Adapting a KD-Tree Based AI to Google Research Football

Yuto Takiguchi s1270138 Supervised by Prof. Maxim Mozgovoy

Abstract

Soccer is one of the better choices for creating multi-agent AI because soccer has many datasets and there is a simulator that can test AI. Google Research Football (GRF) is an open-source soccer game for reinforcement learning and it offers a powerful soccer game engine. Therefore, this environment is useful for testing new soccer AI. Also, there is AI which is the kd-tree based AI. The AI uses an actual soccer game dataset and performs with low computational cost because there are no learning steps. Therefore, this AI excels as a first step of soccer AI.

In this paper, the goal is to integrate GRF with the kd-tree based AI. Since GRF provides a standardized soccer AI research environment, we want to focus on it. We need to modify the kd-tree based AI appropriately to use in the GRF environment because they are incompatible at the gameplay level and the level of player actions. The kd-tree based AI for GRF was able to run as a first step of our soccer AI.

1 Introduction

Soccer is one of the most popular sports and team competition sports. Although simulating soccer is difficult because of a lot of states and actions of players, it is possible to create multi-agent AI because soccer has numerous actual datasets. However, a powerful soccer game engine is necessary to simulate soccer quickly and correctly.

In this paper, the GRF environment is adopted to develop a soccer AI. The GRF environment is an open-source 3D soccer game environment that aims to control players by reinforcement learning. Also, the environment can learn specific situations such as corner kicks, off-sides, fouls, penalty kicks [1].

There is a kd-tree based AI running on the environment, SimpleSoccer [2]. That AI is controlled by learning from actual soccer game data. However, SimpleSoccer is a 2D soccer game and cannot simulate specific situations.

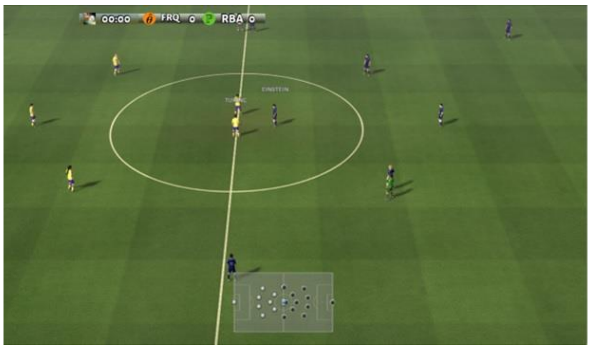
The purpose of this paper is to integrate the kd-tree based AI into GRF as a next step. While the basic movements of soccer should be integrated without many changes, the pass system and actions of specific situations should be integrated by appropriate changes. The reason is that the same basic movements are shared between SimpleSoccer and GRF environments, however the SimpleSoccer environment cannot support certain actions that require 3D.

2 Method

2.1 Google Research Football

The GRF is a 3D soccer simulator as shown in figure 1 and it makes easy-to-simulate soccer AI which is the goal of GRF. It can simulate specific situations such as a corner kick, a shot against an empty goal, a game between a specific number of people, and so on. Also, new situations can be added to the GRF environment. A new AI can be added to use the Player class that is offered by the GRF environment.

Figure 1: Google Research Football Screen.



The important information of the GRF environment is about the field, ball, players, and action. The soccer field size is [−1, 0.42] at the bottom left corner and [1, −0.42] at the top right corner. Table 1 shows the ball and player information. The player has a Boolean flag to show whether the player is on the field or not. The player can disappear if he receives a red card or due to an initial setting. There are 19 default actions such as the following:

- Idle.
- Move in 8 directions.
- 3 types of passes.
- Shot.
- Sprint/Stop sprint.
- Sliding.
- Dribbling/Stop dribbling.
- Reset current movement direction.

Although there is an extension action set, this set is not used because the AI becomes complex.

Table 1: Ball and Player Information

Ball	Coordinate (x, y, z).
	Direction (x, y, z).
	Rotation (x, y, z).
	Who has the ball.
	Which team has the ball.
Left/Right Team Players	Coordinate (x, y).
	Direction (x, y).
	Fatigue factor (0..1)
	Yellow card (0 or 1)
	Active (true or false)
	Roles (0..9)

2.2 KD-Tree Based AI

We adopted the kd-tree algorithm [3] to control AI agents running on the GRF environment. Since the algorithm can learn from an actual soccer dataset, it can make a soccer AI more human-like and decrease computational costs.

In this paper, x and y coordinates are used for the kd-tree algorithm. Since a z coordinate is not included in the datasets, it cannot be used.

The algorithm controls ten players simultaneously excluding a goalkeeper. Since the kd-tree based AI does not generate goalkeeper actions, a goalkeeper just passes to a teammate who is in a direction that he is facing to avoid game stopping. The algorithm [4] runs as follows:

1. Get all players and ball coordinates.
2. Search the players within a certain range of the player controlled by AI (see Figure 2).
3. Find situations from the dataset where the player is located within a certain range based on the coordinates of the player found by step 2 (see Figure 3).
4. If there are two or more situations, determine the best situation by the direction of players. If there are no situations, go to step 2 and decrease the range of step 2.
5. Get an action from the situation.

In step 3, the range of the controlled player is smaller than the found players because the controlled player's position is the most important. If the range of the controlled player is large, the kd-tree may return a situation where the controlled player is far away.

In step 4, if the range of step 2 decreases to the minimum range, the kd-tree returns no result. This means that the player cannot be controlled by the kd-tree. The action is set to the previous movement action or random passing because there is no way to determine an optimal action.

Figure 2: Find Players.

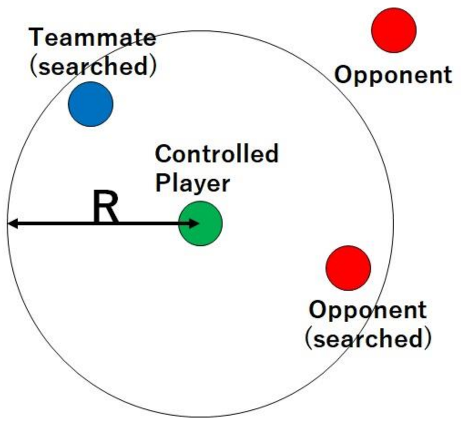
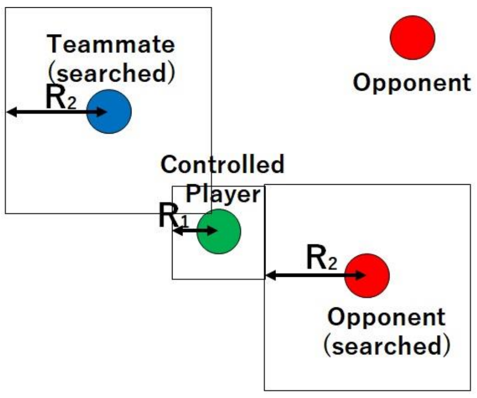


Figure 3: Find Similar Situations.



2.3 Boost.Python

While the GRF environment uses Python for player control, the kd-tree based AI for SimpleSoccer uses C++. Therefore, it is necessary to send data from Python to C++. To solve this problem, we use the Boost.Python library. This library enables C++ and Python linking.

2.4 Pass System

The problem with integrating the kd-tree based AI for SimpleSoccer into the GRF environment are the differences in implementation of each system. One of the differences is the pass system.

The SimpleSoccer environment uses the id of the pass target. Therefore, this environment can select the pass target accurately. However, the GRF environment determines the pass target by the direction that the passer is facing. This means that if there are two or more teammates in the line-of-sight of the passer, the pass target is unclear. Also, the passer needs to first look in the direction of the pass target before passing, even if the environment sends the order to pass.

To solve these problems, first, the direction between the passer and the pass target was calculated. If the passer hasn't looked in the calculated direction yet, the passer turns and moves in that direction. Next, the kind of pass was determined. The GRF environment has 3 types of passes: short pass, long pass, and high pass. Since there are no specific guidelines on categorizing a pass based on its distance in GRF, we keep thresholds tunable. Currently, the passes are categorized as follows (see Listing 1):

- Short pass: less than 15 m
- Long pass: 15 - 30 m
- High pass: more than 30 m

Listing 1: How to Select Pass Action.

```
SelectPassAction (player, target)
1  direction ← GetDirection(player, target)
2  distance ← GetDistance(player, target)
3  if direction = player.direction then
4     if distance < shortPassBorder then
5         return shortPass
6     else if distance < longPassBorder then
7         return longPass
8     else
9         return highPass
10    end if
11 else
12    return moveInDirection
13 end if
```

2.5 Off-Side

Another difference between the environments is the rules implemented. In particular, the off-side rule needs to be modified. While the GRF environment supports the off-side rule, the SimpleSoccer environment does not. Therefore, the kd-tree based AI for SimpleSoccer sometimes believes that passing when off-side is optimal.

To solve this problem, a function was created to determine whether the pass would be off-side or not before passing. The first step is storing distances of the x-axis between opponents, excluding the goalkeeper, and the right sideline of the soccer field. Next, the opponent with the minimum distance is found. The distances of x-coordinates between the opponent who is found in the previous step and the pass target is calculated. Finally, if that distance is less than zero, then the pass can be off-side (see Listing 2). Therefore, the environment can send the player an order not to pass.

2.6 Possession State

One of the system differences is the way to store the possession state. The possession state is which team has the ball. The SimpleSoccer environment maintains the possession state during passing. Therefore, if the ball leaves the player by passing, the possession state is maintained if the pass does not fail. However, the GRF environment does not maintain the possession state during passing. This means that the moment the ball leaves the player, the possession state will be that neither team has the ball (see Figure 4)

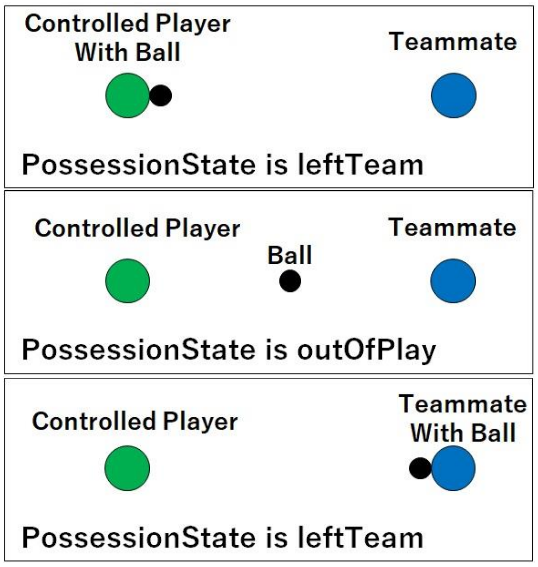
To solve this problem, a new possession state was created based on the old one. Once a team has the ball, the new possession state maintains its value if another team does not take the ball (see Listing 3).

Listing 2: Off-Side Function.

```cpp
bool isOffside (int targetID, GRF::Players& players)
{
  std::vector<double> distances(10);
  // The reason for p=1 is removing GK
  for (int p = 1; p < players.rightPlayers.size(); ++p)
  {
    distances[p - 1] =
    abs(GRF::Filed::RIGHT_OF_METER_FIELD - players.rightPlayers[p].coord.x);
  }
  auto it = std::min_element(std::begin(distances), std::end(distances));
  auto closestPlayerIndex = std::distance(std::begin(distances), it);
  auto distClosestPlayer =
       players.rightPlayers[closestPlayerIndex].coord.x -
       players.leftPlayers[targetID].coord.x;

  return distClosestPlayer < 0;
}
```

Figure 4: Possession State During Passing for GRF.


Controlled Player With Ball
Teammate
PossessionState is leftTeam
Controlled Player
Ball
Teammate
PossessionState is outOfPlay
Controlled Player
Teammate With Ball
PossessionState is leftTeam

Listing 3: Possession State Converter.

Possession State Converter

```
1. newPossessionState ← outOfPlay
2. If possessionState = Defending then
3.     newPossessionState ← rightTteam
4. else if possessionState = Attacking then
5.     newPossessionState ← leftTeam
6. end if
```

3 Results and Discussion

The ten players excluding the goalkeeper were able to be controlled simultaneously by the kd-tree based AI on the GRF environment. Also, the player was able to pass to the target via three types of passes. The off-side system reduced off-side pass. The possession state converter made the possession state behave the same as on SimpleSoccer. Therefore, the AI ran basically as expected.

However, some points to be improved were found. Since the pass system could not pass to the target accurately, the player sometimes passed to the wrong player. To improve this, a deep understanding of the GRF engine library is required and in some cases the library may need to be modified.

One of the other improvements is the behavior when an instruction occurs via the off-side system. For now, the player acts based on the previous action when he cannot pass due to off-side. However, if there are other options from the kd-tree, the player should ideally follow those.

The possession state does not change as long as the opponent team does not get the ball. Therefore, it may be better if the possession state is determined depending on the amount of time the ball is not touched when neither team has the ball.

4 Conclusion

In this paper, we integrated the kd-tree based AI for SimpleSoccer into the GRF environment. Although some possible improvements were identified, to some extent the AI worked as expected. This integration allows for deeper AI development in the GRF environment because many things were not able to be implemented in the kd-tree based AI for SimpleSoccer. For example, since the GRF environment supports many actions, the AI can do various actions. Also, the AI learns specific situations that are not implemented on SimpleSoccer such as corner kicks, penalty kicks.

We hope the kd-tree based AI will be helpful for the development of soccer AI.

References

[1] K. Karol, et al, “Google research football: A novel reinforcement learning environment,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, Apr. 2020, vol. 34, no. 04, pp. 4501-4510.

[2] M. Buckland, “Sports Simulation - Simple Soccer,” in *Programming Game AI By Example*, Massachusetts: Jones & Bartlett Learning, 2005, pp. 133-192.

[3] B. J. Louis, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509-517, Sep. 1975.

[4] G. M. Bogdan and M. Mozgovoy, “Towards Case-based Reasoning with kd Trees for a Computer Game of Soccer,” in *2019 IEEE IUCC and DSCI and SmartCNS*, Oct. 2019, pp. 570-572.