Abstract

RoboCup is an international scientific initiative with the goal to advance the state of the art of intelligent robots. When established in 1997, the original mission was to field a team of robots capable of winning against the human soccer World Cup champions by 2050.
This report covers the design and implementation of a new intelligent kicking strategy for defender. It consists of three parts, namely Field Division, Kicking Direction Strategy and Dynamic Kicking Velocity Strategy. In this summer semester, our group have successfully accomplished it both in simulation environment and on the real robot.

Contents

Chapter 1

Introduction

The Robot Soccer World Cup (RoboCup) is a worldwide robotics competition held annually, where the research goals concern cooperative multiple robot and multiple agent systems in dynamic adversarial environments(See figure 1.1). All robots in this league are fully autonomous. The original mission of it was to field a team of robots capable of winning against the human soccer World Cup champions by 2050. Furthermore, solutions from RoboCup can be also applied in general robotic topics. Our course is based on the RoboCup Standard Platform League. It is a soccer league where all teams participate using the same robot, the NAO robot from SoftBank Robotics. These robots play fully autonomously and each one takes decisions separately from the others, but they still have to play as a team by using communications. The teams play on a green field with white lines and goal posts, with no other landmarks, and the ball consists in a realistic white and black soccer one. These game characteristics generate a very challenging scenario, which allows improving the league every year.
The code part of our course is based on B-Human[1], which is a joint RoboCup team of the Universityat Bremen and the German Research Center for Artificial Intelligence(DFKI). The team was founded in 2006 as a team in the Humanoid League, but switched to participating in the Standard Platform League in 2009.

Figure 1.1: Robocup

1.1 Motivation

With the knowledge background of machine learning, dynamic control and computer vision, this course provides a platform for students to apply their theory into practical scenarios. Based on Nao robots, students can take a first look of the real robots world.
Meanwhile, teamwork is also practiced during the whole practicum. Students will learn how to communicate with each other and work together efficiently.
At last, with the final team representation students can enhance their presentation skills.

1.2 course goal

This course consists of two stage, namely basic introduction lecture and practice on real robot. The first stage lasts 1 month and students will learn together about the framework of B-Human CodeRelease and how to setup and calibration on a real robot.
The second part lasts nearly two month. All students are divided into two teams and each team find their own research field. With teamwork students will finally apply their knowledge learned from lectures and implement their improvement on the real robot.

1.3 Self Contributions

Based on original B-Human CodeRelease, our group found kicking strategy of our defender is too simple. With a lack of supporter, our defender works inefficiently.

The main goal of our group is to set an improvement of the kicking strategy. We will enhance defending area and create a new dynamic kicking strategies.
My work is code implementation of the whole strategy. At first I enlarge the defending area and divide it according to different attributes of the fields. Based on the divided field, My group mates will then find intelligent dynamic kicking strategies in mathematical form. At last I implement it in code form and realize it both in simulation environment and on real Robot.

Chapter 2

Field division

2.1 Motivation

In the real situation we have only three robots in one team, namely striker, keeper and defender. Since the lack of supporter, defender in original small defending area works inefficiently[2]. Enlarging the defending area and setting an intelligent kicking strategy could be a more wise choice to win the game. In the enlarged region, defender will not only play multiple roles, both defender and supporter, but also will contribute more to the whole game.

2.2 Theory Introduction

2.2.1 Field Enlargement

The first step of our multiple roles strategy is to change the border of the defending area. A suitable border should minimize conflict with the opponent defender. In this regard, we set it just the same as the opponent defending area border. See figure 2.1, the red line represents our new defending border. In this way, working area of our defender covers both the defending area and supporting area. Direct collision with the opponent defender will never happen. Meanwhile defener has the possibility to pass the ball much closer to the opponent goal.

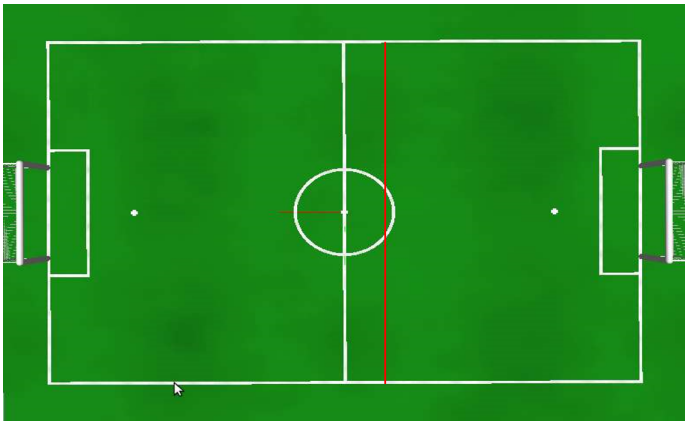


Figure 2.1: Larger defending area

2.2.2 Field Division

In the new larger defending region, kicking ball just away from our own goal is not enough. To enhance the chance of winning a goal we need to do more. Our group decided to add a dynamic kicking strategy, Which means different regions have different kicking directions. Based on the good localization result from our teammates, defender is capable to decide itself, in which direction and which speed to kick. In order to achieve this strategy, we need at first to figure out different sub regions with different attributes. With the help of statistical data[3], we quickly find out 5 different areas in our new defending region. See figure 2.2.

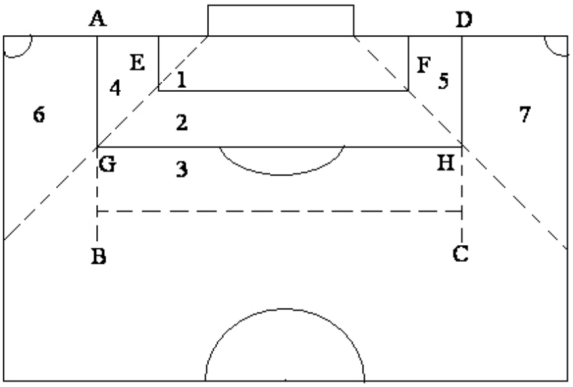


Figure 2.2: Field division in theoretical vision

As the picture showed above, region 1 is the nearest region to our own goal, which has the highest risk of losing a goal[3]. We set the width along y axis to one half of the half field and the width along x axis to one third of the half field. Region 2 is a simplified model of the original defending area, of which we simplify the circle

border to a straight line. The characteristic of region 3 and 5 are the same, since they are in a symmetric position. The only difference is the reference point of kicking direction. Their width along y axis equals respectively to the maximal kicking distance of our defender. Within this distance, the ball has a chance to be kicked out of the field. On the other hand, these two regions are closer to the opponent goal, which means more opportunities of winning a goal. The rest part of the new area is named as region 4. In this region no matter how fast the robot kicks, the ball will never get out of the field.

2.3 Implementation Details

New sub region border is added in different states related to the decision of kicking direction in *Defender.h*. New borders of sub defending areas are all real number in coordinately form. The whole justify process is simple. For example, when robot gets position information of the ball with parameter *theBallModel.estimate.position* in robot coordinate, it can directly translate it to field coordinate and compare it to the the border coordinate of the sub regions. In real code it is represented as new conditions between the translation of states. The framework of only one-kick process is drawn in figure 2.3.

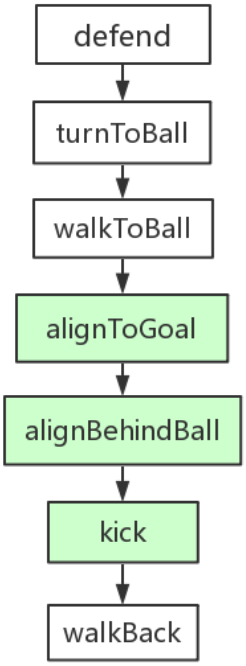


Figure 2.3: One-Kick Process

The states in green color were added with new transition conditions. One sample for it is copied from *Defender.h* as below:

```
1 if(std::abs(libCodeRelease.angleToBoarder) < 8_deg && std::abs(
      theBallModel.estimate.position.y()) < 100.f)

3 goto alignBehindBall_c;
```

Regardless of the already existed kicking direction condition, the function condition less than *100.f* represents that the position of the detected ball is inside the red line in x axis mentioned above, which is the enlarged border of the whole defending area. In this way we can restrain our defender position in a specific region.

Chapter 3

Kicking direction control

3.1 Motivation

Based on a robust self localization result, motion control is another important part in the game. The whole defending process can be regarded as a state transition loop. To make our task more specific we roughly separated the motion part into four subsections, namely Walking, Kicking, Stopping the ball and Other motions [4]. Our group focus mainly on the kicking part. This section will find the best direction in different sub regions respectively. Regards of the attributes of five sub regions mentioned above, our dynamic kicking direction control aims at increasing the probability of winning a goal.

3.2 Theory Introduction

3.2.1 Basic Assumptions

To simplify the problem, we first assume our defender plays on an empty field and there is no detection or communication with other robots.

3.2.2 Different Strategies In Each Region

- region 1 has the highest risk of losing a goal, according to statistical data from World Cup[3].In this condition defender is supposed to quickly destroy the shooting of the opponent players. We can assume the center of our own goal as the center of a circle, due to mathematical theory the best direction could along the radius of this circle. In real task, we simplify the circle border into straight lines. The kicking direction in this region is still along the circle radius.

- Region 2 has the same attributes as normal defending area. We first define a measure of the scoring probability with the concept Open Angle[5]. Then we define a criterion function and find the solution to this optimal problem.With the help of Matlab, we finally get the optimal direction along x-axis.

- Region 3 and 5 are symmetric and is the border of the enlarged area. In this condition, avoiding kicking the ball out of field has more priority than others. Kicking direction is then simply towards the opponent goal. The only difference is the reference point. The left goalpost is for region3 and right goalpost is for region 5.

- Region 4 is in the center of the whole field. Regards that defender is not aim at shooting a goal, we only need to find a safe reference point to pass the ball. So the kicking direction is dynamic and totally depend on this point.

3.2.3 Conclusion of Kicking Directions control

In conclusion, we finally determined a new kicking direction strategy according to the different sub regions. See table 3.1

Table 3.1: Kicking directions according to 5 sub regions.

Region	Direction
1	along radius
2	along x axis
3 and 5	towards opponent goal
4	towards desired reference point

3.3 Implementation Details

3.3.1 Framework of Kicking Direction Control

In order to realize our dynamic kicking direction strategy, we need at first to find out states we want to change. Build a frame block of the code is a brief way. with iteration conditions. In this code, as long as the ball is in our new defending area, defender will keep tracking and kicking.See figure 3.1. with iteration conditions, as long as the ball is in our new defending area, defender will keep tracking and kicking.

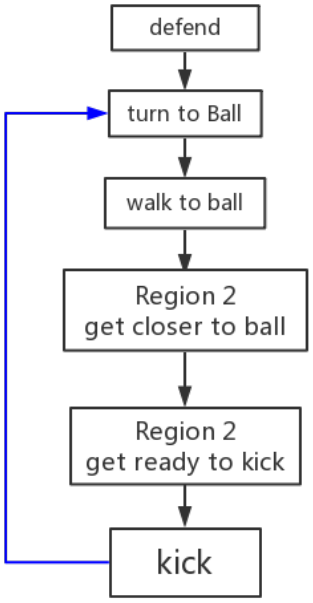


Figure 3.1: Framework of the original kicking process

There are only two states related to the kicking direction control, namely *alignToGoal*, which means get closer to ball, and *alignBehindGoal* parts, which means get ready to kick. The ball may occur in every sub regions, our defender should judge itself where the ball is and which direction to kick. Since different regions have different kicking angles, we built 3 new states in *alignToGoal* part and *alignBehindGoal* part respectively(See figure 3.2).

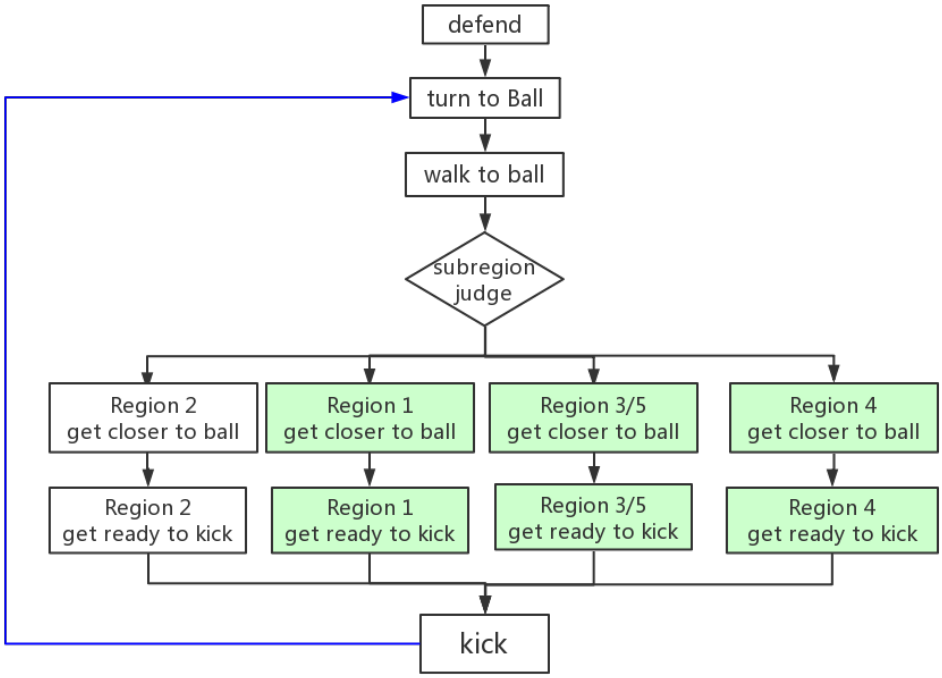


Figure 3.2: Framework of the mainly kicking process

Compared with the original framework, each part have four states now, which can be considered as four models. These models represent four different kicking strategies in corresponding sub regions(strategies in region 3 and region 5 are the same).

3.3.2 Code Accomplishment

For example, new code in *Defender.h* has now 4 states in *alignToGoal* part, namely *alignToGoal in region 1*;*alignToGoal in region 2*;*alignToGoal in region 3 and 5* and *alignToGoal in region 4*. These four states are different only in kicking angle. After finish the last state, robot will first judge which sub region the ball is in and then transits into one suitable sub state of this *alignToGoal* part.
The judgement between state transition is showed below:

```
if( Ballpose[0] > -1412.f && Ballpose[0] <= 200.f && std::abs(Ballpose
    [1]) <= 1388.f && theBallModel.estimate.position.norm() < 500.f)
      goto alignToGoal_c;

if( Ballpose[0] > -1412.f && Ballpose[0] <= 200.f && std::abs(Ballpose
    [1]) > 1388.f && theBallModel.estimate.position.norm() < 500.f)
      goto alignToGoal_l;

if( (Ballpose[0] > -2460.f && Ballpose[0] <= -1412.f && theBallModel.
    estimate.position.norm() < 500.f) || (Ballpose[0] <= -2460.f &&
```

```
    Ballpose[0] >= -3700.f && std::abs(Ballpose[1]) > 1500.f &&
    theBallModel.estimate.position.norm() < 500.f))
      goto alignToGoal_r;

if( Ballpose[0] <= -2460.f && Ballpose[0] > -3700.f && std::abs(
    Ballpose[1]) <= 1500.f && theBallModel.estimate.position.norm() <
    400.f)
      goto alignToGoal;
```

Another code example is for the new state of *alignToGoal*.

```
state(alignToGoal_c)
    {
        transition
        {
            if(theBallModel.estimate.position.norm() >= 1000.f) //if
                someone else kicks the ball away, go back to defending
                goto defend;
            if(libCodeRelease.timeSinceBallWasSeen() > 3000)
                goto walkBack;
            if(std::abs(libCodeRelease.angleToBoarder) < 8_deg && std::
                abs(theBallModel.estimate.position.y()) < 100.f)
                goto alignBehindBall_c;
        }
```

The only difference between different *alignToGoal* states exists in the parameter of kicking angle, which represents kicking direciton. The realization of different direction in every sub regions can be represented in simulation environment. Figure 3.3 is in region 1. Figure 3.4is for retion 2. Figure 3.5 is for region 3 and 5.

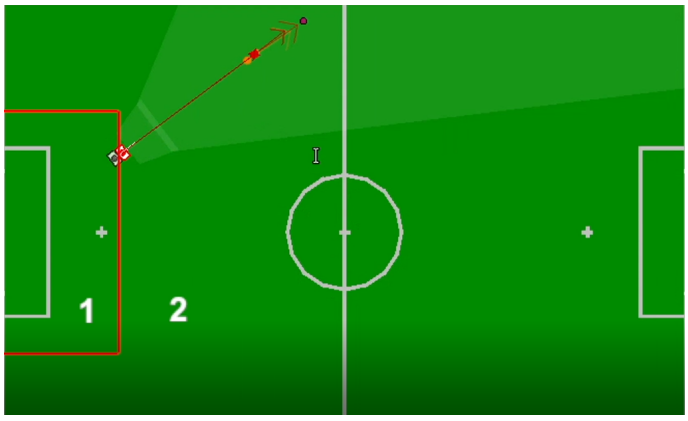


Figure 3.3: region1

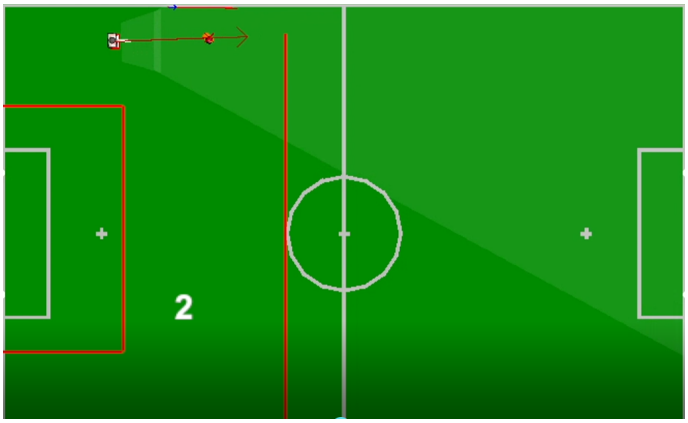


Figure 3.4: region2

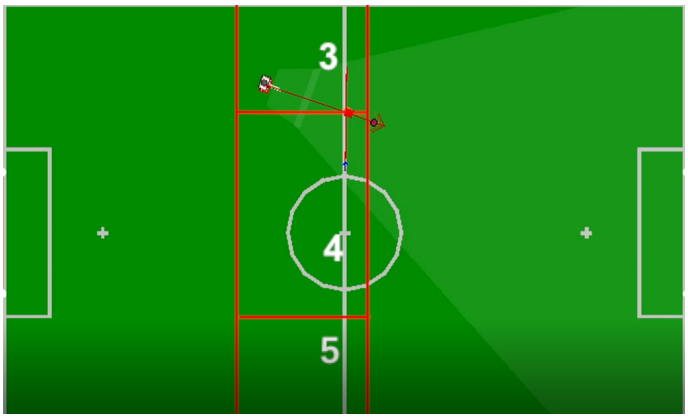


Figure 3.5: region 3 and 5

The three situations above use the same optimal kicking velocity, namely the maximal kicking velocity. Since region 4 has special attributes, we need to use a combination of dynamic kicking velocity and direction. The next chapter will discuss more specifically about this region.

Chapter 4

Dynamic Kicking Velocity

4.1 Motivation

In the sub region 4, defender need to kick the ball to a desired point. In regards to this, we need a dynamic kicking speed. The original kicking function provides only a constant kicking velocity, which can not satisfiy our requirement.

4.2 Theory Introduction

Based on the simulation environment SimRobot, we make several experiments and collect data of kicking speed and corresponding kicking distance. After data collection we analysis it by Matlab and calculate a polynomial function for the relationship between kicking velocity and distance.

4.2.1 Functional Relationship

The mathematical functional relationship is showed as below:

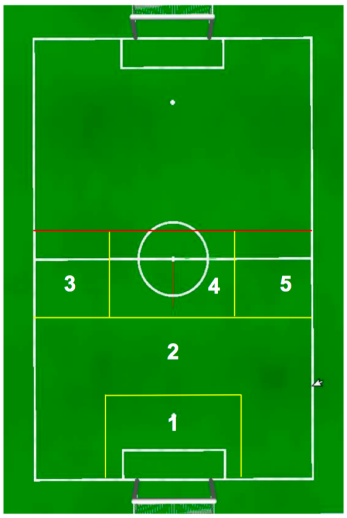
$$v = 0.287 + 0.34\sqrt{0.059d - 79.072}$$

v is kicking speed and d is shooting distance. With this function, robot can calculate how strong he should kick.

4.2.2 Combination of Kicking Direction and Velocity

With the final results in direction and velocity decision, we can get the final dynamic kicking strategy. See figure 4.1. Note that, dynamic kicking speed is implemented

only in region 4.



	Direction	Velocity
Area 1	align to own goal	v_{max}
Area 2	positive direction of X-axis	v_{max}
Area 3	mode1: to left goalpost	v_{max}
	mode2: border of opponent defense area	f(v,d)
Area 4	mode1: to left goalpost	v_{max}
	mode2: border of opponent defense area	f(v,d)
	mode1: to right goalpost	v_{max}
Area 5	mode2: border of opponent defense area	f(v,d)
	mode3: to right goalpost	v_{max}
ps	v_{max} maximal stable speed	
	f(v,d) velocity-shooting distance function	

Figure 4.1: Complete dynamic kicking strategy

4.3 Implement Introduction

4.3.1 Framework of of Dynamic Kicking Velocity Strategy

Compared with the original code, defender has only one kicking speed, because in original small defending area there is no need to have other kicking speeds. Regard of the enlarged defending region, when we want our defender pass the ball to a desired position, it is indeed a good idea to have a dynamic kicking speed.
To realize dynamic kicking speed according to the distance between robot and ball, we first fitted a polynomial of them by the Matlab. According to this polynomial we changed the kicking function in *LibCodeRelease.cpp*[6]. to a dynamic one. This new function is called in the state *Kicking*. See figure 3.2.

4.3.2 Code Accomplishment

The code details of this dynamic function is showed as below:

```
1  float LibCodeRelease::kickSpeedDynamic(float dist ){
       float a1 = 1.4;
3      float a2 = 594.5;
       float a3 = 790720.0;
5      float a4 = 2.87;
       float a5 = 10;
7
       float speed = (a1/a5/a5/a5 * (sqrt((a2 * dist) - a3))) + a4/a5;
9
```

```
        return speed;
11      }
```

We use this dynamic kicking speed in region 4 in order to pass the ball to a desired point. From the result in simulation (See figure 4.2)and on real field(See figure 4.3).

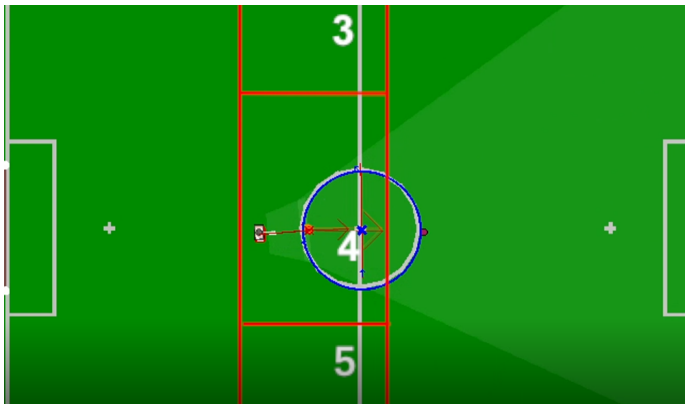


Figure 4.2: Desired position of passing the ball

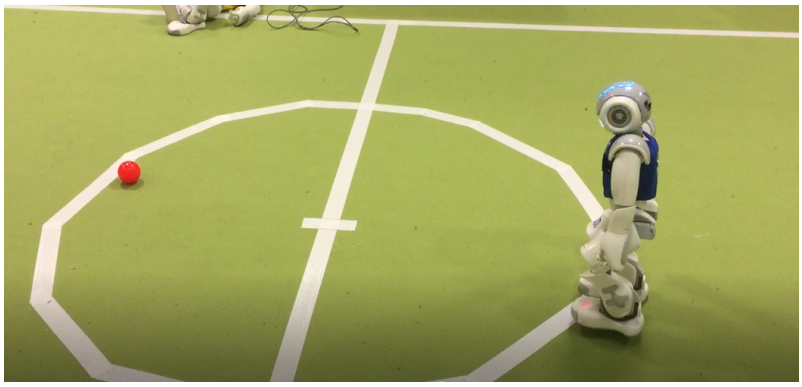


Figure 4.3: Implementation on real robot

As we can see, robot in both environment have successfully passed the ball to the desired position in region 4. This position can successfully avoid direct collisions with the opponent defender. This decreases the risk of losing the ball while passing it to our own striker. In conclusion this strategy outperforms than the original one.

Chapter 5

Conclusion and Future Work

The experiment showed a new feasible dynamic kicking strategy of defender. Regards with different attributes of sub regions defender will kick the ball in an optimal combination of direction and speed. In theory part we realized it in forms of mathematical functions and in implementation part we accomplished the desired movement on real robot.
In general detection of the opponent player and our own team player can be implemented in the future. Robot can translate the position information of the other players in forms of points coordinates on the field. Considering these information, an optimal desired kicking point will then be calculated. With the help of our new kicking strategy, defender is capable to pass the ball to that optimal point. In conclusion our dynamic kicking strategy outperforms the original one and will enhance the chance of winning a goal.

List of Figures

Bibliography

[1] Thomas Röfer, Tim Laue, Judith Müller, Michel Bartsch, Malte Jonas Batram, Arne Böckmann, Martin Böschen, Martin Kroker, Florian Maaß, Thomas Münder, Marcel Steinbeck, Andreas Stolpmann, Simon Taddiken, Alexis Tsogias, and Felix Wenk. B-human team report and code release 2013, 2013. Only available online: `http://www.b-human.de/downloads/publications/2013/CodeRelease2013.pdf`.

[2] Thomas Röfer, Tim Laue, Jonas Kuball, Andre Lübken, Florian Maaß, Judith Müller, Lukas Post, Jesse Richter-Klug, Peter Schulz, Andreas Stolpmann, Alexander Stöwing, and Felix Thielke. B-Human team report and code release 2016, 2016. Only available online: `http://www.b-human.de/downloads/publications/2016/CodeRelease2016.pdf`.

[3] Lei Fang. Research on best region of football shooting.

[4] Matthijs Van Keirsbilck, Roland Schmid, Moritz Berthold, Ricardo Vasquez, Ingrid Ibarra, Robert Darius, and Oscar Wellenstam. Robotum report ss 2016, August 2016.

[5] Joydeep Biswas, Juan Pablo Mendoza, Danny Zhu, Benjamin Choi, Steven Klee, and Manuela Veloso. Opponet-driven planning and execution for pass, attack, and defense in a multi-robot soccer team, 2014.

[6] Thomas Röfer, Tim Laue, Jesse Richter-Klug, Maik Schünemann, Jonas Stiensmeier, Andreas Stolpmann, Alexander Stöwing, and Felix Thielke. B-Human team report and code release 2015, 2015. Only available online: `http://www.b-human.de/downloads/publications/2015/CodeRelease2015.pdf`.

License