i-bots.de
info@i-bots.de

RoboCupJunior Soccer Simulation 2021
Team Description Paper i-bots 5

Maximilian Dubinin, Paul Janosch Frese
Mentor: Marvin Bersiner
Roberta RegioZentrum
Hannover, Germany

Maximilian Dubinin (left) and Paul Janosch Frese(right)

Abstract

In this paper we will describe our tactics and the inner workings of our robots to you. This paper contains an overall strategy, a more detailed explanation of certain strategies, for example why we came to the conclusion of using one of our robots to push away enemies, and an in-depth explanation of every robot. For example, how we're calculating the future positions of our ball, to defend our goal more efficiently, or how we're calculating the best position for the striker, so that the robot can reach the ball as quickly as possible.

Introduction

Hello, we are Max and Paul and we are team i-bots 5 from the Roberta Regiozentrum Hannover. We are both 16 years old and live in Hannover Germany. This year we decided to use the free time we had during the current pandemic to take part in the Robocup junior soccer sim. We chose this category as we already prepared ourselves to continue in the Robocup soccer open league after we both participated in the rescue line category last year at Robocup 2020.

1. Strategy

1.1 Overall Strategy

The most fundamental part of our tactics is that every one of our robots has a specific task. There is a Striker, in charge of striking goals, a goalkeeper, who is defending the goal, and a pusher.

1.2 Certain Strategies

After having the overall tactics out of the way, we focused on preventing own goals, by making sure, that only the goalkeeper can stay in the back quarter, so that the striker can't make any own goals accidentally, and by programming the Goalie to defend the goal from all sides.
We programmed the Striker to shoot the ball with precision instead of pushing it into the goal, but the problem is, that there is almost always an enemy blocking off the goal from the Striker. That's why we also programmed a Pusher who is pushing the enemies out of the way, so that we have a better chance to Strike a goal.

2. Functionality

2.1 Driveto Function

The first thing we worked on was the driveto function, which needed the x and y coordinate of the destination, to find the most efficient way. The function besides if it would be more efficient to drive forwards or backwards, depending on the direction the robot is facing.

2.2 Robot 1 - Striker

The Striker tries to score a goal a any moment, except when the Goalkeeper needs help defending.

Ball Prediction. Every time step, the Striker tries to get behind the ball, by predicting its future positions.
After the ball speed was calculated, the Robot checks, if the ball position in the next time step is reachable, by dividing the distance between the ball and the robot with the average robot speed per time step, and thus getting the time steps needed to reach the position as a result.
We decided to put the average ball speed to around 0,008m per time step, as this number has proven itself to be accurate enough for this application. This speed is quite a bit lower than the maximum speed of 0.0128, because turning while starting had to be taken into account. If the Ball is not reachable in the time step, the calculation gets repeated with the next time step and the next theoretical ball position and distance.

Future Ball position = $$ball\ position(x) + ball\ speed(x) * iterations = future\ ball\ position(x)$$
$$ball\ position(y) + ball\ speed(y) * iterations = future\ ball\ position(y)$$

This process gets repeated until either the process was repeated over 300 times (this acts as a fail safe, so that the process wouldn't be repeated infinitely) , or if the amount of time steps in which the robot will reach the calculated position is lower than the iterations of the program, or rather the amount of time steps the ball needs to reach the position.
To put it briefly, if the latter one is the case, the robot is able to reach this position before the ball.

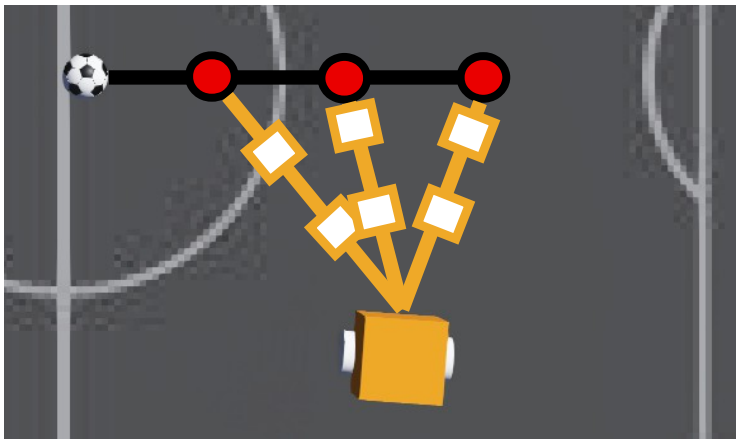

Fig 1. The prediction of the Ball. Future ball positions are shown in red, and the time steps needed to reach every position are shown in white. (not to scale)

As you can see,(see Fig. 1). to reach the position for the first time step the robot would need to cover almost 3 time steps. In the second time step, the Robot would still need more time steps than the Ball to reach the position, and in the third time step, the robot is able to reach the position before the ball, because the time steps the robot needs to reach the point are lower than that of the ball.

Reaching the Ball. The robot will not move to the exact position calculated, but rather 0,1m closer to the teams side, so that the robot ends up behind the Ball, and has better chances of scoring a goal. Every time the Ball changes its velocity, the calculation will be executed another time, to prevent the robot from going into the wrong direction after the ball hit a wall or another robot.

Shooting the Ball. The robot will only shoot the ball if it's located in the purple area. At First, the robot rolled towards he goal along the yellow line, and α was the angle of incidence of the ball when hitting the robot. In this instance, the ball would ricochet off the robot with the angle of α and roll along the green line. To fix this, we programmed the robot to follow the red line, so that the angle is $\alpha/2$ and the ricochets off the robot directly into the goal(yellow line).

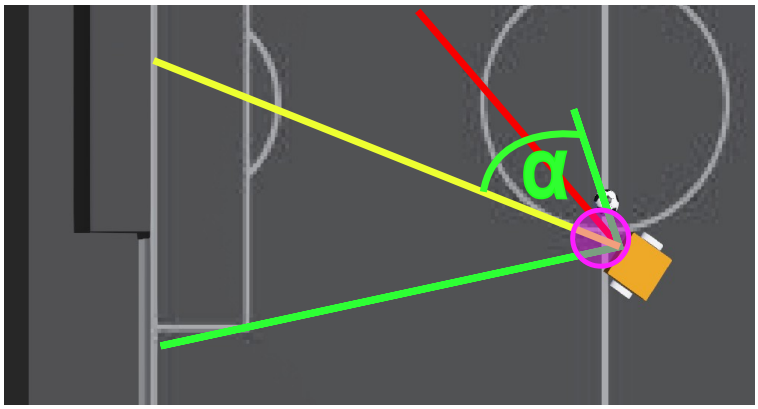

Fig 2. Shooting the ball. The angle of incidence of the ball is shown in green, and the path the ball would roll on if the robot moves into the direction of the goal is also green. The path the robot needs to take is red and the resulting path for the Ball is yellow. The zone in which the robot shoots the ball is shown in purple

Helping the Goalkeeper. After extensive testing we realised that the enemies often got a goal because the opponent overpowered our goalie(see Fig. 3a). In this case the ball together with our robot got pushed into the goal. To prevent this we gave our Striker the task to support the goalkeeper by pushing in the same direction to out-push the opponent during such a situation(see Fig 3b). This didn't cause any other problems with our strategy as the Striker is supposed to wait for the ball to leave the zone around our goal anyway.

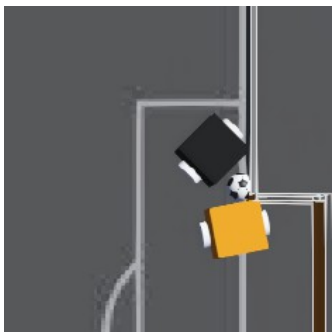

Fig 3a: the Goalkeeper getting pushed into the goal by an enemy robot.

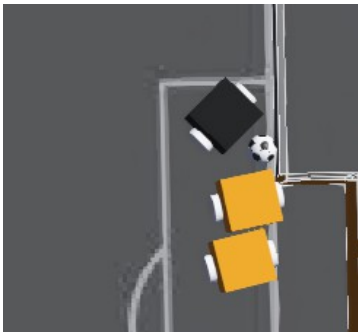

Fig 3b: the Striker helping the Goalkeeper to push away the Enemy.

2.3 Robot 2 – Goalkeeper

For our Goalkeeper we decided to split the field in 5 areas.

As long as the ball is at the height of the goals, near the x-axis (blue area, see Fig. 4.), our defender tries to be between the goal and the ball by taking the balls y-Position and trying to stay at the same y-height just outside the penalty area. This situation also includes the very rare case of when the ball already is behind the robot at goal height. After trying out different approaches we settled with leaving the robot outside the penalty area as to not accidentality push the ball into the net. By staying at the same y-level as the ball our goalie also is able to block robots which are trying to score a goal.
If the ball is on the left or the right side of the field (yellow area), the robot will stop at the height of the near goalpost to prevent leaving the goal wide open, protecting it from angled shots. In this scenario it often happens that the goalie waits at one point for quite some time. To avoid getting the robot reset we programmed it to slightly drive on the spot – just enough movement to not get reset while still keeping a good position. Furthermore, by predicting the balls movement the robot either faces sideways or straight(green dotted line). This ensures the goalkeeper can react as fast as possible to shots from all different directions.

If the ball is located in one of the corner areas (red area) the goalie simply keeps the same x-level as the ball staying at the y-height of the goal post.

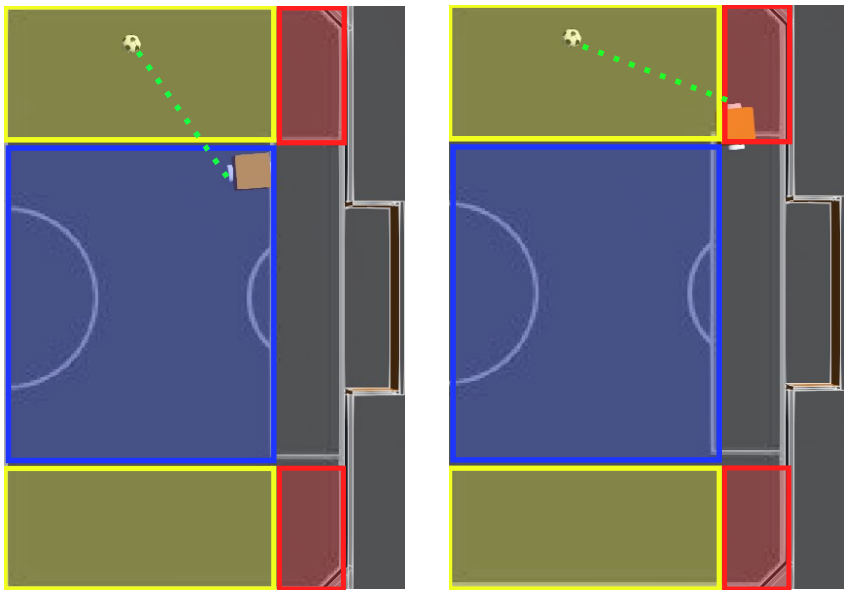

Fig 4. The zones of the goalkeeper. The robot predicting the ball is shown in green.

2.4 Robot 3 – Pusher

For the role of our third robot we decided to use it as a pusher. It's task simply is to push away the opponents goalkeeper so that their goal is widely open. Robot 3 is programmed to locate the opponents robot which is standing the closest to the opponents goal. Then it tries to come in from the side and push the opponent as far away to the side as possible. By checking on which side of the field the opponent is situated we programmed Robot 3 to complete this task as efficiently as possible.

3. Conclusion

In conclusion, we think that our code improved by a lot in contrast to the code we wrote during the demo competition. What helped us the most, was the driveto function, because we didn't need to focus on the robot movement, but rather on the programming and strategy of the robot. Of course, our program isn't perfect, for example, our robots can't detect the walls, but all in all we think we have a pretty good result.

References

1. Simulation platform.: Webots. https://cyberbotics.com/doc/guide
2. Download and Setup.: Rcj-soccer-sim. https://github.com/RoboCupJuniorTC/rcj-soccer-sim/tree/v0.1-alpha
3. Getting started.: The Python Tutorial. https://docs.python.org/3/tutorial/index.html