

# **Machine Learning-Based Team AI in a Cooperative Multi-Agent Game**

Victor Khaustov

A DISSERTATION  
SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY  
IN COMPUTER SCIENCE AND ENGINEERING

Graduate Department of Computer and Information Systems

The University of Aizu

2021



© Copyright by Victor Khaustov, 2021

All Rights Reserved.

The thesis titled

*Machine Learning-Based Team AI in a Cooperative Multi-Agent Game*

by

Victor Khaustov

is reviewed and approved by:

---

Chief Referee

Associate Professor

Maxim Mozgovoy

Maxim Mozgovoy



Senior Associate Professor

Rentaro Yoshioka

Rentaro Yoshioka



Senior Associate Professor

Yutaka Watanobe

Yutaka Watanobe



Senior Associate Professor

Evgeny Pyshkin

Pyshkin Evgeny



THE UNIVERSITY OF AIZU

2021

# Contents

<b>Chapter 1 Introduction</b>	<b>1</b>
<b>Chapter 2 Background and Related Works</b>	<b>5</b>
2.1 Brief History of Game AI . . . . .	5
2.2 Team AI and Cooperative Multi-Agent Games . . . . .	6
2.2.1 Real-World and Virtual Team Sports . . . . .	6
2.2.2 Tactics and Strategy in Sports . . . . .	7
2.2.3 The Challenges of Sports Team AI . . . . .	8
2.3 Soccer Environments . . . . .	11
2.3.1 RoboCup . . . . .	11
2.4 Believable AI . . . . .	12
2.4.1 The Role of Fun in Game AI Competitions . . . . .	13
<b>Chapter 3 Principles of Decision Making</b>	<b>15</b>
3.1 Acting graph . . . . .	15
3.2 Knowledge generalization . . . . .	18
3.3 Learning and acting . . . . .	21
<b>Chapter 4 Data Preprocessing</b>	<b>26</b>
4.1 Related Works . . . . .	26
4.2 Experimental Setup . . . . .	27
4.3 Game Event Detection Pipeline . . . . .	29
4.3.1 Player Movements Analysis . . . . .	29
4.3.2 Ball Possession . . . . .	30
4.3.3 Detecting Passes and Shots on Goal . . . . .	31
4.4 Event Detection Quality Evaluation . . . . .	34
4.5 Discussion . . . . .	36
<b>Chapter 5 Evaluation Results</b>	<b>38</b>
5.1 Method and Parameters . . . . .	38
5.2 Passes . . . . .	39
5.3 Believable Player Movements . . . . .	40
<b>Chapter 6 Conclusion</b>	<b>43</b>

# List of Figures

Figure 2.1 Features of particular team sports in 3 dimensions: number of players, physical contacts, importance of team cooperation . . . . .	7
Figure 3.1 Acting graph . . . . .	16
Figure 3.2 Building an acting graph. . . . .	17
Figure 3.3 Encoding an acting graph. . . . .	18
Figure 3.4 Generalization tree. . . . .	19
Figure 3.5 Node structure. . . . .	19
Figure 3.6 Tree search. . . . .	20
Figure 3.7 Generalization levels. . . . .	20
Figure 3.8 Generalization levels represented via codes. . . . .	21
Figure 3.9 Learning process for action. . . . .	22
Figure 3.10 Learning algorithm. . . . .	23
Figure 4.1 Approximation of player movement speed and direction . . . . .	29
Figure 4.2 A scheme of an unsuccessful pass and a deflected shot . . . . .	34
Figure 4.3 The influence of parameter values on event recognition quality . . . . .	35
Figure 5.1 A heat map of movements of a player controlling the ball. The attacking team's goal line is on the right. . . . .	41
Figure 5.2 Movements of ball-possessing human- and AI-controlled characters. The attacking team's goal line is on the right. . . . .	42

# List of Tables

Table 2.1 Comparison of game AI for different genres . . . . .	10
Table 4.1 Experimental datasets . . . . .	28
Table 4.2 Initial experimental setup: parameters . . . . .	30
Table 4.3 Annotated events . . . . .	34
Table 4.4 Optimal parameter values . . . . .	35
Table 4.5 Accuracy of event detection . . . . .	36
Table 5.1 Pass Length Distribution . . . . .	40
Table 5.2 Pass Direction Distribution . . . . .	40
Table 5.3 Similarity of Passing Patterns . . . . .	40

# Abstract

AI researchers often rely on popular games as testbeds for evaluating new algorithms and approaches. Sports games possess typical traits of a successful AI testbed: they are popular among the general public, they are easy to setup, and they provide sufficient challenge for AI, since the participants are expected to exhibit both athletic abilities and a certain level of tactical and strategic thinking.

In particular, sports games can be used to test the feasibility of machine learning algorithms in the domain of dynamic, game-like environments. Being a prevalent method of creating AI systems nowadays, machine learning is not commonly adopted for game AI systems, and its efficiency in this area still needs investigation.

The dissertation studies the process of constructing a working AI system based on machine learning, able to operate in a cooperative multi-agent game environment. The work outlines the stages of initial training data processing, automatic action markup, learning an agent model (based on a Markov decision process), and evaluation of the obtained agents. The training dataset is based on actual human tracking data, obtained with specialized video capturing and digitization equipment installed at sports arenas.

Experiments are being conducted using a simplified soccer game engine, thus the results can be considered directly applicable to soccer-like virtual environments with possible generalizations to adjacent areas. The resulting agents are evaluated on the basis of their ability to perform reliable ghosting, i.e., to recreate styles and skills of players, comprising the input dataset.

Our results prove that it is possible to convert human tracking data into a dataset with game events, suitable for training a machine learning-based AI system. We show that events can be detected reliably and represented in the game world with a sufficient degree of accuracy. We also demonstrate that the resulting AI system is able to accurately replicate the behavior of real soccer players according to a number of well-grounded metrics. These metrics are able to evaluate various aspects of AI behavior independently, which enables us to identify weaker aspects of our AI system and focus our efforts on them. Our experiments also prove that a machine learning-based AI provides a closer approximation of a human play style than a typical rule-based AI system.

The obtained results will be of interest to game AI developers, sports analysts, and the general AI community. A clear trend in the sports games genre is a convergence between the game and the reality, which in particular practice means the demand for more believable AI-controlled agents. While many traditional game AI systems are based on manual design, it is arguably difficult to create diverse and realistic AI-controlled sports teams without machine learning methods. Another clear trend is a more in-depth, thorough approach to sports analytics, based on detailed analysis of player tracking data. Stronger teams of sports analysts can be now found not only in the world-class teams but also in teams of lower ranks. The ability of an AI system to mimic the behavior of teams and specific players can become an invaluable capability for these specialists. Finally, the creation of multi-agent coordinated behavior, emerging from the decision making processes of individual participants, is an interesting topic for a wide audience of AI researchers.

# Chapter 1

## Introduction

Historically, Artificial Intelligence (AI) researchers have often relied on popular games as testbeds for evaluating new algorithms and approaches. Discussing a famous adage “*chess is a Drosophila of AI*”, J. McCarthy wrote: “*To computer scientists in general I have only one wish to express: let there be more of these Drosophila-like experiments; let us create some more specific examples!*” [1].

Several factors may contribute to the success of a particular testbed. For example, in the case of chess one may note that the game is easy to setup, it has a wide appeal both to general public and to researchers themselves, it poses problems that are perceived as generalizable beyond the game world, and even allows independent study of isolated aspects of the game such as endings.

We can say that most *team sports* have similar features: they are popular among the general public, they are easy to setup, and they require the participating team members to exhibit both athletic abilities and a certain level of tactical and strategic thinking. Although various definitions of team sport can be given, we propose to abide here by that put forward in a recent survey: “a game that typically involves two teams playing against each other, each composed of a set of players with their individual roles and abilities” [2]. These features make team sports an interesting challenge for a game AI system and we, therefore, consider them as a promising testbed for AI.

Sports analytics can serve as a rich source of data for a variety of machine learning tasks. Widely available sports-related datasets range from collections of individual players’ performance indicators and historical game stats to detailed event logs of particular matches. Recent advancements in object tracking technology enabled sports analysts to collect spatiotemporal information, reflecting athletes’ movements over time. Resulting spatiotemporal datasets were used in numerous research tasks, including similar play sequences retrieval [3] and identification of defensive strategies [4] in basketball, and shot prediction in tennis [5]. In soccer, spatiotemporal datasets were used to analyze team formations and play styles [6], as well as learning coordinated defensive strategies [7].

Most available spatiotemporal sports datasets, to the best of our knowledge, are organized as sequences of game field snapshots that include athletes’ locations at specific points of time. However, for many types of research projects, information about game events is also crucial. For example, in the case of team sports games (such as soccer, basketball or rugby) it might be important to know which athlete is currently possessing the ball, whether a snapshot with a ball in the air belongs to a pass or a shot on goal sequence, and so on. One realistic scenario where such information is necessary is the development of a case-based reasoning AI system for playing a sports game, which is described in this work. The AI needs to learn actions performed by the athletes in specific game situations, so both these actions and situations must be reconstructed from a series of snapshots, contained in a dataset. Other possible applications of this kind of data are sports analytics and game visualization [8].

Machine learning is the prevalent approach for creating AI systems nowadays. However, its



adoption is not uniform. While in certain domains, such as speech recognition or image analysis, machine learning is ubiquitous, this cannot be said, for example, about game AI systems. One of the reasons for the present state lies in aims of a typical game AI system, which are often divergent from traditional “performance-oriented” goals of a non-game AI.

Being an integral part of a game project, the game AI has to contribute into game quality. However, it is not easy to figure out which AI features would make the game more successful. Various aspects of “good game AI” are currently being discussed in the community, and include, in particular, believability, fun, and high skill level [9,10].

The diversity of these aspects can be explained with the diversity of computer games and game genres. As Kevin Dill summarizes, “*The one thing that is universally true is that games are about creating a particular experience for the player — whatever that experience may be. The purpose of Game AI... is to support that experience*” [11].

As a result, the prevalent approaches to game AI creation are based on manual behavior design, reflecting game designers’ views of the desired complexity and acting style of AI-controlled game characters. These approaches often rely on finite-state machines [12] or, more recently, on behavior trees [13].

In contrast to manual AI design, machine learning methods typically produce probabilistic results, which in practice might mean somewhat unreliable AI behavior. In a game AI setting such cases might be perceived as major flaws by the users, who can find and exploit shortcomings of their AI-controlled opponents. Manual design, however, produces less advanced but more predictable behavior, which is relatively easy to test and correct. Manual design is also helpful in cases where certain well-defined behavioral traits are required. For example, one might need to create a highly aggressive, highly defensive, or a simple entry-level opponent in a fighting game.

Therefore, the application area of machine learning in game AI systems is limited by the goals of game AI itself and by inherent properties of machine learning methods. At the same time, certain types of game worlds can greatly benefit from machine learning-based AI. One possible example is cooperative multi-agent games, representing sports-like environments.

Sports games represent real-world activities performed by people, thus it is expected that AI-controlled game characters also behave in a human-like, believable manner. Conversely, mechanic, predictable and robot-like reactions are perceived as undesirable by the players. Game strategies in sports cannot be easily classified into “aggressive”, “defensive” or other well-defined types that are easy to implement manually. Behavior of players in a sport game is often idiosyncratic and exhibits specific traits of individuals.

This observation is even more relevant for team games, where individual behavioral traits of athletes are combined with team strategies, resulting in unique combination forming a team’s “play style”, which is hard to design with traditional manual methods. Furthermore, it is expected that game designers have to create the whole line-up of virtual players and teams, providing sufficient challenge and diversity for the users.

From a practical point of view, it is also often a challenge to identify parameters affecting the mentioned “behavior traits”. In other words, it can be unclear for the AI designers and programmers (who are not professionals in sports analytics), which elements of AI behavior have to be altered in order to obtain diverse and realistic characters. This is especially challenging in multi-agent games (such as soccer, basketball, etc.), where the concept of “team behavior” arises. Sports fans know that play styles or team strategies vary across teams, but it is not easy to capture and formalize this difference.

These factors make machine learning a suitable choice for cooperative multi-agent sports games. One can envision AI-controlled characters trained on real-world data, captured from movements of actual athletes, and preserving their idiosyncratic behavioral traits. This goal is becoming more and more realistic with a wide adoption of human tracking technological solutions [14] and their higher availability to sports professionals and game developers.

---

As noted above, machine learning algorithms provide probabilistic results, which makes their proper numerical evaluation crucial. It is necessary to understand whether the input data is sufficient to train desired behavioral elements, and whether the resulting character is actually able to represent the “golden standard” play style with sufficient reliability and accuracy.

However, both the development and the evaluation of a machine learning-based AI in this case is sufficiently challenging. According to “D-SOAKED” classification of AI environments proposed by Russel and Norvig [15], a sports game can be treated as a stochastic, dynamic, sequential, and continuous setting (p. 45-46). This is not the case for most domains where machine learning methods are typically applied, which brings additional challenges into the design of a decision making system.

The dynamic and sequential nature of game environments makes evaluation hard, because we have to analyze decisions made by the AI agents on extended time intervals. It is not sufficient to make sure that the AI is able to choose the right actions in certain predefined test cases. The AI also has to be able to recover from errors, to provide robust decision making in a wide range of scenarios, and to capture the “essence” of player and team strategy. Evaluating the conformance to the desired strategy is a separate research goal, since human behavior is stochastic, and thus “conformance” does not necessarily mean verbatim repetition of any specific decisions found in the training data. The elements comprising individual and team play styles are discussed in literature on sport analytics [6, 16, 17].

We rely on soccer tracking data [7] to identify actions performed by human players in specific game situations and apply the obtained knowledge to act in previously unseen situations. The current work can be seen as one of the steps towards the creation of a human tracking data-driven AI system for the game of computer soccer. The subsequent text is organized as follows.

Chapter 2 is dedicated to the analysis of multi-agent sports-like game environments. We discuss distinctive features, characteristic for this particular type of game world, and how they affect the requirements for the AI systems controlling non-player characters. We study existing literature and identify the challenges associated with such environment. We explain the choice of a soccer-based testbed game project and explore related available systems, examining their goals and application areas. We also discuss the notion of “believability” and “fun” in game AI systems and their role in perceived quality of the resulting game project.

In Chapter 3 we discuss specific machine learning methods used in our system. The choice of methods is not trivial in this case, since they have to satisfy a number of criteria, including the ability to be trained on relatively small input data sets, available to us. We outline how learning process is organized, how agent knowledge is represented in computer memory, how similar case extraction is performed, and how the final decision making process is organized. We provide grounds for our choice and briefly examine possible alternatives and options for the future experiments.

The goal of Chapter 4 is to provide an in-depth review of tasks associated with preprocessing the source human tracking data to make it a suitable input for a machine learning algorithm. Tracking data of soccer players is captured with specialized hardware, installed at a stadium. It provides a series of digitized game snapshots that lack game event information, crucial for AI decision making. Thus, these events have to be reconstructed and represented in a form applicable in a game world (which, in turn, should be seen as a rough approximation of a real soccer game rather than an accurate simulation). We discuss challenges associated with the data processing step and describe our approach to address them.

The final Chapter 5 provides an account of our experiments, performed in the course of designing and improving our AI system. We discuss the configuration of the AI agent (its feature space, its method of ranking probabilistic decisions, its choice of decision making points, etc.), possible improvements, and the methods of evaluating AI behavior. We outline the set of metrics that can be used to assess the conformance of the AI system to the target play style, the basis of

their choice, and their reliability. We discuss challenges associated with the evaluation process and describe our current results. We consider both single-player and multi-player scenarios, where either only the player currently possessing the ball is evaluated, or evaluation is made for the whole team.

## Chapter 2

# Background and Related Works

This chapter is dedicated to the analysis of multi-agent sports-like game environments. We discuss distinctive features, characteristic for this particular type of game world, and how they affect the requirements for the AI systems controlling non-player characters. We study existing literature and identify the challenges associated with such environment. We explain the choice of a soccer-based testbed game project and explore related available systems, examining their goals and application areas. We also discuss the notion of “believability” and “fun” in game AI systems and their role in perceived quality of the resulting game project.

Sport games are among the oldest and best established genres of computer games. Sport-inspired environments, such as RoboCup [18], have been used for AI benchmarking for years. In this chapter we argue that, in spite of the rise of increasingly more sophisticated game genres, team sport games will remain an important testbed for AI due to two primary factors. First, there are several genre-specific challenges for AI systems that are neither present nor emphasized in other types of games, such as team AI and frequent re-planning. Second, there are unmistakable non skill-related goals of AI systems, contributing to player enjoyment, that are most easily observed and addressed within a context of a team sport, such as showing creative and emotional traits. We analyze these factors in detail and outline promising directions for game AI research within a team sport context.

Before suggesting how important is game AI for the team sports area, we first look at how the game AI landscape currently evolves (Section 2.1). Next, we review the differences between real-world and virtual team sports (Section 2.2.1). We also look at the tactics and strategy perspective (Section 2.2.2), before more generally collecting the challenges of sports AI, also in comparison with other genres, such as Multiplayer online battle arena (MOBA) and Real-time strategy (RTS) (Section 2.2.3). As said before, player enjoyment is an important aspect also for game AI, and it is addressed in Section 2.4.1.

### 2.1 Brief History of Game AI

Without doubt, game-related benchmark environments have recently propelled AI progress and led to a huge impact in the media, thereby contributing to the current AI hype. The game of Go was already an unofficial AI benchmark when chess lost much of its appeal for AI researchers as a result of DeepBlue’s victory over Kasparov in 1997. When Silver *et al.* [19] provided an approach that was able to beat professional human Go players, the old Atari console environment has already been used to invent *Deep Reinforcement Learning* [20, 21]. It soon turned out that most of the games of the the so-called *Atari learning environment* are relatively easy to deal with by this new technique, even playing many games with super-human performance was quickly achieved. In search for new challenges, research turned to RTS games.

StarCraft was established as a benchmark and competition environment in 2010 [22], and was still considered very challenging when DeepMind and Blizzard teamed up to provide the

SC2LE environment for StarCraft 2 in 2017 [23]. Following the reports of the AlphaStar team of DeepMind [24], it seems that RTS games are not yet completely done, but getting in reach. This is surprising on one hand, because most researchers had expected it to take years to get that far. On the other hand, there is higher emphasis on generalizability now: we are not satisfied with an AI system that can be used for a single game and in previously seen environments anymore; we expect the AI shall be able to transfer knowledge between situations, environments, and eventually games.

Another aspect, the team play between different AI systems, is emphasized in the cooperating OpenAI Five [25] MOBA game bots. The number of bots in a team is limited to five, but they clearly need to work together well in order to win a game — a capability that is also fundamental for team sports AI. We, therefore, also compare the properties of MOBA, RTS and team sports games and the resulting challenges for AI later on.

With this section, we highlight the need to define good benchmark environments that integrate *team AI*, here understood as multiple game AIs that interact among themselves and with humans in order to reach a common goal. For this, team sport games provide an excellent context with numerous advantages.

## 2.2 Team AI and Cooperative Multi-Agent Games

### 2.2.1 Real-World and Virtual Team Sports

There is no universal definition of “team sports”. According to Collins Dictionary, a team sport is “*a sport in which teams play against each other*” [26]. The current Wikipedia article expands this definition as follows: “*A team sport includes any sport where individuals are organized into opposing teams which compete to win. Team members act together towards a shared objective. This can be done in a number of ways such as outscoring the opposing team*” [27].

For some sports, one may wonder to which extent they should be considered full-fledged team sports. For example, in rowing, relay racing or swimming, there *are* competing teams of athletes, but little communication is required among team members and virtually no contact with the opposing team.

More interesting cases are sports in which teams take on distinct roles, and follow different objectives in different phases of the game, like fielding team and batting team in baseball, or when each team consists of several groups with immutable roles as in quidditch [28], where two largely independent but somehow interacting games (chaser and beater) take place on the same field, extended by a third one (seeker) during the end phase of the game.

Having noted these scenarios, let us narrow down the subsequent discussion to the types of sports in which both teams have the same structure and follow the same goal as, for example, in ice hockey, basketball, soccer, volleyball and many other similar games.

Sport games are also a genre staple of the video games industry. Since *Tennis for Two* (1958), virtual renditions of sport games spread literally to every gaming platform available. Obviously, it is not possible to capture *all* aspects of a physical athletic activity on a computer, and different game projects emphasize different elements of real-world sport events. As extreme examples, departing from a somewhat standard formula of a sport video game, we can mention *Subbuteo* (1990), representing soccer as a turn-based billiard-like strategy, *Captain Tsubasa* (1988), emphasizing role-playing elements of soccer and player relationships, and a number of games like *Super Mario Strikers* (2005) or *Nintendo World Cup* (1990), with unrealistic “fun” elements such as powerful supershots, players’ special abilities, and extreme weather conditions. Even within the same franchise, developers sometimes create games that emphasize “simulation” or “fun” aspects, such as in case of EA’s *NBA Live* vs. *NBA Jam* or *NHL* vs. *NHL Slapshot*. Therefore, the discussion around “team sports as a game AI testbed” requires first

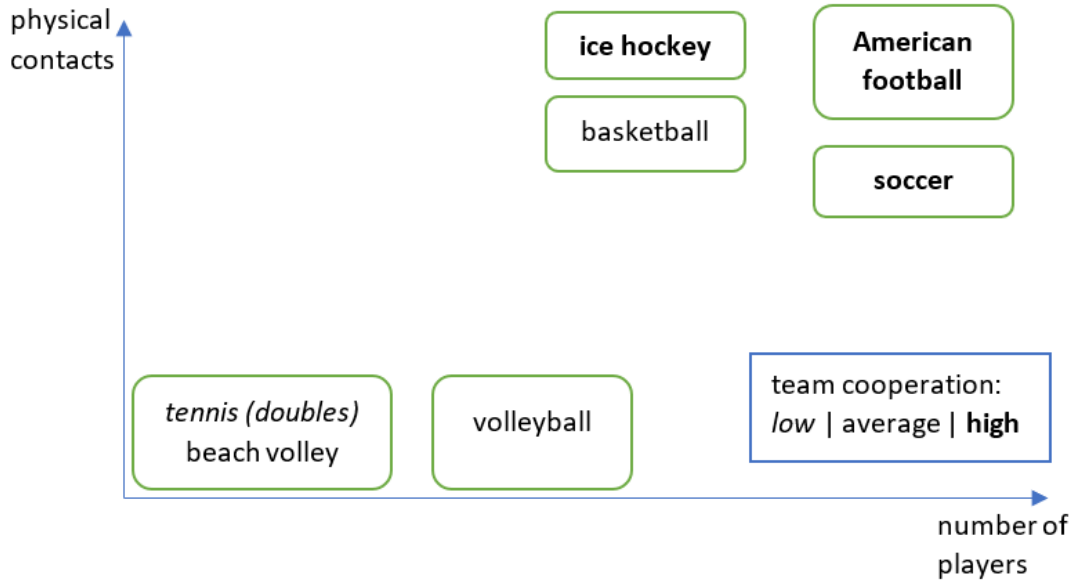


Figure 2.1: Features of particular team sports in 3 dimensions: number of players, physical contacts, importance of team cooperation

deciding which aspects of team sports should be investigated, and what are the ultimate goals.

### 2.2.2 Tactics and Strategy in Sports

An obvious problem with the choice of team sports as an AI benchmark lies in the fact that they require certain physical abilities and athletic skills in addition to tactical and strategic decision making. Since these physical aspects cannot be captured within the game framework, it is necessary to investigate the extent to which sport team behaviors pose a significant challenge for AI. This, in turn, can be further split into questions as:

1. What is the role of tactics and strategy in real-world team sports?
2. What is the complexity of actual decision making in real-world team sports?
3. How well can these experiences translate into a video game setting (see Figure 2.1)?

Obviously, the contribution of tactical/strategic factors highly depends on a particular sport. However, there is a plethora of works dedicated specifically to these game elements in a large variety of team sports [29–32]. In turn, good game strategies are not easy to construct. For instance, soccer history is characterized by a series of paradigm shifts in building up team strategies, as Wilson remarks in his book solely dedicated to the history of soccer tactics [33]: “*football is not about players, or at least not just about players; it is about shape and about space, about the intelligent deployment of players, and their movement within that deployment.*”

The differences in player abilities impose additional constraints on coach decisions: one has to devise an efficient tactical scheme, applicable in the given context, in a match between two specific teams consisting of specific players. We can observe this phenomenon even in relatively uncomplicated games, such as beach volleyball, where each team consists of two players, and there is no physical contact with the opponents. Koch and Tilp [34] show how differences in physical characteristics between male and female athletes (among other factors) encourage them to prefer distinct techniques in order to win.

Speaking of computer game renditions of sports, it seems that the goal of the mainstream projects, such as EA’s FIFA, is “*to make the games even closer to the actual game, that is, to*



*make the computer game converge with the sport*”, as observed by Sicart [35]. Interestingly, in his analysis of the differences between the real soccer and FIFA’12, the most salient feature was FIFA’s AI system, controlling the players and the referee. Sicart is generally satisfied with simulation of physical aspects of the game, but believes that AI falls short in its understanding of soccer. While real soccer rules leave enough room for referee’s interpretation, AI referees of FIFA’12 are scripted in a deterministic way, leaving no space for ambiguity. Similarly, AI-controlled players behave in a logical yet predictable manner, and even superstar players honored for their creativity and tactical vision (like Messi) do not exhibit such abilities in the computer game.

These limitations actually have deep implications for the players. The rigid, predictable nature of AI decision-making encourages people to exploit this knowledge and build their strategy around it. As Sicart puts it, *“FIFA players (...) need to learn how to think procedurally, how to decode the technical implementation of a known set of rules, tactics, and player characteristics, and apply this way of thinking to ways of playing the game.”* This observation reinforces our proposition that sport game AI is still far from satisfactory, and requires further research.

### 2.2.3 The Challenges of Sports Team AI

One of the goals of AI competitions is to develop AI methods, applicable to a wider domain or at least to a wider set of subtasks in the given domain. As expressed by Togelius [36], *“we should try to counteract the tendency of competition participants to overfit their solution to particular problems and problem parameters, so as to make the results of competitions more generally valid.”* This is a relatively recent approach for setting up competitions: it comes with the rise of *artificial general intelligence (AGI)*, which requires a much higher level of autonomy than most AI approaches currently have. A recent example of a step into this direction is the Obstacle Tower Challenge [37] with its procedurally generated, diverse levels that require high generalization skills.

Likewise, team sport games can serve as a good vehicle for researching a diverse set of AI-related challenges, including the following:

1. **What can coaches and athletes learn from AI and vice versa?** One can easily observe that the strategies of the top RoboCup 2D Simulation League teams are substantially different from the strategies of teams in real soccer. Despite the specific goals of RoboCup and their implications for the game (see Section 2.3.1), it is still unclear which parts of the setup can be responsible for these differences. We also do not know who, in principle, plays computer soccer better — skillful human players or AI teams. We can compare AI-controlled characters in games like FIFA with their real-world prototypes, but there has been no comparison between the best real soccer teams and AI-controlled computer soccer teams. It is possible that AI systems can greatly benefit from the arrival of digitized real soccer team behavior data; and it is also possible that real-world teams could benefit from studying team tactics, exhibited by the best RoboCup teams.
2. **Which facets does a good team sport AI integrate?** Sport games almost inevitably have to rely on some type of AI technology. A computer soccer match without AI would require simultaneous presence of exactly 22 participants plus referees. Thus, the role of AI in this setup is to be both an opponent and a teammate, and to provide smooth gaming experience and “suspension of disbelief” for human players. Environments like RoboCup set a straightforward goal for an AI system: it has to win as many matches as possible. However, as AI methods mature and become more skillful, like in chess or fighting games, we will need another benchmark, emphasizing the *user enjoyment* facet, crucial for a *game* system. Player appeal is much harder to evaluate — the goal is elusive, and the benchmarks are subjective, but one cannot just ignore the existence and importance of this factor.

3. **What is emergent and stable team behavior?** While dealing with team sports, one has to clarify what constitutes team behavior, and what are the characteristics of successful teams. It is possible that team behavior can be defined in terms of goal-driven decision-making, where team goals (e.g., scoring) take precedence over individual goals (e.g., showing off). However, real soccer teams possess other important traits, such as adaptability to opponent counter-actions, efficient repetition of the same successful patterns or adjusting strategies on the go. Professional players are able to predict the actions of their teammates and opponents (and quickly adapt their behavior if these predictions turn out to be wrong), as well as perform clearly identifiable elements of team strategy. Teams can quickly regroup after an unsuccessful attack and readjust formations in case of player injury or removal. Thus, team behavior is a complex interplay of individual and group tactics, which makes it an interesting challenge for an AI system with high potential impact.

Intelligent team behavior is an interesting problem of AI research, having great practical significance. Currently there are no established testbeds for benchmarking team AI behavior, and we think that sport games similar to soccer, ice hockey or basketball have a potential to become such a testbed. Arguably, alternative options could include game genres like RTS and MOBA. According to our comparison (see Table 2.1), MOBA games are especially close to sport games in several aspects. However, there are notable differences, too.

While successful strategies in RTS and especially MOBA games require team coordination, arguably, “team behavior” is not the most complex skill to master in these genres. RTS and MOBA are specifically designed to be enjoyable for all participants. Thus, every player typically controls a “hero”, being in charge of their own group of subordinate units. Complex team-based strategies might require some participants to sacrifice their squad for the benefit of the team and/or concentrate on unpleasant menial tasks to support the frontline actually fighting the enemy. However, such patterns are not very common in real game sessions thanks to a carefully constructed design of game economy and team structure.

Team sport games have grown naturally around the concept of team-vs-team competition, and their rules are a product of long evolution, aimed at keeping the game interesting for both participants and spectators. While in each sport there is a number of superstar athletes showing exceptional abilities, generally, sport teams are comprised of people possessing comparable skills (in any case, there is no such “unit diversity” as in RTS), and game elements like “resource mining” or “research and construction” are absent. Thus, one of the ways to make the game complex enough to keep people’s interest is to make room for diverse and non-trivial team strategies. This might mean that some team members have indeed to perform activities that can be perceived as “less exciting”, like goalkeepers in soccer, who normally have no chances to score a goal. However, their somewhat auxiliary team role is compensated by the possibility to exhibit the personal mastery required to perform their role well.

Therefore, we believe that sport games may be at least as challenging as RTS/MOBA games in terms of developing efficient team strategies. Moreover, it is easier to start an AI research project with something relatively simple like two-player-team beach volleyball, continue with five-a-side football or futsal, and then proceed to a more complicated game like soccer or American football. The problem of AI creation is also limited in this case with a relatively uniform set of tasks, dealing with various aspects of team behavior, while in case of RTS/MOBA, a successful AI system has to address diverse issues like resource management, unit choosing, complex hierarchical goal planning, and so on.

In terms of the sheer number of game elements, sport games are much simpler than RTS/MOBA, and thus their game engines should be easier to develop and easier to communicate with. A considerable challenge of modern sport games development lies in the need to implement smooth and complex animated sequences for a diverse set of onscreen actions. However, this task is not relevant for AI research, and thus can be greatly simplified. If we look at existing



Table 2.1: Comparison of game AI for different genres

AI Property	Team sports AI	MOBA AI	RTS AI
<b>Control</b> Are there multiple autonomous agents?	Mostly yes; for RoboCup, 12 per team (11 players + the coach agent directing the overall strategy).	Yes, but mostly at the end of the game — the roles are quite fixed at the beginning, and there is not much coordination.	Central control is possible for lower-level decisions; strategic-level decisions are centrally controlled.
<b>Unit diversity</b> Is it required to find ways to combine different units or to counter the enemy strategy?	Average; units can be quite different, but, in principle, most units are able to replace other units.	High; units are different and need to be combined in suitable ways; only a small subset of possible unit types is used in each game.	High, but there is less diversity in abilities (special powers) than in MOBA.
<b>Agent roles</b> Does the game impose specific roles the agents have to fulfill?	Yes, but there is a variety of “play systems” with slightly different roles (in general, there are offensive and defensive roles, and often dedicated goalkeepers).	Quite diverse roles and very diverse agent templates (heroes); the choice of heroes alone can have a significant impact on team performance.	Roles are not obvious, except in cases of clear specialization (workers / army units / air units); units have little individuality and mostly count as a part of their squad.
<b>Action spaces</b> Does the agent have many micro-level actions available?	Yes, several: movement, different ball actions (passing, shooting, dribbling); also specific actions for interaction with other agents (e.g., fouls and tackles).	Yes, usually the action space is growing during the game as new actions become available; it is slightly higher than for team sports at the start, with a low number of agent-specific actions.	Action space is dynamically growing, but this is less emphasized and not available for all unit types; movement action space is quite limited, certain actions (e.g., shooting) are often done automatically.
<b>Movements/state spaces</b> How free is the AI in controlling movement? What additional data is needed to make up the state space?	In principle, agents can move over the whole field, but roles may restrict movement space; additional restrictions are imposed by other factors, such as physical conditions and penalties of the players.	In principle, agents move freely, but in most situations the movements are strongly restricted; possible movements are also affected by many additional properties, such as physical state and obtained items.	Free movement, but rarely used in practice (e.g., for scouting), since units have to stay together for successful combat; reasoning is mostly done on the level of squads rather than individual units.
<b>Rhythm/independent episodes</b> Is it possible to change the strategy easily?	Most sport games have a restart behavior: game state is relatively independent of what happened in the last episode.	Frequent restarts in the beginning, towers have similar meaning as goals in soccer; the more successful party gains advantage over time.	No restart behavior: once one team has strategic dominance, it is hard to overturn the situation.

team sport-like systems actually used for AI projects, such as RoboCup 2D Simulation platform, WeBots (used in recent AI World Cup competitions) or MuJoCo Soccer Environment (serving as a platform for DeepMind’s experiments in team AI), we observe that all of them downplay animation and player contact, and instead focus on movement and passing behavior.

We must recognize, however, that there is no agreement on what constitutes “core” game elements that must be somehow represented in a computer simulation. For example, in soccer-like games we see environments that implement or ignore elements like physical contact/tackling, overhead passes and the offside rule. Thus, it is difficult to assert which particular elements are crucial for “interesting” team behavior (though, the rules of actual sports should serve as a reasonable approximation, and, probably, should not be ignored without good reasons).

Since sport games represent real sports, serious players have more elaborate expectations about AI, especially as other game aspects reach higher levels of realism. The AI-controlled opponents have to be believable, and the teammates have to be reasonably creative and supportive. And we hope that the growing availability of actual player tracking data can provide valuable insights for reaching these goals.

We can also note that the world of a typical sport game can hardly be characterized as “rich” — there are no fantasy-themed landscapes, intricate dungeons or exciting storylines. As there is not plenty of “decorative” game elements to “distract from” poor AI technology, it has to perform on par with other core elements, such as animations and physics. Maybe even more than in other game genres, some AI individuality is expected, e.g. in a soccer video game, where Messi and Ronaldo shall not only visually look like their real-world counterparts, but also somehow behave similar to them.

Another very interesting aspect of AI for team sport games is related to the diversity of possible goals: take a role of a teammate/opponent/referee; be a superstar forward player or a supportive defender; be strong and efficient, or be fun and inventive. Since the rise of really strong AI systems for a variety of games, it can be expected that these aspects of AI development would get more attention. As a firmly established activity, sport games can be good environments for investigating such goals, as we already know much about motivation of spectators and athletes, principles of good refereeing, and understand the general psychological environment surrounding sports competitions.

## 2.3 Soccer Environments

### 2.3.1 RoboCup

RoboCup [18] is a great reference point for discussing virtual environments for the game of soccer. RoboCup Federation organizes the event series with the ultimate goal stated as follows: *“by the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup”* [38].

On the way to the accomplishment of this grand vision, a number of smaller goals have to be achieved. Thus, RoboCup competitions consist of several independent events, each focusing on a few relatively isolated subproblems, such as hardware challenges, computer vision, and team-based behavior strategies. In practice it means separation into “hardware” and “software” leagues, and organization of different tournaments for different types of robots (including separate competitions for 2D and 3D virtual software-simulated robots).

These tournaments are deliberately designed to support the grand vision of RoboCup organizers, and to ensure smooth knowledge transfer between independent RoboCup events. For example, software robots must deal with the same type of constraints as their hardware counterparts, such as limited visibility, noisy sensory input, imprecise locomotion, and low-level actions (for instance, an agent may not dribble the ball: it has to program each ball kick independently).

Such setup is perfectly reasonable given the ultimate goal of RoboCup competitions. However, RoboCup environments, such as 2D Simulation League [39], were not designed to resemble computer games. While one can connect mouse and keyboard to RoboCup software and take control of one of the 22 players, this experience will be significantly different from a typical sport video game session.

Let us focus on the 2D Simulation League, representing the most video game-like environment among RoboCup events. It is striking that the pool of successful 2D Simulation League participants is very stable. During the last 5 years, only 7 different teams managed to reach a top 3 position. The total number of participants in each of these events ranged from 13 to 19 teams. All these teams were established at least 6 years prior to their prize-winning seasons.

In addition, we also notice a general decline in participation: 24 teams played in each of five consecutive seasons from 2000 to 2004, and since that period no competition could attract more than 20 teams.

We believe these observations can be at least partially explained by the sheer amount of work required to establish a decent team: high entrance barriers work against aspiring contestants. In general, a skillful RoboCup 2D Simulation League AI system must be able to:

- Efficiently and reliably translate high-level decisions (such as “run with a ball for 10 meters”) into a series of low-level kick and dash actions.
- Reason on the basis of noisy and limited audiovisual sensory data.
- Handle teammate-teammate and teammate-coach messages using limited communication channels.
- Efficiently distribute roles between the teammates according to their skill profiles generated by the RoboCup server at the beginning of a match.
- Adjust its strategy if some player is removed from the field by a referee.
- Be good at playing set pieces (kick-ins, free-, penalty- and corner-kicks).

While these diverse abilities are relevant to the field of game AI, it is difficult to develop all of them within a small group having expertise in just one specific subfield of AI research. A public release of the award-winning Helios team’s *agent2d* codebase [40] motivated around 80% of other teams to abandon their own developments and switch to *agent2d* [41]. This process is seen by different authors as either “greatly beneficial” [42] or “detrimental” and leading to lower diversity of teams [43]. Furthermore, RoboCup was never designed as a *game* world; instead, it was conceived as downgraded (software-only) robot training environment.

## 2.4 Believable AI

Efficient game AI systems, able to beat human players, are already developed for many genres of computer games, including multi-agent team environments [25]. However, skill is not the only essential requirement for a game AI: its essential purpose is to facilitate immersion and contribute to user engagement. Studies show that people generally prefer playing with other people rather than with bots [44], and perceive human-like NPCs as more enjoyable; thus, the goal of creating believable, human-like AI agents is important for practical game development [45, 46].

The importance of AI believability is especially high in computer simulations of real-life games, such as sports video games. People playing computer soccer or basketball expect to see a faithful rendition of their favorite sports, including realistic representation of athletes’ behavior. This task can be approached by learning behavior patterns from actual human tracking data, as demonstrated in several recent works [7, 47].

The last years are marked with the growing availability of sports tracking data. Such datasets are already used in professional sports analytics to provide valuable insights and guidance for players by estimating the effectiveness of particular actions [48–51]. Several annotated corpora are now freely available for research purposes, which facilitates exploration of their potential use outside the sports analytics area.

A game AI system is a good example of technology that might benefit from human tracking data in sports games. Virtual representations of real-life sports events belong to one of the oldest and most well-established genres of video games. Apparently, the target audience of computer sports games consists largely of sports fans: major PC and game console franchises, such as *FIFA*, *NBA* or *NFL* rely heavily on licensed teams and players and get yearly updates. Thus, an AI system playing a virtual sports game might also get insight and guidance using data obtained from professional players.

In addition to this general consideration, we must note that game AI systems have to satisfy quite specific requirements, stemming from the fact that their principal purpose is to *entertain the player* rather than *to be successful*. Russel and Norvig [15] discuss classic AI technologies mostly using the model of a “rational agent” that “*does the right thing*,” which “*is expected to maximize its [agent’s] performance measure*” (p. 39). In the case of a computer game, the overall success of an AI system is determined with its contribution to player enjoyment, even if it means “*playing to lose*,” as Johnson [10] puts it.

The apparent goal of nearly all mainstream projects is “*to make the games even closer to the actual game, that is, to make the computer game converge with the sport*” [35]. Thus, implementing a “human-like” AI decision-making system to make virtual players act like their real-life counterparts can be a reasonable goal. It can be pursued even further to the level of “virtual stars,” imitating particular famous athletes or “virtual teams,” playing in style resembling a particular team.

Our ultimate goal is to create human-like teams for the video game of soccer by learning from tracking data obtained from real-life soccer teams. Since this is a complex task involving numerous different objectives, we are trying to address some of them independently. Our experimental setup is based on a SimpleSoccer simulator with a built-in AI engine, developed by Mat Buckland [12].

### 2.4.1 The Role of Fun in Game AI Competitions

Togelius [36] gives the following advice for organizing game AI contests and choosing the game environment: “*choosing a fun game*”, “*it also helps if the game is famous*”, and “*making it really easy to get started*”. Applying this to our domain, we might add as well that it definitely helps if the respective real-world sport is popular worldwide.

RoboCup apparently relies on another formula. Analyzing a 20-year long story of RoboCup, Ferrein and Steinbauer [52] note “*the atmosphere of some three thousand robot enthusiasts*” and “*fascinating outreach to the general public*”, and emphasize the interdisciplinary nature of participants’ work, community building and team building efforts, and a chance for the participants to solve real robotics problems. In other words, it seems that RoboCup is backed first and foremost with people’s interest to physical robots, while “game-like” aspects of this event (especially in case of simulation leagues) play a somewhat secondary role.

The notion of what constitutes fun is largely subjective, but we can at least remark that, while it is *possible* to play RoboCup 2D Simulation League matches between human-controlled teams, people do not really do it; and, in any case, the gameplay of such a match would radically differ from any successful commercial soccer game. It is, therefore, highly likely that (i) RoboCup is not a very fun *video game*, and (ii) there is a niche for sport games in *game AI competitions*, which is not filled by RoboCup by design. One of the candidates for filling this niche could be the game Rocket League [53] which combines ball game and car racing fun elements and is already played by different bots in teams of 3.

The role of fun in a game is clearly broader than just to be an instrument for keeping the public interested in a certain AI competition. But it is one of the factors, shaping the game world, directly affecting the design goals of a game AI system, and the principles of its subsequent evaluation. Long-standing popularity of a certain game genre proves that the core game mechanics is both fun (*game-wise*) and challenging enough to continue fueling interest of both spectators and participants.

For team sports, the battle of team strategies has always been a major part of spectator enjoyment. Sport fans appreciate teams showing both spectacular and efficient play, and the balance between these goals is not easy to achieve. In addition, professional regulating bodies monitor actual developments of common game tactical patterns, and adjust the rules to keep competitions appealing to the audience and fight against degenerate strategies [54]. Given the long history of most popular team sports, one can assume that current rules are well balanced, and encourage inventive, non-trivial team play. For example, the early offside rule of soccer, introduced to prevent “goal hanging” (a degenerate yet efficient tactical pattern) [55], underwent significant revisions in 1925 and 1990, and was last refined in 2005 — every time to encourage attacking play and to limit the abuse of defensive “offside traps” [33, 56]. Similarly, a back-pass rule, preventing the goalkeeper from handling the ball received from a teammate, was introduced in 1992 and extended in 1997 as an anti-time-wasting measure [57].

These considerations make us believe that team sport games are an appealing choice as a game AI testbed: they are fun, spectacular, well-balanced, reasonably complex, easy to setup, rely on simple rules, and enjoy a wide fan base.

While games can be used and are actually used to benchmark general AI technologies, there are important features, characterizing *game* AI systems. It is generally presumed that the main point of a game is to win, and thus the best AI system in a typical AI competition is the system that wins.

However, the purpose of game AI is not necessarily to be *strong*. According to Dill [11], “*The one thing that is universally true is that games are about creating a particular experience for the player — whatever that experience may be. The purpose of Game AI (...) is to support that experience.*” Thus, depending on a particular game, the goal of a *good* AI system might well involve being strong, predictable, erratic, friendly, hostile, and so on. In other words, a good game AI testbed should support, at least, theoretically, various possible goals for AI-controlled characters.

The goal of strong AI development for games like chess or Go can be considered achieved, since computers are able to defeat even the best human players. We can expect that more game genres will be added to this list in the nearest future. For example, a recent work by Oh *et al.* [58] discusses the development of an AI system, able to defeat professional human players in a modern fighting game.

It is difficult to say how good modern AI methods are in playing sport games. Michael and Obst [59] observe that AI teams of RoboCup 2D Simulation League play better than human teams; they remark, however, that RoboCup is not designed to be played by people, which supposedly affects their performance. In any case, there are independent goals of designing a strong AI system, and an AI system that contributes to the overall user enjoyment. Arguably, the latter task is even more important for the needs of practical game development.

Team sports are a good testbed for investigating such non-skill related traits of AI systems as well. There is extensive literature on factors making team sports exciting for both athletes and spectators, for which there are many good examples [60, 61]. Likewise, there is a general understanding of what constitutes fun in the context of an AI system for a sports video game. In particular, we often observe that people prefer playing against other people, because people behave in a certain “human-like” way that is perceived as inherently enjoyable [62]. Thus, striving for a human-like behavior can be a legitimate goal for a sport game AI system, as important and challenging as a highly skilled behavior.

## Chapter 3

# Principles of Decision Making

In this chapter we discuss specific machine learning methods used in our system. The choice of methods is not trivial in this case, since they have to satisfy a number of criteria, including the ability to be trained on relatively small input data sets, available to us. We outline how learning process is organized, how agent knowledge is represented in computer memory, how similar case extraction is performed, and how the final decision making process is organized. We provide grounds for our choice and briefly examine possible alternatives and options for the future experiments.

Goals for the knowledge representation, storage and retrieval mechanism:

1. Provide real-time learning and acting.
  - Avoid processing power intensive searches by providing some kind of indexing for the knowledge. Replace  $O(n)$  searches by  $O(\log n)$  searches.
  - Allow learning on a small learning data set by providing knowledge generalization mechanism.
  - Knowledge generalization mechanism must also be fast  $O(1)$ .
2. Take into consideration sequencing in the situation  $\rightarrow$  action space. System's acting must keep actions sequences profile close to the learned behavior to some extent.
3. Allow different types of actions to have different requirements to the details in situation description.

### 3.1 Acting graph

The described system is an adaptive learning system in which knowledge is represented in the form of semantic directed graph (acting graph) paired with a generalization forest or generalization directed graph. Nodes in acting graph represent situations and arcs represent action instances (*or transitions between situations by means of an action*). As a result, paths in the acting graph can be read as situation  $\rightarrow$  action  $\rightarrow$  situation  $\rightarrow$  action ...  $\rightarrow$  situation sequences as shown on Figure [3.1](#).

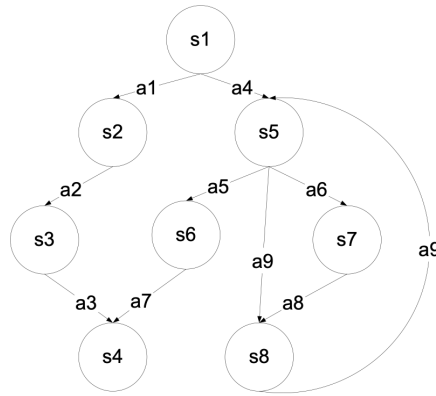


Figure 3.1: Acting graph

This method may be applied to any problem where given a situation the system (computer system/robotic system etc.) must find an appropriate action or actions based on its own or borrowed experience (knowledge) accumulated during training of any kind. Examples of this training may be learning from a teacher or self-learning. Such “situation and action” paradigm may be substituted to any other related paradigm like “input and output” or “state and transition” or any association between concepts etc. One particularly important application in which the invention has already been implemented several times is a creation of an agent in a computer game. This agent collects knowledge from a teacher (designer of game or any other computer game player) or by learning on itself with some kind of goal and then acts using the accumulated knowledge. For this application, situation is a situation in a computer game world and action is an action that the computer game character can do.

In the contrast to other machine learning models, acting graph is a direct recording of a situation- $\rightarrow$  action- $\rightarrow$ ... situation process with a certain (predefined) level of details (see Figure 3.2).



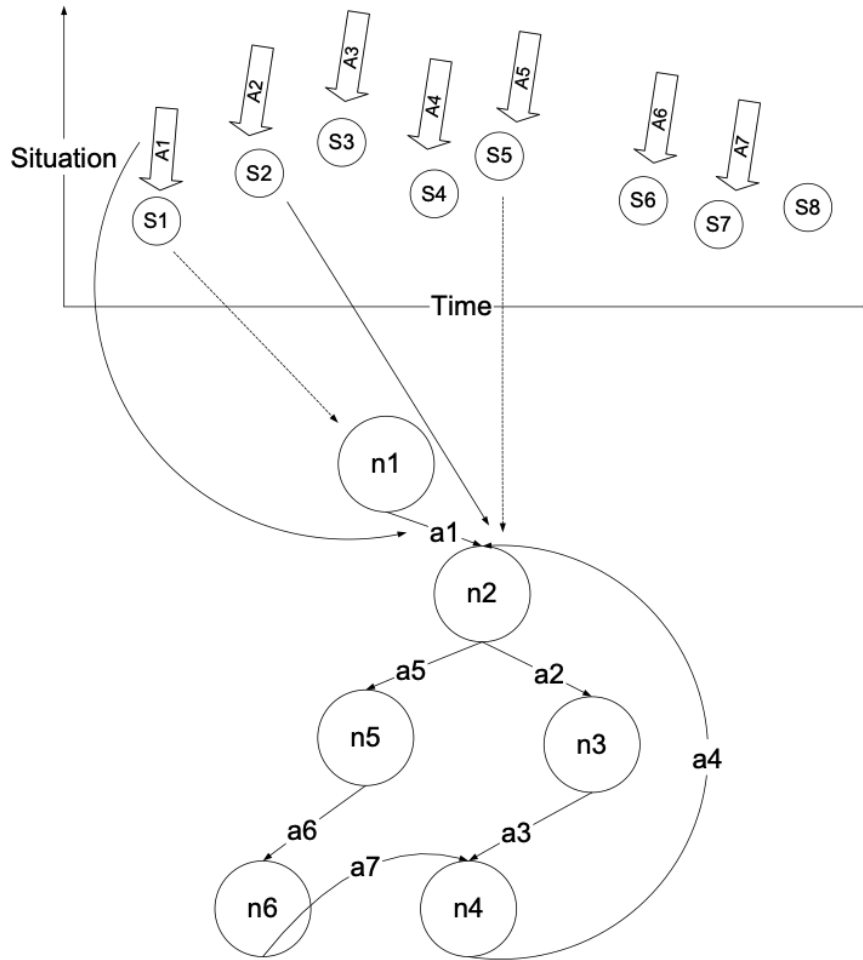


Figure 3.2: Building an acting graph.

Using terms of classification of Artificial Intelligence methods, the present system may be classified as an instance based active learning on semantic directed graphs and instance based active reinforcement learning on semantic directed graphs.

Situations are mapped to nodes and actions are mapped to arcs in the acting graph. If two situations are mapped to one node in the acting graph then it means that the system does not distinguish between two situations, i.e. they are similar in terms of the maximum level of details of particular embodiment of the system. As shown on Figure 3.2, situation S1 mapped to node n1, action A1 mapped to arc a1, etc. Two actions are mapped to one arc in the acting graph only when both pairs  $\langle \text{initial situation, result situation} \rangle$  are mapped to the same pair of nodes and the action is the same in terms of its type and parameters sensed/analyzed by the particular embodiment of the system. Therefore, an arc of an acting graph may be interpreted as a representation for a transition between two situations: initial situation and result situation by means of a particular action. System does not distinguish between actions that are of the same type, have the same parameters and derive the same transition between nodes in the acting graph. Types of actions, action parameters and their granularity are the features of the particular system's implementation.

Besides being a transition between two situations, arcs also store additional information that may include: information about an action type or class, action parameters, and any additional information about initial and result situations. Additional information about initial and result situation allows the system to store and analyze different aspects of a situation depending on a type of the action. This feature is later utilized by a mechanism of retrieval of the most appropriate actions for a given situation.



### 3.2 Knowledge generalization

Knowledge representation method uses static knowledge generalization and knowledge retrieval method uses these static generalizations along with dynamic generalizations (run-time). Dynamic and static generalizations are hierarchically bound together providing means for processing power-efficient and flexible knowledge generalization.

The system is further described for the case of generalization forest. It is possible to change the description of the system to use generalization graph with no restrictions on the structure instead of generalization forest.

The system operates in two modes: learning and acting. During learning the system perceives situations and actions and updates its acting graph and generalization forest. During acting the system finds a list of appropriate actions for a given situation or one - the most appropriate action.

During learning and acting, from each given situation the system derives an ordered list of codes  $\langle code_0, \dots, code_i, \dots, code_j, \dots, code_N \rangle$  that serve as compact situation representations on different levels of abstraction.  $code_i$  is a generalization of a situation on  $i_{th}$  level of abstraction. If  $i < j$  then  $code_j$  is a generalization of  $code_i$ . For generalization forest if  $i < j$  then for each value of  $code_i$  there must be one and only one value of  $code_j$ . This limitation applies restriction further referred as tree restriction to a possible structure of a generalization procedure that derives an ordered list of codes from the situation.

Each node of the acting graph is associated with only one unique  $code_0$  (code of the lowest level of abstraction, thus the highest level of details). Associations between nodes and codes (Code0ActingGraphNodeMap) are stored in random access memory in the form of a map, hash table or other broadly used data structures allowing quick search for an object given a particular code (see Figure 3.3).

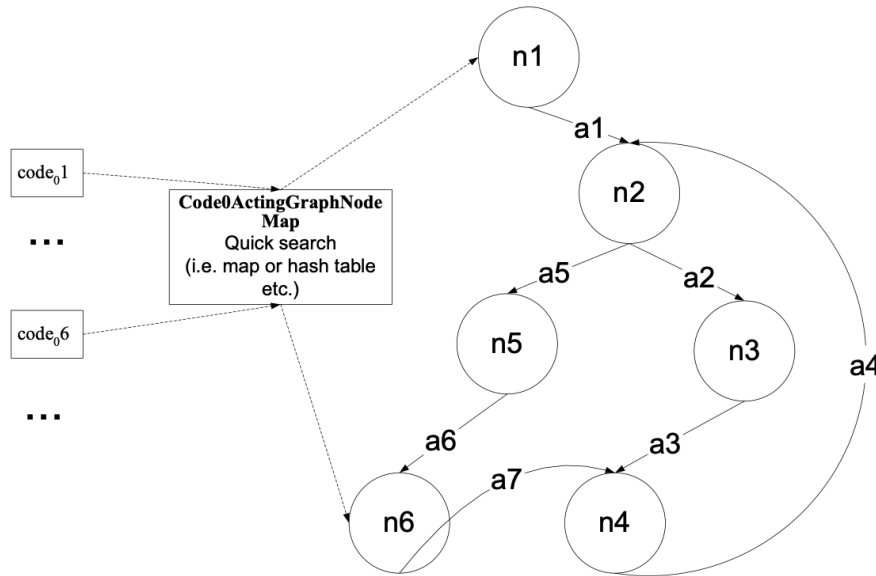


Figure 3.3: Encoding an acting graph.

For  $i$  less than number of levels of generalization, each  $code_i$  is associated with a particular unique  $code_{i+1}$ .  $code_{i+1}$  is a generalization of  $code_i$  and therefore is associated with a list of  $code_{i+th}$ . Associations between codes are stored in random access memory in the form of direct pointer/pointers. This generalization relation between codes together with the tree restriction forms a forest structure composed of codes. Figure 3.4 shows one tree out of the forest. Each tree in the forest has a number of levels equal to the number of levels of abstraction used in the

system.

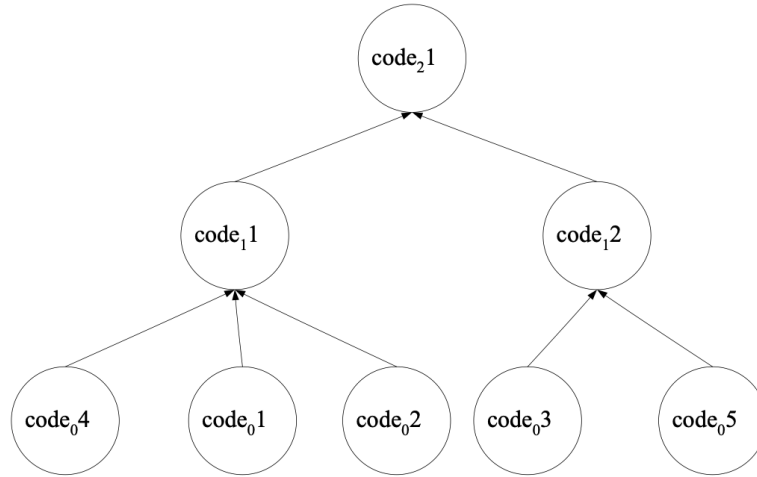


Figure 3.4: Generalization tree.

The association between codes may be substituted by the association between nodes. The association is bundled with information such as counters, weights and any additional information and form a structure in the random access memory. These structures will further be referred as, simply, nodes. Typical node structure is shown on Figure [3.5](#).

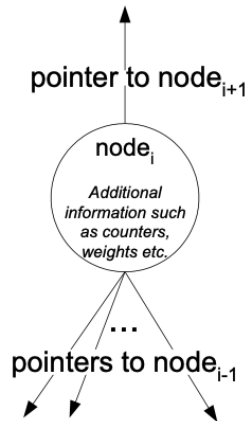


Figure 3.5: Node structure.

Notes:

- if  $i = 0$  then there is no  $node_{i-1}$  pointers
- if  $i = \text{Number of generalization levels}$  then there is no  $node_{i+1}$  pointer
- pointer to  $node_j$  doesn't necessarily mean physical memory pointer, it could be value of  $code_j$  etc.

$Node_0$ s are the nodes of the acting graph and they are also associated and bundled with acting graph arcs. This structure allows quick access to all arcs for a given node.

In one embodiment of the system, associations  $\langle \langle code_i \ code_{i-1} \rangle, \dots \rangle$  and  $\langle code_i \ code_{i+1} \rangle$  are stored bundled together in the random access memory. These bundles are nodes of a generalization forest tree and are associated with the  $code_i$  values in the form of map

(Code<sub>i</sub>GeneralizationTreeNodeMap), hash tables or other broadly used data structures allowing quick search for a bundle given a particular  $code_i$  as shown on Figure 3.6

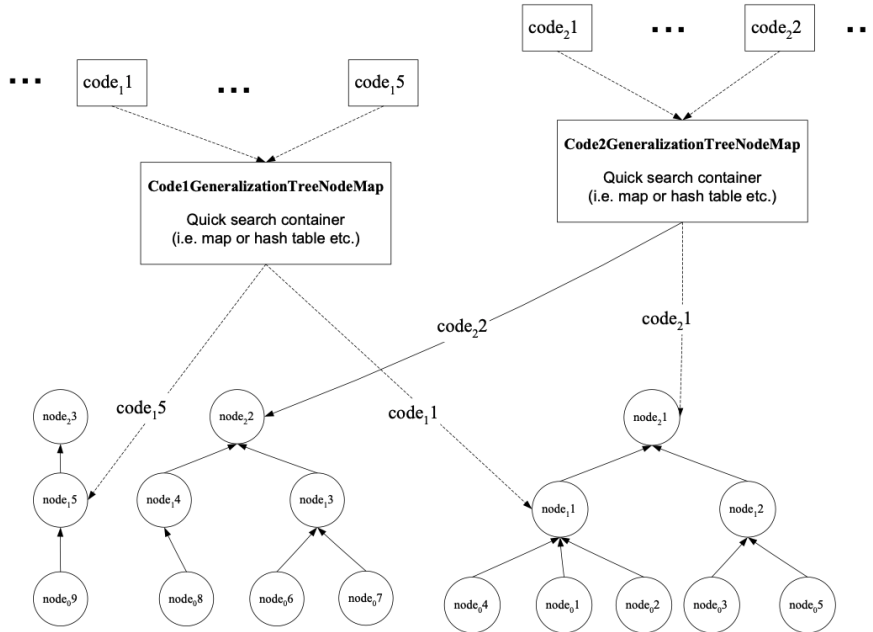


Figure 3.6: Tree search.

Acting graph and generalization forest form a structure that is illustrated on Figure 3.7. Horizontal plane contain an acting graph above which there is a generalization forest.

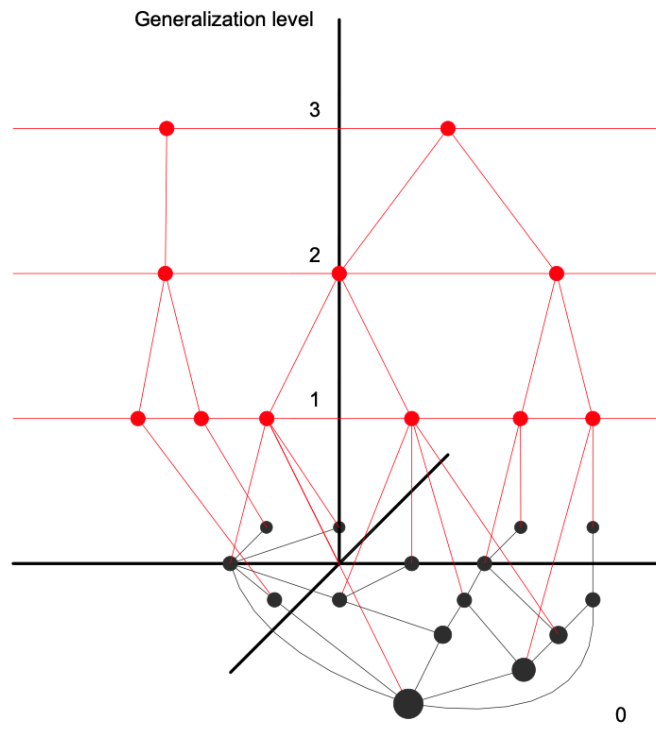


Figure 3.7: Generalization levels.

Ideas of Figure 3.3 Figure 3.6 and Figure 3.7 are shown together on Figure 3.8. Each node

of the acting graph and each node of the generalization forest are associated with codes of the corresponding level (i.e. representations of different level of abstraction) and these associations are addressable by codes (i.e. quick search for a node by a code).

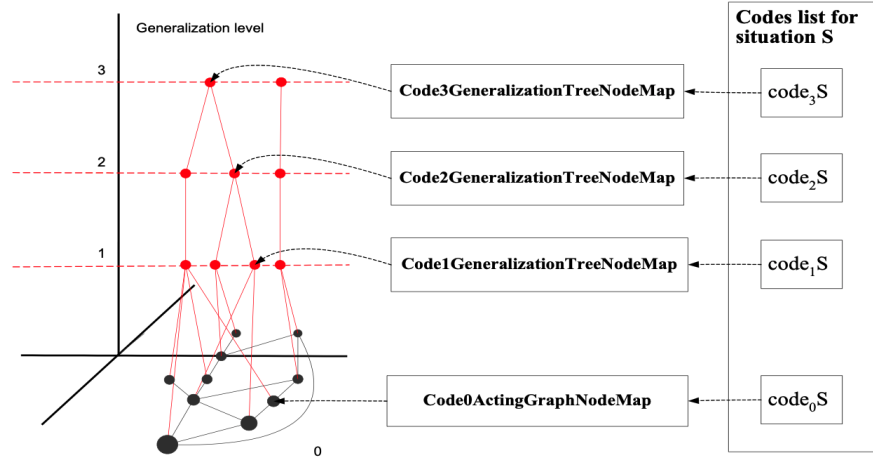


Figure 3.8: Generalization levels represented via codes.

$Node_i$  is the acting graph node for  $i = 0$  or the generalization forest node for  $i > 0$ .  $Code_i S$  is a  $code_i$  for a situation  $S$ , i.e. representation for  $S$  on  $i_{th}$  level of generalization. If  $code_i S$  is already associated with  $node_i$  then it means that the situation  $S$  has already been encountered by the system on  $i_{th}$  level of generalization. Each generalization tree has a number of levels equal to the number of levels of generalization utilized by a particular implementation of the system. For the example on Figure 3.7, number of levels for each tree is equal to the number of levels of generalization and equals 4. Nodes of level 0 of any generalization tree are nodes of the acting graph. For any  $node_i$  of level  $i > 0$  of the generalization forest, children of  $0_{th}$  level –  $node_0$  are the most detailed representations for situations similar to the  $i_{th}$  level of generalization. Besides being the most detailed (to the system) representations of situations, nodes  $node_0$  are nodes of the acting graph and they have arcs – actions associated with them.

Described generalization structure further will be referred as static generalization structure. Static generalization structure forms means for a quick search for similar situations up to a certain level of generalization given a particular situation.

### 3.3 Learning and acting

During learning, the system learns new situations – adds or updates acting graph nodes; learns new transitions – adds or updates new acting graph arcs; and learns new generalizations – adds or updates nodes and arcs of the generalization forest. Figure 3.9 illustrates a typical scenario for the system: after learning of situation  $S$  is completed, the system learns that in the situation  $S$  that already corresponds to an existing node  $n_{01}$  of the acting graph, there is an action  $A1$  that transfers the system to the situation  $R$ . The system generates list of codes  $\langle code_0 R, code_1 R, code_2 R, code_3 R \rangle$  for the resulting situation  $R$ . Through corresponding  $code \rightarrow node$  associations (see Figure 3.7) the system finds existing corresponding nodes for codes and finds out which codes are not encountered yet. If corresponding node exists then the system updates counters and arcs if necessary. If corresponding node does not exist then the system adds a new node and arcs. On the Figure 3.9 nodes for  $code_0 R$  and  $code_1 R$  do not exist which means that the system has not encountered the situation  $R$  generalized to the level 0 and 1. Nodes for  $code_2 R$  and  $code_3 R$  already exist, which means that the system is already familiar with the situation  $R$  generalized to the level 2 and 3. Note: according to the system's structure,

if a situation is found on a level  $i$  then it will be found on all levels  $j > i$ .

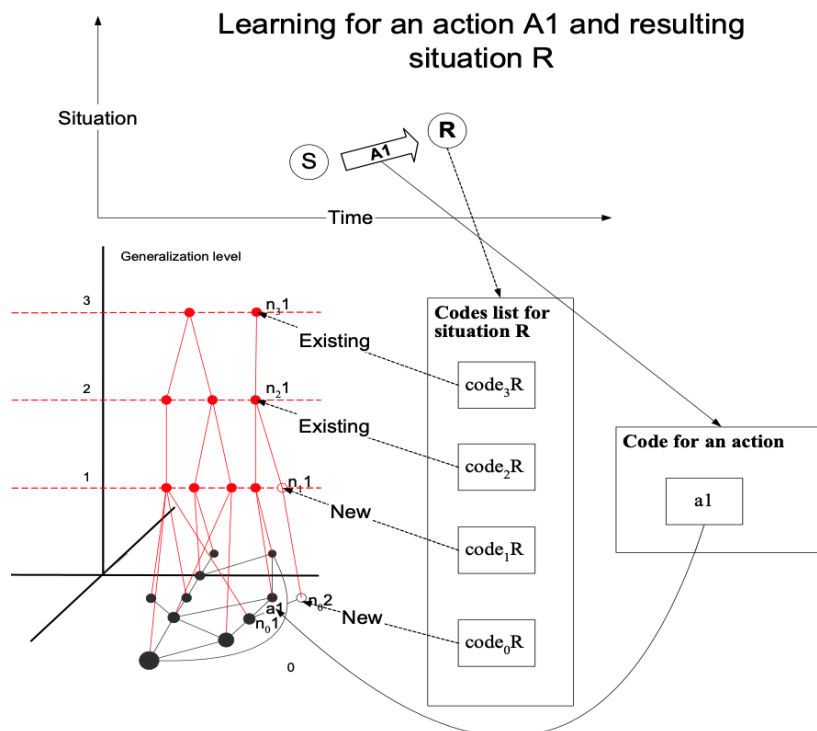


Figure 3.9: Learning process for action.

Learning algorithm is illustrated on Figure [3.10](#).



Figure 3.10: Learning algorithm.

Acting graph along with generalization forest are stored in the random access memory and also periodically (i.e. by the end of training session) stored in the long-term memory.

During acting, the system's task is to find appropriate actions for a given situation. Search for an appropriate action in a specific situation is comprised of two steps: search for an ordered list of similar situations in the knowledge in the form of acting graph nodes (situations search) and choice for the most appropriate action (action choice).

Situations search includes two steps: existing  $i_{th}$  generalization search and collection of siblings of  $0_{th}$  level.

Let's assume that the system is searching for a situation  $S$  in its knowledge. First, the system must derive a list of codes for situation  $S$ . After that, it conducts  $i_{th}$  generalization search which is a consecutive search for  $code_0$  in  $Code0ActingGraphNodeMap$ ,  $code_1$  in  $Code1GeneralizationTreeNodeMap$ , ...,  $code_i$  in  $CodeiGeneralizationTreeNodeMap$  until the  $i_{th}$  code ( $i_{th}$  generalization) is found (i.e. there is an association between  $code_i$  and existing  $node_i$ ).

After  $i_{th}$  generalization –  $node_{i_{th}}$  is found, the system collects all children of  $0_{th}$  level for a  $node_i$ , i.e. the system accesses a list ( $NodeList_{i-1}$ ) of  $i - 1$  nodes associated with the  $node_i$ , and then creates a list ( $NodeList_{i-2}$ ) of  $nodes_{i-2}$  using lists of  $i - 2$  nodes associated with

$i - 1$  nodes from  $NodeList_{i-1}$  and iteratively, through lists of  $i - j$  nodes creates a list of  $node_0$ . Result of the situation search is a list of acting graph nodes  $node_0$ . This process is further referred as the collection of siblings of  $0_{th}$  level.

$Node_i$ , result of  $i_{th}$  generalization search, is a representation of a given situation on the  $i_{th}$  level of generalization. List of siblings of  $0_{th}$  level for  $node_{i_{th}}$  are nodes of acting graph and they have arcs – actions associated with them. Having this list, the system may further consider outgoing arcs for each sibling as possible actions in the given situation.

List of siblings of  $0_{th}$  level for  $node_i$  will be further referred as  $i_{th}$  knowledge source. Arcs for  $nodes_0$  from the  $i_{th}$  knowledge source are  $i_{th}$  knowledge source actions.

Arcs store additional information about situation relevant to a particular class of actions. Action choice mechanism uses this information to order  $i_{th}$  knowledge source actions according to the level of applicability to the current situation. Actions are assigned their rank and some actions are filtered. Action may be filtered/ranked because of the details of current situation are not compatible with the details of the situation stored in the corresponding arc or it may also be filtered/ranked using sequence analysis within the acting graph. Sequence analysis may check if the sequence of actions and situations that followed after the proposed action is likely to be repeated. This analysis also utilizes the information that stored in the arcs – additional details about the situation.

Action filtering may filter all actions from  $i_{th}$  knowledge source, i.e. no actions will be considered by the system as appropriate to the current situation. The action choice consecutively uses  $i + j_{th}$  knowledge sources until the appropriate action is found.

Described  $i_{th}$  knowledge sources are implementations of static generalization mechanism. The system also utilizes dynamic generalizations. To create dynamic generalization the system applies a function  $D_i$  to a  $code_i$ . This function modifies  $code_i$  to  $D_i(code_i)$  and therefore virtually changes the situation that is being searched. For  $D_i(code_i)$  the system may also do a search for  $i + j_{th}$  (for  $j \geq 0$ ) generalization. Result of this search will further be referred as  $D_i, i + j$  knowledge source.

The action choice mechanism hierarchically combines different dynamic and static knowledge sources to create a bigger pool of actions to choose from.

The system's action choice mechanism may also use several special knowledge sources: sequence knowledge sources, blind knowledge source, wait knowledge sources and jump knowledge sources.

Sequence knowledge sources play important role to preserve sequencing of actions. When the system chooses an action  $a1$  for a situation  $S$  from a node  $node_0S$ , it remembers a result of an action  $a1 - node_0R$  of acting graph, i.e. arc  $a1$  connects  $node_0S$  and  $node_0R$  in the acting graph.  $Node_0R$  is called a link node and list of codes corresponding to that node is link codes list. Sequence  $i_{th}$  knowledge source is an empty list if current link node cannot be found in  $i_{th}$  knowledge source for a link  $code_i$ . If current link node is found in  $i_{th}$  knowledge source for a link  $code_i$  then sequence  $i_{th}$  knowledge source is a single node – link node. Therefore, sequence  $i_{th}$  knowledge sources are knowledge sources that may include only one  $node_0$  – link node which is a representation for a situation that the system expects to be a result of the previous action. Sequence  $i_{th}$  knowledge source must be empty if link  $code_i$  cannot be found in  $i_{th}$  knowledge source, i.e. the expected result of the previous action cannot be viewed as  $i_{th}$  generalization of current situation. Using sequence  $i_{th}$  knowledge sources, the system may give preference to actions that belong to a certain path in the acting graph – path that goes through a previous action.

Jump  $i_{th}$  knowledge sources consist directly of actions that may be applied to the current situation  $S$  to transfer to another situation  $R$  in which  $node_0R$  will likely to be found in  $i_{th}$  knowledge source. These knowledge sources allow the system to generate synthetic actions to transfer itself to a known situation.

Blind knowledge source is a knowledge source that contains a link node. Wait knowledge

source includes synthetic wait actions that allows the system to wait for some time. These knowledge sources are possible solutions for the system to act when it is completely lost, i.e. the current situation is completely unknown.

Actions from different knowledge sources may be grouped, ranked depending on the needs of the particular implementation of the system. Action choice may stop on the first acceptable action or it may continue until all knowledge sources had been queried and all possible actions had been compared to find the best action or the most appropriate action group. Each action may have a rank assigned to it depending on the knowledge source it is coming from and also depending on the comparison of the current situation with additional situation information details stored in the arcs (actions).

The action choice mechanism uses knowledge sources in a way allowing setting the processing time limitations. This functionality is achieved by generating and using *nodes<sub>0</sub>* and/or actions from different knowledge sources one by one, i.e. the system does not have to pre-generate all knowledge sources or even all elements for a knowledge source before starting to check the applicability of the actions. Existing broadly used programming techniques may be utilized to meet such a goal.

This system may be utilized for different kinds of learning. For example, the system may learn to act like some kind of a teacher; the system may learn to act on itself in an unknown environment or/and to learn how to act with particular goals. To suit the later cases, the system may use reinforcement learning with delayed rewards. Reinforcement learning is implemented by assigning weights to all *nodes<sub>0</sub>* and arcs of the acting graph. Some special *nodes<sub>0</sub>* are directly assigned positive or negative rewards. Special *nodes<sub>0</sub>* are the nodes corresponding to some special events in the world in which the system is operating. For example, if the system is utilized for learning to effectively play (also known as play to win) as one player in a computer game of tennis then the special *nodes<sub>0</sub>* can be the representations for situations where the player or the opponent scored. If player scores the system gets the positive reward and if the opponent scores the system gets the negative reward in a form of, for instance, a positive or negative weight associated with corresponding special nodes. As soon as at least one special node gets the reward, the system may propagate the weight back using the paths in the acting graph that leads to the special node. As a result, *nodes<sub>0</sub>* and arcs (actions) are assigned weights accordingly to the delayed rewards received by the special nodes. Existing graph-theory techniques may be utilized to implement this weight distribution.

The acting mechanism may use described weights to implement acting with goals. To achieve this, actions ranking takes weights into consideration and, to achieve positive performance, gives preference to the actions with higher weights.

If the system is utilized to learn by trial and error in the unknown environment then the system may be coupled with any kind of external decision making system to provide the initial pool of possible actions for unknown or not-well known situations. When acting in well-known situations the system will utilize its action choice mechanism and when acting in other situations the system will use actions from external decision making system. As weights are assigned to the actions, the system will tune its acting to match the provided goals. External decision making system may be a rule-based system, expert-system or any other system providing actions for a given situation.

When the system already has knowledge from training, acting with goals is also possible. The system may choose actions from its knowledge (not from external decision making system), but affect the ranking of these actions by calculated weights. Therefore, the system may be utilized to effectively act from learned experience, i.e. the system may learn from a teacher, but during acting use learned experience with any kind of preferences.



## Chapter 4

# Data Preprocessing

In this chapter we discuss the procedures we employ for automatic detection of ball events in spatiotemporal soccer datasets. By using them, it is possible to extend existing datasets with information about player movements, ball possession, passes, shots on goals, and tackles, necessary for a variety of sports analysis-related tasks. Our primary goal is to ensure the simplicity and accuracy of the methods, and their easy integration into a dataset processing pipeline. Therefore, in most cases, we rely on straightforward rule-based approaches, if they can be easily configured and show sufficient accuracy.

The chapter is organized as follows. The “Related Works” section briefly discusses the traits of soccer spatiotemporal datasets available today. We note the widespread absence of event markup and examine existing approaches to generate it. We discuss the latest works in this area, which are mostly focused on application and evaluation of specific methods for event recognition. The “Experimental Setup” section provides a description of the datasets we use and their internal structure. We also define the specific list of events we are interested in, and note their properties. The next section titled “Game Event Detection Pipeline” discusses specific algorithms and procedures used to detect each event type. Sufficient details are provided to recreate an independent implementation, if necessary. The “Event Detection Quality Evaluation” section is dedicated to the analysis of accuracy of the proposed approach. We report results independently for each of our datasets and estimate test accuracy using  $F_1$  score. The final section, the Discussion, restates our contribution. We show how the obtained results are connected with our initial goals, what the advantages and limitations of our approach are, and outline possible further research directions.

### 4.1 Related Works

Sports provide a vast amount of diverse information, ranging from statistical tables reflecting the performance of individuals and teams to video streams of competitions. However, different sources of information provide different types of data, and it is not always possible to combine them to obtain a comprehensive picture of certain phenomena.

The reasons for such fragmentation are both technological and legal. For example, the English Premier League compiles and publishes basic statistical facts about their soccer players, and, in general, such data is often reported in newspapers and various “yearbooks”. Some research works are focused on analyzing soccer video streams and detecting the events directly from them [63, 64]. However, video recordings of complete matches of a particular event might be harder to obtain, and their distribution is typically strictly controlled by the copyright holder. Soccer video recordings that capture the whole field are rarely available at all. Often the content of a particular dataset reflects the collectors’ understanding of what constitutes “interesting” information. For example, an often-cited F24 soccer feed [65] provided by Opta Sports [66] includes specific types of in-game events, manually annotated by the experts [67]. Copyright

issues limit the availability of such datasets, and the question of ownership in borderline cases remains open [68].

We are specifically interested in spatiotemporal soccer player tracking data with ball-event markup. Player tracking data is available from several providers, such as StatsPerform.com, DataStadium.co.jp, and Chyronhgo.com. Typically, it is obtained by software-assisted digitization of video streams, recorded by several fixed cameras, installed at a stadium [69]. This technological process presumes no additional post-processing, therefore producing player and ball coordinates only. On the other hand, ball-event datasets, such as the aforementioned Opta's F24 feed or the *Soccer match event dataset* [50] are typically focused on events, and do not provide complete player tracking data. In cases where manual event annotation is available, time synchronization might be inaccurate, and errors in player attribution are common [70].

Event detection in soccer datasets is a subject of several recent research works. A survey made by Gudmundsson and Horton [71] lists numerous tasks related to the spatiotemporal analysis of team sports. However, as of early 2016, only a few attempts were made to deal with ball-event processing. Furthermore, these works are dedicated either to categorizing and labeling known events [72, 73] or predicting future events [74, 75]. Certain approaches to automatic event detection are proposed only in more recent papers [70, 76, 77].

Richly *et al.* [70, 76] apply several different machine learning-based methods to recognize four event types (pass, reception, clearance, shot on goal) in a spatiotemporal soccer dataset. They prepared a “gold standard” manual markup comprising 194 events within 8:47 min of active game time and used it for training and testing different methods, including Support Vector Machine, K-Nearest Neighbors, Random Forest, and Artificial Neural Networks. Their best results are achieved with neural networks, yielding a precision of 89% and a recall of 90%. Interestingly, a significant quality improvement was obtained by applying a smoothing filter that reduced the original 25 Hz sampling rate of game recordings to 10 Hz. It is also worth mentioning that the resulting event markup does not include extended event information, such as a passer's and a receiver's identity in the case of a pass event. Finally, it can be argued that the original training set is relatively small (it includes only seven shots on target, for example), so a more thorough evaluation might be necessary to evaluate the quality of the proposed approach in practical scenarios.

Morra *et al.* [77] experiment with a much larger Soccer dataset comprising 500 minutes of gameplay. This dataset consists of artificial (simulated) soccer matches obtained with an open-source Gameplay Football engine [78]. This approach makes it possible to evaluate their algorithm on thousands of events, annotated with perfect accuracy. However, such data should be treated as an approximation of a real scenario, where camera jitters and imperfect player tracking methods often lead to artifacts that can be observed in digitized recordings of actual soccer matches. The method for detecting game events is based on a set of handcrafted rules, expressed with temporal logic statements. These rules are implemented using the ETALIS library for Prolog. The work of Morra *et al.* deals with a more complex set of events, so a direct comparison of the obtained results with the ones obtained by Richly *et al.* [70, 76] would be inaccurate. Still, the authors report an improvement to precision of 96%, and to recall of 93%.

## 4.2 Experimental Setup

As noted by Morra *et al.* [77], it is difficult to compare results obtained in different research works due to differences in experimental settings, types of events, and used datasets. Therefore, details of the experimental setup are essential, as they might have a significant impact on results. In addition, details of our datasets show what can be expected from acquired player tracking data in typical cases.

We work with two distinct spatiotemporal soccer datasets, obtained by tracking players with several fixed cameras, and subsequent digitization of recorded video streams (see Table 4.1).

The first dataset (“DS”) represents five full matches of the Japanese J1 league captured in 2011. These recordings accurately capture the course of the game, and even significant pauses within the game are not removed. Player tracking is accurate in general, but occasional jitters do occur, so one may observe sudden jumps of player and ball objects. Such situations usually happen when many players are close to each other, particularly during free and corner kicks. Each captured frame is annotated with two additional binary attributes: *ball owning team* (home/away) and *ball status* (dead/alive), so we know which team is the attacker, and whether the game is stopped by the referee.

The second dataset (“ST”) [7] consists of a large number of short game episodes (from 5 to 150 seconds), taken from recent matches played in a top European league. An episode starts when a certain team gets possession of the ball, and ends when the team loses the control of the ball. Player movements are smooth, and jitters are rare. However, this dataset has few shots on goal and few unsuccessful passes. Occasionally the outcome of the attacking team’s last action can be analyzed, but episodes ending with the ball still in the air are also common.

Table 4.1: Experimental datasets

Name (provider)	Sequences	Total time	Frames per sec	Additional features
DS (Data Stadium)	5	320 min (5 games)	25	<i>ball owning team</i> and <i>ball status</i> fields tracking data of a referee z-coordinate of the ball timewasting periods are included player names and roles are known
ST (Stats Perform)	7578	2220 min (45 games)	10	attacking team is always on the left no “dead” segments (stopped by referee) teams are always complete (11 players) player data is anonymized

The ST dataset is organized as a collection of independent sequences, representing game episodes. Each sequence contains a list of frame objects, organized as follows:

$$(G_x, G_y)(P1_x, P1_y) \dots (P10_x, P10_y)(G'_x, G'_y)(P1'_x, P1'_y) \dots (P10'_x, P10'_y)(B_x, B_y)$$

A frame starts with the  $(x, y)$  coordinates of the goalkeeper of the team currently possessing the ball. After the goalkeeper coordinates, the coordinates of the field players of the same team are listed. There is no predefined order of players in this list, but the same order is preserved throughout the episode, so it is possible to trace the trajectory of a certain player in the given episode. The total number of field players is always ten, which means that only episodes featuring complete teams are included in the dataset. The next block describes the opposing team in the same manner. Finally, there are  $(x, y)$  coordinates of the ball. All coordinate values lie within a range of  $[-52.5 \text{ m}, +52.5 \text{ m}]$  along the  $x$ -axis, and  $[-34.0 \text{ m}, +34.0 \text{ m}]$  along the  $y$ -axis.

We convert the DS dataset into the same representation, though the original format contains more information (see Table 4.1).

Current literature shows that there is no universal agreement on the list of events that need to be identified in such player-tracking data, so different authors develop their own schemes suitable for their goals. For our ultimate goal (the development of a case-based reasoning soccer AI system), the following events were identified:

- **Successful pass event.** A player successfully passed the ball to an identified teammate.

- **Unsuccessful pass event.** A player tried to make a pass to an identified receiver, then the ball left the field or was intercepted by the other team. Note that “clearance” events (when the players kick the ball away from their own goal line) often fall into this category.
- **Shot on goal.** A player attempted a shot on goal, characterized by a certain target point.

In addition, we identify 1) the player currently possessing the ball, and 2) player movements, defined as the speed and direction of the given player calculated with the required precision.

### 4.3 Game Event Detection Pipeline

#### 4.3.1 Player Movements Analysis

In our work, we are only interested in the basic approximation of actual player movements. We divide an input game segment into equal intervals of user-specified duration and calculate player speeds using Equations (4.1) and (4.2) according to their positions at the beginning and end of each interval. Note that this approach will represent player movements with straight lines, approximating the general trajectory (see Figure 4.1).

$$v_y = \frac{y(t_1) - y(t_0)}{t_1 - t_0} \quad (4.1)$$

$$v_x = \frac{x(t_1) - x(t_0)}{t_1 - t_0} \quad (4.2)$$

Even such a simple method allows us to study player movements with arbitrary precision and analyze their basic traits. In particular, we were able to distinguish real teams from teams comprised of rule-based AI bots by analyzing probability distributions of player movement directions in different zones of the game field [79].

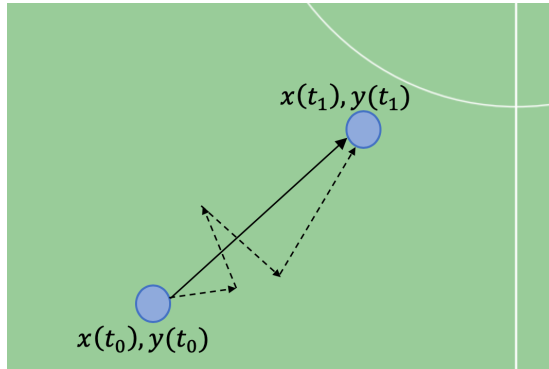


Figure 4.1: Approximation of player movement speed and direction

However, it should be mentioned that in the general case, the analysis of player movements is a more complicated process. In raw data captured by player tracking systems, jitter is inevitable, so certain smoothing algorithms are required. The choice of these algorithms, in turn, is not a trivial process, since sudden changes in speed and direction are very common in soccer, so smoothing may cause undesirable distortions [80]. One way of removing them is to use the Gaussian smoothing kernel [81], where  $x$  and  $y$  components of the trajectory are processed separately and treated as one-dimensional time-dependent signals, according to Equations (4.3)–(4.4).

$$x'(t) = \frac{1}{2N_F + 1} \sum_{i=-N_F}^{N_F} x(t+i) \cdot G(i) \quad (4.3)$$

$$y'(t) = \frac{1}{2N_F + 1} \sum_{i=-N_F}^{N_F} y(t+i) \cdot G(i) \quad (4.4)$$

Here  $2N_F + 1$  denotes the width of the kernel,  $x'$  and  $y'$  are the smoothed components of the trajectory, and  $x$  and  $y$  are the components of the raw trajectory.  $G$  is the set of Gaussian coefficients defining the shape of the kernel.

### 4.3.2 Ball Possession

Both our datasets contain information about the team currently possessing the ball. Each frame in the DS dataset has a special attribute indicating a ball-owning team. Each recording in the ST dataset corresponds to a short game episode, where the team on the left-hand side possesses the ball. However, it is also necessary for us to know which player of the ball-possessing team dribbles the ball at a given moment. In general, three separate cases have to be identified:

1. The player dribbles the ball, so the ball is located in the immediate vicinity of the player.
2. The ball is outside the immediate reach of any player, but it can be treated as “being possessed” by a certain player. For example, a player might kick the ball out of immediate reach, but still keep it under control.
3. The player performs a pass or attempts a shot on goal. During this event, the ball is not possessed by any player, but we still can treat the current team as possessing the ball, until it is intercepted by the other team.

The first case is easy to identify: we simply check the distance between the ball and the closest player of the ball owning team. If this distance is shorter than a certain threshold *VicinityThreshold* (see Table 4.2), the player is treated as possessing the ball. However, if several teammates are located near the ball, we give the preference to the player who controlled the ball on the previous frame to avoid unwanted changes in ball possession.

Table 4.2: Initial experimental setup: parameters

Parameter	Value	Description
VicinityThreshold	1.0 m	Minimal distance between the ball and the ball-possessing player
GracePeriod	1.0 sec	Time required for a player to be considered a ball receiver
GoalpostDistance	5.0 m	Minimal distance from a goalpost for a kick to be considered a pass
MinFailedPassLength	2.5 m	Minimal ball travel distance of an unsuccessful pass
MinTrChangeAngle	12.5°	Minimal change in trajectory direction considered “significant”
MinSpeedChangeFactor	1.5	Minimal change in ball speed considered “significant”

The second case is treated by our pipeline as possession with an additional “*ball far away*” flag set. If the ball leaves an immediate vicinity of a ball-possessing player  $p$ , we run the procedure described in Listing I

---

**Algorithm 1** Detecting possession of a faraway ball

---

**Require:** ball is possessed by player  $p$ ;  $\text{DISTANCE}(\text{ball}, p) \geq \text{VicinityThreshold}$

```
1:  $\text{startFrame} \leftarrow \text{CURRENTFRAME}()$ 
2:  $\text{currFrame} \leftarrow \text{CURRENTFRAME}()$ 

3: loop
4:   if  $\text{DETECTPASSORSHOT}(\text{currFrame})$  then ▷ Defined in Listing 3
5:     mark segment  $[\text{startFrame} - 1, \text{currFrame})$  as a pass or a shot
6:     break

7:   if  $\text{DISTANCE}(\text{ball}, p) < \text{VicinityThreshold}$  then
8:     mark segment  $[\text{startFrame}, \text{currFrame})$  as player  $p$ 's possession with ball far away flag set
9:     break

10:   $\text{currFrame} \leftarrow \text{NEXTFRAME}()$ 
```

---

Special handling of such segments allows us to perform a finer-grained analysis of possible actions in specific game moments. Players possessing a faraway ball cannot perform passes and shots on goal until they reach the ball again.

The third case (passes and shots on goal) requires more complex detection procedures, which will be covered in the next subsection.

### 4.3.3 Detecting Passes and Shots on Goal

When a ball-possessing player attempts to pass the ball to a teammate, we register the occurrence of a pass event. Some passes are successful, while others end with the ball out of bounds or intercepted by the opposing team. A passer, a target receiver, and a pass result (successful/-failed) comprise pass event markup in our system.

A simple approach for detecting a pass, therefore, can be based on detecting two consecutive basic events: 1) the ball leaves the immediate vicinity of a ball-possessing player; 2) the ball goes out of bounds or comes into the possession of another player (either a teammate or an opponent). In the case of the ST dataset, detecting whether the ball is intercepted by a certain player is not an entirely straightforward process due to the absence of a  $z$ -coordinate of the ball. Data does not show whether the ball approaches the player at a low trajectory or flies high above the player's head. A sharp change of the ball trajectory or speed near a player is a good indication of pass reception. However, passes that do not significantly alter ball movement are also common (for example, defenders often make forward passes to midfielders moving in the same direction). Thus, we use both change of ball speed/trajectory and the presence of a certain “grace period” (see Table 4.2) when the ball is near the potential receiver as indications of a pass event (see Listing 2).

**Algorithm 2** Detecting changes in ball possession**Require:** ball enters vicinity of a non ball-possessing player  $p'$ 


---

```

1: function ISPOSSESSIONCHANGED(currFrame)
2:   currFrame  $\leftarrow$  CURRENTFRAME()
3:   prevFrame  $\leftarrow$  GETFRAMEAT(currFrame - 1)
4:   nextFrame  $\leftarrow$  GETFRAMEAT(currFrame + 1)

5:   prevSpeed  $\leftarrow$  CALCULATEBALLSPEED(prevFrame, currFrame)
6:   nextSpeed  $\leftarrow$  CALCULATEBALLSPEED(currFrame, nextFrame)
7:   speedChange  $\leftarrow$  MAX(nextSpeed, prevSpeed) / MIN(nextSpeed, prevSpeed)
8:   if speedChange > MinSpeedChangeFactor then
9:     return true  $\triangleright$  speed changed

10:  prevDir  $\leftarrow$  CALCULATEBALLDIRECTION(prevFrame, currFrame)
11:  nextDir  $\leftarrow$  CALCULATEBALLDIRECTION(currFrame, nextFrame)
12:  if  $|nextDir - prevDir| > MinTrChangeAngle$  then
13:    return true  $\triangleright$  trajectory changed

14:  if ball is still possessed by  $p'$  at currFrame + GracePeriod then
15:    return true  $\triangleright$  ball possessed by  $p'$  after grace period

16:  return false  $\triangleright$  no change in ball possession so far

```

---

A ball going out of bounds is another indication of an unsuccessful pass event. Such passes have to be distinguished from shots on goal. We believe that for most practical tasks, it is enough to treat an event, which ends with a ball passing closer than a certain distance *GoalpostDistance* (see Table 4.2) from the nearest goalpost, as a shot on goal, and all other “out of bounds” situations as resulting from unsuccessful passes. Since shots are characterized by a shot target point in our system (a single value, representing an offset from the goal center), we note the specific location where the ball crosses the goal line. If this point is outside the goal, we correct the value by moving it to the nearest goalpost and treat this new target as the true intention of the attacker.

Finally, we need to identify a target receiver of an unsuccessful pass. This can be a challenging task even for a human observing the game, especially using a 2D visualization. In the current system, we use the following heuristics: an intended receiver is the teammate closest to the ball at the moment when it has left the field or has been intercepted by the opponent. In addition, we filter out all opponent-intercepted passes shorter than *MinFailedPassLength* (see Table 4.2). Such situations are treated as tackles: the ball is merely transferred from one team to another without a pass or shot attempt.

As a result, in the current version of the system, we implement the following procedure for detecting passes and shots on goal (see Listing 3).



---

**Algorithm 3** Detecting passes and shots on goal

---

```
1: function DETECTPASSORSHOT( )
2:   if  $|Ball_y| > 34.0$  then                                ▷ ball crossed the touchline
3:     return VERIFYFAILEDPASS( )
4:   if  $|Ball_x| > 52.5$  then                                ▷ ball reached the goal line
5:     if  $DISTANCE(Ball_y, 0) < GoalpostDistance + GoalLength/2$  then
6:       return shot on goal
7:     else
8:       return VERIFYFAILEDPASS( )
9:   if ball is within vicinity of another player  $p'$  then
10:    if ISPOSSESSIONCHANGED( ) is false then              ▷ see Listing 2
11:      return no event detected
12:    if  $p'$  is a teammate then
13:      return successful pass
14:    else
15:      return VERIFYSHOT( )

16: function VERIFYFAILEDPASS( )
17:   if pass distance is longer than  $MinFailedPassLength$  then
18:     return unsuccessful pass                                ▷ See Figure 4.2(a)
19:   else
20:     return no event detected

21: function VERIFYSHOT( )
22:   if  $p'$  is in the goal area and ball trajectory line crosses the goal line
23:     not further than  $GoalpostDistance$  from the goalpost then
24:       return shot on goal                                  ▷ See Figure 4.2(b)
25:   else
26:     return VERIFYFAILEDPASS( )
```

---



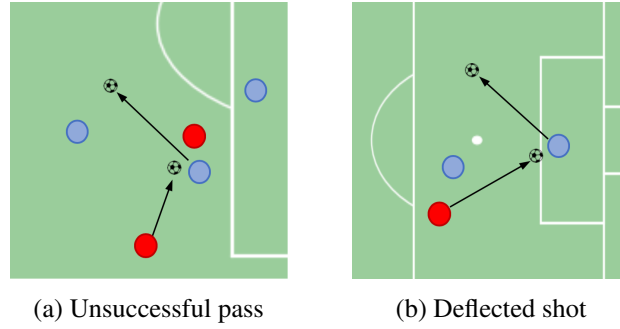


Figure 4.2: A scheme of an unsuccessful pass and a deflected shot

## 4.4 Event Detection Quality Evaluation

The most complicated part of event markup in our pipeline is the detection of passes and shots. Thus, we performed a brief evaluation to estimate its accuracy.

We created a “gold standard” markup by watching the games that comprise our datasets in a 2D soccer simulator and manually annotating successful passes, unsuccessful passes, and shots on goal. In the case of the ST dataset, we analyzed the 40 longest episodes, corresponding to 60 minutes of playing time in total (see Table 4.3). In the case of the DS dataset, all five matches were annotated. Because manual annotations are difficult to synchronize with events, we consider events as correctly recognized if they fall within  $[-0.5s, +0.5s]$  interval of the corresponding “gold standard” event.

Table 4.3: Annotated events

Event	Quantity	
	ST	DS
Successful pass	1097	2910
Unsuccessful pass	95	331
Shot on goal	8	97

Since the proposed algorithm relies on specific parameter values, listed in Table 4.2, fine-tuning them is necessary to achieve high event detection performance. The initial experimental setup is based on manually chosen values, reflecting our general understanding of soccer and observations of the gold standard markup process.

To reveal the optimal combination of parameter values, we applied a greedy search routine. It evaluates the algorithm on the set of parameters  $\{p_1, \dots, p_i, \dots, p_n\}$ , where the value of  $p_i$  is iterated over a predefined range, while the rest of the values remain constant. The same process is repeated for each parameter in the set. This approach is feasible in our case because most parameters are loosely related and affect different types of situations in the game. For example, both *MinTrChangeAngle* and *MinSpeedChangeFactor* values affect the ability of the system to recognize changes in ball possession. However, they are introduced to deal with different types of game episodes (see Listing 2), so an optimal value of one parameter should increase the overall performance of the whole event detection algorithm. It is also easy to choose reasonable ranges and loop steps due to physical constraints of soccer.

The quality of event detection can be estimated with recall, precision, and  $F_1$  score values, calculated according to Equations (4.5)–(4.7). The search routine iterates over a range of permissible values of an individual parameter, while the rest are fixed at their initial values, specified in Table 4.2. We perform this routine for each parameter except *GoalpostDistance*,

which affects shot on goal events only. Since the initial setup already provides optimal recognition of shots on goal, we decided to keep the value of *GoalpostDistance* unchanged. It should also be mentioned that we performed a search for optimal parameter values using the ST dataset only. The DS dataset contains significant player jitter, and thus cannot serve as a reliable ground for fine-tuning.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (4.5)$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (4.6)$$

$$F_1score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (4.7)$$

The evaluation routine<sup>1</sup> allows us to make the following observations (see Figure 4.3). Any choice of *MinFailedPassLength* and *MinSpeedChangeFactor* within the test ranges have virtually no effect on the resulting performance. The optimal value of *MinTrChangeAngle* is approximately 0.1; all higher values cause errors in pass events recognition. The optimal values of *GracePeriod* and *VicinityThreshold* are achieved inside their respective ranges; any deviation from the optimum increases recognition errors. On the other hand, nearly optimal results (with  $F_1$  score higher than 0.9) are obtained on wide range of values. The optimal choice of parameters is provided in Table 4.4.

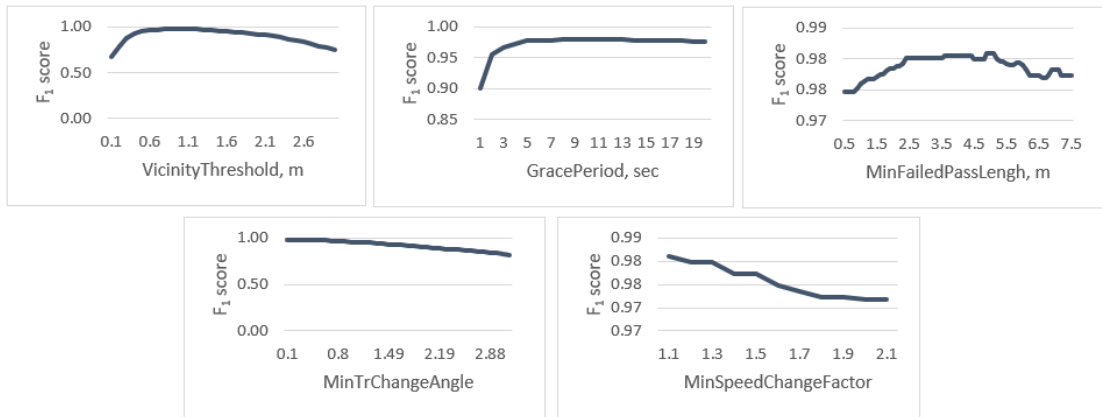


Figure 4.3: The influence of parameter values on event recognition quality

Table 4.4: Optimal parameter values

Parameter	Value
VicinityThreshold	0.9-1.2 m
GracePeriod	0.5-2 sec
MinFailedPassLength	0.9-7.5 m
MinTrChangeAngle	5°-25°
MinSpeedChangeFactor	1.1-1.5

The resulting evaluation of our algorithm, performed with the optimal set of parameters on both ST and DS datasets, is shown in Table 4.5. Since the performance of the algorithm is stable

<sup>1</sup>The source code for event recognition and parameter tuning is available at [github.com/vi3itor/soccer-event-recognition](https://github.com/vi3itor/soccer-event-recognition)

on a wide range of input parameter values, we believe it is able to achieve comparable results on other spatiotemporal soccer datasets.

Table 4.5: Accuracy of event detection

Event	ST			DS		
	Precision	Recall	F <sub>1</sub> score	Precision	Recall	F <sub>1</sub> score
Successful pass	0.9991	0.9973	0.9982	0.9218	0.9470	0.9342
Unsuccessful pass	0.8318	0.9368	0.8812	0.8182	0.9013	0.8577
Shot on goal	1.0	1.0	1.0	0.9336	0.9607	0.9469

We should note that the DS dataset is more difficult to analyze both manually and automatically. This happens because of lower tracking accuracy, especially in the case of overlapping players. DS tracking data is obtained with TRACAB technology available in 2011; the current Generation 5 TRACAB system provides more accurate results in such scenarios [82].

Errors in event recognition, generally, occur in borderline cases. For example, there were situations where a trajectory of a flying ball was slightly changed after passing a teammate's vicinity. The algorithm considered this change insignificant, while a human expert recognized it as a one-touch pass made by a teammate. Similarly, there were discrepancies in recognizing intended ball receivers in cases of unsuccessful long-distance passes.

As seen in Table 4.3, the ST dataset contains few unsuccessful passes and shots on goal. Thus, the evaluation of detection accuracy for these events is not as reliable as for the DS dataset. The lower accuracy of event detection for the DS dataset can be explained by considerable player and ball jitter. The DS dataset is recorded at a 25 Hz rate, and there is no trajectory smoothing, so our algorithms occasionally cannot project actual player movements.

## 4.5 Discussion

Event recognition in player tracking data is a subject of several research works. The best results to date were reported in Richly *et al.* [76] and Morra *et al.* [77]. However, it seems that obtaining the most accurate results was not the main goal of these works. Both papers are focused on the evaluation of specific methods as seen from the conclusions made by their respective authors: “the results showed that neural networks present a viable model to detect events in soccer data” [76]; “we have shown that ITLs are capable of accurately detecting most events from positional data” [77]. Therefore, high accuracy demonstrates the versatility of the suggested methods and provides a firm ground for their use in similar tasks.

Our primary goal was to develop a procedure that would provide a quick and accurate event markup of specific soccer datasets we use. It is hard to say how well the same approach would work in other team sports games. However, in the case of soccer, the obtained results are very close to the “gold standard” manual markup, so we consider our algorithms ready for practical use. We have to repeat that specific precision/recall values reported in the present work should be treated as dependent on the experimental setup, and specific types of events in particular.

In general, we should note that our rule-based approach possesses a number of advantages. It is simple, straightforward, and can be easily implemented in any conventional programming language required in the given project. It does not need a large annotated dataset for learning, which might be important for rare events such as shots on goal, where it is hard to collect a sufficient number of observations. It can be easily updated or modified, and it can serve as a baseline procedure for evaluating other algorithms based on more advanced methods.

One obvious problem with our approach is related to its dependency on specific parameter values listed in Table 4.2. Thus, the quality of event recognition might vary across datasets

(corresponding to teams of different skill levels, for example). However, present parameter values were chosen based on general knowledge of soccer and demonstrated robust performance on wide value ranges, so we believe they are applicable to diverse collections of soccer matches.

Summing up, the resulting procedure is simple, fast and accurate, is able to recognize soccer events with comparable or higher precision than competing approaches. However, its flexibility is limited: while its operation can be adjusted with fine-tunable parameters, any changes in the list of supported events might require significant code update.

## Chapter 5

# Evaluation Results

One of the most interesting research questions in this chapter is to find out how well the patterns of human team behavior can be translated into a simplified world of a video game. Our experimental setup is based on Buckland’s SimpleSoccer simulator, serving as a good model of a simple arcade-style soccer game [12] (Ch. 4).

One of the principal presumptions in our project is the availability of *limited* datasets of player tracking data. With the growing adoption of tracking systems, we can expect a higher availability of such data in general. However, a typical professional soccer team plays only several dozens of matches per season, so if we plan to learn team-specific behavior patterns, we are limited to small data samples, especially for processing relatively rare game events (such as shots on goal or throw-ins). Therefore, it was considered reasonable to rely on the methods used in our previous projects [83, 84] and described in Chapter 3. Next is a succinct description of the system, which also contains the values of the attributes that were used in the experiment.

### 5.1 Method and Parameters

Virtual agents’ knowledge is represented as a graph, having individual game situations as vertices, and actions as weighted edges. This way, it represents the fact that the situation *A* turns into the situation *B* as a result of a certain action during the learning phase. Agents rely on a combination of case-based reasoning decision making with Markov chain-like database of human actions, and the procedure itself can be seen as a variation of a Markov decision process [85]. A decision making algorithm tries to find the best match for the current game situation and acts accordingly.

During the learning stage, our system uses game states and actions from the STATS dataset, taken at the defined time interval (by default, 0.5 second), to create the game graph. The following attributes are used to represent individual game situations:

- **PwbX[range]**, **PwbY[range]**: the coordinates of the player possessing the ball (in the specified range).
- **DMF**: the “danger to move forward” heuristic estimation (0-5); depends on the distance to the nearest opponent in the forward direction.
- **MoveDir**: the current movement direction of the player with the ball (8 directions are supported).
- **CDir**: the direction (0-7) of the closest opponent, from the perspective of the player possessing the ball.
- **CDist**: the distance to the closest opponent, from the perspective of the player possessing the ball (0-2).

- **SPD**: the “safest pass danger” heuristic estimation on the scale of 0 to 5 (depends on locations of both teammates and opponents).
- **SFD**: the “safest forward pass danger” heuristic estimation on the scale of 0 to 5.
- **PassOpt**: the Boolean attribute indicating that at least one safe pass (with danger estimation of 0-2) is found.
- **FFX, SFX**: x-coordinates of two teammates closest to the opponent’s goal line, converted to the range [0, 3].

Every game state in the graph is recorded in three different representations, reflecting different approximations of the actual soccer game state:

- $R_0$ : PwbX[0-17], PwbY[0-9], DMF, MoveDir, CDir, CDist, SPD, SFD, PassOpt, FFX, SFX
- $R_1$ : PwbX[0-17], PwbY[0-9], DMF, FFX, SFX
- $R_2$ : PwbX[0-8], PwbY[0-4], DMF

Chapter 4 thoroughly explains what datasets are used and how the actions are extracted. Each learned pass action is characterized by the coordinates **PwbX** and **PwbY** of both sending and receiving players. Unlike real soccer players, player characters of Buckland’s SimpleSoccer can only move with constant speed in eight predefined directions. Thus, we approximate actual human movements with SimpleSoccer actions. This is done with a simple procedure:

1. Identify the movement performed by the player possessing the ball during the defined time interval (by default, half a second) of the game.
2. Calculate the best matching direction **Dir** for this movement out of 8 possible options.
3. Calculate the duration **D** reflecting the number of frames in SimpleSoccer to cover the required distance.
4. The resulting movement (**Dir, D**) is recorded, and the procedure is repeated for the next time interval.

During acting (decision making) stage, the system tries to find a match for the current on-screen game situation using representation  $R_0$ . If no matches are found, it proceeds to  $R_1$ , and, if necessary, to the “fallback” representation  $R_2$ . If no actions are found or they are not applicable in the given context (the actual onscreen coordinates of senders and receivers are significantly different from the values in the data structure), the system retreats to the built-in AI agent.

## 5.2 Passes

Passes is one of the key elements of a team strategy in soccer. Distinctive passing patterns can be observed even on the level of individual teams and players [17, 86], so, presumably, they are connected to certain human-like behavioral traits, recognizable by the players. Furthermore, passes are abundant in any soccer match (unlike shots on goal, for example), and are easy to classify and compare. For learning passing behavior, we can expect a professional team to make 365-370 passes per game on average [87]. We set the goal to achieve human-like passing behavior by learning from human tracking data using a proposed system.

To evaluate the performance of a new passing algorithm, we compared the characteristics of passes made by the new algorithm, the old (default) algorithm, and by the real-life teams. We

Table 5.1: Pass Length Distribution

Team	Pass length, m %				
	0-10	10-20	20-30	30-40	40+
Default AI	0.00	11.74	52.84	27.84	7.58
New AI	17.44	47.29	20.74	7.36	7.17
Real Teams	28.40	49.36	16.89	3.71	1.37

Table 5.2: Pass Direction Distribution

Team	Pass direction (%)							
	FW	FR	R	BR	B	BL	L	FL
Default AI	8	11	21	11	8	9	17	14
New AI	16	14	17	10	4	7	16	16
Real Teams	10	14	16	12	9	12	15	12

simulated a number of AI vs. AI matches using old and new algorithms until 500 passes are made in each case, and extracted 500 random passes from the STATS.com dataset. The passes were classified according to their length and direction (see Table 5.1 and Table 5.2).

To compare passing patterns, we represented pass length and pass direction values for different teams as vectors and calculated their cosine similarity ratios (see Table 5.3).

The obtained results show that the original AI system does not exhibit human-like passing patterns in terms of distance. However, the distribution of pass directions is close to the distribution found in real data. Our system, based on learning by observation, shows human-like passing behavior according to these criteria.

It is interesting to note that according to the passing direction criterion, both AI systems are reasonably human-like. It is possible that the distribution of passing directions in real matches indeed reflects certain general logic of the game of soccer, and depends on individual team tactics to a lesser degree. It is also possible that our direction/distance classification does not adequately reflect the complexity of passes in real soccer, and passes have to be categorized according to other criteria as well, such as danger estimation or pass quality evaluation, as proposed in [49].

### 5.3 Believable Player Movements

Our primary goal was to confirm that the proposed algorithm is able to control the player possessing the ball and thus can be considered a part of the general AI system, controlling virtual soccer players. Fortunately, SimpleSoccer includes a built-in rule-based AI, so it is possible to set up a match where the player possessing the ball is controlled by our algorithm while the remaining players are controlled by SimpleSoccer AI.

Table 5.3: Similarity of Passing Patterns

	Distance Similarity / Direction Similarity (%)		
	Default AI	New AI	Real Teams
Default AI	-	95.64	97.89
New AI	56.15	-	95.87
Real Teams	43.35	97.45	-

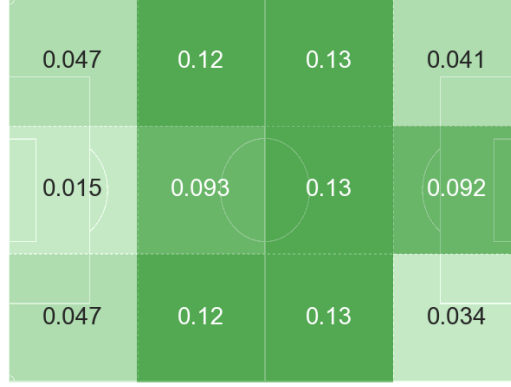


Figure 5.1: A heat map of movements of a player controlling the ball. The attacking team’s goal line is on the right.

In our test runs, the new data-driven AI player exhibits behavior that looks natural. After acquiring both moving and passing behavior (as described in the previous section), the player is able to attack, evade opponents, and make reasonable passes. Its movements are consistent, and there are no erratic changes in the movement direction. The player does not lose the ball unexpectedly, and in general, we are satisfied with the results.

However, we also wanted to perform an objective evaluation of the AI-controlled player’s behavior to make sure it satisfies the stated requirement of “human-likeness.” In order to do it, we decided to compare how AI-supplied actions correlate with movements performed by real athletes in actual games of the STATS dataset.

We divided the original dataset into a subset of 4000 randomly chosen sequences used to train the AI system, and a test set of the remaining 3500 sequences. The decision-making process was initiated on every 5th frame of each test set sequence, and the results were compared with the actions chosen by the original players possessing the ball.

It is important to note that direct comparison cannot be applied in this scenario: there is no predefined “right” action in the given game situation, since player movements are probabilistic in nature, and the same player might decide to behave differently in similar cases. Furthermore, STATS data is anonymized, so it is not possible to learn actions specific to a particular athlete; instead, we acquire “generalized” behavior exhibited by all ball-possessing players.

Therefore, we decided to compare the probabilities of movement directions of human- and computer-controlled players, observed in the same zones of a soccer field. For the purpose of this task we treat the field as consisting of twelve zones (see Figure 5.1).

For each zone, we created a radar chart, showing the probabilities of movement in each of 8 possible directions. The resulting visualization is shown in Figure 5.2. One can note that predominant movement directions of human players indeed depend on their zones: in their own half, players tend to move the ball away from the central zones closer to field flanks and attempt to return the ball closer to the center when approaching the opponent’s goal line.

It is also quite clear that the radar charts of human and AI players are very similar. One way to obtain a numerical similarity ratio estimation is to use the cosine similarity. We can represent the complete behavioral fingerprint of a player as a vector  $V$ , having elements calculated as:

$$V_{8i+j} = p(Z_i) \times p(D_j), \quad (5.1)$$

where  $p(Z_i)$  is the probability of player to be in the zone  $0 \leq i \leq 11$ , and  $p(D_j)$  is the probability of choosing the direction  $0 \leq j \leq 7$  for movement action.

In our case, cosine similarity between the “human vector” and the “AI vector” yields a rate of 0.98. This value should be considered a rough estimation of a “true similarity,” since it does



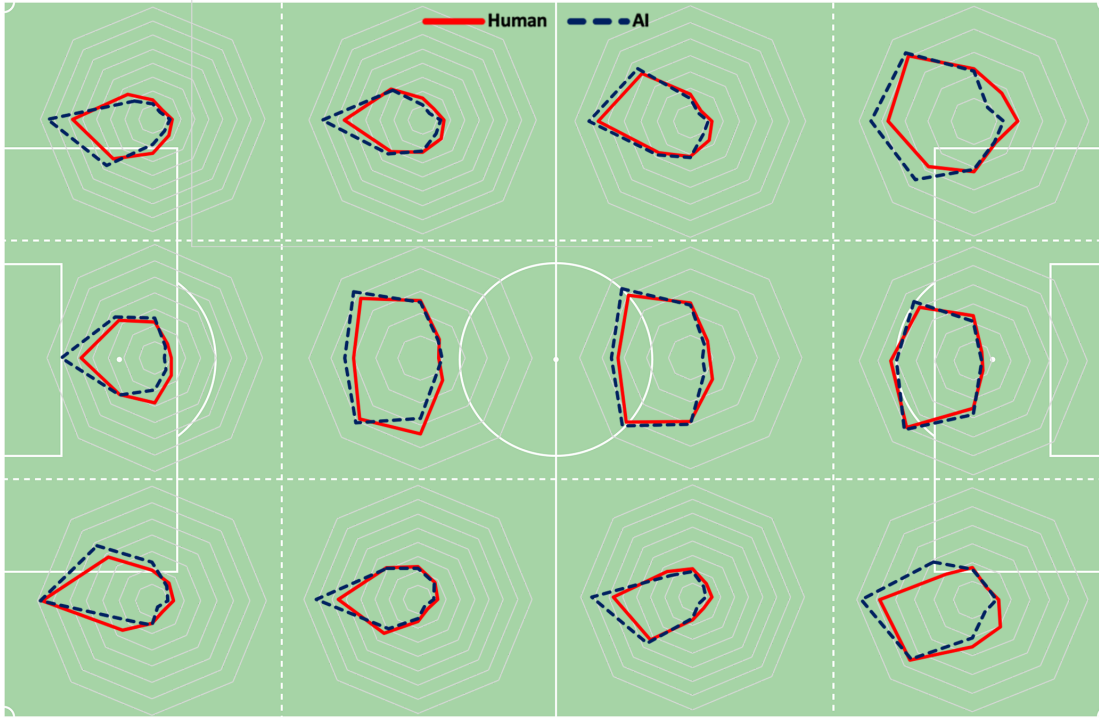


Figure 5.2: Movements of ball-possessing human- and AI-controlled characters. The attacking team’s goal line is on the right.

not take into account fine-grained contextual information about situations where decisions are made, but it still shows that our AI system is indeed able to replicate human behavior to a certain extent.

Our algorithm relies on three handcrafted sets of features, reflecting our understanding of soccer situations, as seen by a player controlling the ball. We recognize that this fact makes our approach a somewhat “ad-hoc solution,” so one of our future priorities will be to make the process of feature selection more transparent and streamlined. In the current experiments, the AI system is able to find a matching game state using  $R_0$  in approximately 19% of cases, while  $R_1$  is responsible for 63% of cases, and “fallback”  $R_2$  is used in the remaining 18% of game situations.

## Chapter 6

# Conclusion

In this work we argued that team sports occupy a special, possibly unique, niche in the AI development landscape, and thus deserve attention on their own, rather than within the context of other multi-agent environments, such as RTS or MOBA games. Sport video games are virtual representations of real-world competitions, and thus hold a borderline position between the real world and the virtual world. On the one hand, they have to capture the essence of real-life events and represent them faithfully to be appealing for sports fans. At the same time, they have to be fun to play and accessible for gamers, who want to be immersed in a make-believe world, where they can play the roles of successful top-class athletes.

A general trend in the sport games genre is convergence between the game and the reality [35], which in practice means more realistic graphics, environments, and AI systems. Good AI is one of the cornerstones of a high-quality team sport game; it is hard to imagine, for example, a soccer-like game set up in a people-only online multiplayer mode, and played without AI.

Currently, the advancements in AI technologies extend the list of games considered “solved” in the sense that AI beats professional human players. In practice, this means that the focus of game AI research can shift to secondary challenges, aimed at maximizing entertainment value of game products. Sport games are especially good for studying such emotional-driven goals of AI: the pool of popular sports events is stable, and we know much about motivation and enjoyment of both participants and fans.

One may argue that the issues related to subjective user enjoyment are hard to deal with: there are no easy ways known to identify AI-related sources of user enjoyment, to implement AI that maximizes fun, nor to evaluate obtained results. However, dealing with *sport game* AI systems, it is impossible to ignore these factors, especially in the light of their growing importance. It is, therefore, both reasonable and necessary to anticipate this trend, by focusing more research effort towards defining an appropriate and long-lasting AI benchmarking environment.

This work describes the design of the adaptive learning system that is capable of decision making in dynamic environments. Such system can learn and act in real-time with a little processor power overhead. Acting graphs allow reinforcement self-learning and learning with goals along with additional sequence analysis by the decision making method.

We have designed and implemented a rule-based algorithm for event detection in a spatiotemporal soccer dataset. Our method achieves high accuracy on two datasets and most probably can be used in other similar scenarios without modification. The proposed scheme can also supply baseline event recognition quality indicators for subsequent research projects. Similar techniques can be used to recognize other significant game events, including offsides, free kicks, and corner kicks. Further research directions may include support for other event types and adaptation of our algorithm for other team sports, such as basketball or ice hockey.

Soccer is a complex multi-agent game, challenging for AI technologies. The diversity and sophistication of technologies used in modern RoboCup AI teams shows how difficult it is to

design a skillful AI system for soccer [88]. However, with the advancements in practical AI development, the importance of other factors such as believability of AI-controlled teammates and opponents will grow. Designing a comprehensive data-driven AI system for soccer is a challenging task, which requires the right choice of methods as well as a good understanding of the game. So far, we have succeeded in using the datasets of player to obtain a reasonably human-like moving and passing behavior of a player controlling the ball. Stable and believable moving behavior is confirmed both with qualitative evaluation and with numerical analysis, yielding a high similarity ratio between human and AI players according to our fingerprint metrics. We analyzed the passing behavior of human teams and compared it with a rule-based AI system according to two features: pass length and pass duration. Our tests show that real teams significantly differ in their passing patterns from the AI. Unfortunately, the dataset contains few shots on goal, so these actions have to be handled separately for now.

Naturally, the most interesting directions for future research are related to multi-agent behavior. Controlling both a player possessing the ball and the rest of the team requires a certain degree of player-player coordination, which makes it especially challenging.

We should also note that the task of translating real-life actions into a game world can be challenging, too. Real athletes possess different abilities, and they operate in a physical world, only partially replicated in a computer game. However, if game designers strive for realism, they might find studying human tracking data insightful.

# Acknowledgments

The completion of my dissertation would not have been possible without the support and nurturing of my research advisor Associate Professor Maxim Mozgovoy. I'd also like to extend my gratitude to the review committee for their valuable comments and feedback. I'm incredibly grateful to my parents, Zhanna and Aliaksandr, for their patience and profound belief in me. And I cannot begin to express my thanks to Veronica for her unparalleled support in everything that I do.

# References

- [1] J. McCarthy, “Chess as the drosophila of AI,” in *Computers, chess, and cognition*. Springer, 1990, pp. 227–237.
- [2] R. Beal, T. J. Norman, and S. D. Ramchurn, “Artificial intelligence for team sports: a survey,” *The Knowledge Engineering Review*, vol. 34, p. e28, 2019.
- [3] L. Sha, P. Lucey, S. Zheng, T. Kim, Y. Yue, and S. Sridharan, “Fine-grained retrieval of sports plays using tree-based alignment of trajectories,” *ArXiv*, vol. abs/1710.02255, 2017.
- [4] C. Tian, V. De Silva, M. Caine, and S. Swanson, “Use of machine learning to automate the identification of basketball strategies using whole team player tracking data,” *Applied Sciences*, vol. 10, no. 1, p. 24, Dec 2019. [Online]. Available: <http://dx.doi.org/10.3390/app10010024>
- [5] X. Wei, P. Lucey, S. Morgan, and S. Sridharan, “Predicting shot locations in tennis using spatiotemporal data,” in *2013 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2013, pp. 1–8.
- [6] A. Bialkowski, P. Lucey, P. Carr, Y. Yue, S. Sridharan, and I. Matthews, “Identifying team style in soccer using formations learned from spatiotemporal tracking data,” in *2014 IEEE International Conference on Data Mining Workshop*, 2014, pp. 9–14.
- [7] H. M. Le, Y. Yue, P. Carr, and P. Lucey, “Coordinated multi-agent imitation learning,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1995–2003.
- [8] Y. Wu, X. Xie, J. Wang, D. Deng, H. Liang, H. Zhang, S. Cheng, and W. Chen, “Forvizor: Visualizing spatio-temporal team formations in soccer,” *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 65–75, 2018.
- [9] N. A. Taatgen, M. van Oploo, J. Braaksma, and J. Niemantsverdriet, “How to construct a believable opponent using cognitive modeling in the game of set,” in *Proceedings of the fifth international conference on cognitive modeling*. Citeseer, 2003, pp. 201–206.
- [10] S. Johnson, “Playing to lose: Ai and civilization,” in *Game Developer Conference*, 2008.
- [11] K. Dill, “What is game AI?” in *Game AI pro*, S. Rabin, Ed. A K Peters/CRC Press, 2013, pp. 3–10.
- [12] M. Buckland, *AI game programming by example*. Wordware Publishing Inc., 2004.
- [13] M. Colledanchise and P. Ögren, *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.
- [14] A. Bastida Castillo, C. D. Gómez Carmona, E. De la cruz sánchez, and J. Pino Ortega, “Accuracy, intra-and inter-unit reliability, and comparison between gps and uwb-based

- position-tracking systems used for time–motion analyses in soccer,” *European journal of sport science*, vol. 18, no. 4, pp. 450–457, 2018.
- [15] S. Russell and P. Norvig, *Artificial intelligence: a modern approach 4th ed.* Pearson Education UK, 2020.
- [16] A. Hewitt, G. Greenham, and K. Norton, “Game style in soccer: What is it and can we quantify it?” *International Journal of Performance Analysis in Sport*, vol. 16, p. 355, 04 2016.
- [17] T. Decroos, J. Van Haaren, and J. Davis, “Automatic discovery of tactics in spatio-temporal soccer match data,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 223–232.
- [18] “RoboCup federation.” [Online]. Available: <https://www.robocup.org>
- [19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, “Playing Atari with deep reinforcement learning,” *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: <http://arxiv.org/abs/1312.5602>
- [21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: <http://dx.doi.org/10.1038/nature14236>
- [22] S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, “A survey of real-time strategy game AI research and competition in StarCraft,” *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 5, no. 4, pp. 293–311, 2013. [Online]. Available: <https://doi.org/10.1109/TCIAIG.2013.2286295>
- [23] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. P. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing, “Starcraft II: A new challenge for reinforcement learning,” *CoRR*, vol. abs/1708.04782, 2017. [Online]. Available: <http://arxiv.org/abs/1708.04782>
- [24] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, Y. Wu, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, K. Kavukcuoglu, D. Hassabis, and D. Silver, “AlphaStar: Mastering the Real-Time Strategy Game StarCraft II,” <https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/>, 2019.
- [25] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, “Dota 2 with large scale deep reinforcement learning,” *arXiv preprint arXiv:1912.06680*, 2019.

- 
- [26] “Collins english dictionary complete and unabridged (13th ed.),” 2018.
- [27] Wikipedia contributors, “Team sport — Wikipedia, the free encyclopedia,” 2019. [Online]. Available: [https://en.wikipedia.org/wiki/Team\\_sport](https://en.wikipedia.org/wiki/Team_sport)
- [28] —, “Quidditch (real-life sport) — Wikipedia, the free encyclopedia,” [https://en.wikipedia.org/w/index.php?title=Quidditch\\_\(real-life\\_sport\)&oldid=985766522](https://en.wikipedia.org/w/index.php?title=Quidditch_(real-life_sport)&oldid=985766522), 2020, [Online; accessed 2-November-2020].
- [29] Gandolfi, G. (Ed.), *NBA Coaches Playbook: Techniques, Tactics, and Teaching Points*. Human Kinetics, 2008.
- [30] Johnston, M. and Walter, R., *Hockey Plays and Strategies, 2nd Ed.* Human Kinetics, 2018.
- [31] Zauli, Alessandro, *Soccer: Modern Tactics*. Reedswain, 2011.
- [32] Tahtouh, Toni Faouzi, *Volleyball: Techniques and Tactics*. Lulu Publishing Services, 2017.
- [33] J. Wilson, *Inverting the Pyramid: The History of Football Tactics*. Orion, 2010.
- [34] C. Koch and M. Tilp, “Beach volleyball techniques and tactics: A comparison of male and female playing characteristics,” *Kinesiology*, vol. 41, pp. 52–59, 06 2009.
- [35] M. Sicart, “A tale of two games: football and FIFA 12,” in *Sports Videogames*. Routledge, 2013, pp. 40–57.
- [36] J. Togelius, “How to run a successful game-based AI competition,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 1, pp. 95–100, 2016.
- [37] A. Juliani, A. Khalifa, V. Berges, J. Harper, H. Henry, A. Crespi, J. Togelius, and D. Lange, “Obstacle tower: A generalization challenge in vision, control, and planning,” *CoRR*, vol. abs/1902.01378, 2019. [Online]. Available: <http://arxiv.org/abs/1902.01378>
- [38] “RoboCup federation: Objective.” [Online]. Available: <https://www.robotcup.org/objective>
- [39] “RoboCup soccer simulation.” [Online]. Available: <https://ssim.robotcup.org/>
- [40] H. Akiyama and T. Nakashima, “Helios base: An open source package for the RoboCup soccer 2D simulation,” in *RoboCup 2013: Robot World Cup XVII*, S. Behnke, M. Veloso, A. Visser, and R. Xiong, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 528–535.
- [41] M. Prokopenko and P. Wang, “Disruptive innovations in RoboCup 2D soccer simulation league: from cyberoos’98 to gliders2016,” *CoRR*, vol. abs/1612.00947, 2016. [Online]. Available: <http://arxiv.org/abs/1612.00947>
- [42] M. Prokopenko, P. Wang, S. Marian, A. Bai, X. Li, and X. Chen, “RoboCup 2D soccer simulation league: Evaluation challenges,” in *RoboCup 2017: Robot World Cup XXI [Nagoya, Japan, July 27-31, 2017]*, 2017, pp. 325–337.
- [43] P. MacAlpine and P. Stone, “UT Austin Villa RoboCup 3D simulation base code release,” in *RoboCup 2016: Robot Soccer World Cup XX*, ser. Lecture Notes in Artificial Intelligence, S. Behnke, D. D. Lee, S. Sariel, and R. Sheh, Eds. Berlin: Springer Verlag, 2017, pp. 135–43.
-



- [44] P. Sweetser, D. Johnson, J. Sweetser, and J. Wiles, "Creating engaging artificial characters for games," in *Proceedings of the 2nd International Conference on Entertainment computing*. Carnegie Mellon University, 2003, pp. 1–8.
- [45] B. Soni and P. Hingston, "Bots trained to play like a human are more fun," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 363–369.
- [46] S. Temsiriririkkul, N. Sato, K. Nakagawa, and K. Ikeda, "Survey of how human players divert in-game actions for other purposes: Towards human-like computer players," in *International Conference on Entertainment Computing*. Springer, 2017, pp. 243–256.
- [47] B. Banerjee and M. E. Taylor, "Coordination confidence based human-multi-agent transfer learning for collaborative teams," in *AAMAS Adaptive Learning Agents (ALA) Workshop*. sn, 2018.
- [48] Lowe, Zach. Lights, Cameras, Revolution. [Online]. Available: <https://grantland.com/features/the-toronto-raptors-sportvu-cameras-nba-analytical-revolution/>
- [49] S. Chawla, J. Estephan, J. Gudmundsson, and M. Horton, "Classification of passes in football matches using spatiotemporal data," *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, vol. 3, no. 2, pp. 1–30, 2017.
- [50] L. Pappalardo, P. Cintia, A. Rossi, E. Massucco, P. Ferragina, D. Pedreschi, and F. Giannotti, "A public data set of spatio-temporal match events in soccer competitions," *Scientific data*, vol. 6, no. 1, pp. 1–15, 2019.
- [51] T. Decroos, L. Bransen, J. Van Haaren, and J. Davis, "Actions speak louder than goals: Valuing player actions in soccer," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1851–1861. [Online]. Available: <https://doi.org/10.1145/3292500.3330758>
- [52] A. Ferrein and G. Steinbauer, "20 years of RoboCup," *KI-Künstliche Intelligenz*, vol. 30, no. 3-4, pp. 225–232, 2016.
- [53] Y. Verhoeven and M. Preuss, "On the potential of Rocket League for driving team AI development," in *Proceedings of IEEE SSCI 2020*, 2020, pp. nn–mm.
- [54] G. Kendall and L. J. Lenten, "When sports rules go awry," *European Journal of Operational Research*, vol. 257, no. 2, pp. 377 – 394, 2017. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0377221716304908>
- [55] G. Bradley and C. Toye, *Playing Soccer the Professional Way*. Harper & Row, 1973.
- [56] Wilson, J., "Why is the modern offside law a work of genius?" 2010. [Online]. Available: <https://www.theguardian.com/sport/blog/2010/apr/13/the-question-why-is-offside-law-genius>
- [57] FIFA, "Goalkeepers are not above the law," 1997. [Online]. Available: <https://www.fifa.com/news/goalkeepers-are-not-above-the-law-72050>
- [58] I. Oh, S. Rho, S. Moon, S. Son, H. Lee, and J. Chung, "Creating pro-level AI for real-time fighting game with deep reinforcement learning," *arXiv preprint arXiv:1904.03821*, 2019.
- [59] O. Michael and O. Obst, "Betarun soccer simulation league team: Variety, complexity, and learning," *arXiv preprint arXiv:1703.04115*, 2017.



- 
- [60] A. Correia and S. Esteves, “An exploratory study of spectators’ motivation in football,” *International Journal of Sport Management and Marketing*, vol. 2, no. 5-6, pp. 572–590, 2007.
  - [61] T. K. Scanlan, P. J. Carpenter, M. Lobel, and J. P. Simons, “Sources of enjoyment for youth sport athletes,” *Pediatric exercise science*, vol. 5, no. 3, pp. 275–285, 1993.
  - [62] P. Sweetser, D. Johnson, J. Sweetser, and J. Wiles, “Creating engaging artificial characters for games,” in *Proceedings of International Conference on Entertainment Computing*, 2003, pp. 1–8. [Online]. Available: <https://dl.acm.org/citation.cfm?id=958734>
  - [63] A. Khan, B. Lazzerini, G. Calabrese, and L. Serafini, “Soccer event detection,” in *4th International Conference on Image Processing and Pattern Recognition (IPPR 2018)*. AIRCC Publishing Corporation, 2018, pp. 119–129.
  - [64] M. A. Russo, L. Kurnianggoro, and K. Jo, “Classification of sports videos with combination of deep learning models and transfer learning,” in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1–5.
  - [65] Opta Sports. F24: Feed specification document. [Online]. Available: <https://www.docdroid.net/ymMGPRQ/opta-playground-f24-documentation-pdf>
  - [66] ——. Homepage. [Online]. Available: <https://www.optasports.com/>
  - [67] M. Strong. How Opta produces and develops its unique data. [Online]. Available: <http://web.archive.org/web/20160117072238/https://www.sportstradingnetwork.com/opta-produces-develops-unique-data/>
  - [68] R. Marlin-Bennett, *Knowledge Power: Intellectual Property, Information, and Privacy*. Lynne Rienner Pub, 2004.
  - [69] Z. McCann. (2012) Player tracking transforming nba analytics. [Online]. Available: [https://www.espn.com/blog/playbook/tech/post/\\_id/492/492](https://www.espn.com/blog/playbook/tech/post/_id/492/492)
  - [70] K. Richly, M. Bothe, T. Rohloff, and C. Schwarz, “Recognizing compound events in spatio-temporal football data,” in *International Conference on Internet of Things and Big Data*, vol. 2. SCITEPRESS, 2016, pp. 27–35.
  - [71] J. Gudmundsson and M. Horton, “Spatio-temporal analysis of team sports,” *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–34, 2017.
  - [72] M. Horton, J. Gudmundsson, S. Chawla, and J. Estephan, “Automated classification of passing in football,” in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2015, pp. 319–330.
  - [73] M. Beetz, N. von Hoyningen-Huene, B. Kirchlechner, S. Gedikli, F. Siles, M. Durus, and M. Lames, “Aspogamo: Automated sports game analysis models,” *International Journal of Computer Science in Sport*, vol. 8, no. 1, pp. 1–21, 2009.
  - [74] Y. Yue, P. Lucey, P. Carr, A. Bialkowski, and I. Matthews, “Learning fine-grained spatial models for dynamic sports play prediction,” in *2014 IEEE international conference on data mining*. IEEE, 2014, pp. 670–679.
  - [75] X. Wei, P. Lucey, S. Vidas, S. Morgan, and S. Sridharan, “Forecasting events using an augmented hidden conditional random field,” in *Asian Conference on Computer Vision*. Springer, 2014, pp. 569–582.
-

- [76] K. Richly, F. Moritz, and C. Schwarz, “Utilizing artificial neural networks to detect compound events in spatio-temporal soccer data,” in *Proceedings of the 2017 SIGKDD Workshop MiLeTS*, 2017.
- [77] L. Morra, F. Manigrasso, G. Canto, C. Gianfrate, E. Guarino, and F. Lamberti, “Slicing and dicing soccer: automatic detection of complex events from spatio-temporal data,” in *Proc. 17th International Conference on Image Analysis and Recognition (ICIAR 2020)*. Springer, 2020.
- [78] B. Schuiling. Gameplay football. [Online]. Available: <https://github.com/BazkieBumpercar/GameplayFootball>
- [79] V. Khaustov and M. Mozgovoy, “Learning believable player movement patterns from human data in a soccer game,” in *2020 22nd International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2020, pp. 91–93.
- [80] J. Perš, M. Bon, and S. Kovacic, “Errors and mistakes in automated player tracking,” *Proceedings of sixth computer vision winter shop*, pp. 25–36, 2001.
- [81] J. Perš, M. Bon, S. Kovačič, M. Šibila, and B. Dežman, “Observation and analysis of large-scale human motion,” *Human Movement Science*, vol. 21, no. 2, pp. 295 – 311, 2002.
- [82] D. Linke, D. Link, and M. Lames, “Football-specific validity of tracab’s optical video tracking systems,” *PLOS ONE*, vol. 15, no. 3, pp. 1–17, 03 2020. [Online]. Available: <https://doi.org/10.1371/journal.pone.0230179>
- [83] M. Mozgovoy and I. Umarov, “Behavior capture with acting graph: a knowledgebase for a game ai system,” in *International Workshop on Databases in Networked Information Systems*. Springer, 2011, pp. 68–77.
- [84] M. Mozgovoy, M. Purgina, and I. Umarov, “Believable self-learning ai for world of tennis,” in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 1–7.
- [85] M. L. Puterman, “Markov decision processes,” *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.
- [86] L. Gyarmati, H. Kwak, and P. Rodriguez, “Searching for a unique style in soccer,” *arXiv preprint arXiv:1409.0308*, 2014.
- [87] P. Luhtanen, A. Belinskij, M. Häyrinen, and T. Vanttinen, “A comparative tournament analysis between the euro 1996 and 2000 in soccer,” *international Journal of performance Analysis in sport*, vol. 1, no. 1, pp. 74–82, 2001.
- [88] M. Prokopenko and P. Wang, “Disruptive innovations in robocup 2d soccer simulation league: from cyberoos’ 98 to gliders2016,” in *Robot World Cup*. Springer, 2016, pp. 529–541.