WITS FC Team Description Paper

Branden Ingram, Andries Bingani, Pravesh Ranchod, Benjamin Rosman
School of Computer Science and Applied Mathematics
University of the Witwatersrand
South Africa

May 2, 2021

Abstract. This document provides an outline of the Wits FC team, as a new entry into the RoboCup 3D Simulation League. We focus specifically on the research emphasis, and the new ideas we have implemented into the team. Furthermore, we describe how we have built off the base code made freely available by UT Austin Villa [5]. This paper describes our two main contributions that distinguish us from the original code base, namely a formation assignment system and our high-level strategy.

1 Introduction

As a new entry in the 3D League of RoboCup Soccer Simulation, the Wits FC team was founded in July 2019 and is a part of the RAIL Lab [3] from the School of Computer Science and Applied Mathematics at the University of the Witwatersrand (Wits) [4] in South Africa. Although our research group has long-standing experience in robotics and reinforcement learning, we are relatively new to RoboCup, with most of the members being undergraduates. In order to avoid starting from scratch, we bootstrapped from the base code release of UT Austin Villa [5]. This allowed us to focus our team development on aspects more closely related to our algorithmic and machine learning expertise.

With 2021 being the first year that Wits FC is planning to compete in the RoboCup 3D simulation league, we have had to build not only the software infrastructure required for the team, but the logistics as well. As we have established our code base, built on that of UT Austin Villa, we have had to establish appropriate computing environments, and document everything to ensure that new students can be easily on-boarded. Additionally, since there are currently no local leagues in South Africa, we are aiming to export this infrastructure to other universities to help them get started. This lack of a local ecosystem has meant that we have had to rely on internal tournaments between our team-members in both 2v2 and 7v7 settings to prototype new ideas. These tournaments also helped us identify weaknesses in the skills we have available as well as the high-level strategy.

2 Overview

As mentioned, we built on the UT Austin Villa Base Code Release as a template for our code base, to focus our efforts on components of the systems better aligned to our current expertise. This code release provides a fully functioning agent, with a number of basic skills which we utilised. Most importantly, it provides many data structures that keep track of the state of the game as well as each agent. Since this is our first time competing in RoboCup we have not yet improved on the basic skills at a low level, however, we plan to do this in the future. These fundamental components are discussed further in Section 3.

Our focus in this first iteration of competing was on adding our own high-level behaviours, which enabled the agent to make appropriate decisions according to different situations. Specifically, we implemented a dynamic formation assignment strategy that uses the Stable Marriage algorithm [2] to pair players to positions on the field. This is discussed in Section 4. In addition, we developed systems to evaluate the viability of kicking to different kinds of target locations. This was utilised in both our kick type selection routines as well as passing behaviours. Lastly, through our own internal smaller format tournaments we developed a system that minimised fouls when it came to tackling. These improvements are all detailed in Section 5.

3 Agent Skills

3.1 Walk

Walking is a requirement of both players with and without the ball, as well being the method for which the agents can traverse the field. This makes it a critical aspect of any 3D simulated soccer team. A walk that is fast, stable and balanced can easily outperform a team with a rudimentary walk engine and can therefore be considered a key to the success of any team. The UT Austin Villa code release has an omnidirectional walk engine based on a double inverted pendulum model and includes slow and stable walk engine parameters. For the time being, our team only utilises this standard walk and has only slightly modified the parameters, however, improving on this is an aspect identified for our future work.

3.2 Kick

Along with walking another skill that is important is that of kicking and being able to interact with the ball in a meaningful way. It is clear that powerful and accurate kicks can be utilised to great success when trying to score or breakthrough defensive structures. Kicking is also an essential ability if you are looking to incorporate any type of team coordinated passing. The UT Austin Villa base code supports a stable kick skill, a faster less reliable kick, and a dribbling skill that keeps the ball at the agent's feet while progressing to a target. Once again, we have only slightly modified the parameters, and have identified that optimising these kicks is an important goal for future development.

3.3 Get Up

A further essential skill is that of simply getting up off the ground after falling, and back into a standing position. For a number of reasons, collisions between robots are bound to happen and are typically rather frequent. This means that any reliable and quick skill for getting up can see an agent taking advantage of a situation where an opposing player is still off their feet. The UT Austin Villa base code supports a stable but slow get up skill. Rather than optimising this skill we instead paid special care to reduce the number of times collisions occur. In particular, we use collision avoidance systems for all the players off the ball. This is discussed further in Section 5.3 where we minimise the number of times falling occurs for agents looking to defend the ball.

3.4 Dive

In order to defend the opponents' shot, we have designed some dive actions for the goalie. As with the kicking skill, we hand-tuned the parameter values from the base code to improve the dive. We used the equations of motion, specifically ($r = r_0 + v_0 t + \frac{1}{2}at^2$) to predict a possible future position of the ball to know when to dive.

4 Formation Assignment

The first of the primary modifications we made to our team involves the question of how to dynamically adjust team formations or positioning while minimising the total distance travelled by all players. A naive solution we first utilised was to have players move to the locations nearest to them in the required formation.

However, this is not an optimal solution, as in many cases one player may be the closest to multiple positions in the desired formation, leaving other positions with no nearby team-members. This can be seen in Figure 1, which shows an example with 7 players where multiple agents (in blue) are all aiming for the same desired formation locations (in white). The red circles represent the selected targets our agents are aiming for, and the red lines indicate the allocations and paths they will travel.

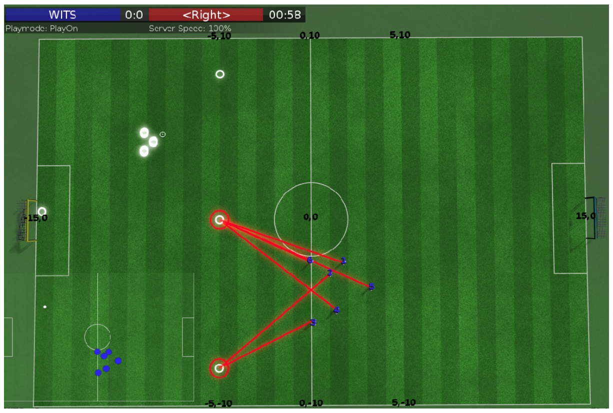

Figure 1: Naive Solution – Go To Closest Target

UT Austin Villa addresses this problem of dynamic role assignment with an algorithm called SCRAM [1]. Here they uniquely map a role to each player on the field. We identified the need for our system to be able to uniquely map players to locations on the field as part of a formation in an efficient manner. We based our approach on the Stable Marriage algorithm [2] , which matches elements of two groups based on preferences for each other. We designed our preferences to be based upon how far a desired location was from a respective player. Therefore, each player's most preferred position is the closest position to them in the formation. Figure 2 shows how this provides a better global mapping, such that each agent now targets a different location in a way that the total overall distance is minimised.

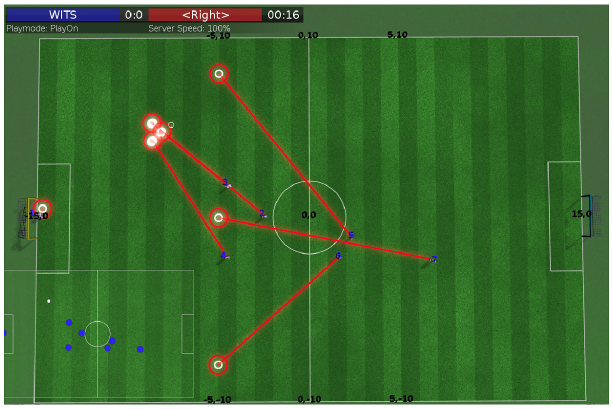

Figure 2: Optimised Solution – Stable Marriage

5 High-Level Strategy

The game of soccer is a very dynamic one with situations changing all the time and as a result, a team needs behaviour that is flexible enough to adapt to the changing environment. For this, we utilised a behaviour tree to account for all the variations in play that could arise. The crux of the tree was identifying whether we were in an offensive or defensive situation which was determined based upon player positions as well as the ball position. We then rely on our formation assignment algorithm to ensure there are always players in good positions, which allow us to perform other high-level aspects of our strategy namely; passing, kick selection and tackling. The exact behaviour was adapted and improved over the course of running multiple 2v2 and 7v7 tournaments within the team.

5.1 Passing

Passing the ball is a key part of soccer. The purpose is to keep possession by moving the ball between teammates while trying to progress the ball up-field into advantageous situations. We determine kick locations based upon many factors including ball position on the field as well as player positions. Once the location of the kick is determined we evaluate the likelihood of its success by checking if the target path is obstructed by another player. If it is obstructed we then iterate looking at other viable locations to kick towards. These locations may simply be open space where we may then follow up and maintain possession or other teammates. Figures 3 and 4 below are visualisations which demonstrate our passing behaviour.

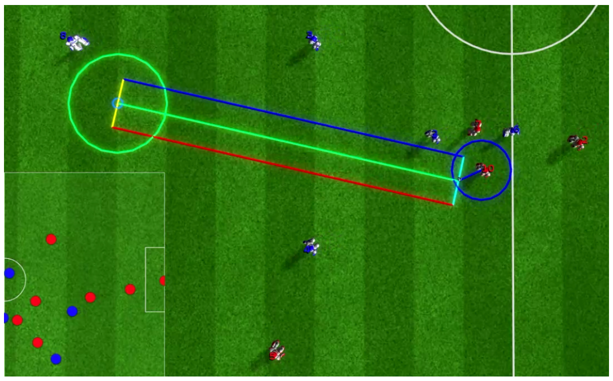

Figure 3: Unobstructed Target Situation

In Figure 3 above, there is a situation where the player in possession (depicted within dark blue circle) is trying to move the ball up-field to the target location (depicted by green circle). In order for our system to determine if the line to this target is obstructed we define a obstruction zone depicted by the multicoloured lines forming a rectangle between ball and target. For the situation depicted in Figure 3 there are no other players within that rectangle therefore the target is not obstructed and our player in possession can proceed to move the ball to the target.

In Figure 4 below we can see that this situation has changed since a blue player has moved within the rectangle. Our player therefore has to determine an alternate target to kick towards. This is done by iterating through the 5 closest teammates to determine if one of them is a viable passing target. One of the criteria is if they are in fact obstructed as well, we use the same obstruction zone approach to determine this. In Figure 4 we can see that after processing we have determined an adjusted target towards the southerly player (depicted by the light blue line and circle).

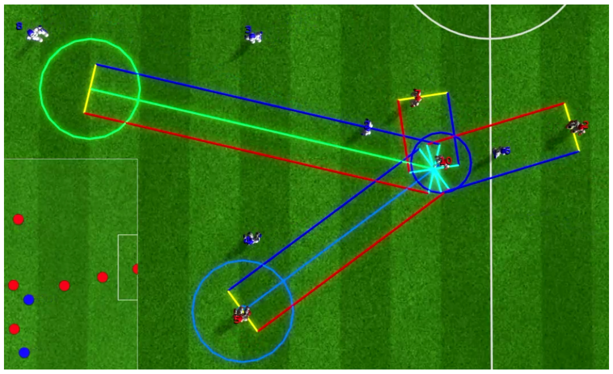

Figure 4: Obstructed Target with Resulting Pass

5.2 Kick Selection

We make use of custom data structures which keep track of players' distances relative to each other and the ball. These data structures are calculated and updated independently by each agent without the use of any communication between agents. This update occurs every server tick while the agent is standing. The distances themselves are computed using the computer vision system built into the original code-base [5].

This information is utilised to determine the viability of our three kicks:

- Long Stable Kick – which takes a while to execute.
- Fast Kick – which is unreliable but quick to execute.
- Dribble Kick – which is fast and very reliable but does not allow for passing or shooting.

Therefore based on the situation of the weigh each of the pros and cons of each kick in order to determine our course of action. Algorithm 1 describes the decision making process and agent will take when considering which kick to use.

Algorithm 1: Determine Appropriate Kick Type

```
Result: Kick Type to use
while While closest teammate to ball do
    Determine distance of closest opponent to ball;
    if Opponent Player Close then
        Kick Type = Dribble;
    else
        if Opponent Player Far then
            Kick Type = Stable;
        else
            Kick Type = Fast;
        end
    end
end
```

5.3 Tackling

Tackling is a difficult technique to incorporate in a simulated team where the agents are prone to falling over or being fouled. We have implemented a system that minimises the risk of our players from being teleported off the field by the server due to a reckless collision. Our approach ensures that our players firstly move in-between our goal and the ball before moving in to try to dispossess the opponent. This ensures that our players never tackle the player with the ball from behind, but rather they opt to move around the opponent first to come at them from the front.

Algorithm 2 below describes how we determine the location a player should move to in order to avoid collisions and still defend from attacks. In general we determine if our closest player to the ball is in-between the line of the goal and the ball. If this this closest teammate is, then they can proceed to go to the ball, otherwise they should move to be in-between the goal and ball first. This is utilised in conjunction with the built-in collision avoidance system provided by the code-base [5].

Algorithm 2: Tackle Behaviour

```
Result: target position to move to
while While closest teammate to ball do
    Determine midpoint between our goal and ball;
    if my position is in between ball and midpoint then
        target = ball;
    else
        target = midpoint;
    end
end
```

6 Future Work

Having such a rich simulation environment, which allows for the development of high-level strategies as well as low-level control, is an exciting prospect for research. Our team is largely experienced in the area of machine learning, and in particular decision theory. As such, we have focused on the high-level strategy for our RoboCup team. However, throughout the process of developing this system, we have seen more ways of defining research-related topics within the domain of RoboCup. This has resulted in interest in using machine learning techniques to help optimise our primitive skills such as walking and kicking. The goal of these projects will most urgently be to achieve faster and more stable walks as well as a longer kick. Specifically, since both the kick and walk skills are defined by a set of features we want to look to optimising these values. In the short term, this will be treated simply as an optimisation problem, but we plan to extend on that to learn the skills through reinforcement learning.

References

[1] Patrick MacAlpine, Eric Price, and Peter Stone. "SCRAM: Scalable collision-avoiding role assignment with minimal-makespan for formational positioning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.

[2] David G McVitie and Leslie B Wilson. "The stable marriage problem". In: *Communications of the ACM* 14.7 (1971), pp. 486–490.

[3] *Robotics, Autonomous Intelligence, and Learning (RAIL) Lab.* `https://www.raillab.org/`. (accessed: 16.04.2021).

[4] *University of the Witwatersrand, Johannesburg.* `https://www.wits.ac.za/`. (accessed: 16.04.2021).

[5] *UT Austin Villa Robocup 3D Simulation Team.* `https://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/`. (accessed: 16.04.2021).