2009

STRATEGY, ROBOT SOCCER IN FIRA MIROSOT CATEGORY

Kushtrim Dragusha
University for Business and Technology - UBT

University for Business and Technology
Faculty of Computer Science and Engineering

STRATEGY, ROBOT SOCCER IN FIRA MIROSOT CATEGORY

A Bachelor Thesis submitted for the degree of
"Bachelor of Science in Management and Mechatronic"
at the University for Business and Technology

UBT

supervised by

o.Univ.Prof.Dipl.-Ing.Dr.Dr.h.c.mult. Peter Kopacek

Kushtrim Dragusha

2006070106/B

Prishtinë 2009 Signature

Content:

ABSTRACT

Robot Soccer strategy is main part of robot soccer system, where strategy makes briefing of the entire system. Where, using different functions and programming hardware features in the way of batter usage of system, should improve the soccer game of robots. Functions combined between each other to make so called strategy where it is called main part of entire system. Because of this beautiful impact on system, any way entire system is just to make strategy working, and for sure devices on system will have also their impact on strategy behaviour. Thesis mainly has to deal with the way of functionality of Robot Soccer strategy. Also here will be described the differences between the functions and strategy. Consist an explanation, about the structure of the entire software functionality, hardware cooperation with software and their features together. Due to these, will continue describing the old strategy and new one including them functions, also them difference. How the function will have impact on robot behaviour and strategy efficiency using same hardware (system). Also the different combination of different functions, in case to full fill the strategy, how are effecting them on final goal. In other hand the goal is to make new functions and full fill strategy logic of the game, ending with a conclusion on this work that is done and it will suggest new steps forward on future.

1 INTRODUCTION

Robot Soccer is games that involve different sides of engineering where them should work properly and make possible the football game of robots. Starting from mechanics where entire robot body is mechanics, then going to electric engineering where robots should have batteries, wires, battery charges and stuff like that.

Other field that is involved on is computer science. This field makes brain of the robot where takes responsibility on programming the intelligence of a robot. This intelligence makes robot play, but never mind all fields have their impact n game not just programming one. Each movement depends on selecting solution from every field that are involve on developing of a robot, where they will shows on his efficiency and his results on playground.

The best field that fits on this is Mechatronic Engineering, which includes different fields of engineering's: Mechanics, Electrics, Computer Science and Control Engineering. This makes possible to work as one in fields of Robotics and technology sides.

2 SYSTEM DECRIPTIONS

When it is mention “System Description” it means that here will be a described every detailed of Robot Soccer system, in term of hardware and software. How they are working and collaborating together, shows as one system.

Robot soccer system is working like: from digital camera is taking the picture and using computer is processing it in case to be able computer to make decision on next steps, so here we have in one way direction of communication between computer using radio transmitter to mobile robots that moves on playground. In other term this is called also MAS which means “Multi Agent System-MAS”.

2.1-Hardware description

Hardware of robot soccer system is made up from mobile robots, digital camera, computing centre and transmitting part.

Mobile robot include in it’s self two drivers (motors), two wheels, body, microcontroller, cover with colours that are used for detection of robots and identifying robot ID, robots are carrying a battery with them self and also other parts that are using for robot security on crashing and so.

Short description about hardware devices functions:

- **Robot drives** are a DC driver that works in low voltage and they have very big moment, where support the robot to have very fast acceleration, three times faster than “Formula 1”, also with very fast velocity, where in FIRA is allowed just 3.5 m/s that are really high speed.

- **Wheels** are mechanic devices, they are made from steel and covered by rubber, rubber is very important in this case due to robot efficiency.

- **Body** of robot has to deal with two functions, first to keep every robot device together, second to be stable and strong fighting in game field.

- **Microcontroller**, makes communication between computer and robot using module of radio frequency, translate the instruction that came's from computer and makes orders to robot devices (robot drivers).

- **Colours covers** needed in case of power unit to know the robot number, robot angle and robot orientation.

- **Digital Camera** makes recording of robots and ball position, in other hand capture the soccer playground and sent the images to power computing centre in this case PC computer.

- **Power computing station** or brain of entire system, from here system takes decisions/orders and execute them. Power computing centre using image processing unit makes detection of robot and ball position on playground and makes decision for robot movement depending on its functions and strategy.

- **Transmitting station** sends orders to robots.

2.2 -Software description

Software structure is made up from three main parts: image processing communication part and the last one is decision make mechanism (system brain).

There for if there is any problem communicating between any of these parts the system will not work properly in terms of very efficiency system to handle the opposite team.

Short description of each part of software, their goals:

- **Image processing**, to make system one decision it should have same information from playground, as where are robots and ball (robot and ball position). This is made up from image processing unit. How it works? Taking the picture from camera and processing each frame, in one frame is taking information of the colours on the field. Before of the system running we are preparing colour identification of each colour what is it goal. Than the system

during the game will identification these colours and bring to us this information. Goal of this part is to make able to identify robots from each other.

- **Decision making mechanism**, when system gets information from image processing unit, it is analysing the information and using different functions and its strategy makes decision for specify robot. This part is making the robot game that we are seeing in ground and purpose of it is to make as much as goals it can to win the game.

- **Communication unit** sends to robots the decision that is taken from Decision make mechanism.

3 STRATEGY AND FUNCTIONS DESCRIPTION

Defining the strategy and function generally is little bit hard, in case of, that both of them sometimes has same goal to full fill. Maybe taking some particular problems to describe the entire idea of them goal will be better idea.

3.1 -Definition of strategy and function

Strategy and function, as mention before they are so similarly on definition, but they have general differences where these differences are in general goals, that means functions has more focused goal, what they should do and strategy means so general, on the game level, how the game will go on, how we will play on different situation and stuff like that, where function how I am going to shoot the goal, how will execute the Free Kick or problems of this level.

Due to description before functions are caring limit of the duties and they fulfil the strategy goal. In this case if we are looking them in hierarchy prism function will be under strategy or child of strategy.

3.2 -Role of function on strategy

Function as a child of strategy should make different jobs to fulfil the game purpose and depending to this duty they can fulfil in different ways, depending to the way of programmer did his job.

As in real cases you can done a job in different ways, but in some cases there are so important how you are doing something and in robot soccer case there are so important how you are making any decision on field so the function should be so strict, depending to FIRA rules also in your goal that you are making to yourself (your team).

What the team is going to do, to have impressive game or efficient one. So in some cases you have to decided, if you want to have smart game or you want to have good game to win the competition depending on hardware possibilities.

3.3 -Different strategies and them goals

Strategies can be different just to improve better game in different situation and to make possible wining of the game. This case is so often when you are planning the losing threats and trying to avoid them.

Some strategies that can be are: defending one, offensive defending, defensive attacking, offensive attacking and so one. And for reasons there you are able to make different combination between them and you can create some of the special movement where you think that you need such of these thinks.

It is crucial to make it possible function that for special cases you have different strategy that makes you in advantage in front of opponent team and gives you more security during the game.

4 OVERVIEW OF EXISTING STRATEGY

4.1 -Structure of existing strategy

In existing strategy robots plays a simple game where they should react in programming logic that they has, not so intelligent and so simple function where decisions are depending from some few different situations.

In this case there are included some predictions and also there is programmed a very good algorithm to avoid any situation out of control.

Out of control means to make possible the wining of the game in championship and to have good position during competition. In European Championship 2008 of Robot Soccer was the first time of our team that we competed and the entire strategy was developed only for championship.

4.2 -State of functions

Comparing to purpose of the team, functions are not so intelligent and they does not represent any high artificial intelligent and sophisticated algorithm but they are very efficient one.

4.2.1 –Function used

Function that was used, are so general and makes duty for more than one position on playground. This makes it less flexible and it cannot use in more cases exact for what it created.

Some of these functions are representing below:

void Deffenders (int whichrobot,int whichrobot1, int whichrobot2)

void DeffendersKO2 (int Goaly, int Deffender,int MidDefenderR,int MidDefenderL)

void DeffendersKO3 (int Goaly, int Deffender,int MidDefenderR,int MidDefenderL)

As it seems in function parameters two of the functions makes possible to play with midfielder robot, defenders and goalie. This makes that called general function and this is not so efficiency and it cannot be used in more cases except one case that provide with the game of these parts of playground.

void Middfielders(int whichrobot)

void Attacers(int whichrobot)

4.2.2 –Analysis and description

Functions Attackers and Middfielders are more less the same one that are describe on next chapter with some changes on some of the functionality and their algorithm but in general they are same. So here will be shown just three of other function that are: Deffenders, DeffendersK02 and DeffendersKO3, which are completed removed from new strategy.

```
void Deffenders ( int whichrobot,int whichrobot1, int whichrobot2)
{
        double sv = 0;
        double sa = 0;
        char st[20];
        char st1[20];

        DPoint Ball;

        Ball.init();
        Ball.x = dat.ball.PPos(0.04,whichrobot).x;
        Ball.y = dat.ball.PPos(0.04,whichrobot).y;

        DPoint from,to,add_from,add_to,from1,to1,add1_from,add1_to;
        DPoint from2,to2,add_from2,add_to2;

        from.x = 0.4;
```

```
from.y = 0.03;

to.x = 0.7;
to.y = 1.78;

from1.x = 0.2;
from1.y = 0.50;

to1.x = 0.35;
to1.y = 1.30;

from2.x = 0.06;
from2.y = 0.65;

to2.x = 0.08;
to2.y = 1.10;

double dx = fabs(dat.hr[whichrobot1].x - Ball.x);
double dy = fabs(dat.hr[whichrobot1].y - Ball.y);

double dx1 = fabs(dat.hr[whichrobot].x - Ball.x);
double dy1 = fabs(dat.hr[whichrobot].y - Ball.y);

double dx2 = fabs(dat.hr[whichrobot2].x - Ball.x);
double dy2 = fabs(dat.hr[whichrobot2].y - Ball.y);

//Centre Defender Robot condition of superkick area
// HT3
if (Ball.x > 1.2)
{
        from.x = 0.4;
        from.y = 0.45;
```

```
        to.x = 0.7;
        to.y = 1.35;

        parallelMove(whichrobot,from,to,0.08);
}
if (Ball.x <= 0.4 && (Ball.y < 0.5 || Ball.y > 1.3))
{
                SuperKick(whichrobot,false);
}
else
{
        if (Ball.x > 0.4)
        {
                if (dx1<0.2 && dy1<0.2)
                {
                        SuperKick(whichrobot,false);
                }
                else
                {
                        parallelMove(whichrobot,from,to,0.08);
                }
        }
        else
        {
                parallelMove(whichrobot,from,to,0.08);
        }
}
//Defender robot condition for area of
if (Ball.x>0.9)
{
        from1.x = 0.2;
```

```
        from1.y = 0.75;

        to1.x = 0.35;
        to1.y = 1.05;

        parallelMove(whichrobot1,from1,to1,0.08);
}

else if (dx<=0.10 && dy<=0.10)
        {
                if ((dat.hr[whichrobot1].y < Ball.y))
                {
                        if (dat.hr[whichrobot1].x < Ball.x)
                        {
                         ParadeCircle(whichrobot1, 45, 0, 45);
                        }
                        else
                        {
                         ParadeCircle(whichrobot1, -45, 0, 45);
                        }
                }
                else if ((dat.hr[whichrobot1].y > Ball.y))
                {
                        if (dat.hr[whichrobot1].x < Ball.x)
                        {
                         ParadeCircle(whichrobot1, -45, 0, 45);
                        }
                        else
                        {
                         ParadeCircle(whichrobot1, 45, 0, 45);
                        }
                }
```

```
        }
else
{
        parallelMove(whichrobot1,from1,to1,0.08);
}
//Conditionn for area of goalie
if (Ball.x>0.75)
{
        from2.x = 0.06;
        from2.y = 0.85;

        to2.x = 0.1;
        to2.y = 0.95;
        parallelMove(whichrobot2,from2,to2,0.07);
} else if( Ball.x<0.1){
        from2.x = 0.03;
        from2.y = 0.5;

        to2.x = 0.1;
        to2.y = 1.3;
        parallelMove(whichrobot2,from2,to2,0.07);
}
if (dx2<=0.09 && dy2<=0.09)
{
        if ((dat.hr[whichrobot2].y < Ball.y)){
                if (dat.hr[whichrobot2].x < Ball.x)
                {
                        ParadeCircle(whichrobot2, 45, 0, 45);
                }
                else
                {
                        ParadeCircle(whichrobot2, -45, 0, 45);
```

```
                }
        }
        else if ((dat.hr[whichrobot2].y > Ball.y))
        {
                if (dat.hr[whichrobot2].x < Ball.x)
                {
                ParadeCircle(whichrobot2, -45, 0, 45);
                }
                else
                {
                ParadeCircle(whichrobot2, 45, 0, 45);
                }
        }
    }
    else
    {
        parallelMove(whichrobot2,from2,to2,0.07);
        if (Ball.x<0.15 && ((Ball.y>=0.5 && Ball.y<0.65) ||
(Ball.y>=1.15 && Ball.y<1.3)))
        {
            from2.x = 0.06;
            to2.x = 0.1;
            parallelMove(whichrobot2,from2,to2,0.08);
        }
    }
}
```

As it is shown Deffender function is made for three robots and it cannot be used for example to command just one robot and in this case it work but for one structure of the game so in terms of constructing entire strategy there will shown some gaps of flexibility and efficiency.

As well as Deffender function also the other two functions are nonflexible and non efficiency. But this case maybe is worst in terms that these two function

caring also for other part of the field like for example Goalie, Defenders and Midfielder, this makes possible to use just ones.

```
void DeffendersKO2 (int Goaly, int Deffender,int MidDefenderR,int
MidDefenderL)
{
        DPoint Ball;
        Ball.init();
        Ball.x = dat.ball.PPos(0.04,MidDefenderR).x;
        Ball.y = dat.ball.PPos(0.04,MidDefenderR).y;
        DPoint from,to,from1,to1,from2,to2,from3,to3;

        //Goaly
        from2.x = 0.06;
        from2.y = 0.65;
        to2.x = 0.08;
        to2.y = 1.10;

        //Deffender
        from1.x = 0.2;
        from1.y = 0.50;
        to1.x = 0.35;
        to1.y = 1.30;

        //MidDefenderR
        from.x = 0.4;
        from.y = 0.03;
        to.x = 0.7;
        to.y = 0.85;

        //MidDefenderL
        from3.x = 0.4;
        from3.y = 0.9;
```

```
to3.x = 0.7;
to3.y = 1.78;

double dx2 = fabs(dat.hr[Goaly].x - Ball.x);
double dy2 = fabs(dat.hr[Goaly].y - Ball.y);

double dx1 = fabs(dat.hr[Deffender].x - Ball.x);
double dy1 = fabs(dat.hr[Deffender].y - Ball.y);

double dx = fabs(dat.hr[MidDefenderR].x - Ball.x);
double dy = fabs(dat.hr[MidDefenderR].y - Ball.y);

double dx3 = fabs(dat.hr[MidDefenderL].x - Ball.x);
double dy3 = fabs(dat.hr[MidDefenderL].y - Ball.y);

//Centre defender Robot condition of superkick area
// HT3
if (Ball.x > 1.3)
{
        from.x = 0.4;
        from.y = 0.25;

        to.x = 0.7;
        to.y = 0.65;
        parallelMove(MidDefenderR,from,to,0.08);
}
if (Ball.x <= 0.4 && Ball.y < 0.5)
{
        SuperKick(MidDefenderR,false);
}
else
```

```
{
        if (Ball.x > 0.4)
        {
                if (dx<0.35 && dy<0.35)
                {
                        SuperKick(MidDefenderR,false);
                }
                else
                {
                        parallelMove(MidDefenderR,from,to,0.08);
                }
        }
        else
        {
                parallelMove(MidDefenderR,from,to,0.08);
        }
}
//Defender robot condition of superkick area -- HT2
if (Ball.x > 1.3)
{
        from3.x = 0.4;
        from3.y = 1.05;

        to3.x = 0.7;
        to3.y = 1.55;
        parallelMove(MidDefenderL,from3,to3,0.08);
}
if (Ball.x <= 0.4 && Ball.y > 1.3)
{
                SuperKick(MidDefenderL,false);
}
else
```

```
{
        if (Ball.x > 0.4)
        {
                if (dx3<0.35 && dy3<0.35)
                {
                        SuperKick(MidDefenderL,false);
                }
                else
                {
                        parallelMove(MidDefenderL,from3,to3,0.08);
                }
        }
        else
        {
                parallelMove(MidDefenderL,from3,to3,0.08);
        }
}
//Defender Robot condition of rotate area
if (Ball.x>0.9)
{
        from1.x = 0.2;
        from1.y = 0.75;

        to1.x = 0.35;
        to1.y = 1.05;
        parallelMove(Deffender,from1,to1,0.08);
}
else if (dx1<=0.10 && dy1<=0.10)
        {
                if ((dat.hr[Deffender].y < Ball.y))
                {
                        if (dat.hr[Deffender].x < Ball.x)
```

```
                        {
                        ParadeCircle(Deffender, 45, 0, 45);
                        }
                        else
                        {
                        ParadeCircle(Deffender, -45, 0, 45);
                        }
                }
                else if ((dat.hr[Deffender].y > Ball.y))
                {
                        if (dat.hr[Deffender].x < Ball.x)
                        {
                        ParadeCircle(Deffender, -45, 0, 45);
                        }
                        else
                        {
                         ParadeCircle(Deffender, 45, 0, 45);
                        }
                }
        }
else
{
        parallelMove(Deffender,from1,to1,0.08);
}
// Goalie Robot condition of rotate area
if (Ball.x>0.75)
{
        from2.x = 0.06;
        from2.y = 0.85;

        to2.x = 0.1;
        to2.y = 0.95;
```

```
        parallelMove(Goaly,from2,to2,0.07);
} else if( Ball.x<0.1){
        from2.x = 0.03;
        from2.y = 0.5;

        to2.x = 0.1;
        to2.y = 1.3;
        parallelMove(Goaly,from2,to2,0.07);
}
if (dx2<=0.09 && dy2<=0.09)
{
        if ((dat.hr[Goaly].y < Ball.y)){
                if (dat.hr[Goaly].x < Ball.x)
                {
                        ParadeCircle(Goaly, 45, 0, 45);
                }
                else
                {
                        ParadeCircle(Goaly, -45, 0, 45);
                }
        }
        else if ((dat.hr[Goaly].y > Ball.y))
        {
                        if (dat.hr[Goaly].x < Ball.x)
                        {
                                ParadeCircle(Goaly, -45, 0, 45);
                        }
                        else
                        {
                                ParadeCircle(Goaly, 45, 0, 45);
                        }
```

```
                }
        }
        else
        {
                parallelMove(Goaly,from2,to2,0.07);
                if (Ball.x<0.15 && ((Ball.y>=0.5 && Ball.y<0.65) ||
(Ball.y>=1.15 && Ball.y<1.3)))
                {
                        from2.x = 0.06;
                        to2.x = 0.1;
                        parallelMove(Goaly,from2,to2,0.08);
                }
        }
}
```

The structure of the next function is the same as DeffendersK02, but just includes some different functions, is using more or less intelligent and prediction, also the structure of robot calling, which robots go to commanding function.

5 NEW STRATEGY, IT'S FUNCTIONS AND GOAL

New strategy is really based on new functions and makes different behaviour of mobile robots on playground. Is based more on specific function where they are making their goal better on creating new face of robots game.

Defining behaviour of the robot game and making prediction which can bean next moves depending on ball moves, also in future development hopefully will be done prediction of the opponent robots.

Here are shown some of the really development where makes the game much better on terms of smart game, makes more intelligent robots, or make the system look like. This duty is not so easy, but it has to full fill in order to compete.

New strategy wants to achieve this goal and makes smart game where robot team should work proper and collaborating on them purposes.

Except this achievement there tend to increasing also the prediction of the movement, ball movement for the first step and in next step to make possible prediction of the behaviour (Strategy) of the opposite team. That means during the opposite team game, our team will makes decision for them behaviour. This will be as final solution of development on our team.

5.1 –Overview of new function

In new strategy there are some functions that are used before, especially on European Champion Ship 2008 (Euroby2008) but main of them, are new one that allows new behaviour of system and makes it better solution for the robot soccer game, including this hardware and basic software solution.

Functions description:

Goalie Function:

First function has to deal with Golly where it should follow the ball and save the goal in case of any attack.

```
Void GoulieAdvance (int robot)
{
        Dpoint Ball;

        Ball.init();
        Ball.x = dat.ball.Ppos(0.04,robot).x;
        Ball.y = dat.ball.Ppos(0.04,robot).y;

        from.x = 0.06;
        from.y = 0.65;

        to.x = 0.08;
        to.y = 1.10;
```

This part makes the field of golly robot operation, where the robot should react and defend. In case to make the more attract game as real he shouldn't go in attack field except in specially cases. So this will make better security for our team and more stable on competing in championship.

```
        Double dx = fabs(dat.hr[robot].x – Ball.x);
        double dy = fabs(dat.hr[robot].y – Ball.y);
        //First condition of following the ball from goalie
        if (Ball.x>0.75)
        {
                from2.x = 0.06;
                from2.y = 0.85;

                to2.x = 0.1;
                to2.y = 0.95;
                parallelMove(robot,from,to,0.07);
        } else if( Ball.x<0.25){
```

```
        from2.x = 0.03;
        from2.y = 0.5;

        to2.x = 0.1;
        to2.y = 1.3;
        parallelMove(robot,from,to,0.07);
}
if (dx<=0.09 && dy<=0.09)
{
        if ((dat.hr[robot].y < Ball.y)){
                if (dat.hr[robot].x < Ball.x)
                {
                        ParadeCircle(robot, 45, 0, 45);
                }
                else
                {
                        ParadeCircle(robot, -45, 0, 45);
                }
        }
        else if ((dat.hr[robot].y > Ball.y))
        {
                if (dat.hr[robot].x < Ball.x)
                {
                        ParadeCircle(robot, -45, 0, 45);
                }
                else
                {
                        ParadeCircle(robot, 45, 0, 45);
                }
        }
}
else
```

```
	{
		parallelMove(robot,from,to,0.07);
		if (Ball.x<0.15&&((Ball.y>=0.5 && Ball.y<0.65)||(Ball.y>=1.15 &&
Ball.y<1.3)))
		{
			from2.x = 0.06;
			to2.x = 0.1;
			parallelMove(robot,from,to,0.08);
		}
	}
}
```

Second part of the code shows us where the goalie will react in field. How is the playground divided for our goalie? In this case our goalie will have some condition where he will react, first one is, when the ball is in opposite side of the playground he will have small part of following the ball in case that it is not necessary to follow in entire his area.

The second condition is when the ball passes the middle part of the playground he will start to follow the ball and save entire his area that is given to him just before. The main idea is to have strict strategy to save the goal as better as it can, so better defending strategy.

Defenders Functions:

To this part is development some of the functions and this makes a lot of choices in developing strategy, as we want to organise the defender strategy.

What does it mean? How it is possible to make such choices?

There are developed three different types of defenders starting from centre defender, right one and the last one third one.

Why is developed like this?

Because, it will help to make different way to organised defending. If you want to have three defenders, or you want another way of organising of the game you are free to make it, because you can choose it. For more detail we will have describing each of them.

Centre Defender:

Main function of Centre Defender is to defend the attacks in our goal and make easier the goalie life, in terms of his duty. This function will make robot to stay in front of goalie eyes, and it have small area where he will operate, but in some cases he will also attack and probably shot a goal.

These cases are just when our attackers are not doing their duty well and for any reason they are off from detection also they may have problems with communication to full fill entire function, here will come up Centre Defender to make his shots.

In normal cases centre defender will pass to any robot or kick off the ball from area of our goal that is his main function.

```
//Function for centre defender
void CentreDeffender (int Deffender)
{
	Dpoint Ball;
	Ball.init();
	Ball.x = dat.ball.Ppos(0.04,MidDefenderR).x;
	Ball.y = dat.ball.Ppos(0.04,MidDefenderR).y;
	Dpoint from,to;

	//Deffender
	from.x = 0.2;
	from.y = 0.50;
	to.x = 0.35;
	to.y = 1.30;
```

As in the goalie function also here first part of the function describes the area of the robot where he can operate. Where the second part is his duty what he will make on situations that are set just for him. But always he has to coordinate his movement with other team members.

```
Double dx = fabs(dat.hr[Deffender].x – Ball.x);
double dy = fabs(dat.hr[Deffender].y – Ball.y);

//Deffender Robot area conditions and rotate one
if (Ball.x>0.9)
{
        from.x = 0.2;
        from.y = 0.75;

        to.x = 0.35;
        to.y = 1.05;
        parallelMove(Deffender,from,to,0.08);
}
else if (dx<=0.10 && dy<=0.10)
        {
                if ((dat.hr[Deffender].y < Ball.y))
                {
                        if (dat.hr[Deffender].x < Ball.x)
                        {
                                ParadeCircle(Deffender, 45, 0, 45);
                        }
                        else
                        {
                                ParadeCircle(Deffender, -45, 0, 45);
                        }
                }
```

```
                else if ((dat.hr[Deffender].y > Ball.y))
                {
                        if (dat.hr[Deffender].x < Ball.x)
                        {
                                ParadeCircle(Deffender, -45, 0, 45);
                        }
                        else
                        {
                                ParadeCircle(Deffender, 45, 0, 45);
                        }
                }
        }
else
{
        parallelMove(Deffender,from,to,0.08);
}
}
```

Right Defender:

The Right defender will work specially in right side of the playground and done his purpose. He should defend attackers that come from the right side of the playground, also in specially cases to make some attacks in his side.

```
//Function for right defender
void RightDeffender (int Deffender)
{
        Dpoint Ball;
        Ball.init();
        Ball.x = dat.ball.Ppos(0.04,MidDefenderR).x;
        Ball.y = dat.ball.Ppos(0.04,MidDefenderR).y;
        Dpoint from,to;
```

```
//MidDefenderR
from.x = 0.4;
from.y = 0.03;

to.x = 0.7;
to.y = 0.85;
```

Like in function before first part always describe the area of that special robot and the rest of it describe the functionality depending on different situation.

```
Double dx = fabs(dat.hr[Deffender].x – Ball.x);
double dy = fabs(dat.hr[Deffender].y – Ball.y);

//Deffender Robot area conditions
if (Ball.x>0.9)
{
        from.x = 0.4;
        from.y = 0.25;

        to.x = 0.7;
        to.y = 0.65;
        parallelMove(Deffender,from,to,0.08);
}
else if (dx<=0.10 && dy<=0.10)
        {
                if ((dat.hr[Deffender].y < Ball.y))
                {
                        if (dat.hr[Deffender].x < Ball.x)
                        {
                                ParadeCircle(Deffender, 45, 0, 45);
                        }
                        else
```

```
                        {
                                ParadeCircle(Deffender, -45, 0, 45);
                        }
                }
                else if ((dat.hr[Deffender].y > Ball.y))
                {
                        if (dat.hr[Deffender].x < Ball.x)
                        {
                                ParadeCircle(Deffender, -45, 0, 45);
                        }
                        else
                        {
                                ParadeCircle(Deffender, 45, 0, 45);
                        }
                }
        }
        else
        {
                parallelMove(Deffender,from,to,0.08);
        }
}
```

Left Defender:

This function, more/ less is to full fill the coverage of the playground with defenders. In other hand he has same functionality like Right and centre defender but he has just to save left side.

```
//Function for left defender
void LeftDeffender (int Deffender)
{
        Dpoint Ball;
        Ball.init();
```

```
Ball.x = dat.ball.Ppos(0.04,MidDefenderR).x;
Ball.y = dat.ball.Ppos(0.04,MidDefenderR).y;
Dpoint from,to;

//MidDefenderL
from.x = 0.4;
from.y = 0.9;

to.x = 0.7;
to.y = 1.78;

double dx = fabs(dat.hr[Deffender].x – Ball.x);
double dy = fabs(dat.hr[Deffender].y – Ball.y);

//Left defender area condition and rotate
if (Ball.x>0.9)
{
        from.x = 0.4;
        from.y = 1.05;

        to.x = 0.7;
        to.y = 1.55;
        parallelMove(Deffender,from,to,0.08);
}
else if (dx<=0.10 && dy<=0.10)
        {
                if ((dat.hr[Deffender].y < Ball.y))
                {
                        if (dat.hr[Deffender].x < Ball.x)
                        {
                                ParadeCircle(Deffender, 45, 0, 45);
                        }
```

```
                    else
                    {
                        ParadeCircle(Deffender, -45, 0, 45);
                    }
                }
                else if ((dat.hr[Deffender].y > Ball.y))
                {
                    if (dat.hr[Deffender].x < Ball.x)
                    {
                        ParadeCircle(Deffender, -45, 0, 45);
                    }
                    else
                    {
                        ParadeCircle(Deffender, 45, 0, 45);
                    }
                }
            }
        else
        {
            parallelMove(Deffender,from,to,0.08);
        }
}
```

Midfielder Function:

This is especially case where this function allows the robot to be attacker also in other situation defender, where is completed depended from situation, position of our team robots also the ball position.

This function in some ways is the key function, depending to, and its behaviour more/less will describe entire team game.

What is the meaning?

Actually if we want to make good game and for sure to win it, then we have to be so careful on this area. Completed understandable that there is involve and has big impact also the technical devices, but comparing to this hardware solution that we have the defending game will be more realistic and strong in front of the opposite team.

```
//Function for middfilder
void Middfielder( int robot)
{
	Dpoint Ball;
	Ball.init();
	Ball.x = dat.ball.Ppos(0.08,robot).x;
	Ball.y = dat.ball.Ppos(0.08,robot).y;
	Dpoint from,to;

	from.x = 0.9;
	from.y = 0.03;

	to.x = 1.2;
	to.y = 1.78;

	double dx = fabs(dat.hr[robot].x – Ball.x);
	double dy = fabs(dat.hr[robot].y – Ball.y);

	if (Ball.x > 1.8)
	{
		from.x = 0.9;
		from.y = 0.25;

		to.x = 1.2;
		to.y = 1.55;
		parallelMove(robot,from,to,0.08);
	}
```

```
        if (Ball.x > 1.2)
        {
                 if (dx<0.35 && dy<0.35)
                {
                        SuperKick(robot,false);
                }
                else
                {
                        parallelMove(robot,from,to,0.08);
                }
        }
   else
        {
                parallelMove(robot,from,to,0.08);
        }

        if (Ball.x<0.7)
        {
                SuperPosition(robot,1.1,0.3,0.);
        }
}
```

Attacker Function:

This function in our case will be used just in some cases because of hardware solution that we have.

Reasons are that first of all, vision sensor is to slow and also the communication centre is not very efficient to make possible fast reaction on the playground. This is main reason that this function is not so useful in our case but in specially cases this will be so useful where it will be used and never mind it will be so efficient.

```
//Function of Attacer
```

```
void Attacer( int attacer)
{
	Dpoint Ball;
	Ball.init();
	Ball.x = dat.ball.Ppos(0.08,attacer).x;
	Ball.y = dat.ball.Ppos(0.08,attacer).y;
	Dpoint from,to,add_from,add_to;

	from.x = 1.3;
	from.y = 0.03;

	to.x = 1.75;
	to.y = 1.78;

	double dx = fabs(dat.hr[attacer].x – Ball.x);
	double dy = fabs(dat.hr[attacer].y – Ball.y);

	if (Ball.x > 1.2)
	{
		if (dx<0.3 && dy<0.3)
		{
			SuperKick(attacer,false);
		}
		else
		{
			parallelMove(attacer,from,to,0.08);
		}
	}
   else
	{
		parallelMove(attacer,from,to,0.08);
	}
```

```
        if (Ball.x<0.8)
        {       if (Ball.y < 0.8)
                {
                        SuperPosition(attacer,1.4,1.2,0.);
                }
        else if (Ball.y > 0.9)
                {
                        SuperPosition(attacer,1.4,0.6,0.);
                }
        }
}
```

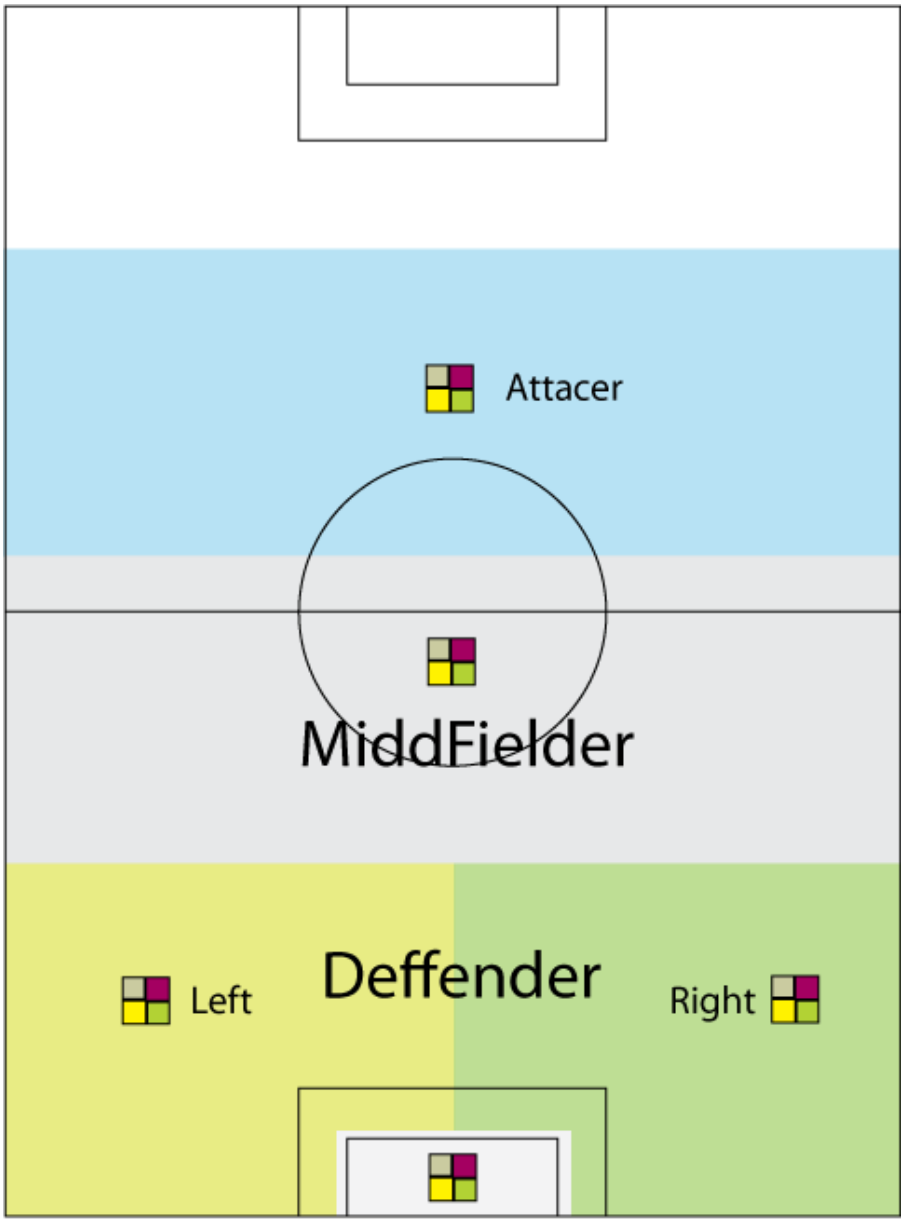

Fig. 1 Robot Soccer strategy robots positions

Other Functions:

Such functions support to complete the entire functionality of the robot game, and make possible to represent real game of football. In this case w have presented two of other functions that makes first one rotating a specific robot to another robot and the other one rotate robot to Ball .

The first function can be use for deferent purposes and fulfilling the goal on strategy. For example it can be used to pass the ball, to prepare robot for any attack or defending etc.

```
//Function to rotate the robot to other robot
void TurnRobotToRobot(int robot1,int robot2)
{
	Dpoint rRobot1, rRobot2;

	rRobot1.init();
	rRobot2.init();

	rRobot1.x = dat.hr[robot1].x;
	rRobot1.y = dat.hr[robot1].y;
	rRobot2.x = dat.hr[robot2].x;
	rRobot2.y = dat.hr[robot2].y;

	double angle = calcAngle(rRobot1, rRobot2);
	RotateRobot(robot2, angle);
}
```

In other hand seecond function also can be used for some purposes like defending ball and stealing one.

```
//Function to rotate the robot to the ball
void TurnRobToBall(int robot,Dpoint Ball)
{
	Dpoint rRobot ;
	rRobot.init();
	rRobot.x = dat.hr[robot].x;
	rRobot.y = dat.hr[robot].y;

	double angle = calcAngle(Ball, rRobot);
	RotateRobot(robot, angle);
	}
```

5.2 –Description their goals

Functions developments, are mainly focused on hardware opportunities and takes the best features from them. In this case we are trying to make best solution based on hardware and also based on vision sensors which is to slow comparing to the opponent teams.

Goalie Function:

On goalie function is build to handle most of the cases where our goalie is in danger from the opponent robots and he should make the best think to avoid these situations.

When the ball is in opponent side of the playground he will have just a part of his area of following the ball and if the ball is coming near and in our side of playground, goalie will follow the ball in entire his area where he should to protect the goal and to be able to react so fast in this side following the ball entire time will help to react faster but should be careful in cases where the direction of the ball will change fast and near of the goal.

```
Void GoulieAdvance (int robot)
 {
        .
        .
        .
        double dx = fabs(dat.hr[robot].x – Ball.x);
        double dy = fabs(dat.hr[robot].y – Ball.y);
        //First condition of ball followed from goal kipper
        if (Ball.x>0.75)
        {
                from2.x = 0.06;
                from2.y = 0.85;
                to2.x = 0.1;
                to2.y = 0.95;
```

```
        parallelMove(robot,from,to,0.07);
} else if( Ball.x<0.25){

        from2.x = 0.03;
        from2.y = 0.5;

        to2.x = 0.1;
        to2.y = 1.3;
        parallelMove(robot,from,to,0.07);
}
if (dx<=0.09 && dy<=0.09)
{
        if ((dat.hr[robot].y < Ball.y)){
                if (dat.hr[robot].x < Ball.x)
                {
                        ParadeCircle(robot, 45, 0, 45);
                }
                else
                {
                        ParadeCircle(robot, -45, 0, 45);
                }
        }
        else if ((dat.hr[robot].y > Ball.y))
        {
                if (dat.hr[robot].x < Ball.x)
                {
                        ParadeCircle(robot, -45, 0, 45);
                }
                else
                {
                        ParadeCircle(robot, 45, 0, 45);
                }
```

```
                }
        }
        else
        {
                parallelMove(robot,from,to,0.07);
                if (Ball.x<0.15&&((Ball.y>=0.5 && Ball.y<0.65)||(Ball.y>=1.15 &&
Ball.y<1.3)))
                {
                        from2.x = 0.06;
                        to2.x = 0.1;
                        parallelMove(robot,from,to,0.08);
                }
        }
}
```

Defenders Functions:

Defenders functions are development on three functions, where each function has his part of the field and they will react on his part of his field also in special cases will make the job of the others robots (attackers for example).

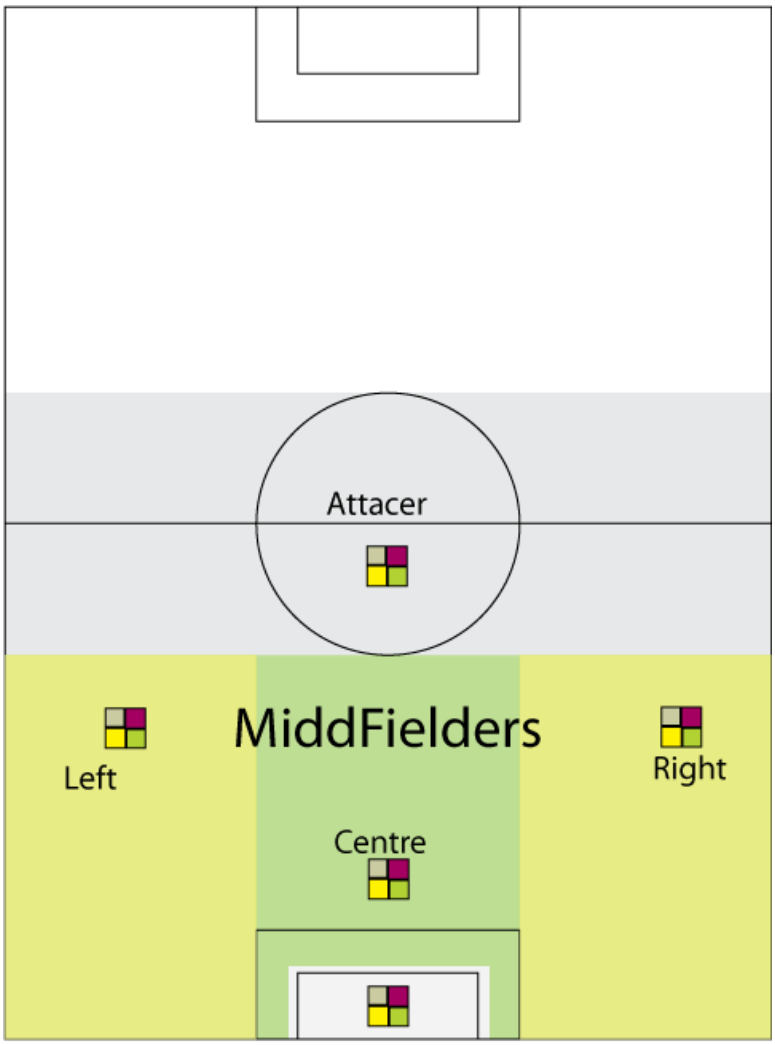

Fig. 2 Next strategy that is usable

Centre Defender:

Centre defender goal is to protect the centre defending part of the playground. Assisting the goalie on his purpose and trying to make better defender of the goal. In this issue the centre defender will be main supporter on having different part of functions where he should play. To play in defending mean while to play midfielder and attacker this is little bit harder to specify the entire these specific function on single robot.

```
Void CentreDeffender (int Deffender)
{
        .
```

```
.
.
double dx = fabs(dat.hr[Deffender].x – Ball.x);
double dy = fabs(dat.hr[Deffender].y – Ball.y);

//Defender Robot area condition and rotation
if (Ball.x>0.9)
{
        from.x = 0.2;
        from.y = 0.75;

        to.x = 0.35;
        to.y = 1.05;
        parallelMove(Deffender,from,to,0.08);
}
else if (dx<=0.10 && dy<=0.10)
        {
                if ((dat.hr[Deffender].y < Ball.y))
                {
                        if (dat.hr[Deffender].x < Ball.x)
                        {
                                ParadeCircle(Deffender, 45, 0, 45);
                        }
                        else
                        {
                                ParadeCircle(Deffender, -45, 0, 45);
                        }
                }
                else if ((dat.hr[Deffender].y > Ball.y))
                {
                        if (dat.hr[Deffender].x < Ball.x)
                        {
```

```
                        ParadeCircle(Deffender, -45, 0, 45);
                    }
                    else
                    {
                        ParadeCircle(Deffender, 45, 0, 45);
                    }
                }
            }
        else
        {
            parallelMove(Deffender,from,to,0.08);
        }
        }
```

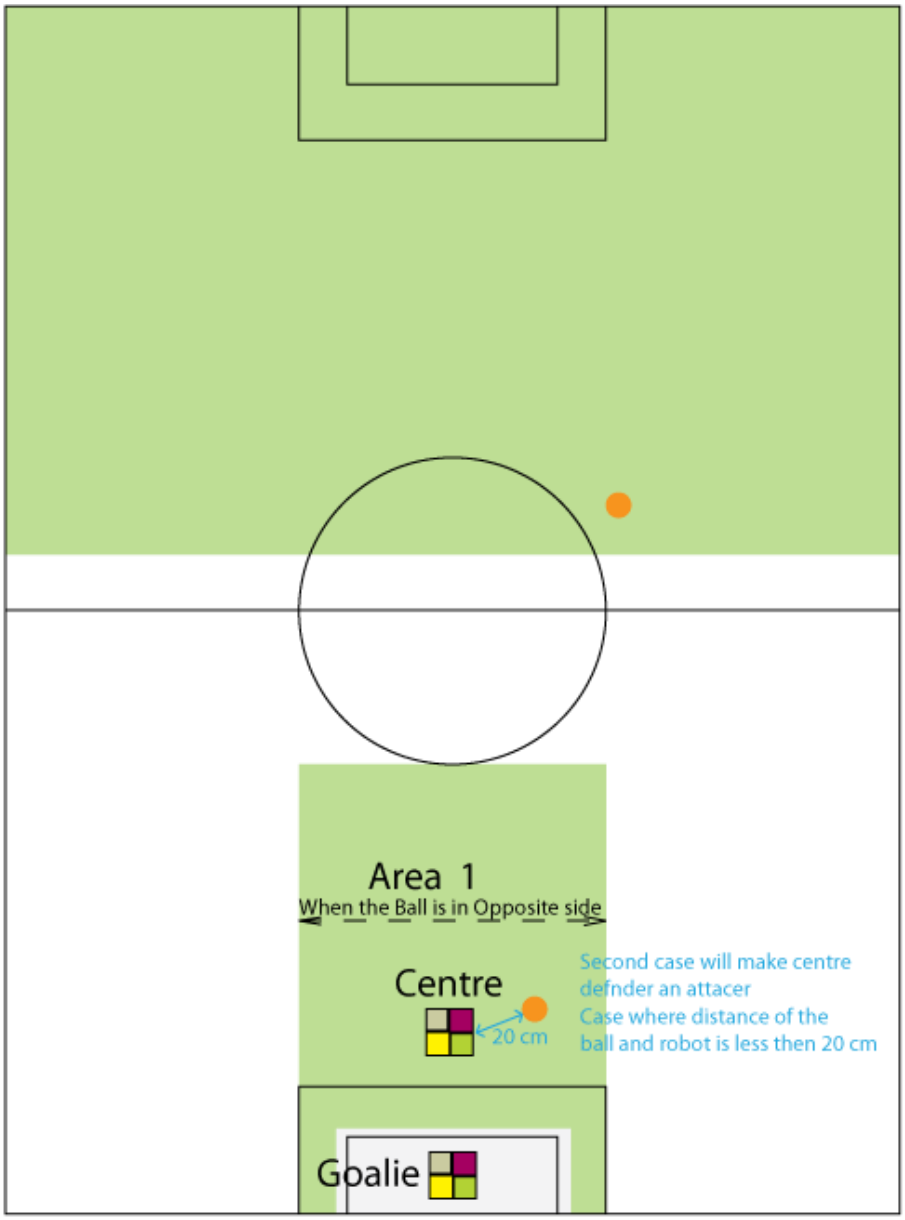

Fig. 3 Centre Defender Robot Area

Right Defender:

Right defender purpose is to support the goalie on defending the goal form opponent attackers and makes better infrastructure to attack on field for our attackers where we can use maybe also some dangers strategy of function on attacking where we know that our goal is safety. Its goal is mainly defending but in some cases to attack fast and immediately come back to defending position.

Void RightDeffender (int Deffender)

```
{
        .
        .
        .
        double dx = fabs(dat.hr[Deffender].x – Ball.x);
        double dy = fabs(dat.hr[Deffender].y – Ball.y);

        //Defender Robot area condition and rotation
        if (Ball.x>0.9)
        {
                from.x = 0.4;
                from.y = 0.25;

                to.x = 0.7;
                to.y = 0.65;
                parallelMove(Deffender,from,to,0.08);
        }
        else if (dx<=0.10 && dy<=0.10)
                {
                        if ((dat.hr[Deffender].y < Ball.y))
                        {
                                if (dat.hr[Deffender].x < Ball.x)
                                {
                                        ParadeCircle(Deffender, 45, 0, 45);
                                }
                                else
                                {
                                        ParadeCircle(Deffender, -45, 0, 45);
                                }
                        }
                        else if ((dat.hr[Deffender].y > Ball.y))
                        {
```

```
                        if (dat.hr[Deffender].x < Ball.x)
                        {
                                ParadeCircle(Deffender, -45, 0, 45);
                        }
                        else
                        {
                                ParadeCircle(Deffender, 45, 0, 45);
                        }
                }
        }
        else
        {
                parallelMove(Deffender,from,to,0.08);
        }
}
```

Left Defender:

Like right defender also the left defender has the same purposes where he mainly defends our goal and some special cases he will be main attacker. This is position where strategy that will be played will be defending one and to have possibilities to shoot goals robots should attack. When they should attack, they should have good decision mechanism in case not to make possible that our goal to be in danger.

```
Void LeftDeffender (int Deffender)

{

.

.

.

        double dx = fabs(dat.hr[Deffender].x – Ball.x);
```

```
double dy = fabs(dat.hr[Deffender].y – Ball.y);
//Defender Robot area condition and rotation
if (Ball.x>0.9)
{
        from.x = 0.4;
        from.y = 1.05;
        to.x = 0.7;
        to.y = 1.55;
        parallelMove(Deffender,from,to,0.08);
}
else if (dx<=0.10 && dy<=0.10)
        {
                if ((dat.hr[Deffender].y < Ball.y))
                {
                        if (dat.hr[Deffender].x < Ball.x)
                        {
                        ParadeCircle(Deffender, 45, 0, 45);
                        }
                        else
                        {
                                ParadeCircle(Deffender, -45, 0, 45);
```

```
                    }
                }
                else if ((dat.hr[Deffender].y > Ball.y))
                {
                    if (dat.hr[Deffender].x < Ball.x)
                    {
                        ParadeCircle(Deffender, -45, 0, 45);
                    }
                    else
                    {
                        ParadeCircle(Deffender, 45, 0, 45);
                    }
                }
            }
        else
        {
            parallelMove(Deffender,from,to,0.08);
        }
}
```

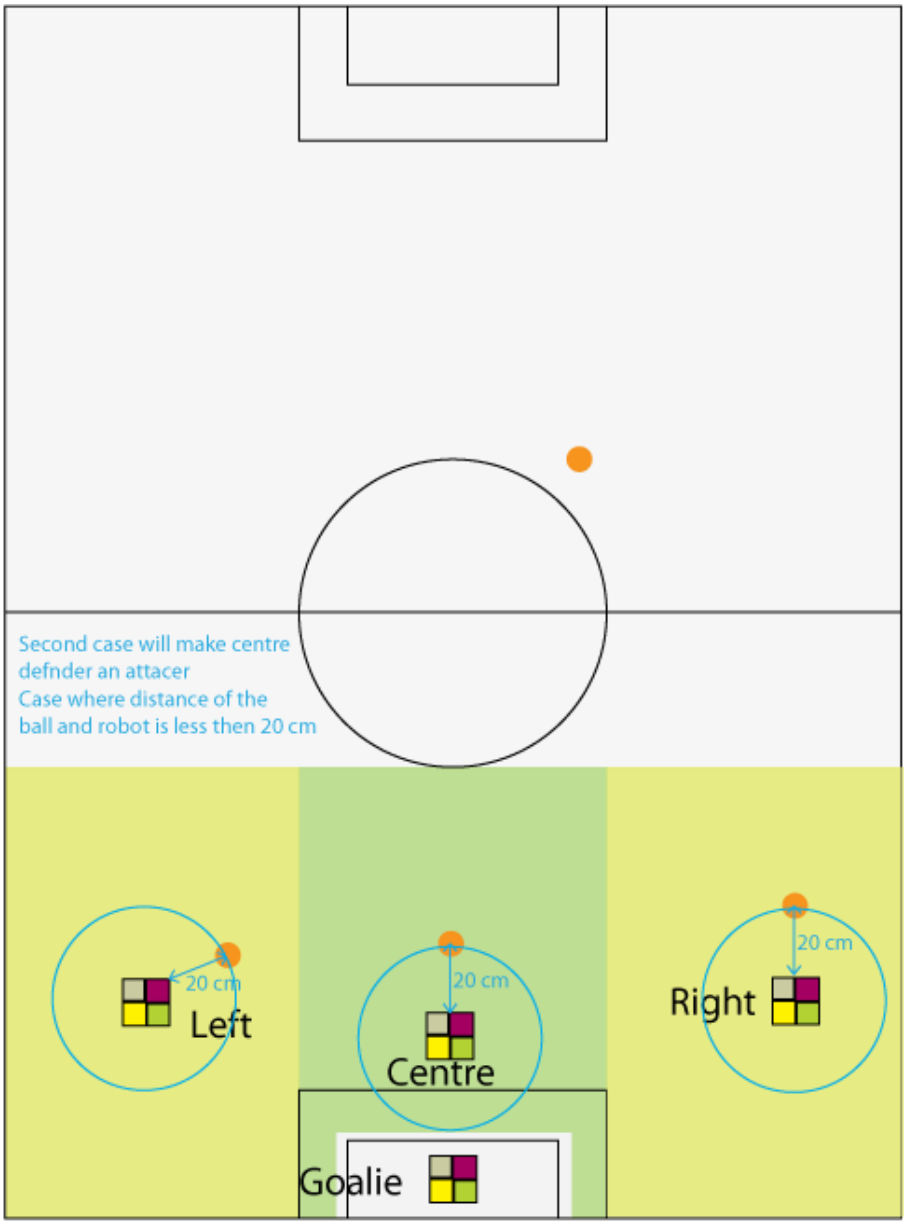

Fig. 4 Defenders Positions

Midfielder Function:

Midfielders in more cases are defensive attackers or offensive defenders. They make possible to handle starting of fast opponent attacks and start fast offensive for our team. This sometime gives us better position and makes possible shooting the goals. These functions are so flexible to avoid the gaps in cases when we are playing offensive strategy, also when we need some robots to be attackers and defenders. This is best solution to use like neutral position for one function or another one.

```
Void Middfielder( int robot)
{
        .
        .
        .
        double dx = fabs(dat.hr[robot].x – Ball.x);
        double dy = fabs(dat.hr[robot].y – Ball.y);

        if (Ball.x > 1.8)
        {
                from.x = 0.9;
                from.y = 0.25;

                to.x = 1.2;
                to.y = 1.55;
                parallelMove(robot,from,to,0.08);
        }
        if (Ball.x > 1.2)
        {
                 if (dx<0.35 && dy<0.35)
                {
                        SuperKick(robot,false);
                }
                else
                {
                        parallelMove(robot,from,to,0.08);
                }
        }
   else
        {
                parallelMove(robot,from,to,0.08);
        }
```

```
        if (Ball.x<0.7)
        {
                SuperPosition(robot,1.1,0.3,0.);
        }
}
```

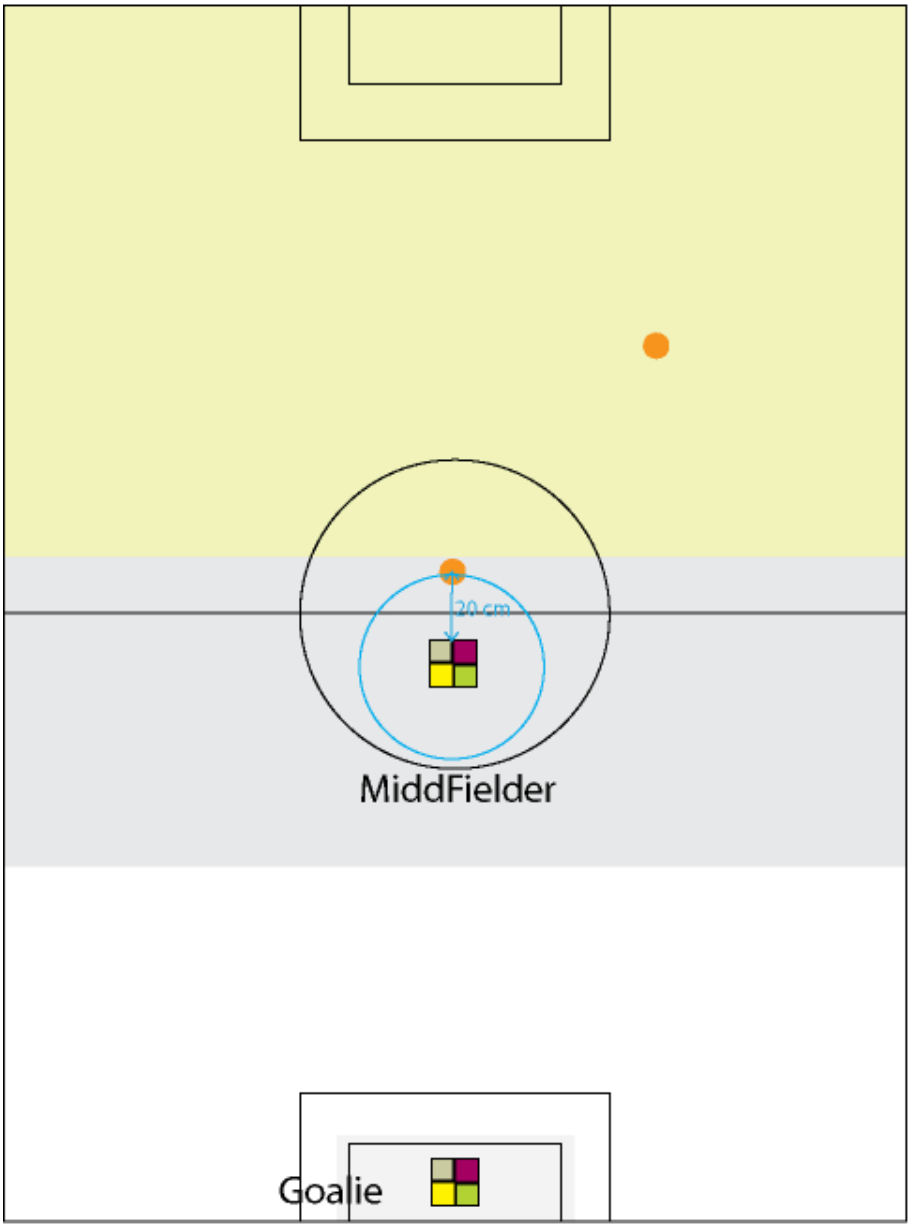

Fig. 5 Midfielder Function Area

Attacker Function:

Attacker function mainly is for attacking but sometimes he will be back to midfielder but never to defending position, he, should be ready anytime for attacking, never mind to prepare any offensive without supporting from defenders and midfielders. He should be specialist for attacks that is his entire job.

```
Void Attacer( int attacer)
{
        .
        .
        .
        double dx = fabs(dat.hr[attacer].x – Ball.x);
        double dy = fabs(dat.hr[attacer].y – Ball.y);
        if (Ball.x > 1.2)
        {
                if (dx<0.3 && dy<0.3)
                {
                        SuperKick(attacer,false);
                }
                else
                {
                        parallelMove(attacer,from,to,0.08);
                }
        }
   else
        {
                parallelMove(attacer,from,to,0.08);
        }
        if (Ball.x<0.8)
        {       if (Ball.y < 0.8)
                {
                        SuperPosition(attacer,1.4,1.2,0.);
```

```
            }
        else if (Ball.y > 0.9)
            {
                SuperPosition(attacer,1.4,0.6,0.);
            }
        }
}
```

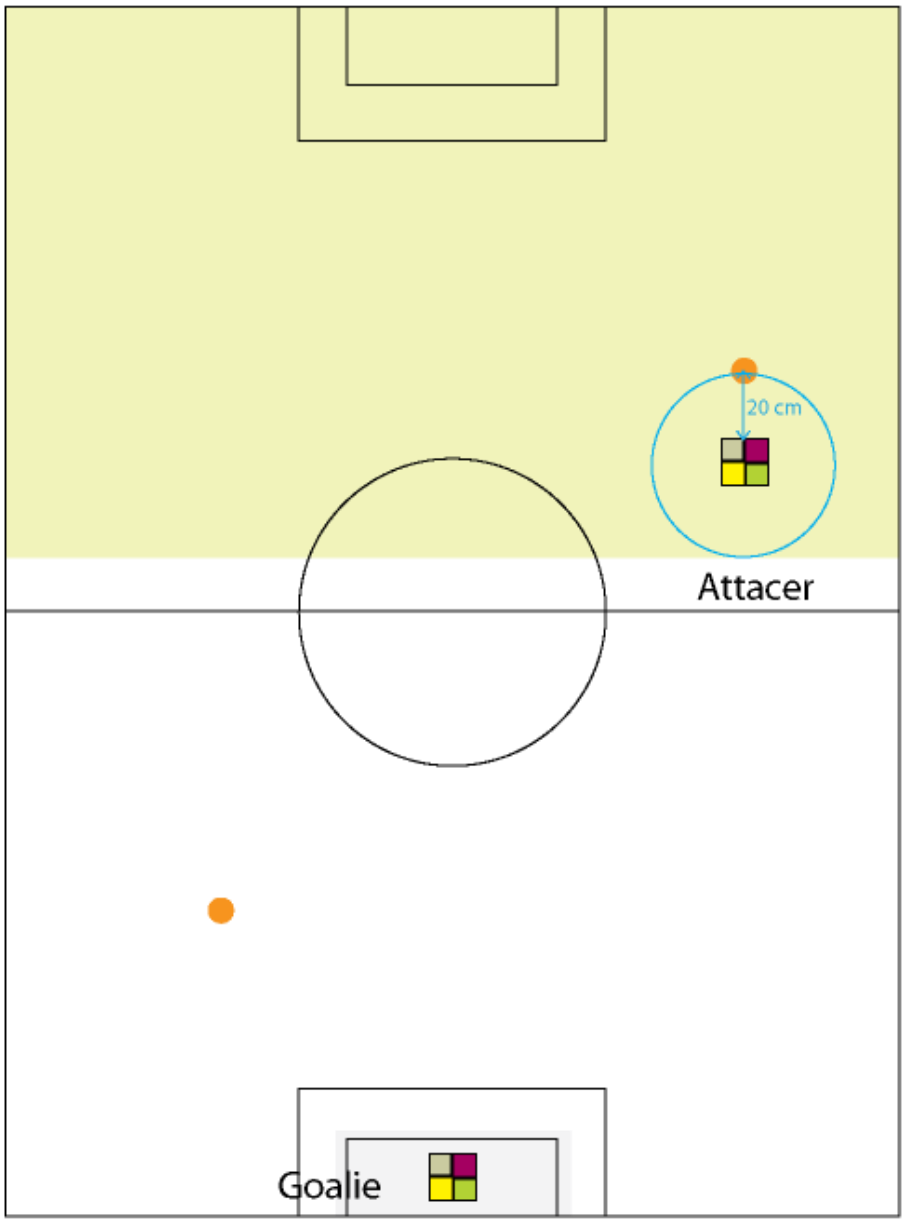

Fig. 6 Attacker Area Function

Other Functions:

Other function goal is to support the main function on their purposes. To represent these support function here are shown two of them that makes possible some calculation for any situation. These functions are used in deferent situation and they can support in many cases.

```
Void TurnRobotToRobot(int robot1,int robot2)
{
        Dpoint rRobot1, rRobot2;

        rRobot1.init();
        rRobot2.init();

        rRobot1.x = dat.hr[robot1].x;
        rRobot1.y = dat.hr[robot1].y;
        rRobot2.x = dat.hr[robot2].x;
        rRobot2.y = dat.hr[robot2].y;

        double angle = calcAngle(rRobot1, rRobot2);
        RotateRobot(robot2, angle);
}
void TurnRobToBall(int robot,Dpoint Ball)
{
        Dpoint rRobot ;
        rRobot.init();

        rRobot.x = dat.hr[robot].x;
        rRobot.y = dat.hr[robot].y;

        double angle = calcAngle(Ball, rRobot);
        RotateRobot(robot, angle);
        }
```

6 EVALUATION OF NEW STRATEGY

Differences between old and new strategy are mainly in the way of new access on constructed and efficiently, where new functions gives more opportunities on constructing new and new strategies and reacting in more of situations depending from some condition that we think that are necessary to react on these situations. We had win speed and safety during the game.

Instead of high intelligent algorithm there are installed algorithm that provide low and medium intelligent in order to provide efficiency due to hardware issue that we have. This is little problem in developing off algorithm where you always should think about hardware issue and stuff like that in case to succeeded with good results.

Old strategy was development more less to provide game and fulfil the requirements from FIRA, after the experiences in championship we had a possibilities to see entire way of developments and our gaps from opponent teams, where we was on this area, we was not so far from the other teams in strategy algorithm that we had provide but there is enough place to develop the system where new function are just a part of them.

New strategy provides very efficiency flexible function and makes possible to work on this infrastructure that is too old and should develop, update on new devices and with these updates to develop new functions that are more intelligent in case to have better position on this field and also to follow the developments on other institutions, running after them or telling opponents power of development.

Another advantage of new strategy is decision make mechanism that is more efficient and makes better decision, more opportunities for decision making also little more intelligent than older one.

7 CONCLUSION

New strategy tend to make more efficiently game and possibilities to use functions to make up different strategies on different ways, to have the best solution in right time, just in case where we need it.

7.1 -Used and new functions solution

Used function was created more hard programmed and they are so solid and really hard to play with in different ways to find the best solution and make one used for. Instead of new one they are more flexible and easy to provide different strategies and have more choices on situations that we can be during the game.

Also they have some similarities where we are talking about intelligent function, where depending on hardware solution that we have and the purpose of the team that we want to have positive result on competition they should be not so smart and slow down the decision maker mechanism. So in this area we have to so careful and make decision what we want to do.

7.2 -Old strategy, advantages and disadvantages

The old strategy was so efficiently when we are talking for relation between decision mechanism and smart (intelligent) strategy (robots), but the efficiently to use them for different purposes is too small and they are in some ways really just for one usage.

7.3 -New strategy

New strategy is focused also in intelligent strategy but also in elastic function where we can provide different behaviour in different situation where and how we need it.

This makes new strategy advantages and makes possibilities to improve more in thinking of different situations that will happen in field, the case to develop

intelligent strategies is depended more in hardware opportunities that are mention before also the level and technology of developing itself.

This makes difficult to have smart strategies without improving the hardware side. The costs on time in robot soccer game are so important and you have to think every think on this parameter in case to more efficiently, otherwise your robot game will be so stupid in front of opponent robots where they will provide speed and high level of decision make mechanism.

In more cases new strategy fulfil this condition and makes easy life where you can talk about next competition.

But if we are going to compare the opponent team with our situation they have experience more than ten years and for our team is second time that we will compete on championship and it is second year that we are working with these problems.

8 SUMARY AND OUTLOOK

Due to support that we had from team Austria in first championship we had succeeded with place four and for next championship we feel more safety, because we know with what we will do.

Our next competition will find us ready and more prepared for it. Also the steps of development of the team itself it is little good feeling to see it growing in professionalism prism and they work in robot soccer system.

First strategy and the second one just makes more strong decision that the team is growing and they have good perspective in future also they start good on this field.

REFERENCES

My references was not from much different sources but so much information in professionalism way.

These sources are:

- Austrian Team software basic source including image processing,
- Professor o.Univ.Prof.Dipl.-Ing.Dr.Dr.h.c.mult. Peter Kopacek lectures and supports on laboratory
- Professor Dr.Bernhard Putz lectures and supports on laboratory
- Professor Mr. Artan Dermaku Lectures and supports on laboratory