XLC-Innovation 2021

Jakub Gál[1] Šimon Peter [2] Rastislav Gaži [3[Mentor]]

[1] SPŠSE Nitra, Slovakia, bukajlag@gmail.com
[2] Gymnázium Golianova, Slovakia
[3] XLC team, Slovakia
Homepage: xlc-team.info
Contact Email: bukajlag@gmail.com

Abstract. In this paper we aim to present the solution for Robocup Junior Soccer Simulation Challenge 2021. In case of any questions or suggestions please do not hesitate to contact us on the contact email address bukajlag@gmail.com.

1 Introduction

1.1 Background

We are the team XLC Innovation, which is a subteam of a bigger organization of the team XLC. Our team consists of two members Jakub Gál and Šimon Peter and a mentor Ing. Rastislav Gaži.

Jakub Gál is the captain of our team and is in charge of all Software and Electronics. He participated in Robocup Junior 2020 virtual poster session in the Open category where he won the best poster award in his category, in RoboCup Junior 2019 in Sydney in the Light Weight category he won a prize for the best poster & presentation and 2nd place in superteam games, as well as 14th place in individual games. He was also at RoboCup 2017 Nagoya in the Light Weight category where he won 2nd place in superteam games. He also won several prizes at a national level in Soccer Open, Soccer Light Weight, as well as rescue.

Šimon Peter is in charge of Mechanical Engineering and 3D printing. He was also a participant of RoboCup 2017 in Nagoya in the Light Weight category where he won 2nd place in superteam games and he also won several prizes in RoboCup Junior at a national level.

We as a team also ended in 4th place in Soccer Simulation Demo 2021.

Fig. 1 Members of XLC-Innovation team consisting of Jakub Gál(left) and Šimon Peter(right)

1.2 Highlights

We think that our biggest highlight is our goalie and overall strategy. Also, we are proud that we were able to create these robots even during the Covid-19 pandemic.

2 Robots and results

2.1 Software

Methods

For the programming of our robots, we used Python and as a text editor, we used visual studio code. We collaborated using mainly GitHub but we also used other software such as Zoom for video calls to discuss strategy.

Strategy

There are two different roles that the robots have: one is an attacker and one is a defender. We always calculate the distance of each robot in our team from the middle of a goal and the robot which is nearest to the middle of the goal is the defender and the other two are attackers. We do not need to deal with communication between the robots because we know that all our robots will get the exact location of every other robot, which helps us greatly.

```
distance_to_y_goal_1 = distance(-0.75,data['Y1']['x'],0,data['Y1']['y'])
distance_to_y_goal_2 = distance(-0.75,data['Y2']['x'],0,data['Y2']['y'])
distance_to_y_goal_3 = distance(-0.75,data['Y3']['x'],0,data['Y3']['y'])
golier_y_name = 'Y1'
if distance_to_y_goal_1<distance_to_y_goal_2 and distance_to_y_goal_1<distance_to_y_goal_3:
    golier_y_name = 'Y1'
elif distance_to_y_goal_2<distance_to_y_goal_1 and distance_to_y_goal_2<distance_to_y_goal_3:
    golier_y_name = 'Y2'
elif distance_to_y_goal_3<distance_to_y_goal_1 and distance_to_y_goal_3<distance_to_y_goal_2:
    golier_y_name = 'Y3'


distance_to_b_goal_1 = distance(0.75,data['B1']['x'],0,data['B1']['y'])
distance_to_b_goal_2 = distance(0.75,data['B2']['x'],0,data['B2']['y'])
distance_to_b_goal_3 = distance(0.75,data['B3']['x'],0,data['B3']['y'])
golier_b_name = 'B1'
if distance_to_b_goal_1<distance_to_b_goal_2 and distance_to_b_goal_1<distance_to_b_goal_3:
    golier_b_name = 'B1'
elif distance_to_b_goal_2<distance_to_b_goal_1 and distance_to_b_goal_2<distance_to_b_goal_3:
    golier_b_name = 'B2'
elif distance_to_b_goal_3<distance_to_b_goal_1 and distance_to_b_goal_3<distance_to_b_goal_2:
    golier_b_name = 'B3'
```

Fig. 2 Part of role assignment code

Defence

When the robot is in defence mode it always tries to get to the intersection of the line that connects the ball and the middle of the defended goal with a predefined spline that informs of the goal area(Fig. 4). We also found out that when all of the robots are in the corner with the ball and lack of progress occurs, the opponent can score a goal after the ball resets to a neutral spot(Fig. 3). We handled this problem by detecting this situation using a calculating average speed of the ball and if the speed goes under a certain threshold for a certain time the defender resets to be in the middle of the goal area to be able to block the opponent's attack as soon as possible. For regulating robot movement we used a proportional regulator, which turned out to be sufficient.

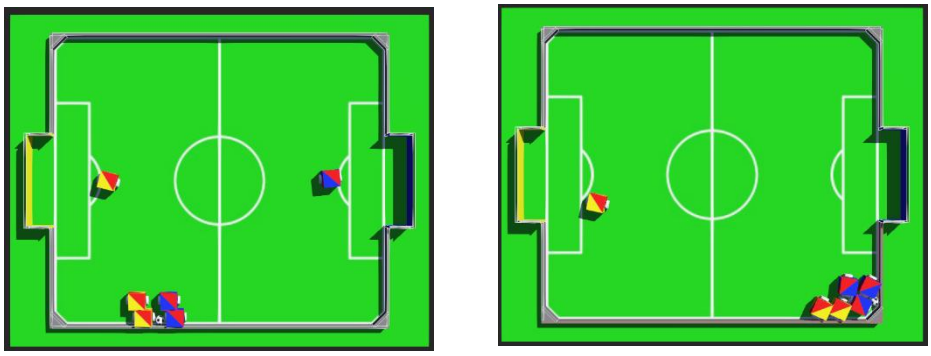

Fig. 3 Different game situations. Defender reset before lack of progress on the left and defender in the suboptimal position on the right

```
#defense_curve
#-0.526* x^2 - 0.5 = y
fake_x = ball_pos['y']*-1
fake_y = ball_pos['x']*-1
#smerovi vector
s_x = 0 - fake_x
s_y = 0.85 - fake_y
#normalovi vector
n_x = s_y
n_y = s_x * -1

if n_y == 0:
    n_y = 0.000000001
if n_x == 0:
    n_x = 0.000000001
#vseobecne vijadrenie priamky
c = (n_x * fake_x  + n_y *  fake_y )*-1
#priesecnik
#-0.526* x^2 - 0.5  = y
#(n_x/n_y *-1))(n * x + (c/(ny*-1)) = y
a = steepness
b = -((n_x/(n_y *-1)))
c = 0.5 - (c/(n_y*-1))
D = math.sqrt((b*b)-(4*a*c))
x_1 = (-b +D)/(2*a)
x_2 = (-b -D)/(2*a)
y_1 = steepness*(x_1*x_1)+0.5
y_2 = steepness*(x_2*x_2)+0.5

distance_1 = math.sqrt(((fake_x-x_1)*(fake_x-x_1)) + ((fake_y-y_1)*(fake_y-y_1)))
distance_2 = math.sqrt(((fake_x-x_2)*(fake_x-x_2)) + ((fake_y-y_2)*(fake_y-y_2)))
if distance_1 < distance_2:
    real_intersection_x = y_1 * -1
    real_intersection_y = x_1 * -1
else:
    real_intersection_x = y_2 * -1
    real_intersection_y = x_2 * -1
```

Fig. 4 Code for defender position calculation

Attacker

When the robot is in attack mode it always tries to get to the point on the line that connects the ball and middle of the opponent's goal where he would be able to push the ball toward the opponent's goal (Fig. 5). For this movement, we used a proportional regulator, which turns out to be sufficient. When the attacker gets to this point with certain toleration then it tries to go to the centre of the opponent's goal. While doing so it also regulates his movement based on the relative angle of the ball to this attacker to be able to manipulate the ball. Both attacking robots use the same code so they are often near each other, which on the first look might look like a problem but after testing, we found out that this strategy is great for defence as well as to push the ball through the opponent's defending robot, which would not be sometimes possible with only one attacking robot.

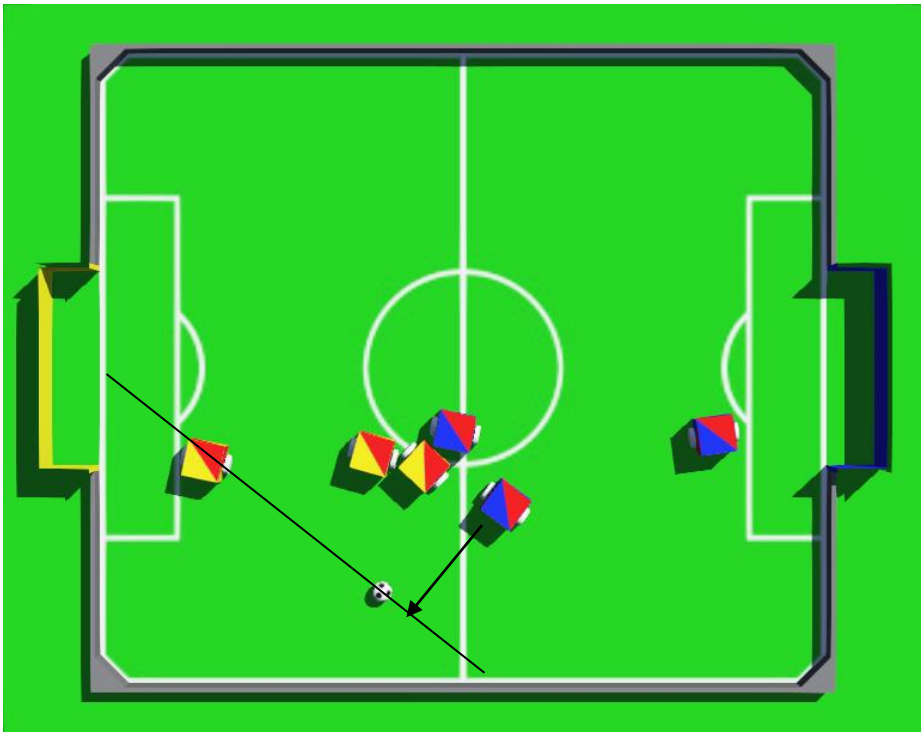

Fig. 5 Illustration of attacker movement

2.2 Results

When creating this robot's strategy we conducted many experiments that we then evaluated and adjusted the robots' behaviour based on the results. In the beginning, we used our own robot strategies for testing, we then compared them and used the best one for the demo competition. But after the demo competition, other teams' strategies were available so we used them to test our robot again to find all flaws and fix them. After qualification, we analysed the matches to find out new problems that needed to be fixed. For example, we found out that every goal we received happened after the ball reset to the middle neutral point and the opponent robot was waiting nearby to attack as fast as possible and all of our defenders were in the corner and were not fast enough to reverse the attack. So we added to robot strategy functionality that monitors if this situation occurs and if it occurs defender goes to the middle of the goal area to be able to stop the attack.

3 Conclusions and Future Work

3.1 Summary

While creating this robot we learned a lot of new information and gained many new experiences ranging from finding more about the strategy creating process, to knowing how to use the Wbots simulator. We also learned to manage long-distance work by using GitHub, Zoom and other programs because of the Covid-19 pandemic.

3.2 Future

We have many ideas about what we want to try in the future. First, we want to create a robust multiagent reinforcement learning neural network that would get in the input positions of robots and ball in the field and the output speeds of motors for each robot. Also for training, we would use all available strategies from the demo competition and this competition to get the best results possible. We think that this is the ideal approach to dealing with this problem but it is very hard to create a robustly working reinforcement learning neural network. If this approach would not work we would use a similar approach but would divide this problem into multiple subproblems such as ball handling or defending, which would be much easier to train using, for example, Soft Actor-Critic architecture. But this approach would lack the ability to collaborate between robots and would again fully depend on a hand-coded strategy. Either way, it is certainly worth testing.

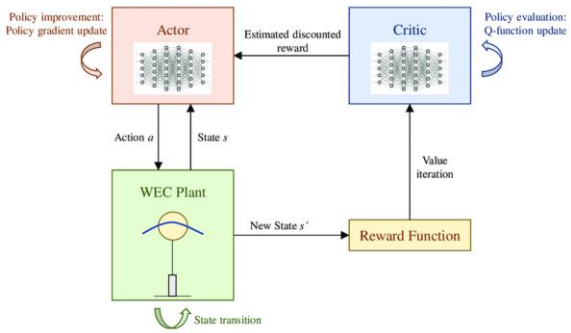

Fig. 6 Diagram of the soft actor-critic algorithm

4 Reference

1. Code of our robots, https://github.com/xlcteam/Soccer_Simulation_2021
2. Webots website, https://cyberbotics.com/
3. Starting guide, https://robocupjuniortc.github.io/rcj-soccer-sim/
4. Robocup Junior forum, https://junior.forum.robocup.org/c/robocupjunior-soccer/
5. Robocup Junior Soccer Simulator, https://github.com/RoboCupJuniorTC/rcj-soccer-sim
6. Code from Demo competition, https://github.com/RoboCupJuniorTC/rcj-soccer-sim-2021-robot-code