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
MohammedYounes

Supervisor

Dr. Hussam Khasawneh

Date

28/02/2021

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**		
2	**Laith Hudaib**	**0160946**		
3	**Yazan Kraek**	**0163947**		
4	**Mohammed Younes**	**0162765**		

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 INTRODUCTION

1.1 Background

Many football teams nowadays tend to play with a low block defense, a style depends on closing and protecting space which make it difficult for strikers to score goals.

It is called zonal defense. It is a type of defensing style where the players are assigned to zone not player.

We want to create training robots that can simulate any certain defending style.

Most of the studies were just about collecting the data for analysis purposes. we are trying to create a product that helps teams to apply the analysis they did to the data in the pitch by creating robots that stimulate the certain defensive playing style.

These robots will move accordingly to the movement of the offensive team, creating a certain defensive formation for the attackers to counter them.

It is close to robo-cup where the robots play against each other. These robots could be just AI robot on the screen or actual robot on the field. All these robots have independent decision making and they rely on AI to decide.

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.

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.

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

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.

1.5 Report Organization

Chapter 2 Design of player positions prediction model

In this chapter we present the step-by-step design process of designing a prediction model for the player positions. We walk throw data cleaning, the exploratory data analysis, feature selection and building the model.

Chapter 3 Robots design options

In this chapter we should walk throw the design options for the robots. The communication system, location detection, component selection, and power circuits all should be presented.

Chapter 4 Controller design

In this chapter we should design our control system according to the selected components and prediction model. The selection of hardware controller, control algorithm, tuning and coding should be presented.

Chapter 5 Testing and validation

Chapter 6 Conclusion & Future work

Chapter 2 DESIGN OF PLAYERS POSITIONS 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.

2.1 Getting and Cleaning Data

In order to have a good prediction model, the first step is to have the right data. We spend a lot of time searching for datasets online and luckily, we found that Metrica-sport has published sample tracking data. The data contains the x and y coordinates for each one of the 28 players in the game at a certain moment of time. This makes the data consists of 53 columns and almost 150,000 observation. Table 2-1 shows a sample from the original data.

Table 2-1 Sample of the original data set

Period	Frame	Time [s]	p1x	p1y	p2x	p2y	p3x	p3y
1	1	0.04	0.90509	0.47462	0.58393	0.20794	0.67658	0.4671
1	2	0.08	0.90494	0.47462	0.58393	0.20794	0.67658	0.4671
1	3	0.12	0.90434	0.47463	0.58393	0.20794	0.67658	0.4671
1	4	0.16	0.90377	0.47463	0.58351	0.20868	0.6764	0.46762
1	5	0.2	0.90324	0.47464	0.58291	0.21039	0.67599	0.46769
1	6	0.24	0.90275	0.47464	0.58214	0.21123	0.67537	0.46727

To make the data more usable, we remove the first three columns that gives information about time as we deal with a time series any information about time will not be meaningful. We also removed the substitute players data as we only need the in-pitch 22 players to predict any player movement. Finally, we remove any rows that have missing data in it.

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. Even though we still need to determine the velocity for each player to fully describe the movement. In result we will have the x-y position and the x-y velocity for each player. The following figure shows the positions of the players as obtains from the data.

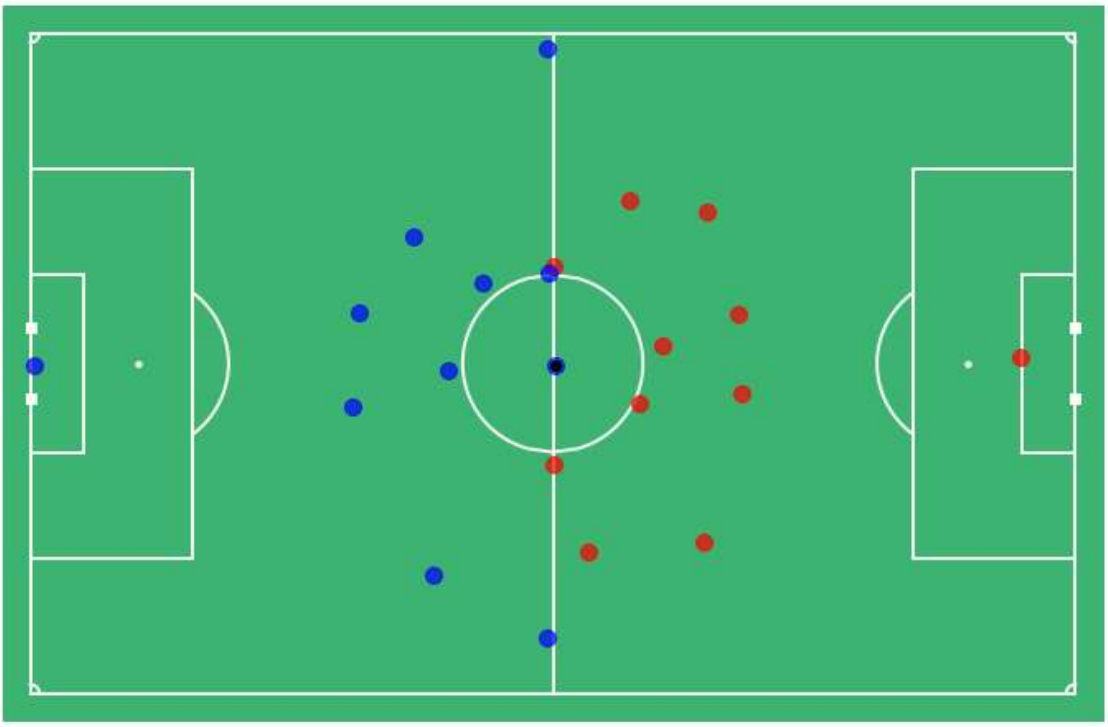

Figure 2-1 Example of player positioning from the data.

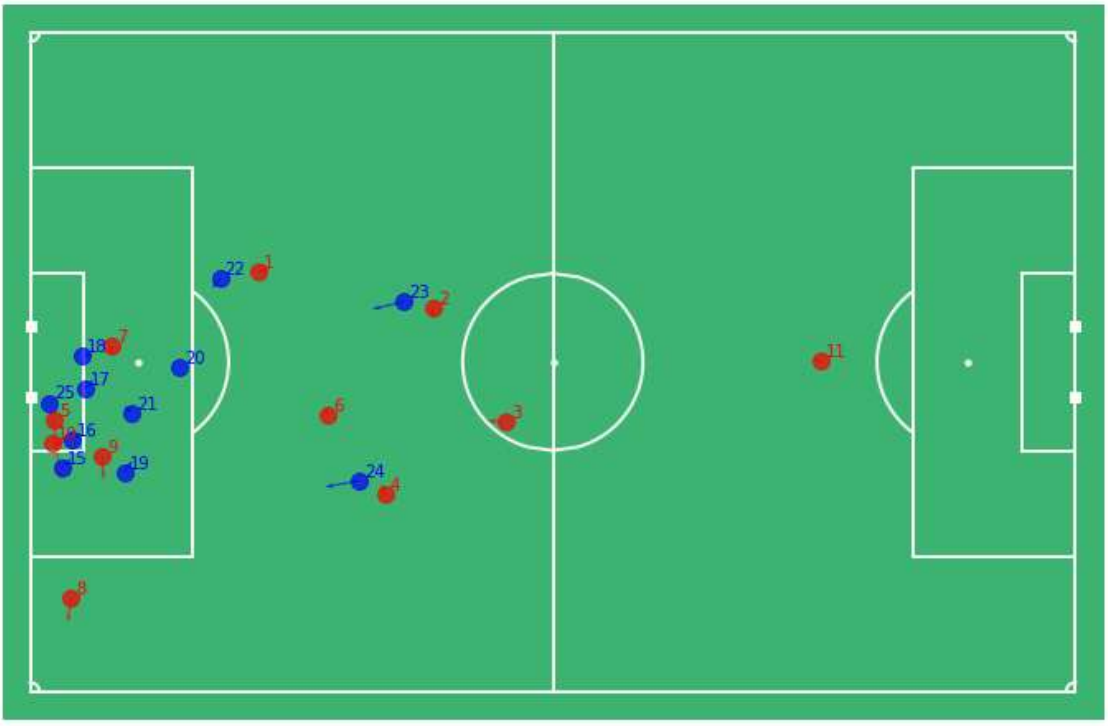

Figure 2-2 adding velocity vector for the frame

2.2 Exploratory Analysis & Features Selection

In order to build the model, we need to determine the inputs and outputs. The first idea was to build a model that has 22 inputs which is the x-y coordinates for all the attacking team players positions in certain moment and predict the 22 x-y coordinates for the robots. This idea is facing two problems, we will get to the second problem later, but the first one is that the player position does not depend on the opponent team positions only, but also on his teammates positioning. This can be seen from the correlations between the variables. We can see a clear pattern in which the variables correlate, and it obvious that the players position highly correlated with the positions of his teammates and the opponent as well. Also, if you take one frame of the game, the position does not tell you all the story, you also need to know where the players are going and, in any speed, so we extended the original data by calculating the velocity vector for each player. The correlation plot in Figure 2-2 shows that there is a relation between the velocity of all players but there is no correlation between the velocity for the players and the positions which is make sense.

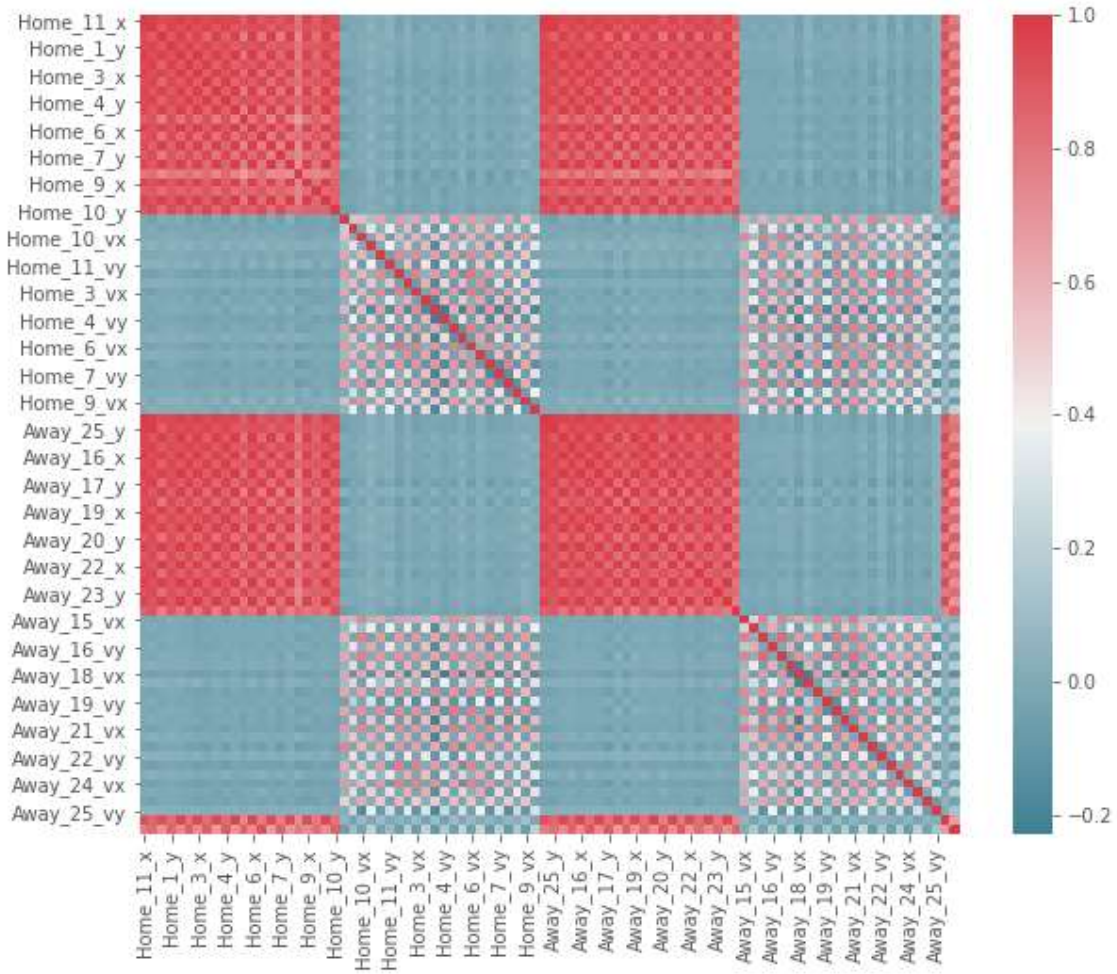

Figure 2-3 Correlation matrix

Now getting back to the other problem for the previous idea, and it is that each player has his own unique behavior on the pitch, and each player covers different areas in the pitch and has different tasks than other players. Figure 2-3 shows how different players behaves in the same period. In result our model should consider all the players positions and velocities including the teammates (in this case the other robots) and the unique behavior for each player.

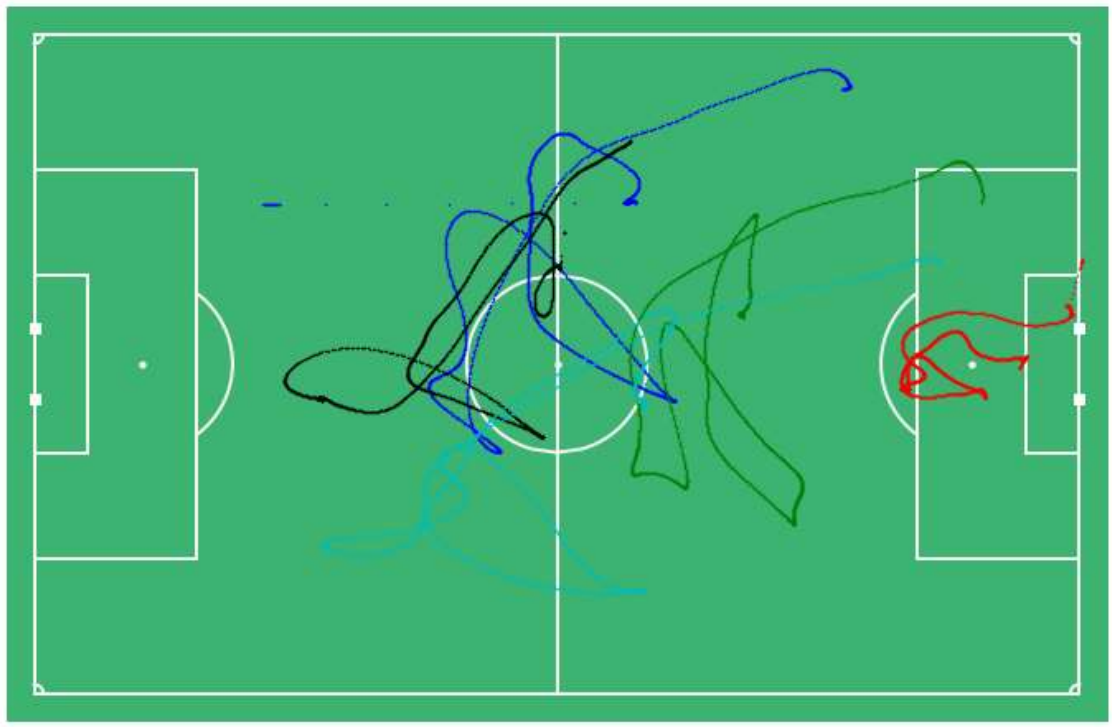

Figure 2-4 Different players movements in the same time period

Now what about the output? We con predict the next position or we con predict the velocity, the prediction of the next position is not very meaningful, because you want the player to move and the movement is difference between two positions, so predicting one position is not actually a movement. Instead, we need to predict velocity, which gives the direction and speed of movement.

Finally, the optimal solution is to build 22 models, each model predicts one component of the velocity vector for one player from the positions and velocities of all players including the position of the player himself. Thus, each model has 88 inputs which is the current positions and velocities for all players and one output which is the x or y component of the velocity for the player.

Other concern is that the same player may behaves differently over the match. Figure 2-4 shows the movements of one player in different periods. Even though the movements are not the same but there is some sort of pattern that the model should be able to capture.

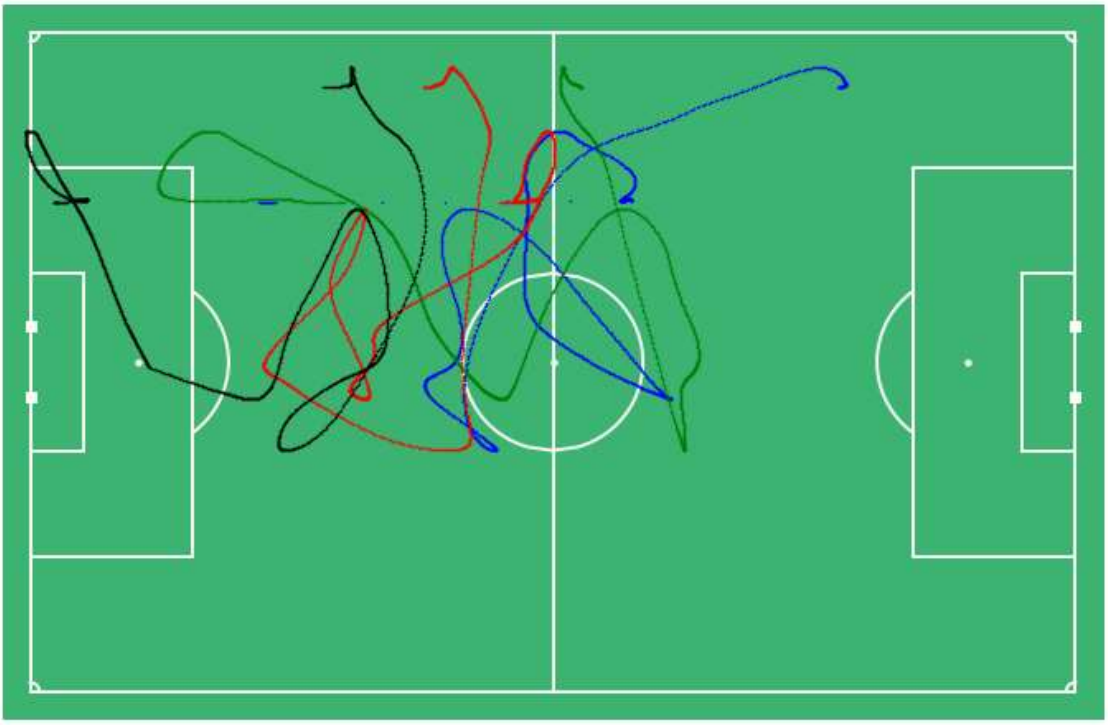

Figure 2-5 Position tracking for the same player in different time periods

2.3 Initial Model: Linear Regression

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 positions and velocities. the models represented by the equation below:

$$y_1 = a_{1,0} + a_{1,1}\,px1 + a_{1,2}\,py1 + \cdots + a_{1,44}\,py22 + a_{1,45}vx1 + a_{1,45}vy1 + \cdots + a_{1,88}\,vy22$$

$$y_2 = a_{2,0} + a_{2,1}\,px1 + a_{2,2}\,py1 + \cdots + a_{2,44}\,py22 + a_{2,45}vx1 + a_{2,45}vy1 + \cdots + a_{2,88}\,vy22$$

.

.

.

$$y_{22} = a_{22,0} + a_{22,1}\,px1 + a_{22,2}\,py1 + \cdots + a_{22,44}\,py22 + a_{22,45}vx1 + a_{22,45}vy1 + \cdots + a_{22,88}\,vy22$$

The parameters of the models were trained by OLS (ordinary least square) function from statsmodels package in python. After we split the data into 75% training set and 25% test set, we then calculate the R squared for the model and it was in the range from 0.53 to 0.81 for all the models. The overall accuracy is average accuracy for all 22 models on the test set

Table 2-2 Linear regression accuracy and computation time

The overall accuracy on test set	Time needed for one prediction
62.8 %	0.03 sec

Even though the linear model has a poor performance, but it gives us an indication that the output can be predicted from the input.

2.4 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.

We build the model using tensorflow package in python. We have tried many structures until we get the final model. We have used adam optimizer and ReLU activation function in hidden units.

Table 2-3 Neural Network Architecture

No. of layer	No. of nodes	No. of parameters
Input layer	88	None
First hidden	120	10680
Second hidden	80	9680
Third hidden	60	4860
Fourth hidden	30	1830
Fifth hidden	10	310
Output layer	2	22

The total number of parameters is the network was 27,382 parameters.

Table 2-4 Neural network performance

The overall accuracy on test set	Time needed for one prediction
79.3 %	0.17 sec

The NN gave us better accuracy but can we go further?

2.5 Final model: Recurrent Neural Network

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

We build the model using tensorflow package is python after reshaping the data to contain the last second of playing which is 25 observations.

The model still in training.

2.6 Constrains

There are some problems we may face during the design of the robots, and we might modify the prediction model or maybe then we need more complex model.

Sampling rate

In the original data there are an observation every 0.04 seconds this makes the data very accurate and we maybe cannot take reading for position every 0.04 seconds and that maybe affect our prediction accuracy.

Players switching

Sometimes, teams tend to switch players between left and right, and as the linear model has a static factor for each feature, this might cause some confusion for the model.

Chapter 3 DESIGN OPTIONS

3.1 Identifying the locations

to identify the location of each robot and player there is various ways to achieve that. we have many options to choose from, and they vary in cost and sufficient. The location of the players and the robots will allow us to have feedback for our controller.

GPS method

One of the options is using GPS sensors on each player and robot. then read each sensor for the location of the player or robot.

Figure 7GPS module

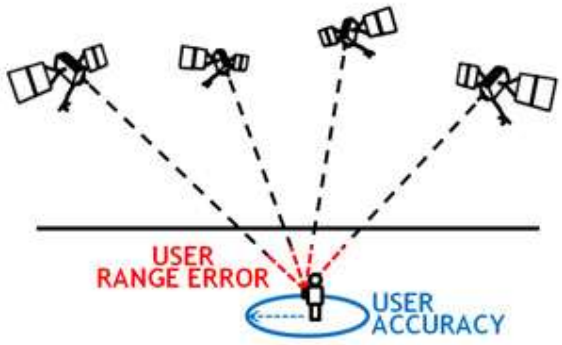

Figure 6:GPS accurcy

But the GPS sensors has many disadvantages. The first disadvantage is the accuracy of the sensors. GPS sensors do not have high accuracy.

The highest accuracy that GPS sensor could have is more than 1 meter squared (from our search of GPS sensors online). This is very low accuracy for our system since the players and the robots will be in same field which is not the whole field but the part where the players going to train.

Another disadvantage of the GPS sensors is the cost of the sensors. as we mentioned before the accuracy of the GPS sensors is low and to get GPS sensor that have 1 meter accuracy would be very expensive to get, and we need twenty-two sensor to put on each player and robot, so it is not cost efficient to use GPS sensors in our project.

Ultra-sonic and laser method

Another choice for identifying the location is to use ultra-sonic sensor or laser distance sensor. by using these sensors on the robot to measure the distance from the edge of the field to the robot. the one robot will have two sensors one for the x-position and one for the y-position.

Figure 9: ultra-sonic module

Figure 8: laser distance module

The advantages of using these sensors are also few but the main advantages are that they are easy to use and cheaper than the GPS sensors.

But it has a lot of disadvantages when using it in our project. The first disadvantage is that we have multiple robots and players on the same field, so the robots will interfere with each other's sensor's readings causing to have wrong position information.

That occur due to the possibility of two robots to be at the same x-location but different y-location (in line with each other) or the opposite case where the robots be at the same y-location but different x-location. This problem could happen while they are moving and could happen with other robots and players.

Another disadvantage is these sensors are hard to mount at the robots and the players. the sensors must always face the same direction all the time, any change in the sensor's direction will give wrong information.

we can solve this by using robots that never change their body direction, but they change the wheels direction. we cannot solve this problem with the human players. in addition to all these disadvantages, mounting these sensors on human players will affect their training and it will not be comfortable to wear.

shaft encoder method

we also found a mechanical way to measure the distance by using two rotary encoders. one on the x-axis and one on the y-axis. it is like the old computer mouses. The robot should have a reference point to be able to determine the location. so, all the robots should enter the field from the same point.

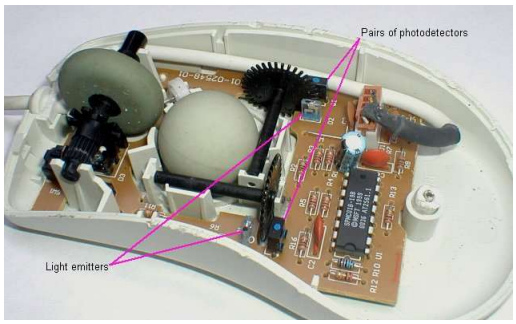

Figure 11: old mouse that uses 2 shaft encoders

Figure 10: shaft encoder module

The disadvantages of this method are close to the previous ultra-sonic and laser method. they must face the same direction all time .An addition problem is the reference point. that means the robots cannot enter the field at one time, they should do sequentially which is time consuming. and if the robot falls it needs to go back to the reference point.

this method only works on the robots and cannot be used on the human players.

Camera method

After more search about the best way to identify the location of the players and the robots we found that is 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 13 : camera held on the ceiling

Figure 12: camera held by a drone

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

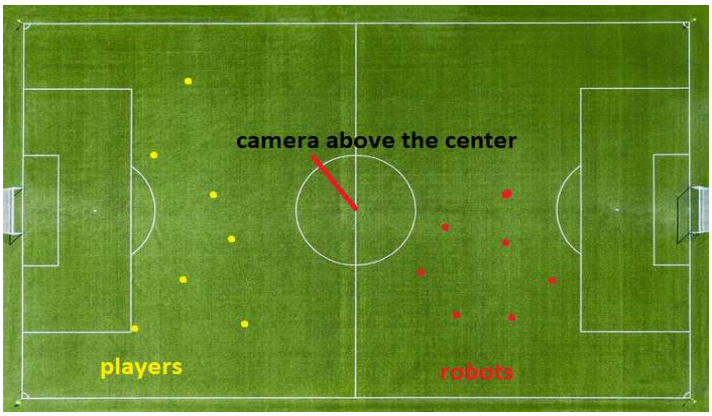

Figure 14: the video the camera captures

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 15 : the camera module we used

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.

3.2 Mechanical Design and Modeling of an Omni-directional RoboCup Player

Introduction

Players are designed for forward movement with ability in all directions, and the goalkeeper is designed to move along with the ability in all directions equipped with a local vision camera. We will explain the details of the mechanical design process of the player, the dynamics of kinematics of the final design.

Design

The robot had to be agile but could not be pushed by other teams on the field. The robot should also be able to "kick" the ball relatively hard, but we will try it, we used a DC-geared motor, which gives us a maximum speed equal to 125 rpm equivalent to 13 m/s and that speed we need, we used the overall direction wheel, Arduino, motors controller, a 12-volt 4000 battery what rechargeable lithium-ion battery with charger.

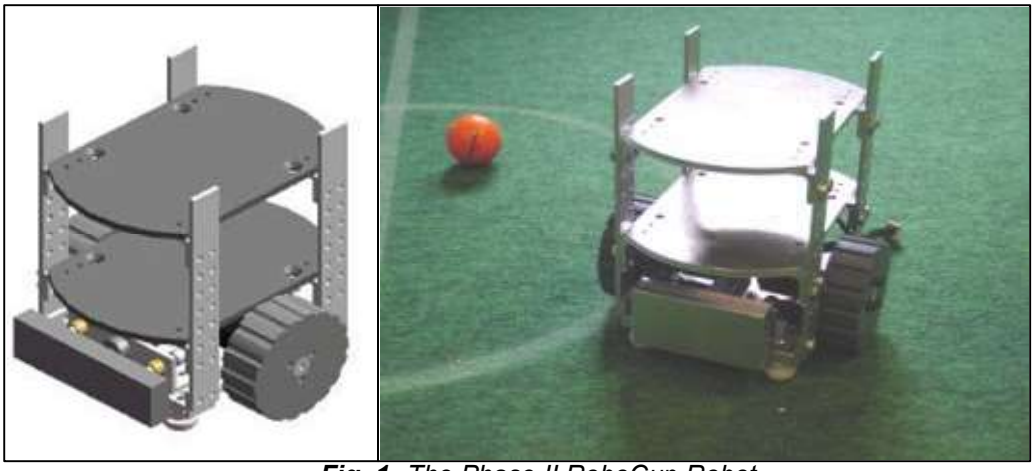

Fig. 1. *The Phase II RoboCup Robot.*

Fig. 2. *The Phase III RoboCup Robot*

Fig. 3. *The Transwheel™, used on the Ohio University player*

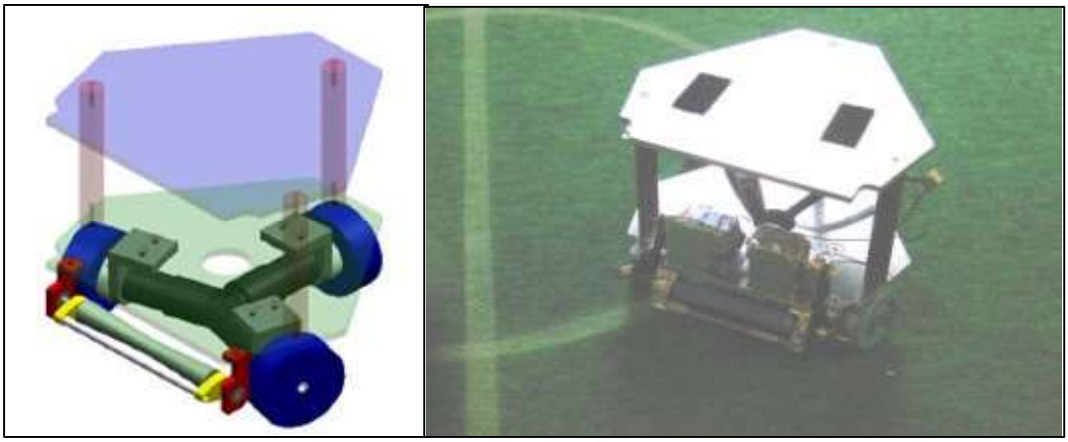

Fig. 4. *The Phase IIIB RoboCup Robot.*

EDUCATIONAL LESSON

- **FORWARD =** 2 motors running in the forward direction, 1 stationary
- **BACKWARD =** 2 motors running in the backward direction, 1 stationary
- **RIGHT OR LEFT** = 2 motors running in the same direction, 1 in opposite

Kinematics of the Phase IV RoboCup Robot

The two front wheels are presented by a points 1 & 2 and are offset from the y_m by fixed angle δ and the moving frame located is the center of gravity of the robot $[x_m, y_m]^T$ the arrows in the figure below indicates the velocity vector generated by the wheel.

By using a simple calculation for the Jacobian matrix and having the velocity generated by each wheel:

δ + Φ → wheel 1

δ - Φ → wheel 2

90 - Φ → wheel 3

The velocity vector = RΘ· i → wheel i.

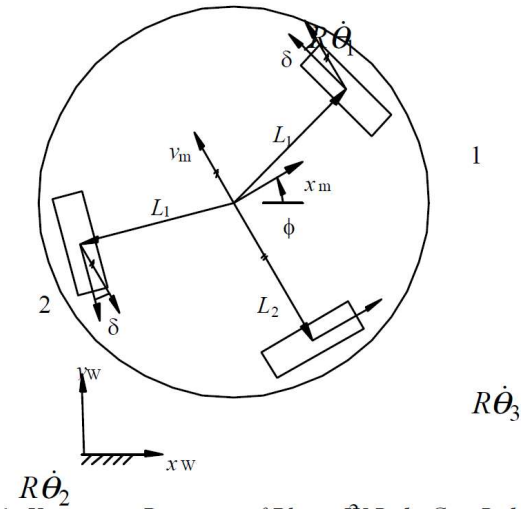

Fig. 6. *Kinematic Diagram of Phase IV RoboCup Robot.* 3

$$R\dot{\theta}_1 = -\sin(\delta + \Phi)\,\dot{x_w} + \cos(\delta + \Phi)\,\dot{y_w} + L_1\dot{\Phi}$$

$$R\dot{\theta}_2 = -\sin(\delta - \Phi)\,\dot{x_w} - \cos(\delta - \Phi)\,\dot{y_w} + L_1\dot{\Phi}$$

$$R\dot{\theta}_3 = \cos(\Phi)\,\dot{x_w} + \sin(\Phi)\,\dot{y_w} + L_2\dot{\Phi} \qquad \text{equ (1), (2), (3)}$$

L_1 → distance from the center of gravity of the robot to the center of the two front wheels (1 and 2) along a radial path.
L_2 → distance from the center of gravity of the robot to the center of the rear wheel (3) along a radial path.
$[x_w, y_w]$ → fixed world frame
Φ → orientation of the robot with respect to the fixed world frame
R → is the radius of the wheels
θ_i → the angular position of each wheel
δ → wheel orientation with respect to the mobile robot frame. For the Ohio University design, this value is 15°

$$\begin{Bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{Bmatrix} = \frac{1}{R}\begin{bmatrix}-\sin(\delta + \Phi) & \cos(\delta + \Phi) & L_1\\ -\sin(\delta - \Phi) & -\cos(\delta - \Phi) & L_1\\ \cos(\Phi)\,\dot{x_w} & \sin(\Phi) & L_2\end{bmatrix}\begin{bmatrix}\dot{x_w}\\ \dot{y_w}\\ \dot{\Phi}\end{bmatrix} \qquad \text{equ (4)}$$

Dynamics

The derivation began with Newton's Second Law:

$\mathrm{M}\ddot{x_w} = F_X$

$\mathrm{M}\ddot{y_w} = F_y$

$I_v\ddot{\Phi} = M_I$ equ (5), (6), (7)

M → mass of the mobile robot

$I_v \rightarrow$ moment of inertia of the mobile robot

F_x & F_y → Cartesian forces acting upon the robot with respect to the world frame

M_I → moment acting upon the center of gravity of the mobile robot.

After transformation, the following equations are revealed.

$\mathrm{M}(\ddot{x_w} - \dot{y_m}\dot{\Phi}) = fx$

$\mathrm{M}(\ddot{y_w} + \dot{x_m}\dot{\Phi}) = f_y$ *equ (8), (9)*

f_x & f_y → Cartesian forces acting upon the mobile robot with respect to the world frame in terms of the mobile robot frame
x_m and y_m → Cartesian position with respect to the world frame in terms of the mobile robot frame.

f_x, f_y, and M_I are given by:

$f_x = -\sin(\delta)\, D_1 - \sin(\delta)\, D_2 + D_3$
$f_y = \cos(\delta)\, D_1 - \cos(\delta)\, D_2$ equ (10), (11), (12)
$M_1 = (D_1 + D_2)L_1 + D_3L_2$

D_i → driving force due to each wheel assembly.

The driving system dynamic model for each wheel is assumed by the equ. below:

$I_w\ddot{\theta_i} + c\dot{\theta_i} = ku_i - RD_i$ equ (13)

i = 1,2,3.

$I_w \rightarrow$ moment of inertia of the wheel assemblies.

$q_i \rightarrow$ angular position of each wheel.

$c \rightarrow$ viscous friction factor of the wheel assembly.

$k \rightarrow$ driving gain factor.

$u_i \rightarrow$ driving input torque.

$R \rightarrow$ radius of the wheels.

The inverse kinematic equations for the mobile robot with respect to the mobile robot frame:

$$\begin{Bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{Bmatrix} = \frac{1}{R} \begin{bmatrix} -\sin(\delta) & -\cos(\delta) & L_1 \\ -\sin(\delta) & \cos(\delta) & L_1 \\ 1 & 0 & L_2 \end{bmatrix} \begin{bmatrix} \dot{x_w} \\ \dot{y_w} \\ \dot{\Phi} \end{bmatrix} \quad \text{equ (14)}$$

Using equation (14) and its derivatives and plugging this into equation (13) and solving for D_i calculates the driving force due to each wheel. Substituting these values into equations (10)-(12) and substituting these into equations (7)-(9) allows the equations of motion to be represented in the standard format.

$$[P]\{\ddot{X_m}\} + \{N(X_m, \dot{X_m})\} = [A]\{U\} \quad \text{equ (15)}$$

$$P = \begin{bmatrix} \frac{MR^2+2I_w sin^2(\delta)+I_w}{MR^2} & 0 & \frac{-2I_w\, sin(\delta) I_w + I_w I_2}{MR^2} \\ 0 & \frac{MR^2+2I_w cos^2(\delta)}{MR^2} & 0 \\ \frac{-2L_1 I_w\, sin(\delta)+L_2 I_w}{I_v R^2} & 0 & \frac{I_v R^2 + I_w {L_2}^2 + 2 I_w {L_1}^2}{I_v R^2} \end{bmatrix} \quad \text{equ (16)}$$

$$N = \begin{bmatrix} \frac{(2c - c\, \cos(2\delta)\dot{x_m}}{MR^2} + \frac{(cL_2 - 2c\, sin(\delta))L_1\dot{\Phi}}{MR^2} - \dot{y_m}\dot{\Phi} \\ \frac{2c\, cos^2(\delta)\dot{y_m}}{MR^2} + \dot{x_m}\dot{\Phi} \\ \frac{(-2L_1 c\, sin(\delta)+L_2 c)\dot{x_m}}{I_v R^2} + \frac{(2c{L_1}^2 - c\, {L_2}^2)\dot{\Phi}}{I_v R^2} \end{bmatrix} \quad \text{equ (17)}$$

$$A = \begin{bmatrix} \frac{-sin(\delta)k}{MR} & \frac{-si\ (\delta)k}{MR} & \frac{k}{MR} \\ \frac{cos(\delta)k}{MR} & \frac{-cos(\delta)k}{MR} & 0 \\ \frac{L_1 k}{I_v R} & \frac{L_1 k}{I_v R} & \frac{L_2 k}{I_v R} \end{bmatrix} \quad \text{equ (18)}$$

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.

Appendices