Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, →Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. *Stable Baselines*. https://github.com/hill-a/stable-baselines, 2018.

[Irving *et al.* 1987] Robert W Irving, Paul Leather, and Dan Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM (JACM)*, 34(3):532–543, 1987.

[Kalyanakrishnan *et al.* 2007] Shivaram Kalyanakrishnan, Yaxin Liu, and Peter Stone. Half field offense in robocup soccer: A multiagent reinforcement learning case study. In *RoboCup 2006: Robot Soccer World Cup X 10*, pages 72–85. Springer, 2007.

[Kober *et al.* 2013] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[Kyrylov and Hou 2007] Vadim Kyrylov and Eddie Hou. While the ball in the digital soccer is rolling, where the non-player characters should go in a defensive situation? In *Proceedings of the 2007 conference on Future Play*, pages 90–96, 2007.

[Lanctot 2017] Marc et al. Lanctot. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages XX–XX, 2017.

[Li *et al.* 2020] Meng-Lin Li, Shaofei Chen, and Jing Chen. Adaptive learning: A new decentralized reinforcement learning approach for cooperative multiagent systems. *IEEE Access*, 8:99404–99421, 2020.

[Lowe *et al.* 2017] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390, 2017.

[Lyu *et al.* 2021] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.04402*, 2021.

[Mnih *et al.* 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Ning and Xie 2024] Zepeng Ning and Lihua Xie. A survey on multi-agent reinforcement learning and its application. *Journal of Automation and Intelligence*, 2024.

[Panait and Luke 2005] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[Papoudakis *et al.* 2019] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019.

[Raffin *et al.* 2021] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[Schulman *et al.* 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[Schulman *et al.* 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Shalev-Shwartz *et al.* 2016] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[Silver *et al.* 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[Smit *et al.* 2023] Andries Smit, Herman A Engelbrecht, Willie Brink, and Arnu Pretorius. Scaling multi-agent reinforcement learning to full 11 versus 11 simulated robotic football. *Autonomous Agents and Multi-Agent Systems*, 37(1):20, 2023.

[Smith and Nau 2005] John Smith and David Nau. Communication and coordination in multi-agent reinforcement learning. *Journal of Artificial Intelligence Research*, XX:XX–XX, 2005.

[Stone *et al.* 2005] Peter Stone, Richard S Sutton, and Gregory Kuhlmann. Reinforcement learning for robocup soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[Sutton and Barto 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Sutton *et al.* 1998] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[Taylor and Stone 2009] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

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

[Trott *et al.* 2019] Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

[Watkins and Dayan 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[Yang 2018] Alice et al. Yang. A study on scalability of multi-agent reinforcement learning in cooperative tasks. In *Proceedings of the International Conference on Autonomous Agents*, pages XX–XX, 2018.

[Yu *et al.* 2022] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[Zhang and others 2019] Chongjie Zhang et al. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, Volume Number(Issue Number):Page Numbers, 2019.

[Zhang *et al.* 2021] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.

[Zhao *et al.* 2016] Dongbin Zhao, Haitao Wang, Kun Shao, and Yuanheng Zhu. Deep reinforcement learning with experience replay based on sarsa. In *2016 IEEE symposium series on computational intelligence (SSCI)*, pages 1–6. IEEE, 2016.

Appendix

Hyperparameters in the marking strategy 1vs1 scenario

The PPO model was trained over numerous episodes in the 1vs1 scenario. The hyperparameters are defined in the table 1 below:

Hyperparameters	**Value**
Learning Rate	0.0003
Discount Factor	0.99
Batch Size	64
Number of Epochs	10
Total number of time steps	2 millions
Clip Range	0.2

Table 1: Hyperparameters used in training the Central PPO model

Hyperparameters in the marking strategy 2vs2 scenario

During the training, the following parameters were used, see Table 2 and Table 3.

Hyperparameters	**Value**
Learning Rate	0.0003
Discount Factor	0.99
Batch Size	64
Number of Epochs	10
Total number of time steps	2.5 millions
Clip Range	0.2

Table 2: Hyperparameters used in the 2vs2 training

Hyperparameters	**Value**
Learning Rate	0.0003
Discount Factor	0.99
Batch Size	128
Number of Epochs	10
Total number of time steps	6 millions
Clip Range	0.2

Table 3: Hyperparameters used in the 2vs2 training after fine-tuning

Hyperparameters in the general defensive strategy 1vs1 scenario

When incorporating ball interaction in the 1vs1 scenario during training, we adjusted our model to include specific parameters, see Table 4, tailored to account for the ball's dynamics.

Hyperparameters in the general defensive strategy 2vs2 scenario

In the 2vs2 scenario with ball interaction, we refined our training approach by integrating specific parameters, as depicted in Table 5, designed to address the complexities of ball dynamics and team coordination.

Hyperparameters	Value
Learning Rate	0.0003
Discount Factor	0.99
Batch Size	128
Total number of time steps	10 millions
Clip Range	0.2

Table 4: Hyperparameters in the 1vs1 with ball interaction

Hyperparameters	Value
Learning Rate	0.0003
Discount Factor	0.99
Batch Size	256
Number of Epochs	10
Total number of time steps	10 millions
stats_window_size	200
n_steps	4096
Clip Range	0.2

Table 5: Hyperparameters used in the 2vs2 training of the General Defensive Strategy

Hyperparameters in the general defensive strategy 3vs2 scenario

In the 3vs2 scenario, the following parameters were used, see Table 6.

Hyperparameters	Value
Learning Rate	0.0003
Discount Factor	0.99
Batch Size	256
Number of Epochs	10
Total number of time steps	10 millions
stats_window_size	400
n_steps	4096
Clip Range	0.2

Table 6: Hyperparameters used in the 3vs2 training