THE UNIVERSITY OF JORDAN

School of Engineering Engineering

Department of Mechatronics

**Bachelor of Science in Mechatronics Engineering**

**Senior Design Graduation Project Report**

# The Design of Virtual Football Training System Using Data Science


Report by

**Suhaib Alhlabiah**
**Laith Hudaib**
**Yazan Kraek**
**Mohammed Younes**

Supervisor

Dr. Hussam Khasawneh


Date

09/06/2021

# DECLARATION STATEMENT

We, the undersigned students, confirm that the work submitted in this project report is entirely our own and has not been copied from any other source. Any material that has been used from other sources has been properly cited and acknowledged in the report.

We are fully aware that any copying or improper citation of references/sources used in this report will be considered plagiarism, which is a clear violation of the Code of Ethics of University of Jordan.

In addition, we have read and understood the legal consequences of committing any violation of the University's Code of Ethics.

| | Student Name | Student ID | Signature | Date |
|---|---|---|---|---|
| 1 | **Suhaib Alhlabiah** | **0162878** | | 9/6/2021 |
| 2 | **Laith Hudaib** | **0160946** | | 9/6/2021 |
| 3 | **Yazan Kraek** | **0163947** | | 9/6/2021 |
| 4 | **Mohammed Younes** | **0162765** | | 9/6/2021 |

# ABSTRACT

In this report we go through the design of a virtual training system for soccer players. We will discuss the artificial intelligence we used, then we will go through the model that we chose, and we discuss the design options we choose and why we chose it. Then we will give the result we got from the model.

# ACKNOWLEDGEMENTS

We would like to thank and convey our gratitude to everyone who assisted us in completing this project and gave us with the chance and means to do so. A particular thanks to our project supervisor [Dr. Hussam Khasawneh], whose stimulating recommendations and encouragement aided us in continuing till the project was successfully completed, and we appreciate his time and continued collaboration.

We'd also like to express our gratitude to our families for their unwavering and unwavering support, as well as their warm and overwhelming love and care.

Thank you to all of our friends and colleagues who have always been a source of encouragement and drive.

Also, we thank metrica sport for their free and clean data.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# GLOSSARY

| ABBREVIATION | DESCRIPTION |
|---|---|
| RNN | Recurrent neural network |
| LSTM | Long short-term memory |
| DNN | Deep neural network |

# Chapter 1 INTRODUCTION

## *1.1 Background*

### Neural networks

neural networks are made to act like human brain. With small elements called neurons. Neurons are connected by lines called links. Then the neurons in the middle form hidden layers, and the outer neurons form the input and the output layers.

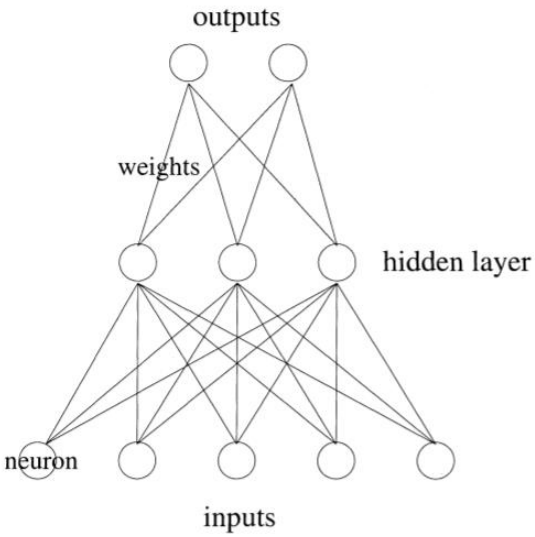


**Figure 1-1 Neural network**

Each link has weight, which is numeric number associated with the links. The output of any hidden layer is multiplied by an activation function.

The activation function objective:

- including nonlinearity to the neural network
- to include a boundary condition so that the divergent values does affect the whole network

Types of activation functions:

- step function
- sigmoid function
- tangent function
- hyperbolic function

Depending on the application, different function type is used.[9]

## Recurrent neural network (RNN)

Recurrent neural network is created to give the neural network the ability to learn sequential and time varying patterns. A recurrent net is a neural network with feedback connections.

The recurrent neural network could be fully connected or partially connected.

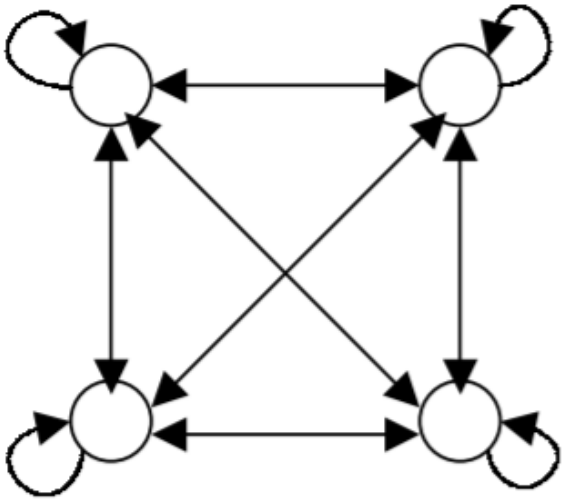

**Figure 1-2 fully connected recurrent neural network**

From the figure above we see the fully connected recurrent neural network, each neural has input from all neurons (including the feedback).

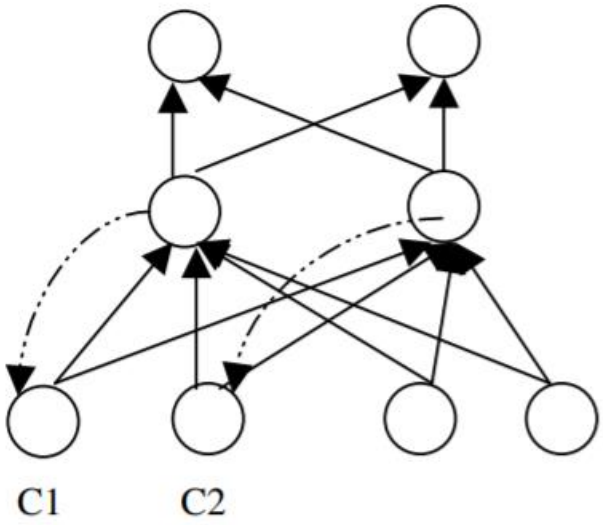


**Figure 1-3 partially connected recurrent neural network**

As from the figure above the partially connected not all the neurons has feedback.

The two neurons (C1 and C2) called context units. In above example the context units get the feedback from the middle layer, which is delayed feedback since it is from the next layer.

There is two ways to use the feedback:

- from the hidden layers to the context units in the input layer
- from the output layer to the context units in the inputs layer.[10]

## Long short-term memory (LSTM)

In the RNN there is a problem arises when using long-term components that the norm of the gradient drops down to zero exponentially or goes to infinity exponentially. That make the long-term correlations hard to reach.

To avoid this problem the LSTM is used.

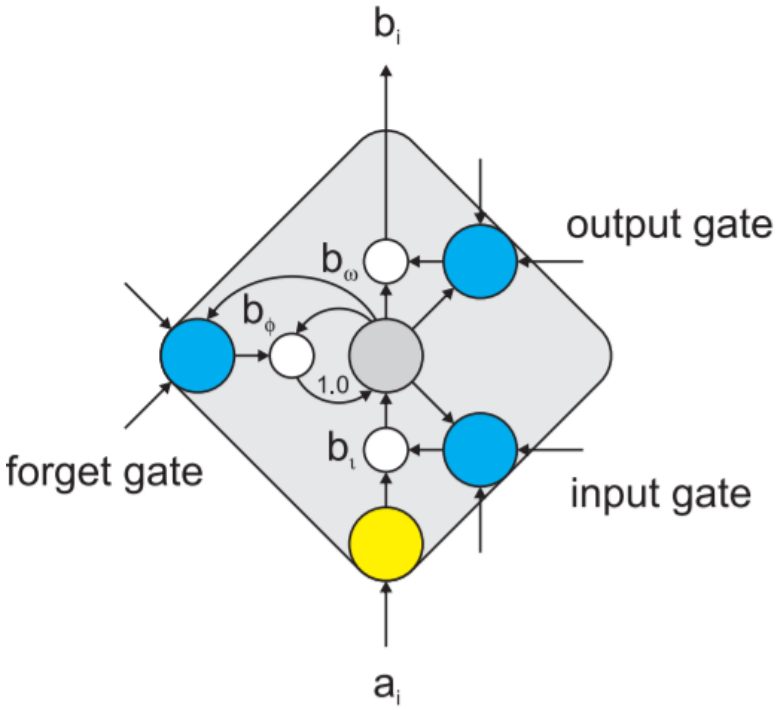


**Figure 1-4 LSTM memory cell**

The LSTM is type of RNN. LSTM unit is consisting of:

- a cell.
- input gate.
- output gate.
- forget gate.

The cell act as memory and save values over any time intervals. The gates control the flow of the data in the cell and control the output and the input flow. [11]

Many applications can use the LSTM like:

- machine translation
- speech recognition
- time series prediction

We used the LTSM to predict the positions for the players and the robots on the field.[11]

## Object tracking

object tracking is important for visual understanding in videos.

The purposes of using object tacking are:

- to distinguish different objects into different tracks
- to keep the same object in a single track.

Since we have multiple objects to track and give the location of each object, multi-object tracking system is used.

The multi-object tracking has:

- fast and stable runtime speed.
- precise and robust localization accuracy.
- The capability for online processing.

Old multi-object tracking algorithms has three-stage scheme:

- Detection
- Track let generation
- Linking

Detection is done by per-frame object detect to know the location of the object in the screen. Then the Tackle generation stage comes next. It combines the results that obtained from stage one, to create the track let (means short tracks, based on short-term cues).

The tractlets include:

- motion
- appearance features
- geometrical affinity

The at last the linking stage. In this stage the algorithm links all the tracklets that been created in the previous stage to form long-term tracking outputs.

This old scheme has three stages and could get good results in tracking quality. but it has many limitations.

One, it depends highly on detection and the generation of the track lets. So, if the detection and the generation of the track lets failed the output will be affected by that.

Two, if something missed by the detection stage there is no way to recover it.

three, the track let generation process usually apt to fail the matches.

So, to avoid these limitations single shot multi object tracker (SMOT) is used. It is system created to avoid the limitation that the older system has. It consists of two-layer stages:

- track let generation
- track let linking.

Compare to the old schemes there is no detection stage. In this method tracking by re-detection has been used.  it focusses on performing detection and tracking at the same time.

The track let generation stage begin with finding or detecting the objects in the first frame then after taking the results and use it to re-detect it in the second frame. with the assumption there is no motion between the two frames. [6]

So, this will be used to detect the players and the robots on the field after we get the video from the camera, then display it on the monitor.

## *1.2 Problem Definition*

To train players to play against teams with zonal defending style, coaches need players to have the knowledge of the defensive style of the team they are facing. So, it is time consuming to train players to this defensive style and hard to create different defensive styles to train against. Having a complete system that can take the data form the opponent defenders about their positions to simulate the defensive style using group of robot player that have high response time to the players on the field. This will give teams attackers the ability to practice playing against any type of defending style they want.

## *1.3 Literature Review*

one of the related topics to our project is the 2D soccer simulator. An open-source package called HELIOS Base. HELIOS base involves a common library, a sample team, a visual debugger, and a formation editor, which help us to develop a simulated soccer team.

HELIOS base software components are all coded using the standard C++. The code depends on the POSIX API, the boost C++ libraries1, and Qt2, but never includes environment specific dependencies. This makes HELIOS compatible with all operating systems like: Windows, Mac OSX and Linux.

HELIOS base has four components: librcsc, agent2d, soccerwindow2 and fedit2.

Librcsc is the common library for developing 2D soccer simulation software. It has multiple library files like geometry, network interface, communication and synchronization with simulator, world model, basic actions, log parser, debug message management, and formation model. librcsc contain all the all things related to the communication between the simulator and agent programs.

Agent2d is a sample team that uses librcsc and contains a data set of the simulated soccer team. This sample team is used as a template when starting team development. Agent2d's process of decision making consist of three layers: agent class, role class, and behaviour class. agent class is responsible for any decision making. role class and behaviour class are usually implemented by the developers.

soccerwindow2 is a viewer program for the 2d soccer simulation. It could work as a monitor client, stand-alone log player, and as visual debugger.

fedit2 is a formation editor for the agent2d. It is a GUI application to edit the formation data. agent2d supports the formation framework provided by librcsc. This formation framework is using Delaunay triangulation. so fedit2 helps editing this formation framework.[1]

The other part of our project depends on the dual wheel robot. it is inspired from the Segway personal transport. A mechanically unstable, two-wheel self-balancing vehicle. this vehicle depends on sensors and intelligent control system so, we can call it a robot. The controller used here was PI-PD controller.

The Two-wheel self-balancing robot has three components : sensors, motor, and motor control, and develop board.

The sensors are used to measure the robot’s tilt angle and angular velocity. These data are needed to balance the robot. The sensor are MEMS sensors including a gyroscope, an accelerometer, and wheel-angle encoders. The gyroscope is for measuring the angular velocity. The accelerometer is for measuring the s the total external acceleration of the balancing robot, which includes the gravitational and motion accelerations. The encoders are used measuring the rotation angle of individual motor shafts as digital signals.

The gyroscope is used to measure the angular velocity of the robot, and an accelerometer measures the Y and Z-components of gravitational acceleration, and encoders measure the distance travelled by the wheels.

When choosing the motor, it should have high torque rather than higher velocity, because it must oppose the rotational moment that gravity applies on the robot. The maximum torque should be enough to correct a tilt angle of 20 degree. For power supply, the motors need a voltage between 12V to 16V, and the development board needs between 5V to 15V. so, two batteries are incorporated: for the motors we select a 14.8-volt L-battery, and for the development board, they choose a four-cell Ni-MH battery pack.

The developed board they used was a mega Arduino for its fast response time and its large number of I/O pins.[2]

Another robot design was a robot called Omni-directional Robo-Cup Player. The robot should be light but not to the point to be pushed by the other teams. The robot also must have the ability to kick the ball with certain accuracy. The design they came up with was The Phase IIIB Robo-Cup Robot. The robot was made by Mechanical Engineering undergraduate students in their university who were working on this project as their graduation project.

The robot has at the front a dribble bar. To get the dribbler to kick the ball the bar rotates with certain speed. The robot has three wheels. The first two wheels has 105° angle from the back wheel, this gives the robot an easier contact with the ball.

Then they improved the design by recessed wheels, and another improvement on the outside design.

The model has an easy inverse kinematic model. The location of the moving frame [xm, ym] ^T at the centre of the gravity of the robot. Although the kinematic of the model was easy but the dynamic of the model was difficult to derive. It was based on the Newton's Second Law.

After inserting the dynamic equation, they derived into the MATLAB Simulink, they obtained the second-order step-input behaviour, they got that the overshoot is 10% for the X position and they Y position around 5%. The settling time is 0.08 s. The settling value is 1.57 rad (90°). [3]

## *1.4 Aims and Objectives*

Many we hope to create a full product that help teams to practice more against certain defensive style by making the robots simulate the defensive style.

Our first objective is to collect data form different match. These data about the players' positions on the x-axis and y-axis. Then we use the position for each player to predict where the defensive robot should be. the prediction is AI program. after inserting the data into the program, it will start doing trails and compare it with the data that we collected then the program will be able to predict the position for the defensive team.

Then our second objective is to apply these predictions to a team of robots ( 11 robots). the players and the robots will have position sensors attached to them so we can get the position of all players ( including the robots). then the data that we collected form the players and robots, the AI program will use it to give the prediction of how to defend against these players.

our third objective is to make a fast robot to keep up with the players. our robots are dual wheels robot. dual wheel robots can change their direction in short period of time. And the dual wheel robots resemble the human body (at least very close compared to other wheeled robots). This requires a suitable control system to make the robots balance and move at the same time.

# Chapter 2 Design Discerption and Analysis

## *2.1 Introduction*

### The solution Idea

If your team is playing against defensive opponent, probably we will suffer from a boring and stressful game, and your team might lose after all even if he was the better. This is because most of the teams tend to use what called “Zonal Marking” or “Zonal Defence”.

In zonal defence every player is assigned to cover some area in the pitch, and all the players should move relatively to prevent their opponent from moving the ball to their area. This style should keep the opponent away from the areas where he can really make danger. Notice that the defenders is not asked to retain the ball as a high priority, but the first priority is to prevent the ball from reaching certain area.

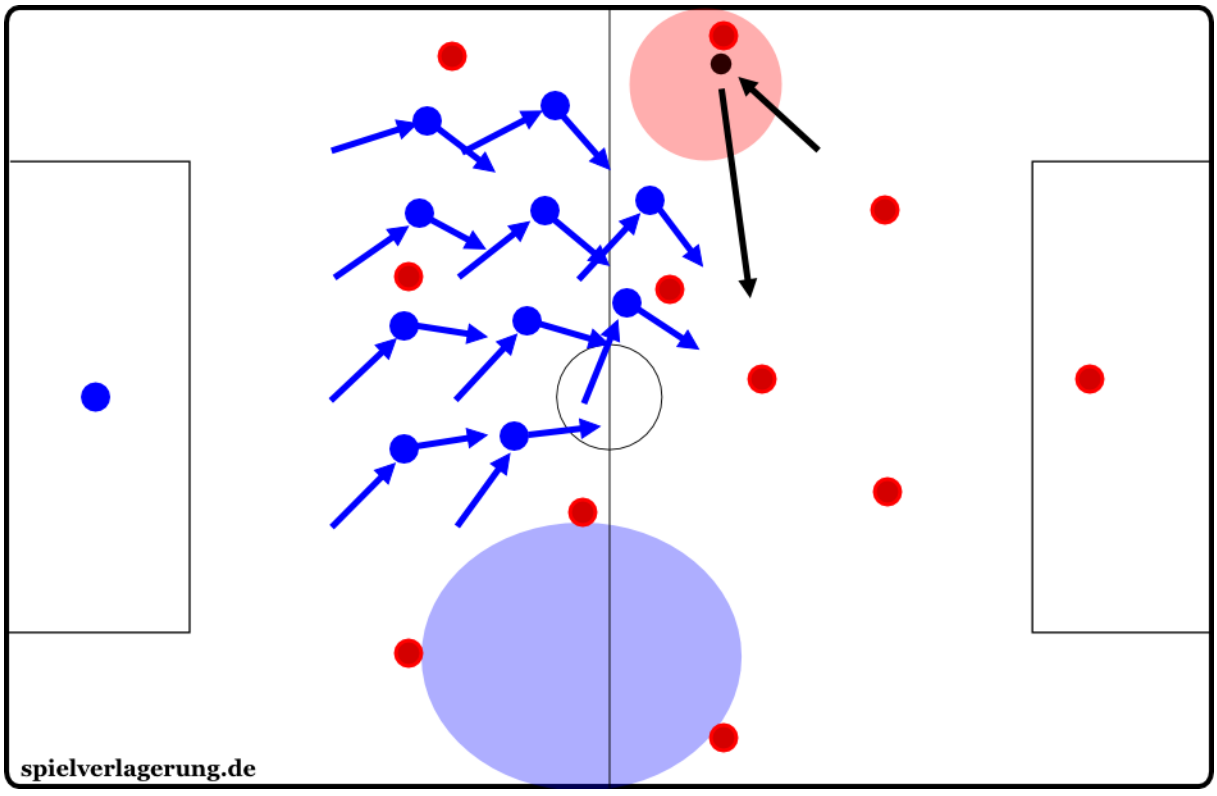


**Figure 2-1 Zonal Marking explained**

The solution we are proposing is to make a team of robots that can simulate the style for the opponent defence, and then we can use the robots to train our team to be familiar with the opponent defensive style.

**Figure 2-2 The solution is a team of robots to be used in training**

### Our part from the solution

The solution is a team of robots. This means that this is a big project to be done. The whole solution will consist of four main components:

- The sensor, to identify the locations of all players and robots and feed them to the prediction model.
- The prediction model, to be the brain for the robot and decide where the robot should move.
- The controller, to make the robots move as the prediction model has decided.
- The robots, the actual hardware that will move in the pitch.

In this project we are going to just build and train the prediction model.

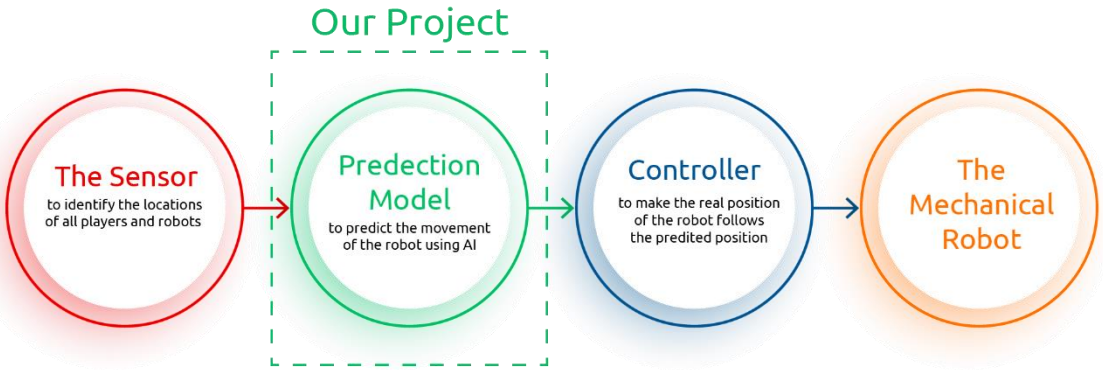


**Figure 2-3 Our project is just part from the whole idea**

## *2.2 Necessary Design Options*

### The sensor to be used: options

We need to identify the locations of the players, robots and the ball. We found many options to locate the players and the robots.

The first was the GPS module were many GPS modules mounted on the players and the robots to identify their locations.

Other option was for finding the robots locations by using a shaft encoder on their wheel and measure their distance from a fixed point on the field.

The last option was the distance sensors (ultrasonic and laser sensor) by mounting them on the robots and measure the distance from the edge of the field.

### Ball location problem

The ball location is part of the input data to the system. but the ball location identification is hard since we cannot place any sensor on the ball. so, most of the location sensors cannot be used because we need the location of the ball.

### The camera sensor

After more search about the best way to identify the location of the players and the ball we found that by using camera that stream the whole field from above. the camera then lifted by drone or if the training field is closed the camera could be mounted on the ceiling.

**Figure 2-4 camera held by a drone**

These drones differe from each outher in price, size, and the camera resulotion. but they all can be used in our project as the camera holder.

In professional soccer matches they usually use these drones, multiple of them so it could capture many angles of the field, and to zoom on the field when needed.

After searshing for suitable drone we reach for many good options that varies in price, size, and the camera resulotion.

The main 3 we found was :

- MAVIC 2  ZOOM
- MAVIC 2 PRO
- PHANTOM 4 PRO

The camera will give a video of the whole field with the players and the robots on it. the players will wear some colourful clothes and the robot will have certain colour on them, to make them easy to identify.

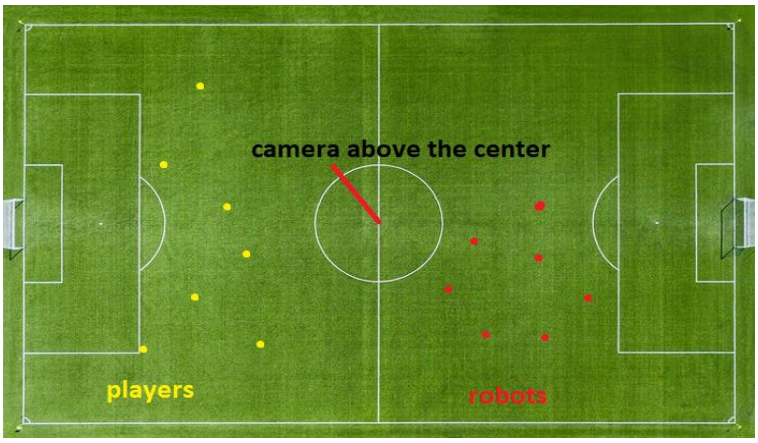


**Figure 2-5 the video the camera captures**

the camera will stream the video to the computer. the we will use a program to identify the location of all the players and the robots from the video.

The camera should have high frame rate per second to be able to follow the players and the robots clearly by the program. also, it must have high quality video so the program could be more accurate about the positions.


**Figure 2-6 camera module**

we look around for good camera that have these specifications and we came across multiple choices. the one we choose was a web cam that has 720 resolution and 60 fps at that resolution.

It is USB web cam. Well use for eye tracking devices, robot, observing golf, high speed video shooting device.

we will have to mount it on the ceiling and then use long USB cord to connect it to the computer.

**Figure 2-7 camera held on the ceiling**

### Mechanical design: options

The robot is the finish line in the project. To play with humans the robot should have certain specifications like its speed, weight, and size.

Two robot design we found that can be used as humanoid player:

- dual wheels robot
- omni-directional wheels robot

The dual wheels robot is not very practical in our project since it needs a balancing system and the robot should rotates before moving the desired direction.

### The omni-directional robots

Other design we found was the omni-directional wheels robot. What makes this robot design special is the ability to move forward to any direction without rotating first, which make it faster to move in any direction. Also, the movement to the certain location could be both translation and rotation movement.

**Figure 2-8 Omni-directional wheel**

This type wheels provides no friction in direction of the axis of the wheels. So, the wheel has two motions, one rotating around its axis (rotation if the large wheel). Two, the rotation along the wheel axis (the small wheel rotates to move in the axis direction).

Usually the omni-directional wheels robot has four wheels, which position in a way to make the robot move in any direction in most efficient way possible.

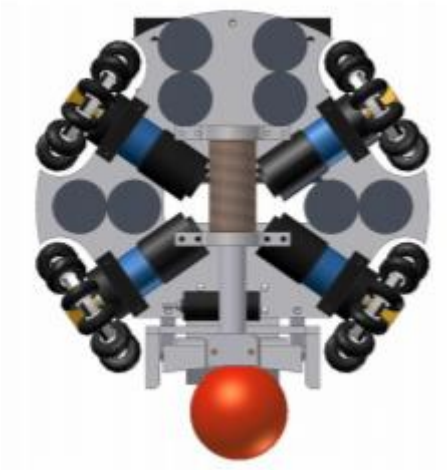

**Figure 2-9 Omni-directional wheels robot**

To control this robot PID controller must be used. The slipping wheels need a PID controller designated to this feature only.

The robot has four motor to drive each wheel. To control these motors anther PID controller is used.

The observation of the robot can be done from an external element. an example of the external element is a camera mounted on the ceiling or attached to drone.

But the robot can also hold the camera (as eyes) to get information from the surrounding. but this will cause a delay in the system. So, sending commands and receive the effects of the commands on the

system. To avoid this problem a prediction system is used. This system predicts the position and orientation after the delay depending on the previous position and orientation of the robot.

The robot could move with three motors accurately and without much loss in the speed.[8]

So, the robot is durable as training robot and could function as instructed. This robot is suitable for our training system.

## *2.3 The Design of AI Prediction Model*

The 11 robots should be able to simulate the movement of the opponent team players, thus the first step is to build a prediction model that predict the behavior and movements of the players. If the robots can move as close as possible to the opponent team players, then our team players will be familiar with the opponent style before the game and in result they can do better on the pitch.

First, we should find data and preprocess it, then we are going to do some exploratory data analysis to gain better understanding of the data, after that we need to make an initial model to see if our output is predictable by the data. The initial model will be a simple linear regression. If the output is predictable from the data, we are moving forward with more complex models to improve the accuracy. A position prediction using recurrent neural networks (LSTM) and a velocity prediction using deep neural network will be discussed.

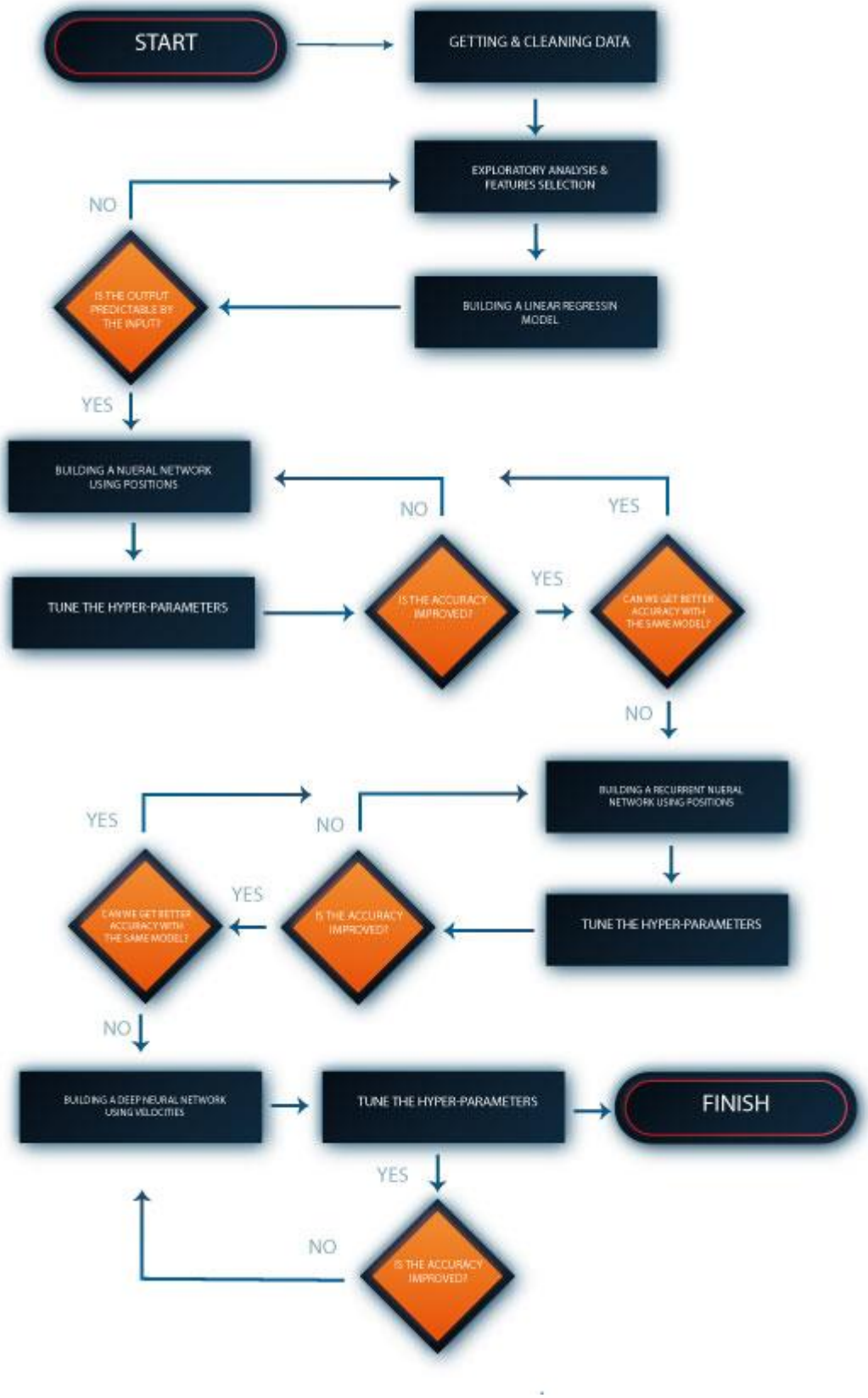


**Figure 2-10 The Main Flowchart of The Project**

## *2.4 Getting and Cleaning Data*

To make a good prediction model, the first step is to have the right data. We have spent a lot of time searching for datasets online and luckily, we have found that Metrica-sport has published sample tracking data. The original data from Metrica-sport is for two matches. For each match there is tracking data for the home team and away team, each in separate file, and an events data file for the hole match.

### Tracking data and events data

Football data usually are two types, the tracking data is the data in which all the players position are sorted for each sample of time. In other words, tracking data is about players locations during the match regarding if the player has the ball or not. Events data is more about the ball not the players. It gives all the actions that happens on the ball during the match, for example, pass or shot. Also, it gives information about the event like the players involved and location.

Table 2-1 shows a sample from the original events data and table 2-2 shows example from the tracking data for the home team.

**Table 2-1 Sample from the original events data**

| Team | Type | Subtype | Period | Start Frame | Start Time [s] | End Frame | End Time [s] | From | To | Start X | Start Y | End X | End Y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Away** | SET PIECE | KICK OFF | 1 | 51 | 2.04 | 51 | 2.04 | Player23 | | NaN | NaN | NaN | NaN |
| **Away** | PASS | | 1 | 51 | 2.04 | 87 | 3.48 | Player23 | Player20 | 0.5 | 0.5 | 0.4 | 0.51 |
| **Away** | PASS | | 1 | 146 | 5.84 | 186 | 7.44 | Player20 | Player18 | 0.43 | 0.5 | 0.44 | 0.22 |
| **Away** | PASS | | 1 | 248 | 9.92 | 283 | 11.32 | Player18 | Player17 | 0.47 | 0.19 | 0.31 | 0.28 |
| **Away** | PASS | | 1 | 316 | 12.64 | 346 | 13.84 | Player17 | Player16 | 0.29 | 0.32 | 0.26 | 0.58 |
| **Away** | PASS | | 1 | 395 | 15.8 | 423 | 16.92 | Player16 | Player15 | 0.26 | 0.66 | 0.27 | 0.91 |
| **Away** | BALL LOST | INTERCEPTION | 1 | 451 | 18.04 | 504 | 20.16 | Player15 | | 0.26 | 0.92 | 0.64 | 0.93 |
| **Away** | CHALLENGE | GROUND-LOST | 1 | 504 | 20.16 | 504 | 20.16 | Player23 | | 0.61 | 0.93 | NaN | NaN |
| **Home** | CHALLENGE | GROUND-WON | 1 | 504 | 20.16 | 504 | 20.16 | Player3 | | 0.62 | 0.92 | NaN | NaN |
| **Home** | RECOVERY | INTERCEPTION | 1 | 504 | 20.16 | 504 | 20.16 | Player3 | | 0.62 | 0.92 | NaN | NaN |
| **Home** | BALL OUT | | 1 | 504 | 20.16 | 534 | 21.36 | Player3 | | 0.62 | 0.92 | 0.54 | 1.01 |
| **Away** | SET PIECE | THROW IN | 1 | 672 | 26.88 | 672 | 26.88 | Player15 | | NaN | NaN | NaN | NaN |

**Table 2-2 Sample of the original tracking data set**

| | | | Home | | Home | | Home | | Home |
|---|---|---|---|---|---|---|---|---|---|
| | | | 11 | | 1 | | 2 | | 3 |
| **Period** | Frame | Time [s] | Player11 | | Player1 | | Player2 | | Player3 |
| **1** | 1 | 0.04 | 0.94275 | 0.50413 | 0.64812 | 0.28605 | 0.67752 | 0.42803 | 0.69882 |
| **1** | 2 | 0.08 | 0.94275 | 0.50413 | 0.64812 | 0.28605 | 0.67752 | 0.42803 | 0.69882 |
| **1** | 3 | 0.12 | 0.94275 | 0.50413 | 0.64812 | 0.28605 | 0.67752 | 0.42803 | 0.69882 |
| **1** | 4 | 0.16 | 0.94275 | 0.50413 | 0.64812 | 0.28605 | 0.67752 | 0.42803 | 0.69882 |
| **1** | 5 | 0.2 | 0.94275 | 0.50413 | 0.64812 | 0.28605 | 0.67752 | 0.42803 | 0.69882 |
| **1** | 6 | 0.24 | 0.94275 | 0.50413 | 0.64812 | 0.28605 | 0.67752 | 0.42803 | 0.69882 |

## Pre-processing data

To make the data more usable, we did some pre-processing and reshaping for the data as following:

- Removing the first two rows since they do not contain any information that cannot be included in the next rows.
- Renaming columns for the players to be on the form (team _ player id _ coordinate) for example (away_22_x)
- Removing substitutes and preserve just the starting XI.
- Removing any rows that contains NA value.
- Merging the home and away files.
- Transforming the positions to metric values instead of normalized form.
- Flipping the positions in the second half so each team is shooting at the same direction the hole match.
- As we deal with a time series any information about time may not be meaningful except if we decided to calculate velocities, so we are going to keep them.

The tracking data after cleaning shown in table 2-3

**Table 2-3 Tracking data after cleaning**

| Period | Frame | Time | home_11_x | home_11_y | home_1_x | home_1_y | home_2_x | home_2_y |
|---|---|---|---|---|---|---|---|---|
| **1** | 51 | 2.04 | 0.94791 | 0.48986 | 0.64787 | 0.27031 | 0.67763 | 0.4263 |
| **1** | 52 | 2.08 | 0.94779 | 0.49005 | 0.64797 | 0.27005 | 0.67765 | 0.42629 |
| **1** | 53 | 2.12 | 0.94766 | 0.49025 | 0.64805 | 0.26978 | 0.67767 | 0.42637 |
| **1** | 54 | 2.16 | 0.94756 | 0.49039 | 0.64815 | 0.26943 | 0.67769 | 0.42634 |
| **1** | 55 | 2.2 | 0.94746 | 0.49049 | 0.64826 | 0.26908 | 0.6777 | 0.4263 |
| **1** | 56 | 2.24 | 0.94736 | 0.49059 | 0.64835 | 0.26881 | 0.67768 | 0.42633 |
| **1** | 57 | 2.28 | 0.94725 | 0.49069 | 0.64845 | 0.26854 | 0.67766 | 0.42628 |
| **1** | 58 | 2.32 | 0.94714 | 0.49079 | 0.64856 | 0.26827 | 0.67762 | 0.42623 |

## Is the data sufficient?

Some other available datasets give information about the facing angle, traveling angle, and speed of the player. As the design of our robots will be circular there will be no need to determine the facing angle.

The velocity vector can be obtained from the position vector and time, and the velocity vector will fully describe the movement. As a result, if we used velocities or RNN and time series, both would give information about the change in position which is the velocity vector, and we can say that the data is sufficient.

Figures 2-1 and 2-2 showed two sample frames from the data, the dots represent the positions of the players and the ball.

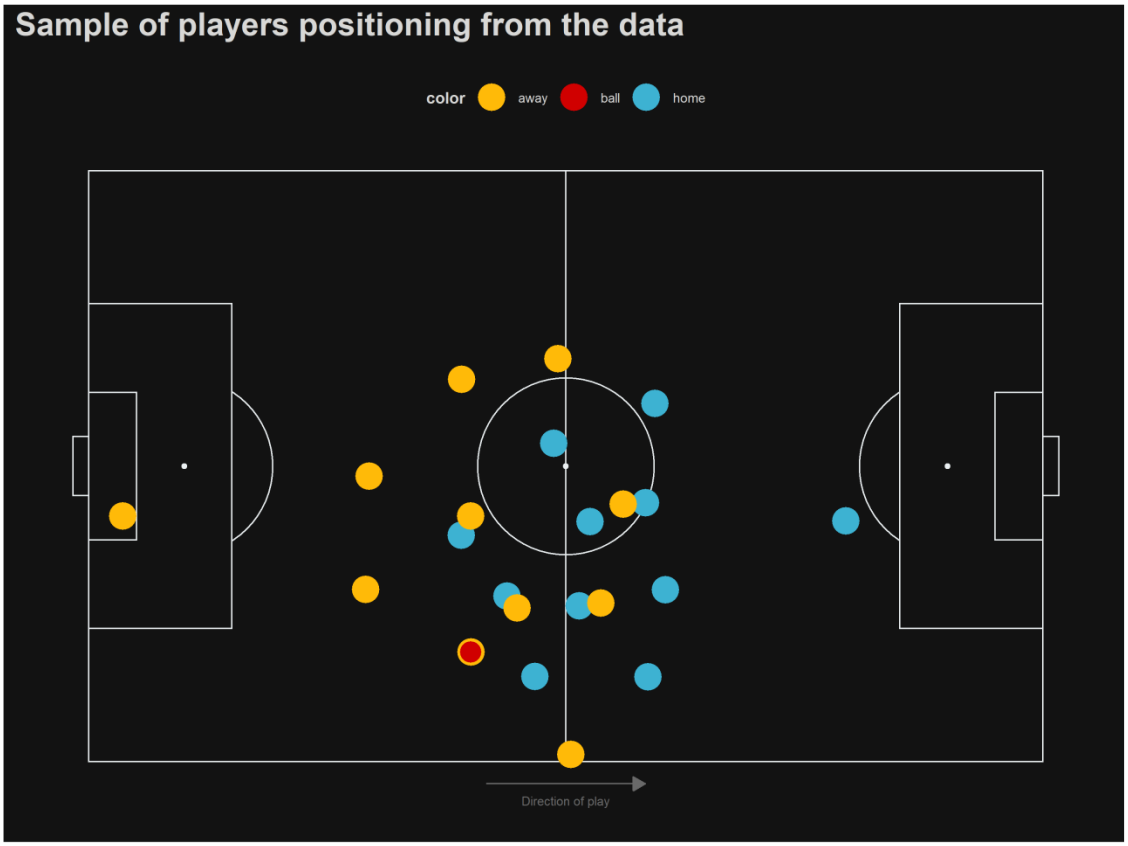


**Figure 2-11 Example of player positioning from the data.**

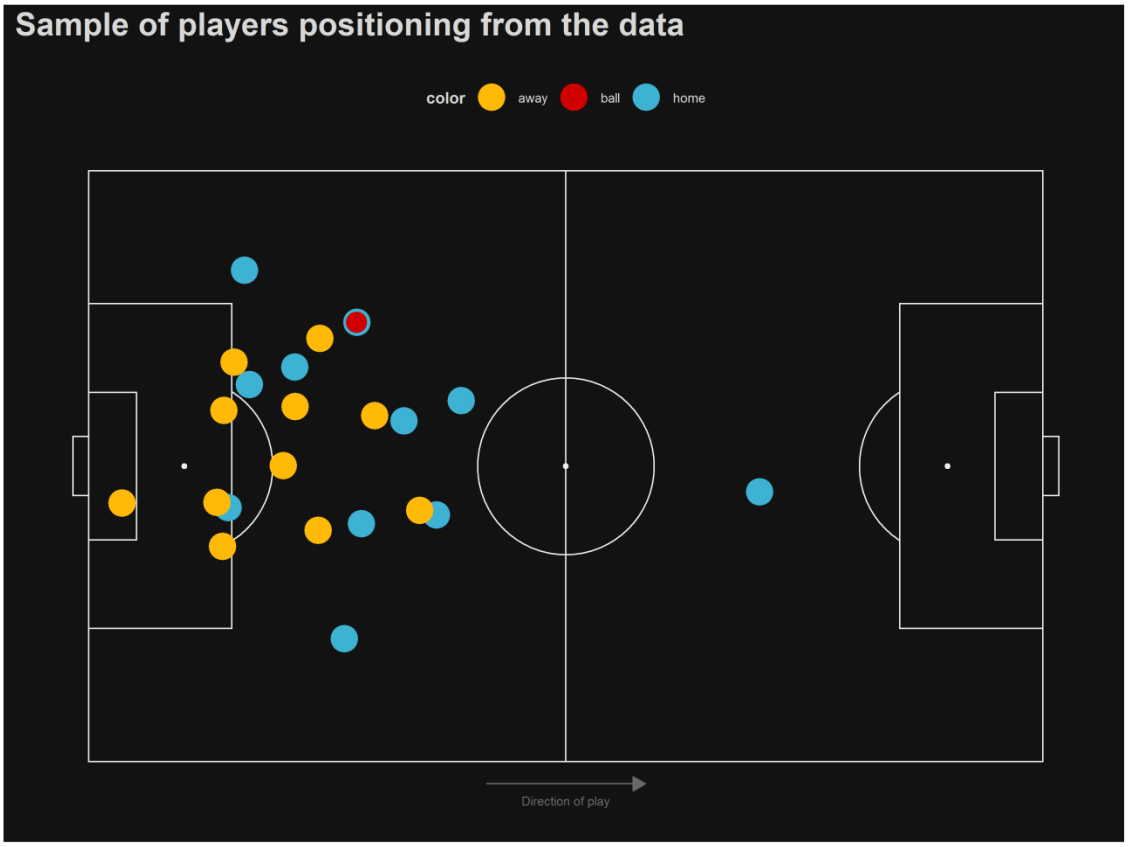


**Figure 2-12 another sample**

## *2.5 Exploratory Analysis*

Understanding the data is the first and maybe the most important part of building a prediction model. Thus, after the data has been read and cleaned, we must develop a better understanding for it.

### How players move

A player usually moves depending on the ball position, his teammates positions, and the opponent players positions. Figure 2-3 shows a player movement in some part of the match.

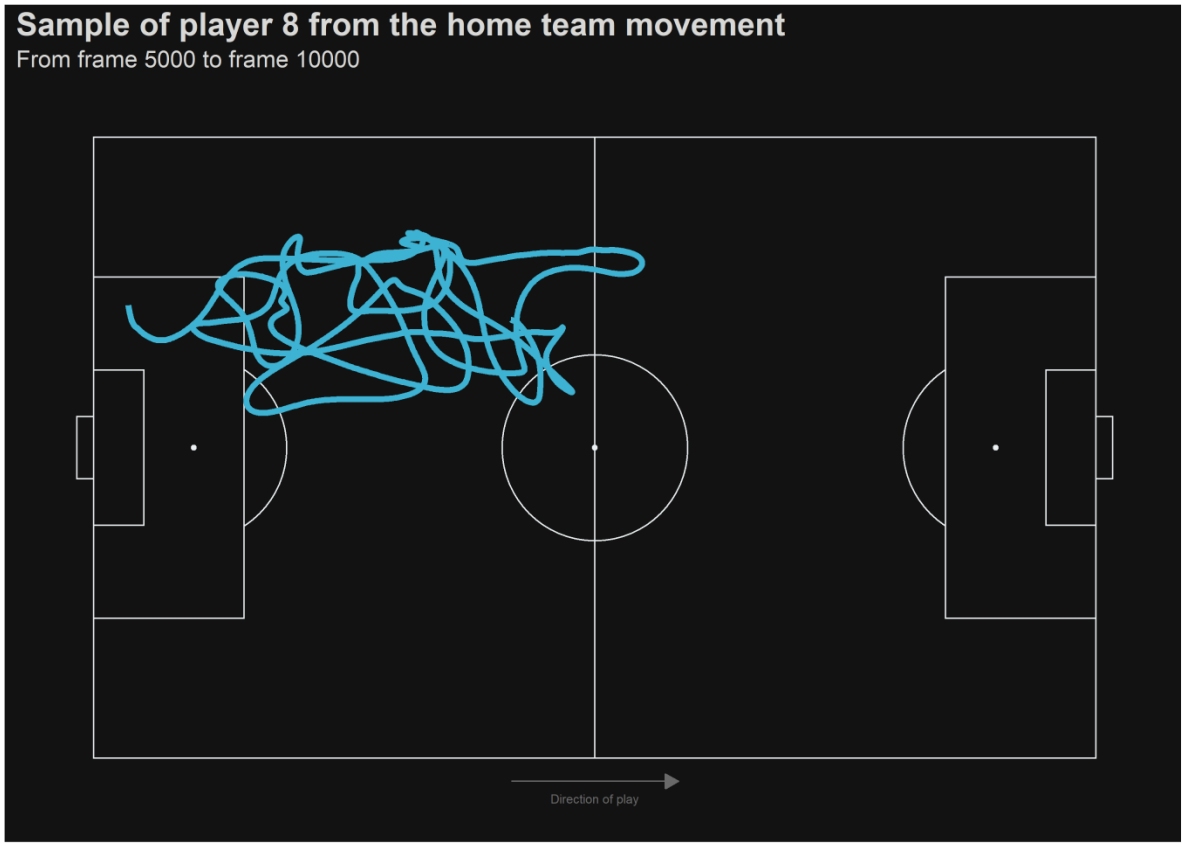


**Figure 2-13 Sample of one player movement**

But not every player has the same tasks on pitch, defenders and attackers will move in different areas since they have different tasks. for example, this is a comparison between four players movements shown in figure 2-4.

We can see that even during the same time, players are moving in different ways, and it's obvious that the home team is attacking in this sample from the game, so even defenders (player 2 for example) are moving into the opponent area.

If we take a look on longer period, we can see that the same players have both offensive and defensive responsibilities, this called the total football where the goalkeeper is the first attacker, and the striker is the first defender as it explained by the legendary Johan Cruyff. Figure 2-5 demonstrates this concept.

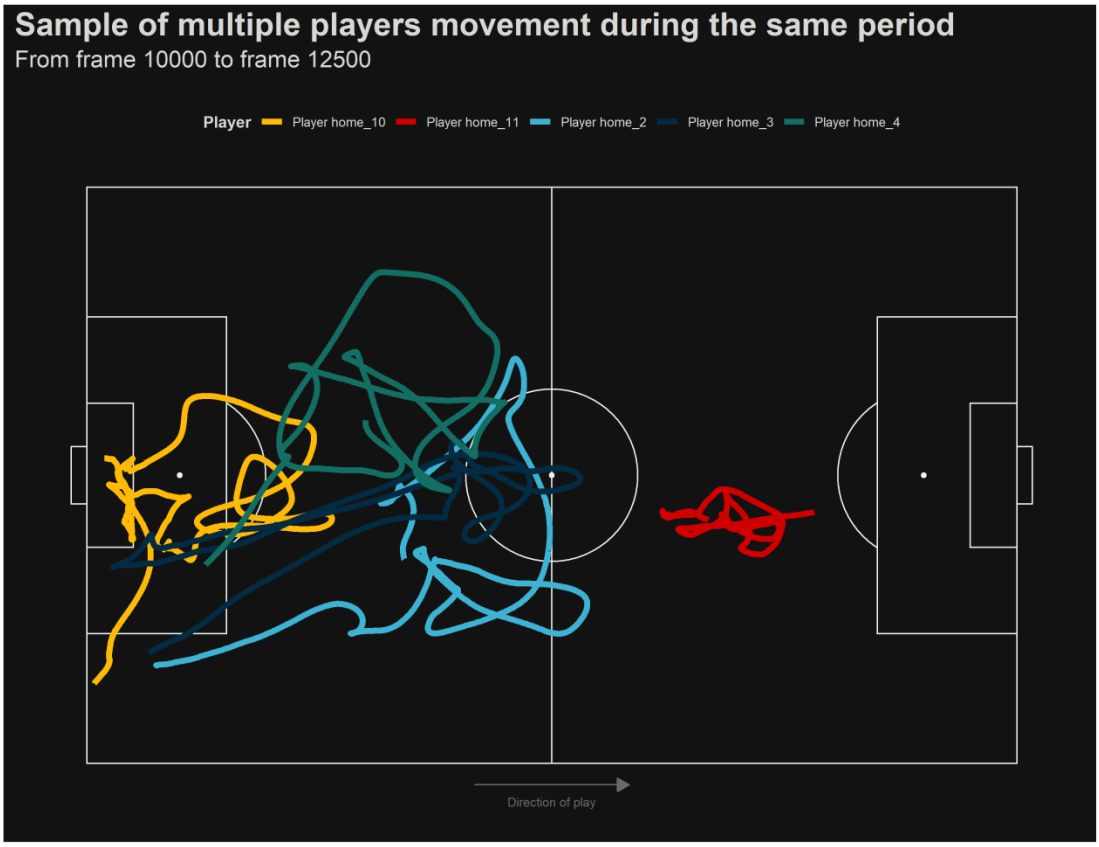


**Figure 2-14 Multiple players movement during the same period**

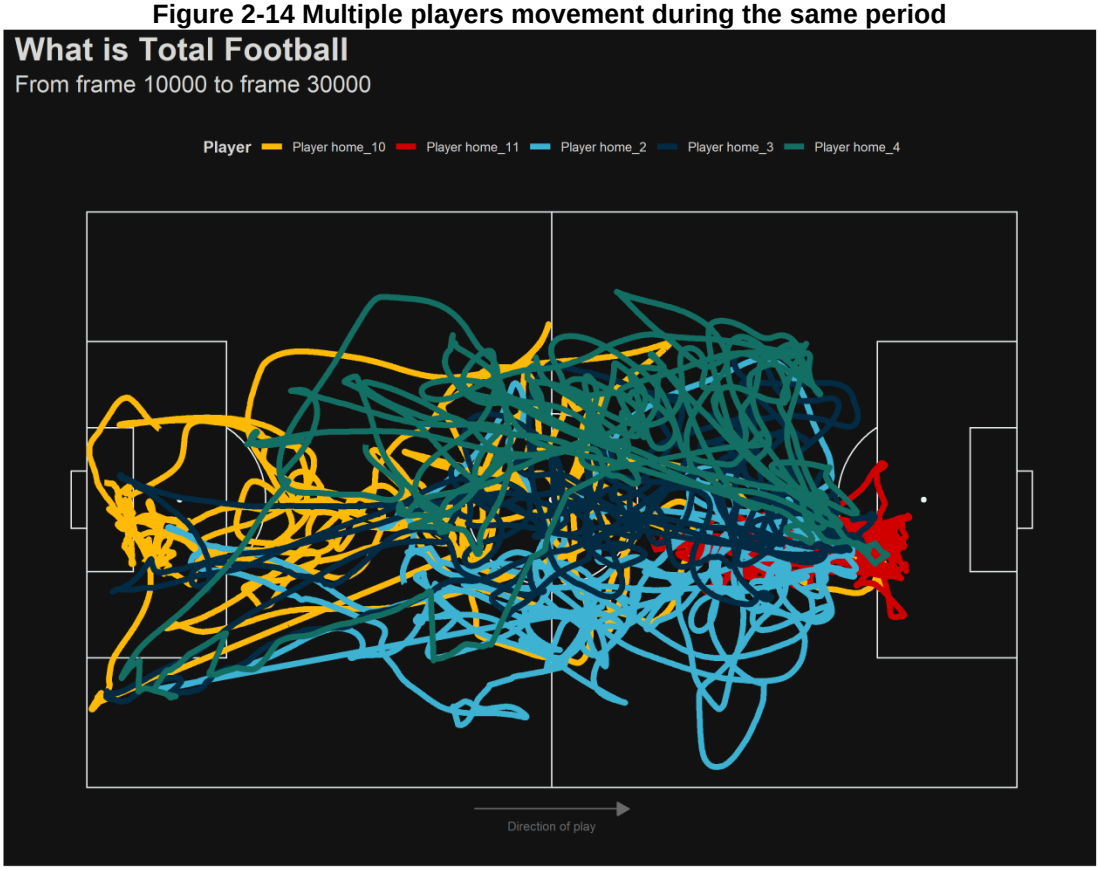


**Figure 2-15 Total football demonstration**

## How are players movements related

The correlation is a measure of how two variables is related. In the correlation matrix we see how some players are related, and we can see that most of the players depend on their teammates as much as they depend on the ball and opponents.

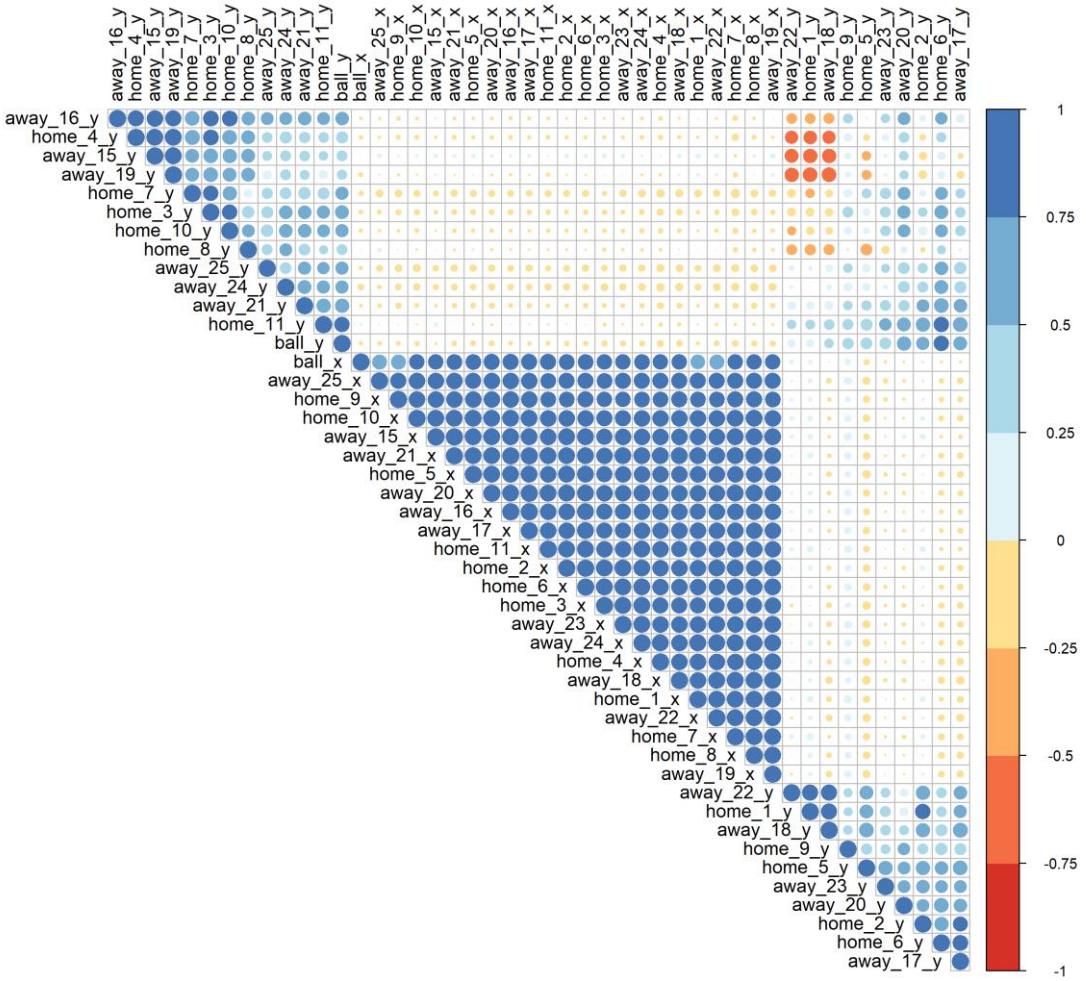


**Figure 2-16 Correlation matrix between all players**

After this trip we can say that:

- Players move relative to the ball, their teammates, their opponents.
- Every player has different responsibilities, so even though all of them are depending on the same factors to decide the next move, they still act differently because of their different responsibilities.
- All players defend and all players' attack.
- It is not necessary that the same player always moves in the same area.

### Feature's selection

We finally conclude from the exploratory analysis the following:

- We should build a model for each player, so we will have 11 models.
- The inputs for every model should be all the players locations (teammates and opponents) and the ball location.
- The output is the location for the player.

## *2.6 Initial Model: Linear Regression Using Positions*

The purpose of the initial model is to ensure that the output is predictable by the data. So, it will be a simple model, quick and dirty.

### Model definition

We will start with linear regression as our initial model. The model consists of 22 separate models, each predict one coordinate for one player from all other players position, and the ball position. the models represented by the equation below:

$$y_1 = a_{1,0} + a_{1,1}\,px1 + a_{1,2}\,py1 + \cdots + a_{1,44}\,py22 + a_{1,45}ballx + a_{1,46}bally$$

$$y_2 = a_{2,0} + a_{2,1}\,px1 + a_{2,2}\,py1 + \cdots + a_{2,44}\,py22 + a_{2,45}ballx + a_{2,46}bally$$

.

.

.

$$y_{22} = a_{22,0} + a_{22,1}\,px1 + a_{22,2}\,py1 + \cdots + a_{22,44}\,py22 + a_{22,45}ballx + a_{22,46}bally$$

### Training and testing sets

In order to build a prediction model, we divide the data into two sets: training set and testing set. The training set is used to train the program. The testing set is for testing the system after the training process.

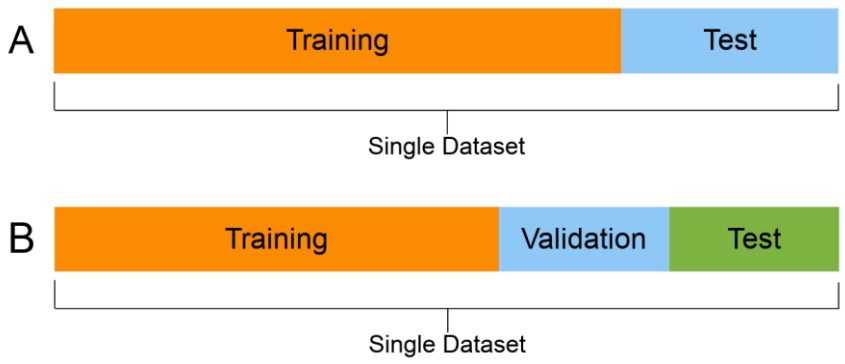


**Figure 2-17 Training and testing splitting**

Some times we split the data into three sets, the training set to train the model, the validation set to tune the hyperparameters, and the test set to test the accuracy of the model.

The number of observations in the data after cleaning is 58795, we need to split it into train and test sets, we used 70% of the data for the training set.

**Table 2-4 Training and testing sets**

| Training Set | Testing Set |
|---|---|
| 41156 observations | 17639 observations |

## Model building and testing for one sample variable

We have trained the model using (lm) function in R. At the first we start with a sample player we choose player 8 from the home team the results were as follows:

**Table 2-5 Statistics for home 8 x linear model**

| | |
|---|---|
| **R-squared** | 0.9696 |
| **F-statistic** | $2.912 \times 10^4$ |
| **p-value** | < 2.2e-16 |

So far, this result is a good indication, but we still need to analysis residuals. By looking at Figure 2-8 we cannot see any special patterns in residuals vs fitted plot, so residuals are probably independent, also from Q-Q plot we can see that residuals are normally distributed.

Also, we can see how the values of residuals are distributed, the reality that the median of residuals is -0.0757 is very promising. We can say that the output home_8_x is predictable by the data, but is that the case for all the variables?

**Table 2-6 Residuals for home 8 x**

| Min | 1Q | Median | 3Q | Max |
|---|---|---|---|---|
| -11.3542 | -2.2601 | -0.0757 | 1.9937 | 19.5416 |

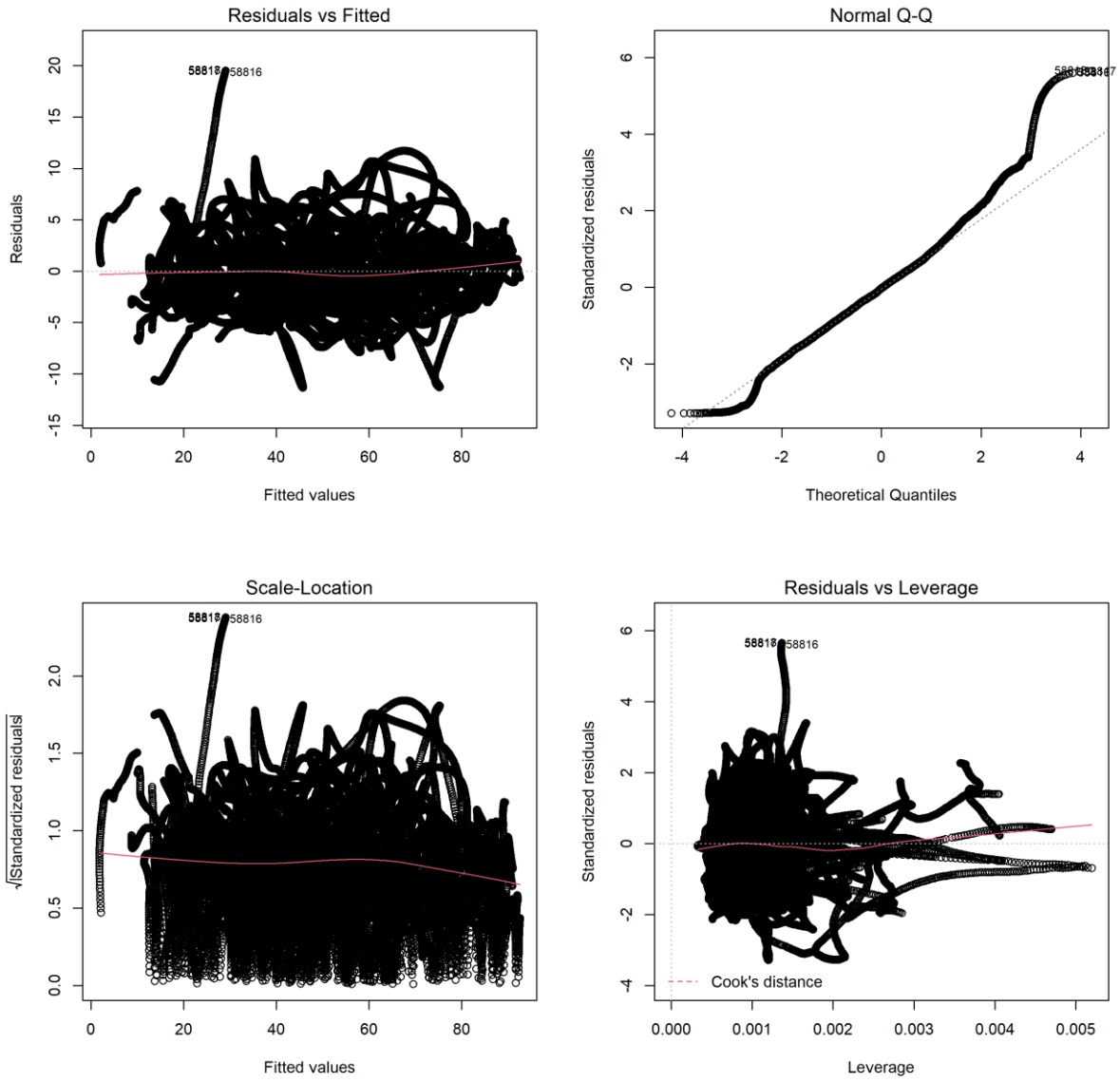


**Figure 2-18 Residuals for home 8 x**

## Many players model

We built some models for other players to see if the results are similar to the model above. We see from table 2-7 that the different models are good enough to conclude that the outputs are predictable by the data.

**Table 2-7 Many models**

| Model | R-squared | Residuals Median |
|---|---|---|
| **Home 8 x** | 0.9696 | -0.0757 |
| **Home 6 x** | 0.9614 | 0.2454 |

| | | |
|---|---|---|
| **Home 10 y** | 0.7867 | -0.1687 |
| **Home 7 y** | 0.8071 | 0.1553 |
| **Home 2 y** | 0.8885 | 0.1183 |

## *2.7 Improvement: Deep Neural Network*

In order to increase the accuracy of the model we are going to try deep neural network. Neural networks are good in fitting complex functions and coping human activities. For a very complex activity like football, we think the neural network will be much better than simple linear regression.

### Building deep neural network for one player

We build the model using Keras package in R. The outputs from the networks are the x and y coordinates for one player, the input is all other players locations and the ball location. We have tried many structures until we get the final model. We have used adam optimizer and ReLU activation function in hidden units.

**Table 2-8 Neural Network Architecture**

| No. of layer | No. of nodes | No. of parameters |
|---|---|---|
| **Input layer** | 44 | None |
| **First hidden** | 80 | 3600 |
| **Dropout** | Rate = 0.3 | |
| **Third hidden** | 60 | 4860 |
| **Dropout** | Rate = 0.3 | |
| **Fourth hidden** | 30 | 1830 |
| **Dropout** | Rate = 0.3 | |
| **Fifth hidden** | 10 | 310 |
| **Output layer** | 2 | 22 |

The total number of parameters in the network is 10,622 parameters.

### Training the neural network

We have trained the neural network for 30 epochs and with 32 batch size. Figure 2-8 shows the training process for player home 6. Models for different players are vary in their response, some of them has very good accuracy but over all we can see clearly that we **under fit** the data, so we need to get **more data** or to make **more complex model**.

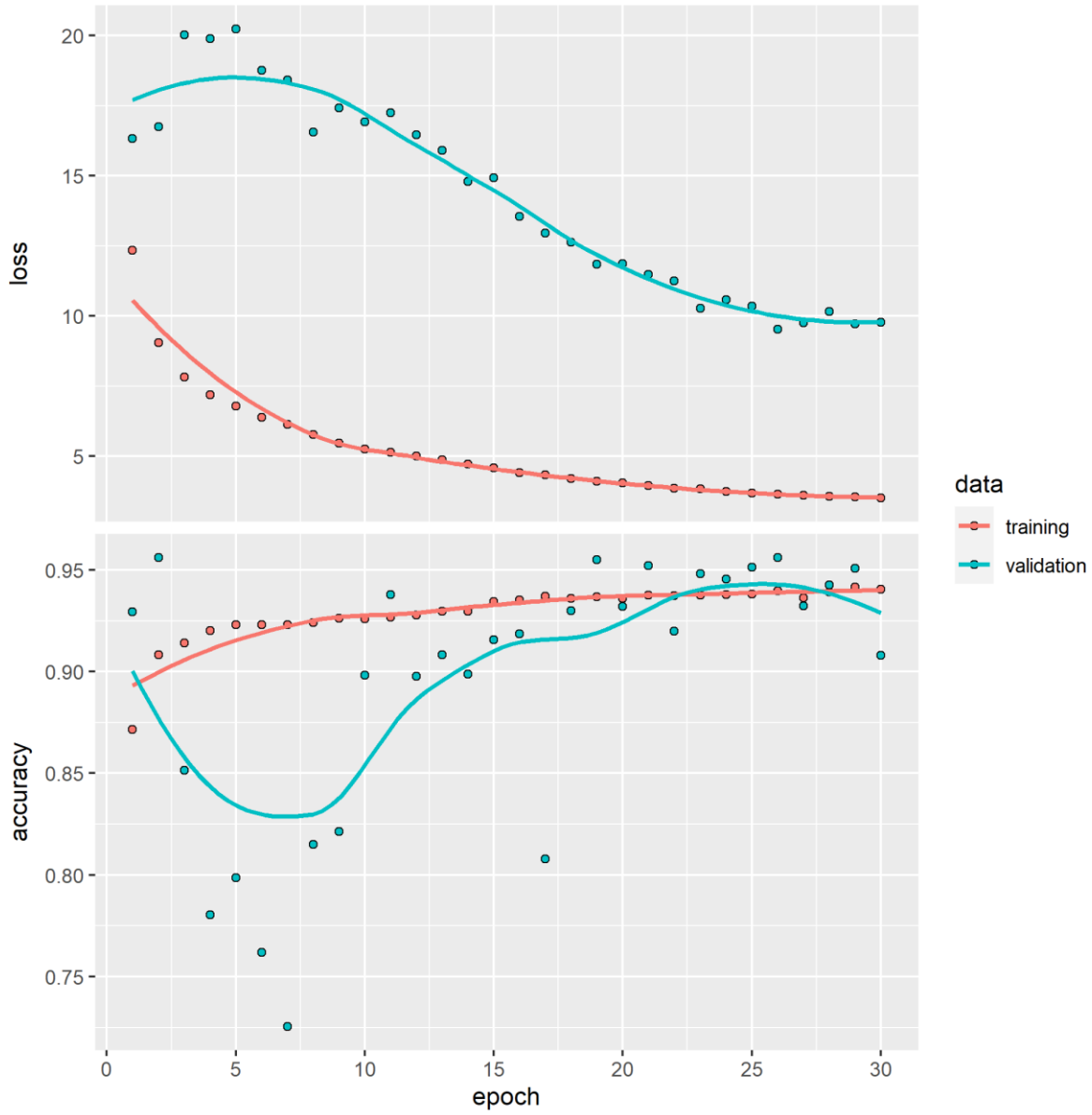


**Figure 2-19 Training process**

**Table 2-9 Models for different players**

| Model | Training accuracy | Test accuracy |
|---|---|---|
| **Home 8** | 93.07 % | 80.72 % |
| **Home 6** | 93.9 % | 87.19 % |
| **Home 10** | 90.77 % | 83.12 % |
| **Home 3** | 96.99 % | 69.77 % |

## *2.8 Advanced Model: Recurrent Neural Network*

The movements of the players in the field can be consider as a time series, so if we take only one frame to predict the output, we are missing some important information that is in the previous movement and here comes the use of RNN which is the perfect model for the time series data.

### Building and training RNN

We build the model using keras package is R, after reshaping the data to contain the last second of playing which is 25 observations. Throw experimentation we tried many architectures, for example this is the training history for 4 layers LSTM network with 100 units in each. This model is clearly **overfit** the data and this is an indication that we need more data to go further with our model.

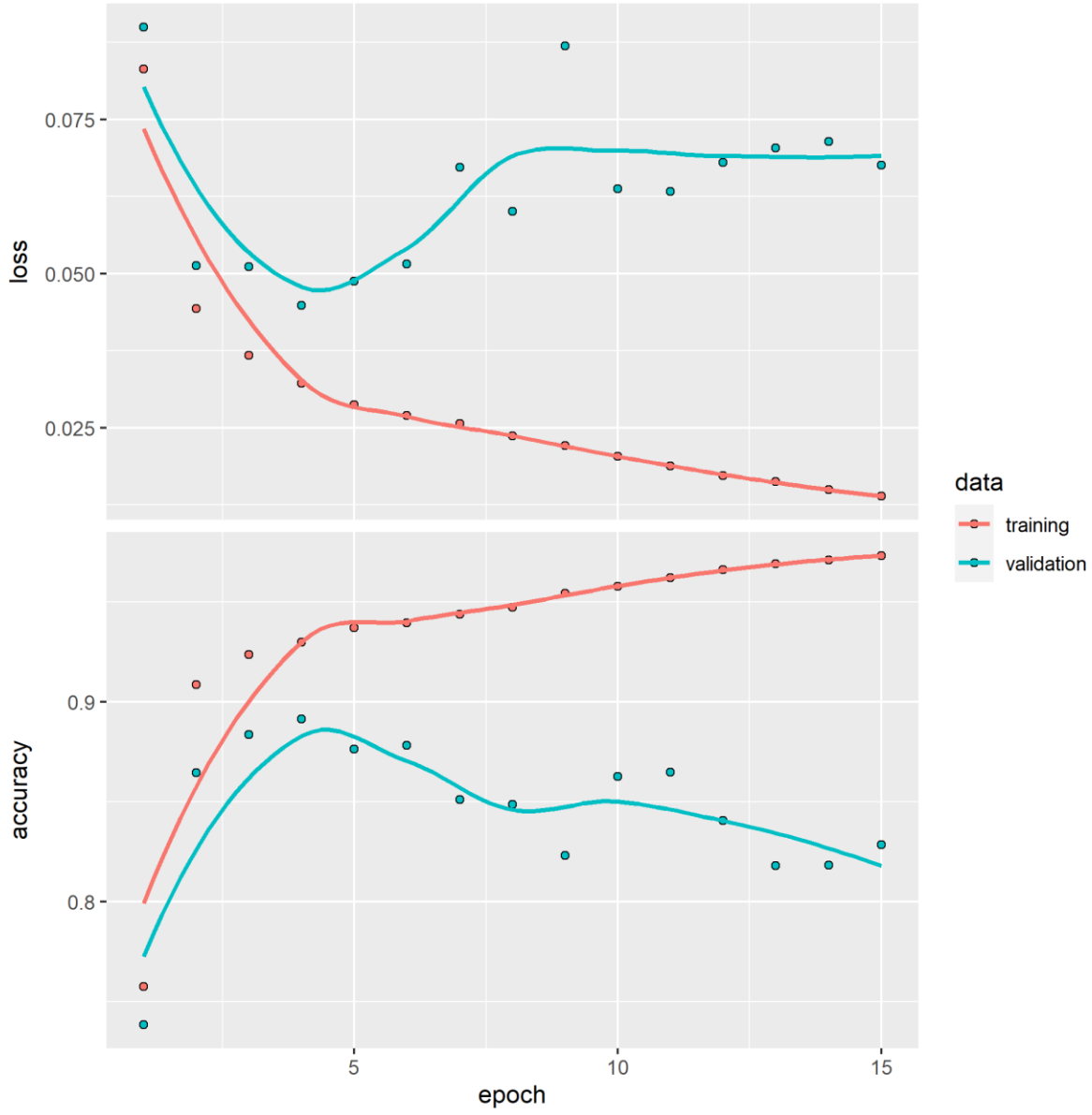


**Figure 2-20 Training history for 4 layers LSTM with 100 units in each layer**

We have tried many models on different players, none of them exceeds the performance of the deep neural network. Even though we believe that a RNN will suit the problem better we don't have enough data to make a good model from it.

So, if we cannot improve our predictions using RNN because the lack of data, what we can do?

## *2.9 Final Model: Deep Neural Network with velocities*

The main idea of using RNN is to include the previous observations into the prediction, but is there a way that we can include information about past in other way?

There is the velocity is the difference between two positions, which means that a velocity observation holds information about the past. So, we are going to add velocity vector to the data and then use a DNN.

### Calculating velocities

The velocity is the change in position, the time step between each observation is 0.04, so we calculate it like this:

$$V_x = \frac{x_{prev} - x_{current}}{dt}; dt = 0.04\ sec$$

$$V_y = \frac{y_{prev} - y_{current}}{dt}; dt = 0.04\ sec$$

### Building the model

We build a sample model on player away 16 and we get a very good results using this model. By looking to the training history, we can see that we have solved the overfitting problem that we have faced with the RNN, and, we have improved the accuracy.

The training accuracy is **96.44 %** and the testing accuracy is **95.25 %** and this is a satisfying result.

**Table 2-10 DNN architecture with velocities**

| No. of layer | No. of nodes | No. of parameters |
|---|---|---|
| **Input layer** | 88 | None |
| **First hidden** | 200 | 17800 |
| **Third hidden** | 120 | 24120 |
| **Fourth hidden** | 60 | 7260 |
| **Fifth hidden** | 30 | 1830 |
| **Sixth hidden** | 10 | 310 |
| **Output layer** | 2 | 22 |

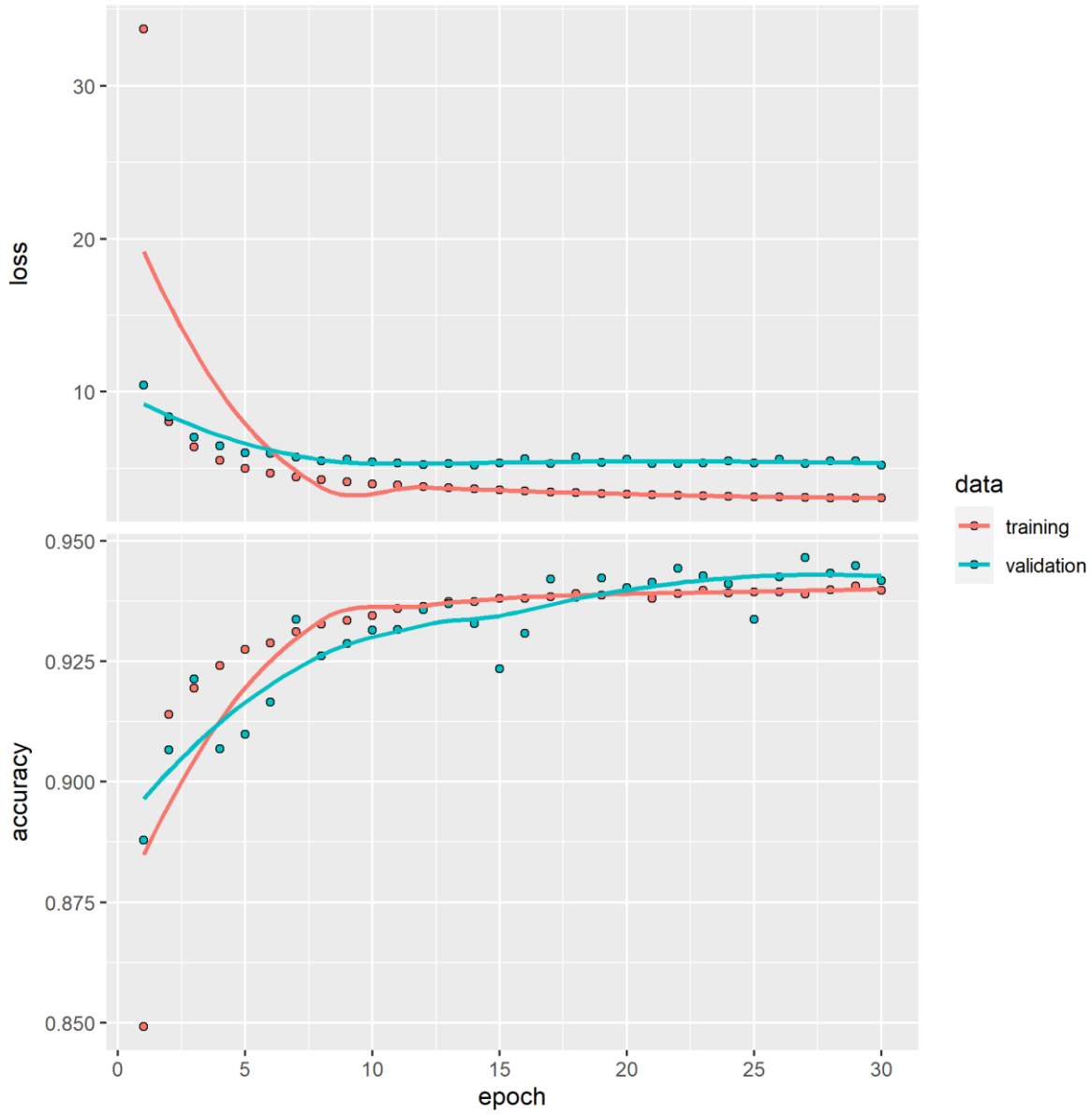


**Figure 2-21 Away 16 training history on DNN with velocities**

**Table 2-11 Models for different players, DNN with velocities**

| Model | Training accuracy | Validation accuracy | Test accuracy |
|---|---|---|---|
| **Away 16** | 96.44 % | 93.79 % | 95.25 % |
| **Away 20** | 96.74 % | 89.75 % | 88.44 % |
| **Away 24** | 98.56 % | 96.47 % | 86.29 % |
| **Away 19** | 94.05 % | 94.02 % | 95.17 % |

# Chapter 3 Design Testing & Results

Our purpose was to predict the opponent team players movements as accurate as possible in order to build the controller for the training robots. We have built 4 types of model, and in each type, we tried many different models and parameters to reach better accuracy. Finally, our DNN using velocities did a great job and there are the results.

## *3.1 Sample player*

This is the player away 19, the figure shows the actual player movement and the predicted, we can see that the prediction model doing well at predicting the player behaviour. Of course, it is not perfect but if we can get this result from just one match data, then this is satisfying.

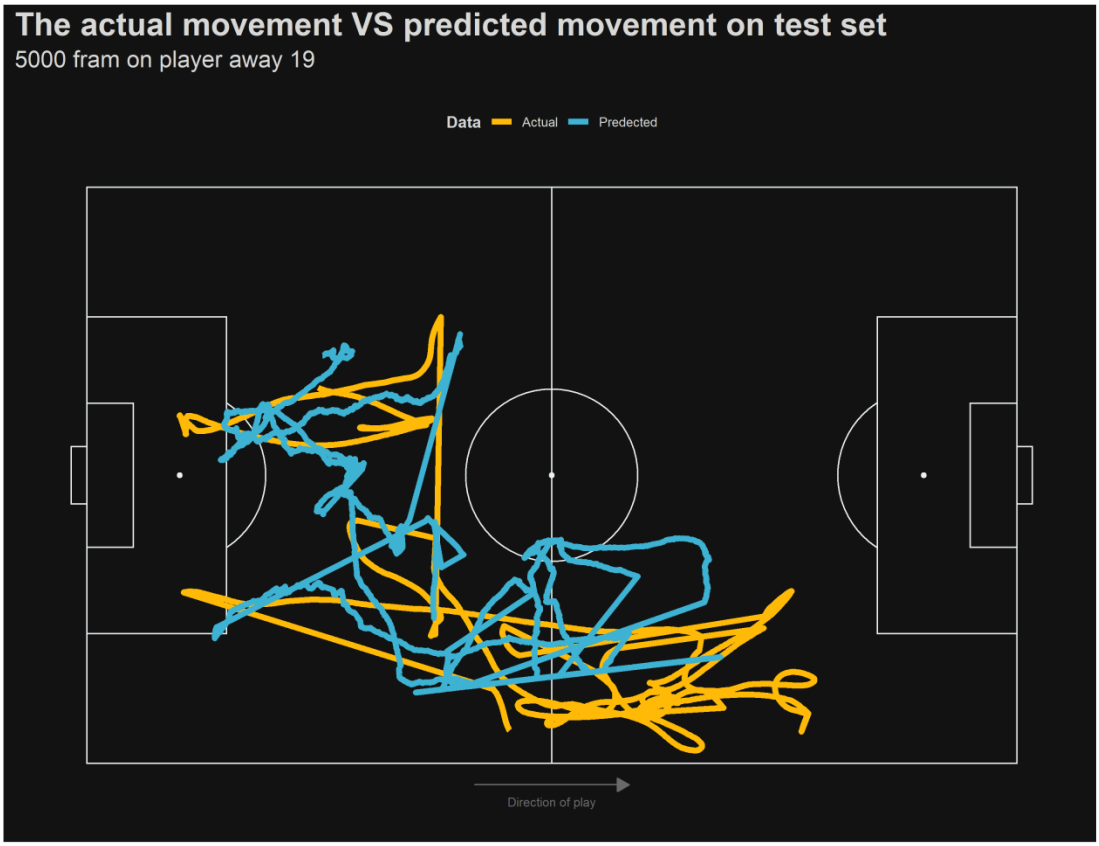


**Figure 3-1 Predicted VS Actual for player away 19**

## *3.2 Accuracy for all players models*

After training all the 11 models we calculated the accuracy on the test set for each of them and there are the results:

**Table 3-1 Accuracy for all models**

| **Player** | **Test set accuracy** |
|---|---|
| Away 25 | 98.43 % |
| Away 15 | 87.26 % |
| Away 16 | 96.30 % |
| Away 17 | 78.39 % |
| Away 18 | 64.59 % |
| Away 19 | 92.47 % |
| Away 20 | 84.93 % |
| Away 21 | 88.06 % |
| Away 22 | 73.05 % |
| Away 23 | 90.40 % |
| Away 24 | 92.32 % |
| **Average accuracy for the whole system** | **86.01 %** |

# Chapter 4 CONCLUSION & FUTURE WORK

## *4.1 conclusion*

Coaches need players to understand the defensive style of the team they are playing against to prepare them to play against teams that use a zonal defending technique. As a result, it takes a long time to educate players in this defensive style, and it is difficult to come up with alternative defensive systems to practice against. Having a full system that can collect data from opposing defenders about their locations to imitate defensive style utilizing a group of robot players with a fast response time to the players on the field. As a result, attackers will be able to practice playing against any form of defence they wish.

So, we designed a virtual training system. Using the data we collected about the players positions, we used the Artificial intelligence programs to create a prediction model to train robots to move like the players in the data we collected. Then the system could be used to train human players by playing against the robots.

The average accuracy for the whole system was **86.01 %**.

## *4.2 Future work*

Our focus in the project was on the Al and to create the prediction model. So, other field in the project we could not give enough time.

First, the robots and players localization.

After searching for many options, the best was the camera system where the camera is mount to the ceiling or on a drone. Then the data from the camara is sent to the computer, then an object tracking program will track all the players and the robots and give their position on the field.

second, the hardware of the robots.

The best robot option is the omni-directional wheels robot, since it is faster and more durable.

# REFERENCES


[1] Akiyama, Hidehisa, and Tomoharu Nakashima. "Helios base: An open-source package for the robocup soccer 2d simulation." *Robot Soccer World Cup*. Springer, Berlin, Heidelberg, 2013.

[2] H. -S. Juang and K. -Y. Lum, "Design and control of a two-wheel self-balancing robot using the arduino microcontroller board," *2013 10th IEEE International Conference on Control and Automation (ICCA)*, Hangzhou, China, 2013, pp. 634-639, doi: 10.1109/ICCA.2013.6565146.

[3] Carter, Brian, et al. "Mechanical design and modeling of an omni-directional robocup player." *RoboCup AI Conference*. Vol. 10. Seattle WA, 2001.

[4] how stuff works website. https://electronics.howstuffworks.com/gadgets/travel/gps.htm.

[5] Shawn, seed studio website : https://www.seeedstudio.com/blog/2019/12/23/distance-sensorstypes-and-selection-guide/?fbclid=IwAR2mcgoiQxQAZKVnoMQ4VcFEWFDCV9EqxDr7_t5HLFyCeX6NB862sTvIdmI

[6] Li, Wei, et al. "SMOT: Single-Shot Multi Object Tracking." *arXiv preprint arXiv:2010.16031* (2020).

[7] Adeel, Umar, et al. "Autonomous dual wheel self-balancing robot based on microcontroller." *Journal of Basic and Applied Scientific Research* 3.1 (2013): 843-848.

[8] Rojas, Raul, and Alexander Gloye Förster. "Holonomic control of a robot with an omnidirectional drive." *KI-Künstliche Intelligenz* 20.2 (2006): 12-17.

[9] Wang, Sun-Chong. "Artificial neural network." *Interdisciplinary computing in java programming*. Springer, Boston, MA, 2003. 81-100.

[10] Medsker, Larry R., and L. C. Jain. "Recurrent neural networks." *Design and Applications* 5 (2001).

[11] Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. "LSTM neural networks for language modeling." *Thirteenth annual conference of the international speech communication association*. 2012.

[12] Data from metrica sport web site https://metrica-sports.com/

[13] Github.com/metrica-sports/sample-data