MC Soccer: Simulating a Soccer Game using Monte-Carlo Methods

NERS44 Final Project

Ravi Shastri

December 16, 2025

Contents

1 Introduction

A game of soccer can be extremely unpredictable, and on any given day, the score of the game will be different even if the players are the same. There is already a lot of randomness in soccer. Where will the ball deflect off an opponent? What happens if I shoot from 50 meters out? What if the goalie makes a howler? What happens if I don't track back to defend an opposition attack? What is my best action right now? There are so many things that are possible in soccer - can the game even be simulated?

The general idea for this project is to use Monte-Carlo (MC) methods to simulate a game of soccer while answering the question, "What is my best option right now that could lead to a goal?". This project will incorporate situation-dependent sampling with action dependent probability density functions (PDFs). Mainly, we want to build a model for expected goals (XG) [1]. The XG model assigns a value between 0 and 1 for the probability that a shot becomes a goal [2]. Given that one knows everything about the state of the shot, including location of the defenders, distance to goal, angle between the goal and the shot, weather conditions, etc., an XG is assigned to that situation. Many sophisticated models exist that are built on an enormous amount of data [3] - I do not have that data, nor can I easily obtain that data. However, using MC sampling techniques, I can generate synthetic data based on weighted PDFs to calculate situation dependent XG. There are a few assumptions needed to build this simplified model.

- **Four Players per Team:** Each team has four outfield players that all interact with the ball. Players are currently attributes of the driver class.
- **Reflecting Boundary Conditions:** Players must remain on the pitch at all times, and the ball can never leave the pitch. There are no throw-ins, goal kicks, or corners. There are also no set plays, fouls, or disciplinary actions.
- **2D Straight-Line Simulation:** The ball moves in a straight line on the group from point A to B.
- **Similar Players:** No players have better or worse attributes than other players. This can be modified by weighted-sampling the decay constant λ in a PDF. Players do not Tire Out.
- **The PDF for Each Action is Known:** These PDFs will be explored in Sec. 2.
- **Goalie Mechanics:** The goalie saves the ball with some probability dependent on the distance to the goal.

This paper uses MC methods to build, run, and analyze a game of soccer. The analysis will compile data from a full season's worth of minutes and $\approx 20,000$ shots to create an XG map similar to Fig. 3 in [3]. Python is used as the coding language, and a full game simulation is linked in Sec. 4.

2 Background

There are three **Action** states that a player can be in: **In-Possession, Attacking, or Defending.**. When the player is In-Possession, they can perform one of four actions: *Shoot, Pass, Carry, or Turnover*. When the player is not in possession, they perform unique off-the-ball actions. Secs. 2.1 through 2.3 will describe the mechanics of the player that is In-Possession. Sec. 2.4 describes the mechanics of the rest of the players. Sec. 2.5 describes the XG calculation.

2.1 Sample Shooting Distribution

The probability of scoring a goal should decreases exponentially as the distance to the goal increases. The PDF is normalized from 0 to the maximum shooting distance d_{max}, where $d_{max} = \sqrt{L^2 + (W/2)^2}$, where L is the length of the pitch and W is the width of the pitch. Using inversion sampling, the normalized shooting PDF is

$$f(x) = \xi = C \int_0^{d_{max}} e^{-\lambda x} dx \Rightarrow x = \frac{-1}{\lambda} \ln\left(1 - \xi[1 - e^{-\lambda \cdot d_{max}}]\right), \tag{1}$$

where λ is a parameter chosen by the user, and ξ is a random number from a pseudorandom number generator. Fig. 1 shows the dependence of the PDF on λ, where the probability of scoring a goal decreases more sharply as λ increases.

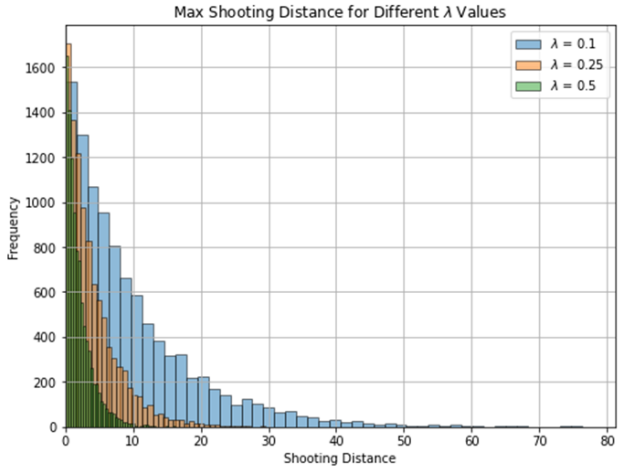


Figure 1: Exponential PDF histogram for different λ parameters.

The exponential distribution is sampled at the beginning of the event with $\lambda = 0.10$. If the distance to goal sample x is greater than the distance to the goal, then the player performs the shooting action. Another exponential distribution is sampled with $\lambda = 0.25$. The initially small λ encourages the player to shoot; however, the higher λ in the second sample ensures that the goal is scored only if the distance to goal is small. If the player does not score, the ball is given to the nearest defender, and the defending team becomes the attacking team. This double sampling method ensures that better chances have a higher chance of scoring, but it doesn't guarantee that a goal will be scored. There are many situations where the distance to the goal is small, but the goalie is there to make a save, or the player just misses. Also, the double sampling accounts for the small chance that a player shoots from far away and actually scores.

2.2 Sample Passing

The passing PDF is also a function of the distance to the nearest teammate. In real soccer, a player is more likely to pass to a teammate that is close to them, but not very close to them. There is a sweet-spot for the passing distance that is modeled in a Gamma distribution, $f(x) = xe^{-\lambda x}$. Initially, a Planckian distribution was chosen because there is a optimal distance away from the player with the ball, and it mimics the desired shape. However, this sampling scheme using a slightly expensive rejection method, and does not capture the desired passing mechanic very well. Unfortunately, a true Gamma Distribution, although popular in MC methods, is expensive to sample. However, a simplification of this distribution process is possible. We first must normalize the distribution such that the normalization constant $C = \lambda^2$. Then we calculate the sample

$$F(x) = \int_0^\infty Cte^{-\lambda t}dt \Rightarrow x = \frac{-1}{\lambda} \cdot \ln(\xi_1 \xi_2), \tag{2}$$

as the product of two pseudo-random numbers. Fig. 2 shows the most passing PDF as a function of distance to the nearest teammate.

If sampling the Gamma-like distribution with $\lambda = 0.15$ returns the passing action, then another uniform distribution is sampled to determine which teammate receives the ball. The distances to each teammate is inverted and normalized to one. The player with the smallest distance occupies the largest portion of the sample space. For example, if one teammate is 5m away from the ball, and the other two teammates are 10m away from the ball, the player 5m away has a 50% chance of receiving the ball, and the other two players each have a 25% chance of receiving the ball. In practice, a higher weight is added to players who are closer to the opposition's goal. A sample is drawn uniformly between 0 and 1 and mapped to the CDF.

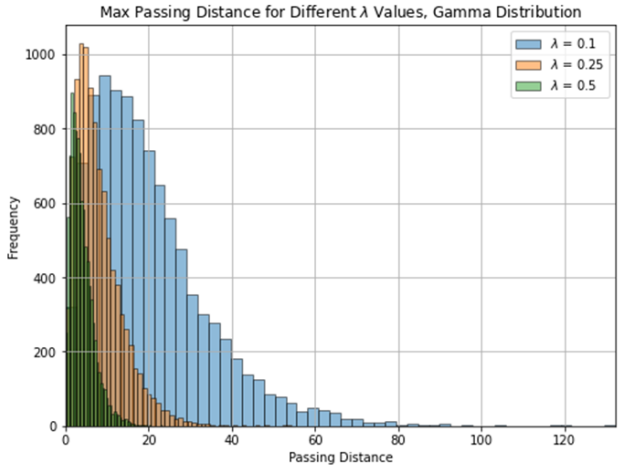


Figure 2: Gamma Distribution PDF histogram for different λ parameters.

2.3 Turnover and Carry Actions

There are many ways a ball is turnover in soccer. The two major turnover mechanics are looped into a single code block, where the ball is somehow stripped from the player in possession. The sample is drawn from the defender nearest to the ball from the exponential distribution shown in Fig 1, with $\lambda = 0.25$. If the sample is larger than the distance to the ball, the ball is turned over via an interception or a tackle. The defender then becomes the player with the ball.

If the shooting, passing, or turnover actions fail, then the player carries the ball forward. A time is sampled from an exponential distribution with λ=0.5. A speed is sampled from a Gaussian distribution centered around 2m/s with an uncertainty of one-fifth of the speed. When the player has the ball, their maximum speed is limited because they are carrying the ball.

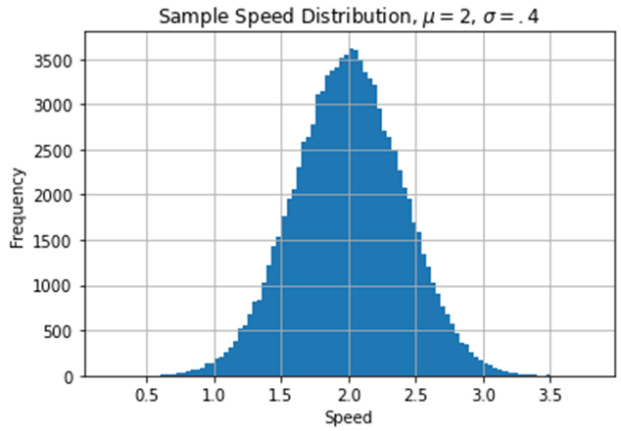


Figure 3: Gaussian Distribution for sampling the players' speed.

Then, a point on the goal-line is selected uniformly, and the player moves towards the goal a distance $d = vt$.

2.4 Attacker and Defender Mechanics

Now that we have described what happens to the player on the ball, we can discuss what happens to the players off the ball. These mechanics are slightly simpler to describe. This can be further complicated with better decision making game theory, but a simplified approach works well for this model's purposes.

If the players are attacking, then 2 of the 3 remaining players move towards the goal. Their speed is sampled from Fig. 3, but the uncertainty in their speed is reduced to $\frac{\mu}{5}$, allowing them to move at a potentially higher speed than the player with the ball. Their direction of travel is determined by a randomly selected point on the goal line. The player closest to their own goal is tasked with staying on defense at the top of their goalie box. There is no offside trap, so players can get as close as they can to the goal line to try and score.

If the players are defending, 2 of them will close down the player with the ball, hoping to force a turnover. The other two players head back to their goal to defend a shot. All players cannot chase the ball because the attacking team can just pass around the defenders and score many goals. More sophisticated game-theory models are possible to implement here, but this model is sufficient for the simple model.

2.5 XG Model

The XG model is a useful tool for determining which team generated the better opportunities in a game. If the game were to be replayed many times, the XG model should predict the average result of the game. It is a measure of how a team performed on a given day independent of the ability to finish off chances. The important parameters from the Mead [3] model include distance and angle to the center of the goal, as shown in the heat map in Fig. 4.

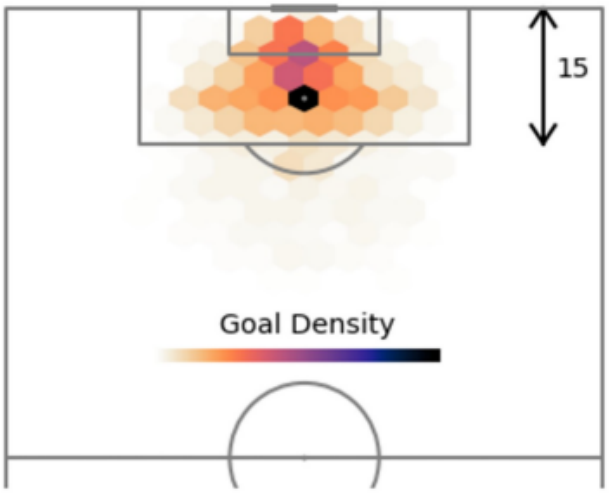


Figure 4: XG heat map from the Mead [3] XG model

Higher quality chances correspond to darker areas on the map, and lower quality chances are ligher areas. The big black mark in the center is where penalty kicks are taken, so it is much hotter than the rest of the map. This work will build a similar model from synthetic data. After many game simulations are run, the shooting data is stored in a *.csv* file, and a simple calculation is performed to find distance dependent XG. We expect to see similar results to Fig. 4, but a much higher density closer to the goal, and a much lower density around the penalty spot.

3 Methods

The GitHub repository for this simulation can be accessed here. *init.py* and *params.py* contain the library, and Algorithm 1 describes the full simulation of the game contained in **Soccer.py**. The *verbose* flag allows the user to plot the positions of each player and show their movement with the ball. If the flag is turned off, a 90 minute game of soccer can take as little as 5 seconds to run.

First, the players are created and the initial position is set. Fig. 5 shows a configuration where the blue team starts with the ball. Then a timestep *dt* is sampled from the Exponential 2.1 distribution in seconds. The player, *soccer.b1* for example, is described by a vector of size 3, where the first two indexes indicate x and

Algorithm 1 MC Soccer Model

Create Geometry and Set Starting Positions
while $t < T_{\text{total}}$ **do**
    Sample time increment Δt.
    Identify the player and team currently in possession of the ball.
    Compute:

- Distance from the ball to the goal
- Distance from the ball to each teammate
- Distance from the ball to each opponent

    Sample **Shooting** action from a "flatter" exponential distribution.
    if action = SHOOT **then**
        Sample outcome using a "sharper" exponential distribution.
        if shot succeeds **then**
            Reset players; conceding team restarts with the ball.
        else
            Ball moves to nearest defender.
            Defending team begins attacking.
        end if
    end if
    Sample **Passing** action from a Gamma-like distribution.
    if action = PASS **then**
        Ball may move to any attacker.
        Probability of receiving the pass $\propto 1/(\text{distance to teammate})$.
    end if
    Sample **Turnover** (STRIP) from a "sharp" exponential distribution.
    if action = STRIP **then**
        Pass or carry fails; defending team begins attacking.
    end if
    if no action occurs **then**
        Carry the ball toward the goal.
    end if
    if $t > T_{\text{total}}/2$ **then**
        Reset ball and players for half-time.
    end if
end while

y position respectively, and the third index indicates whether or not the player is in possession. The player vectors are searched until the player with the ball is determined. The distance from this player to the goal and each other player is calculated, and the shooting action is sampled. If the player shoots, then the shot information is tabulated and saved to a *.csv* file. If the player does not shoot, the Gamma distribution 2.2 is sampled to see if the player passes the ball. If the player holds the ball, the Exponential distribution is used to determine a turnover event, and if nothing happens, the Gaussian 2.3 distribution is sampled for the players speed, and the player moves towards the goal. The rest of the players follow the actions described in Sec. 2.4. This process repeats until a player scores, and the ball resets to the center with the conceding team now in possession, halftime, where the opposite team now in possession, or the game ends. Fig 8 in Appendix 5 show the four actions in progress.

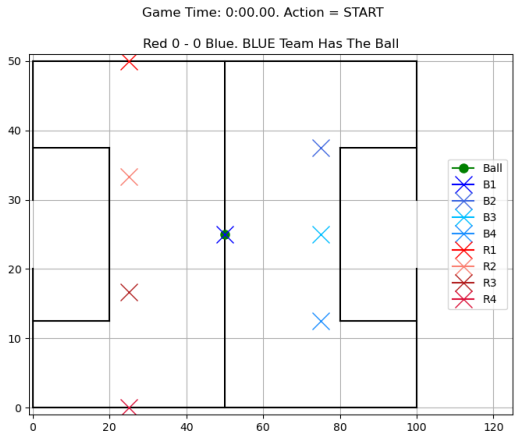


Figure 5: Starting position of the game, Blue team has the ball.

4 Results

4.1 Full Game Visualization

The **Full Game** link can be found in the GitHub repository. The user is encouraged to change the seed at the top of the *Soccer.py* class and run simulations with *verbose=True*. The random seed changes the result of the game due to MC randomness.

After running 20 game simulations, we notice a worrying trend. Blue team seems to outscore the red team at rate higher than what is expected from Monte-Carlo noise. Tab. 1 in Appendix 5 and Fig. 6 demonstrate this point. The blue and red teams should win about one-quarter of the time each, and the most probably result should be a tie. However, the blue team wins more than half of the time. Looking at the XG data, there is very little difference in the number or location of shots taken by either team. This indicates an error in the second sampling exponential function between the Blue and Red teams.

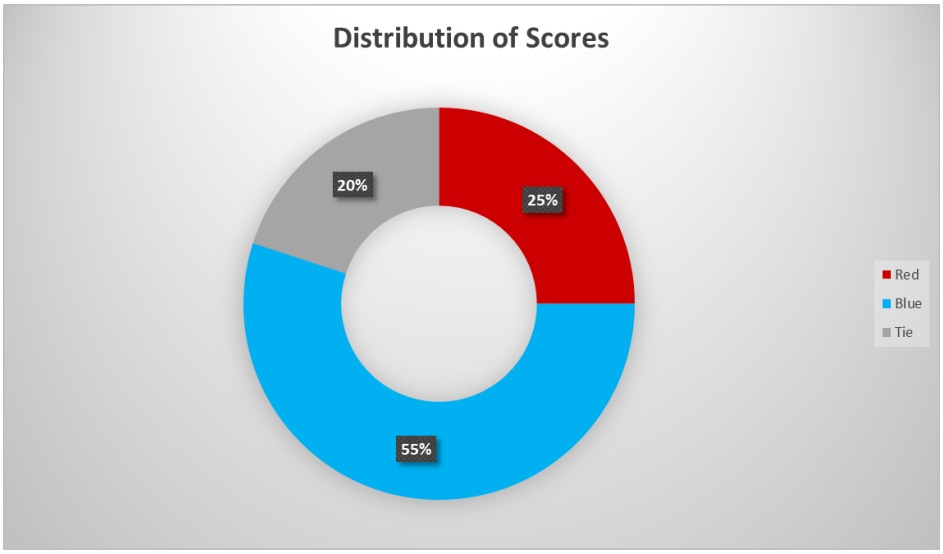
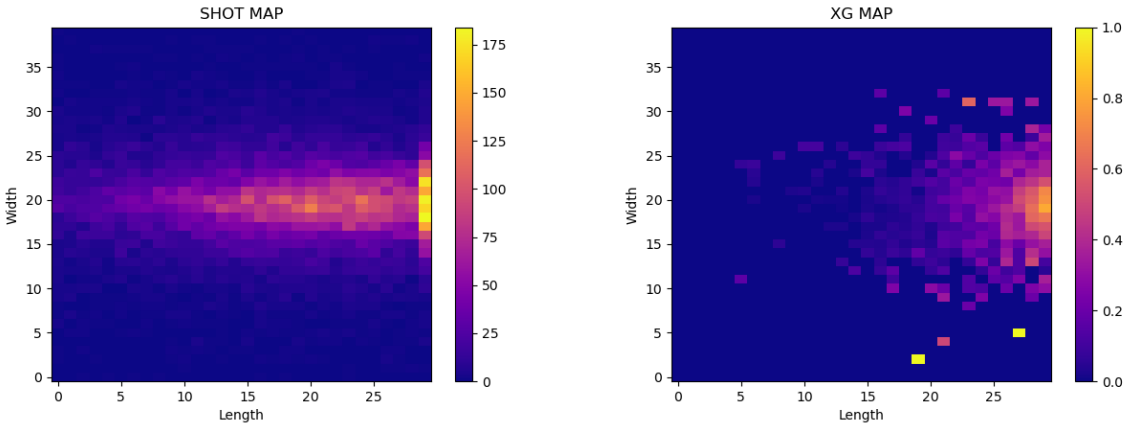


Figure 6: Winner Pie Chart from 20 MC Soccer Runs

4.2 XG Model

XG is calculated from data from $\approx 20,000$ shots. The total attempts is stored on a grid with a grid size of 1m×1m. The number of goals from that position is divided by the number of shots from that position to determine the XG from that spot. Fig. 7 shows the resutls from ≈50000 minutes worth of simulated game time. Each team English Premier League team plays about 3200 minutes per season, so this is about one season's worth of XG data. More data can easily be generated to further refine the XG estimation.



(a) Total attempts heatmap.

(b) MC Soccer XG heatmap.

Figure 7: Total attempts and XG heatmaps from ≈36,000 minutes of simulated games.

We notice that the majority of the shots that turn into goals are near the center of the goal. Our model solely depends on distance to the goal, so this is a good trend to see. However, we do see a few outliers near the edge of the box. Due to a slightly low number of samples, some of the traditionally low XG shots show up as high XG because of the few shots that were taken at that position, they just happened to go in. This is clearly visible when comparing the heatmaps of Figs. 7(a) and 7(b).

5 Conclusion

This work demonstrates that MC methods provide a flexible and intuitive simulation capabilities for simulating the inherently random nature of a game of soccer. By modeling player actions and ball movement through situation-dependent PDFs, this simulation captures many basic features of real soccer, including unpredictable scoring, turnovers, and spatial shot distributions. MC methods were used to sample many PDFs quickly and efficiently. Despite many assumptions, including identical players, straight-line motion, and the absence of set pieces, the model is capable of generating full 90-minute matches efficiently and producing large synthetic datasets suitable for the development of an XG model.

The resulting XG heatmaps exhibit physically reasonable trends, with the highest scoring probabilities concentrated near the center of the goal and rapidly decreasing with distance, consistent with established XG models in the literature. This is most clearly seen when comparing Figs. 7(b) and 4. We see the expected result of higher quality chances being concentrated near the goal, but our model is even further concentrated. This is because, due to the lack of a physical goalie, the players take many more shots much closer to the goal than they do in real soccer. Localized outliers in low-sample regions highlight Monte Carlo estimates are only as reliable as the number of samples available, reinforcing the need for more data generation.

There was also an observed bias in match outcomes, where the blue team consistently outperforms the red despite symmetric initial conditions. This highlights the need for some future work on the part of the developer to find this bug. Another potential avenue of interest would be to use variance reduction techniques, but the application of these methods still seem uncertain. ML can be used, with more data, to develop a predictive XG model for this particular model configuration, or with perturbed λ weighting factors. Overall,

this project shows that even a simplified MC soccer model can reproduce meaningful emergent behavior and generate interpretable performance metrics such as XG. The model can only capture so many features of the game, but the random nature of soccer greatly compliments MC methods. The hope is that this model is somewhat useful to visualize how distance to the goal correlates to the development of an XG model.

Appendix

Table 1: MC Soccer Match Results

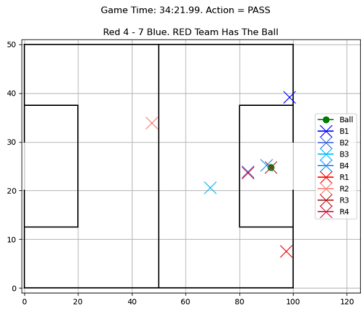
Seed	Red Score	Blue Score	Winner
1	7	4	Red
4	7	6	Red
6	7	9	Blue
7	6	7	Blue
8	6	13	Blue
9	7	9	Blue
10	7	8	Blue
11	8	9	Blue
15	6	4	Red
19	3	6	Blue
20	10	6	Red
23	6	7	Blue
26	4	8	Blue
28	3	12	Blue
38	6	6	Tie
42	10	8	Red
66	6	6	Tie
77	4	10	Blue
86	6	6	Tie
100	6	6	Tie
Number of Goals	**125**	**150**	

References

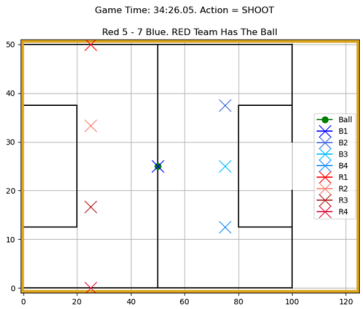
[1] A. Rathke, "An examination of expected goals and shot efficiency in soccer," *Journal of Human Sport and Exercise*, vol. 12, pp. 514–529, 2017.

[2] I. Umami, D. H. Gutama, and H. R. Hatta, "Implementing the expected goal(xg) model to predict scores in soccer," *International Journal of Informatics and Information Systems*, vol. 4, pp. 38–54, 2021.

[3] J. Mead, A. O'hare, and P. McMenemy, "Expected goals in football: Improving model performance and demonstrating value," *PLoS ONE*, vol. 18, 2023.

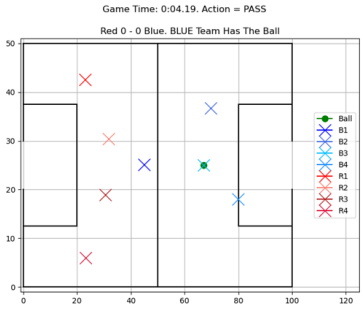


Score Initial Position

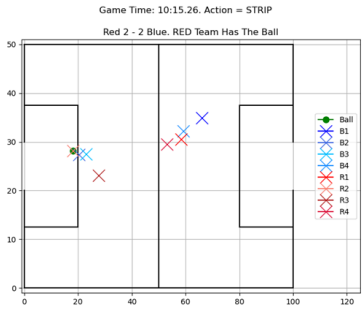


Score Final Position

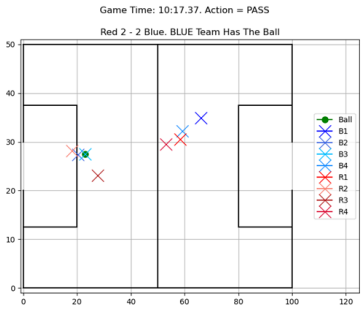
Pass Initial Position



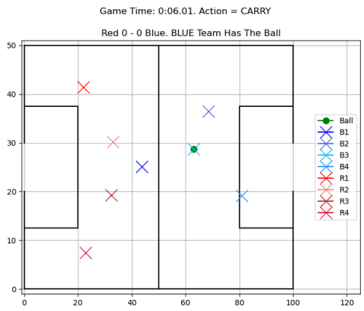
Pass Final Position



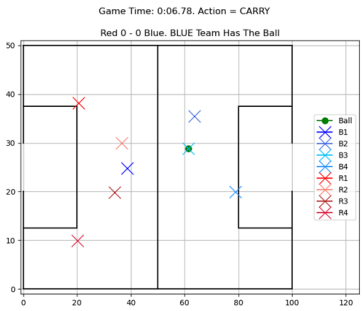
Strip Initial Position



Strip Final Position



Carry Initial Position



Final Initial Position

Figure 8: Each action visualized on the field for a given time-step