RoboCupJunior Soccer Simulation Challenge 2021 Team Description Paper i-bots 4

Eduard Fell and Malte Gembus

Mentor: Leonard Till Häberle
Roberta RegioZentrum
Hannover,Germany

www.i-bots.de
Email: info@i-bots.de

Abstract. In the RoboCupJunior Soccer Simulation Challenge the goal is to code a program which makes as much goals as possible. We tried to achieves this by creating a program in python which can constantly adapt to the enemies' robots play style and use preprogrammed solutions to defend them. Moreover, we predict the position of the Ball, which allows us to intercept the ball very efficient.
Keywords: Robocup Junior, Soccer Simulation Challenge, i-bots

1 Introduction

All of us have already participated multiple times in the RoboCup Junior Line/Soccer and the Demo Competition League. Malte have also participated in the World Robot Olympiad. We have been a team for 2 years now and participated in several competitions together. We also plan to participate in the RCJ Soccer Open in the future. In this competition, Eduard took care of the ball tracking as well as the defense robot. Malte took care of the tactics.

Fig. 1. Team picture. Malte on the left-hand side, Eddie on the right

2 Field zoning

For a better categorization and understanding we sectioned the Field into different Zones (See **Fig. 1.**).

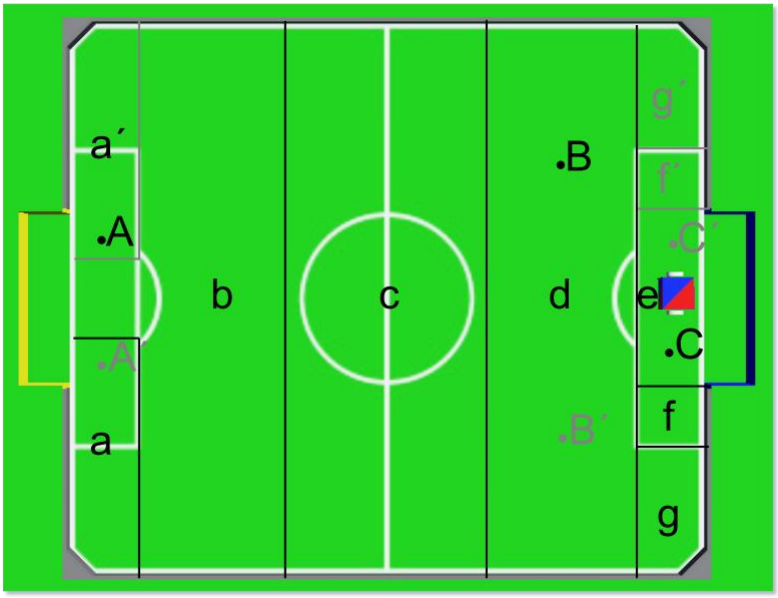

Fig. 2. Divided zones and points. In this case, the enemies are the yellow side. Moreover, all zones and points are mirrored at the x-axis and displayed in grey.

3 Tactics

Our robots decide independently which role is best suited for them (See **Fig. 2.**). These roles have different tasks depending on the current game situation. Because the defense of the goal always has the highest priority, the role of the goalkeeper is always assigned first. The second highest priority has the striker. The third player should always push the opponent's goalkeeper away when we have a chance to score a goal, unless our goalkeeper is pushed away by the opponent, in which case it becomes a decoy and drives into our goal (See **5. Pushing Robots**).

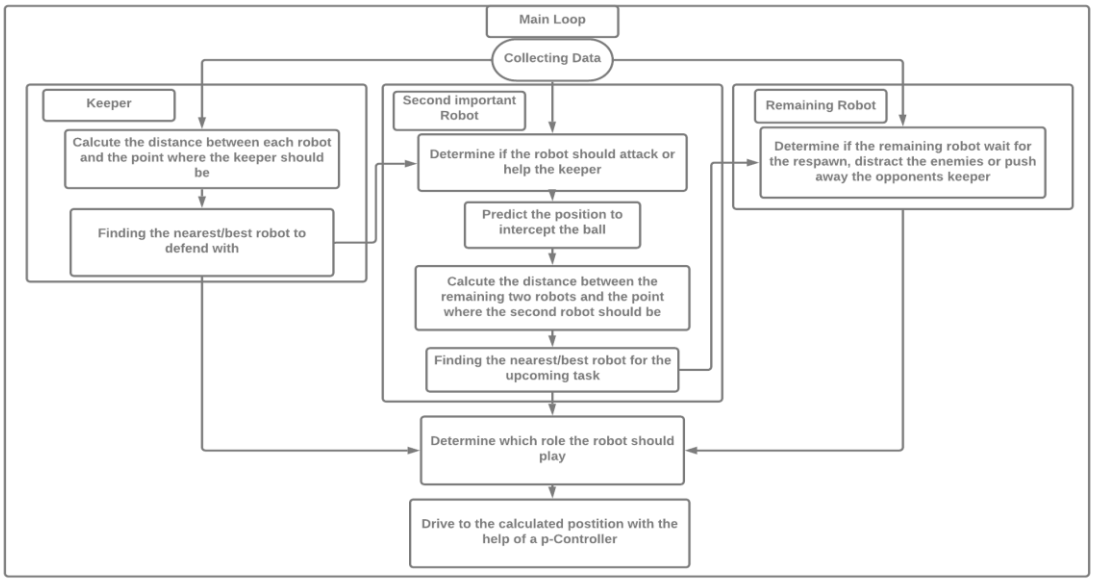

Fig. 3. Code structure of the dynamic role changing

4 Defending

Since the goalkeeper plays the most important role in our program, it is also built accordingly elaborately. We decide how to defend depending on the position of the ball. Also, the direction in which the ball rolls play a big role.

4.1 Defence Point

Normally the robot always tries to be between the center of the goal (Defence Point) and the ball. It also checks if the ball is rolling towards the goal. In this case, the Defence Point is not set to the center of the goal, but to the point where the ball rolls into the goal.

4.2 Zones

When the ball is in area a or b (See **Fig. 2.**), the robot stays on the border of area c and d and only changes the y-position. This allows the robot to block approaching balls early, but at the same time it is not too far out, so it can move freely, not colliding with opposing robots. When the ball is in area c, the robot tries to place itself exactly between the ball and the defense point. This way we cover as large an angle as possible without being too far away from our goal, so we can still react quickly to changes in the direction of the ball. If the ball is in area A, the goalkeeper drives directly towards the ball to reduce the angle at which the ball can roll into the goal and to actively play the ball away from his own goal. If the ball is in area f or g, there is a high probability that the ball will be pushed into the goal along the side of the goal. Therefore, in this situation, the goalkeeper moves directly along the wall in the direction of the ball. This way we push against the ball and prevent it from being pushed into the goal.

5 Attacking the Ball – Predicting the ball

The striker has the important task of scoring the goals. That is why we have written a program that allows him to foresee the future and thus quickly intercept the ball and shoot it into the opponent's goal.

5.1 Get Velocity

First, we calculate the speed of the ball for the x- and y-axis. This is done with the following formula

$$velocity_{axis(x,y)} = \frac{position - position_{last}}{time - time_{last}}$$

The program calculates the time elapsed and the distance in x- and y-direction between the last and the current time step. Then we divide the two values and get the velocity of the ball on the respective axis.

5.2 Prediction

Then we calculate at which point we will intercept the ball. For each axis we calculate the position of the ball in the next timestep.

$$position_{axis(x,y)} = velocity_{(x,y)} * time$$

Since the robot has a fixed maximum speed, we only check if the robot can reach this coordinate within the given time window (**Fig. 3.**). If this is not the case, this calculation is repeated for the following time step. This is done until the coordinate is either not reachable, because it is outside the playfield, or until we would predict more than 2.5 seconds. Also, the point where we intercept the ball must be closer to the opponent's goal than the original position of the robot. This ensures that we always shoot the ball in the direction of the opponent's side. And therefore, no own goals will be scored.

If the prediction fails for one of the reasons given, the robot will drive directly towards the ball.

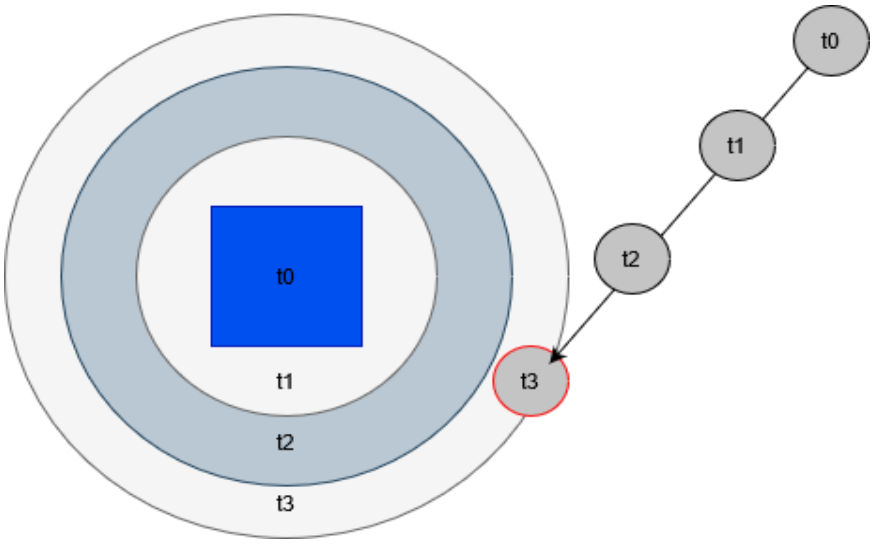

Fig. 4. Visualization: ballprediction. The blue Box is the current position of the Robot. "$t + n$" stands for the number of Timesteps

6 Pushing Robots (Pusher)

Some Teams (including us) have implemented a pushing robot. This robot often tries to push away our goalkeeper, by detecting the rearmost player and push him away. This tactic hurts us a lot, because our keeper is blocked and therefore can't defend. Therefore, we decided to counter this tactic.

6.1 Recognize a Pusher

We can tell if a player is intentionally pushed away by two conditions, both of them must be satisfied. 1. an opponent player is close to our back player for more than 30 seconds. 2. the ball is in the opponent's half. This condition is important to know that the opponent is not only close to the player to follow the ball, but to press against us. The long time was also chosen only to avoid false detections.

6.2 Countering the Pusher

After we have recognized that we are being pushed away, we drive with our least important robot into our own goal to block it. Thus, luring the pusher away from the keeper into our own goal and helps us defending our goal by blocking it.
So, we managed to make the previously useless player usable again and in addition we use the enemy robot to defend for us.

7 Conclusion

Since the Demonstration Competition our code has improved very much und we are very happy with our results. But of course, there are things to improve. For example, some Teams have something like a ballhandling program. This allows them to control the ball better and kick the ball in the needed direction.
But all in all, we have learned very much during this competition and improved our coding skills pretty much.

References

1. https://cyberbotics.com/doc/guide/controller-programming (last visited 22.06.2021)
2. https://docs.python.org/3/tutorial/ (last visited 22.06.2021)