RoboFEI 2021 Team Description Paper

Álvaro D. Neto, Bruno Bollos Correa, Gabriel M. Schiavetto, Gabriel S. L. Agune,
Giovani S. Pereira, Guilherme S. de Amorim, Guilherme W. Cardoso, Henrique B. Simões,
Isabella R. Moscardo, João P. L. de Albuquerque, João V. L. Aguiar, Leonardo da S. Costa,
Luiz F. L. Baptistella, Thiago G. dos S. Vieira, Wesley de S. Motta,
Flavio Tonidandel, Plínio T. A. Junior, Reinaldo A. C. Bianchi

Robotics and Artificial Intelligence Laboratory
Centro Universitário da FEI, São Bernardo do Campo, Brazil
{flaviot, rbianchi, plinio.aquino}@fei.edu.br

***Abstract*—This paper presents the current state of the RoboFEI Small Size League team as it stands for LARC/CBR Small Size League competition 2021, online. The paper contains descriptions of the new robot decision to pass and shoot to the goal and the positioning of the attacker robots during normal game.**

1. Introduction

For LARC 2021, the RoboFEI team intends to use mostly the same electronics and mechanical design that have been used over the last years.

Some significant advances were made in our software, the improvements were verified in the 2021 Small Size League RoboCup, which resulted in our best placement ever in the competition, getting us in the hall of fame of the league [1]. The objective now is to improve furthermore the software system and start replacing our current robots. However, due to the current circumstances, this TDP focuses mainly on the software features. With our researches, we hope to bring innovations and new ideas for the community.

2. Software

In the strategy software, we have have mostly improved the robot decision to pass and shoot to the goal, the offensive positioning of the robots and the dynamic between defenders and attackers during the game.

Additionally, we have started to open source some of our works, as we believe that sharing software is equivalent to sharing knowledge, which is important for educational purposes. Everything mentioned in this paper can be found at our GitLab page [2].

2.1. New log analyser tool

Inside the group of projects that were released in the past year, we have the LogAnalyserRoboFEI-SSL, which is a software that has the main purpose of dealing with log files from Small Size League RoboCup matches [2]. Its main features are: read and play a log file, and, send the referee/vision messages in the network using the UDP protocol. It also contains its own graphical client, which contains several informations of the match sent by the game-controller and by the vision (the field drawing).

Alongside with this, the software is capable of detecting when a pass or a shoot to the goal occurs and, once the move ends, evaluate its quality, giving it a score from 0 to 250. A 0 score means that the pass or shoot decision was bad while a score of 250 means that the move was very good.

The evaluation of a shoot move follows the decision tree [3] shown in Figure 1. It can be seen that the maximum score of 250 is given only if the move resulted on goal and the worst score of 0 occurs only if the ball did not even get to the opponent defense area.

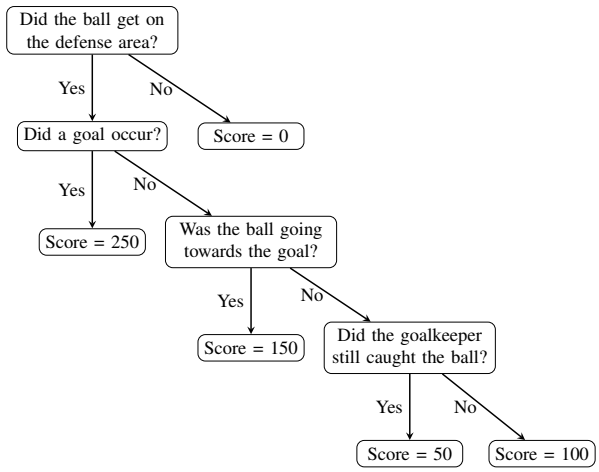

Figure 1. Shoot move evaluation decision tree

The evaluation of a pass move is more complex. If the receiver does not get the ball the score will be zero, otherwise, the score will depend on what the receiver did with the ball. The analysis of the receiver move is called parallel move. The decision trees that evaluates the score of a pass move and the parallel move can be seen in Figure 2 and 3, respectively.

When a move is evaluated, some extra parameters are measured and written into a file alongside with the score of the move. That way, at the end of the process, this files has the information of all pass and shoot moves that occurred during the match analysed.

The parameters evaluated on a shoot to goal move are:

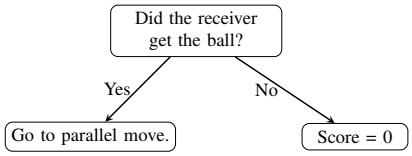

Figure 2. Pass move evaluation decision tree

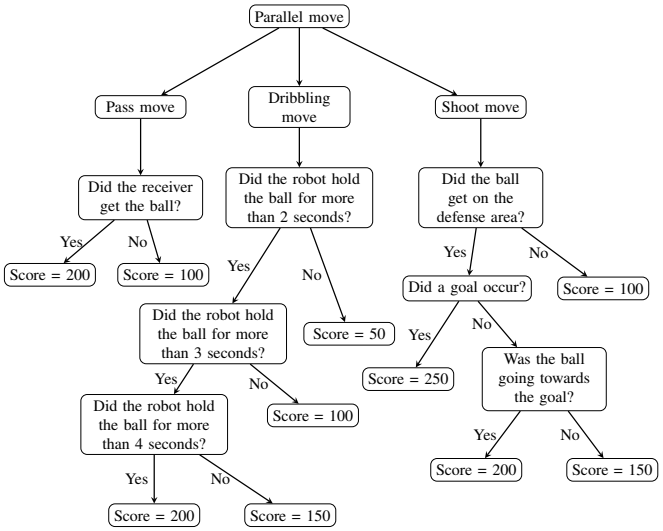

Figure 3. Parallel move evaluation decision tree

- Angle of the free path (without opponents) from the ball to the opponent goal;
- Distance from the ball to the opponent goal;
- Distance of the closer opponent to the ball.

The parameters evaluated on a pass move are:

- Angle of the free path from the ball towards the receiver;
- Distance from the ball towards the receiver;
- Distance of the closer opponent to the receiver;
- Angle between the line from the ball to the receiver and the line from the receiver to the opponent goal;
- Angle of the free path from the receiver towards the goal;
- Distance from the receiver to the opponent goal;
- Distance of the closer opponent to the ball;
- A parameter that mesures if the ball is getting closer or farthest from the opponent goal.

At the end, the parameters of the pass and shoot moves are normalized into a scale from 0 to 250 of only integers numbers. The normalization of all parameters is done according to the curve on Figure 4, where y is the parameter normalized and x is the parameter before the normalization. That way, $y_0 = 0$ and $y_1 = 250$.

For the distance parameters, $x_0 = 0$ and x_1 is one half of the width of the field (disregarding the offset), i.e. $x_1 = 6\ m$ for division A rules, and $x_1 = 4.5\ m$ for division B rules. For the parameters that measure the angle of the free path, $x_0 = 0$ and $x_1 = 45°$.

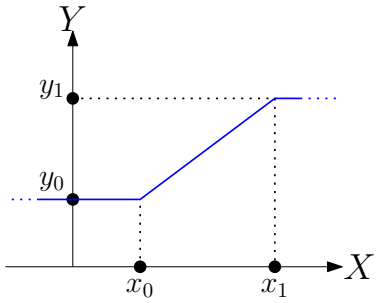

Figure 4. Normalization curve

2.11. Database of passes and shoots from RoboCup matches

With the files generated from the analysis of the match, it is possible to create a database containing the information of all passes and shoots from older matches of the league using its log files. This was done for the most outstanding teams from RoboCup 2017, RoboCup 2019 (Division A only) and LARC 2020 (online); the results were uploaded to a GitHub Repository [4].

2.2. Pass and shoot success prediction

Until RoboCup 2019, our team did not have an elaborate strategy with the ball rolling, the robot was programmed to always kick towards the opponent goal when it had possession of the ball. As the league evolved, teams have increasingly valued the passing strategy in their software; the team's behavior in rolling ball situations is of supreme importance and essential for any team.

Knowing this, we decided to implement a way for the robots to decide, on the most efficient way, when to pass the ball to other robots and shoot to the goal. For that purpose, a supervised machine learning regression model was designed [5] using the database mention on the last section. As result, given the input parameters from the field, the model is able to predict which move would be better in given circumstances.

In order to create the machine learning model, the scikit-learn [6] python module was used. There are 4 types of models implemented in the strategy software: Linear regression [5], AdaBoost [7], Gradient boosting [5] and Random forest [5].

Since the strategy software is written in C++, a python interpreter had to be embedded into the C++ application. This has been accomplished with the help of the *pybind11* library [8].

2.21. Pass and shoot success prediction in the strategy software

With these new changes, when the robot approaches the ball, if the ball is not being possessed by the opponent team, the software evaluates the score of the move to pass to another robot on the field and to shoot on the opponent goal. Robots too close to our defense area and too close to the ball are ignored. At the end, if the shoot score is greater than 150, the robot shoots to the goal, otherwise, it chooses the move that resulted in the greater score.

2.3. *Mines positioning*

With the pass strategy implemented, it was also needed that some of the robots in the field could get into a good position to receive a pass close to the opponent goal and, preferably, free of marking. For that purpose, the Mines Positioning algorithm was developed.

The Mines positioning is a simple algorithm used to determinate the point, inside a matrix of n_rows rows and n_lines lines, further away from the objects called mines and closest to the object called fortune.

It works by giving a score from 0 to 9 to each point of the matrix. The mines acts decreasing the score of its closest points, while the fortune decreases the score of its furthest points.

In the software, this is used to position the attackers during normal start and all robots during opponent ball placement. In order to use it, the matrix must be proportional to the dimensions of the field, this can be seen on Equation (1).

$$\frac{n\_rows}{n\_lines} \propto \frac{field\_width}{field\_height} \tag{1}$$

That way, each element inside the matrix represents one square field area. The area of each square is given by Equation (2).

$$square\_area = \frac{field\_width \cdot field\_height}{n\_rows \cdot n\_lines} \tag{2}$$

The inverse of $square\_area$ is called the scale of the matrix, current scale is $0,01\ mm^{-2}$. For better results, it is important that the $square\_area$ must not be much larger than the area occupied by a robot on the field.

Every position of the matrix is evaluated, depending on the position of the mines and of the point of fortune, with a score from 0 to 9, where the bigger the score the better is the position.

2.31. *Mines positioning on normal start*

During the normal start, the attackers must be as close as possible of the opponent defense area. For that purpose, the point of fortune is located on the center of the opponent goal. The matrix of the algorithm can be visualized with a heat map, thus, a game situation can be seen in Figure 5, the ally robots are represented by a big black circle, opponents are represented by a big white circle and the robot whose destination is being calculated by the algorithm is being represented by a big black square, the points with higher scores are more green, while the points with lower scores are more red and the points with medium score are yellow, the location of the mines are represented by little blue dots.

In Figure 5, it can be seen that one mine was added on the position of each ally and opponent robots. That way, the attacker will get into a position as distant as possible from ally and opponent robots. Mines are added on the destination of ally robots as well, to avoid that the algorithm sets the same destination to two different robots. In the situation shown in Figure 5, the destiny of the robot would be on the center of the most green area.

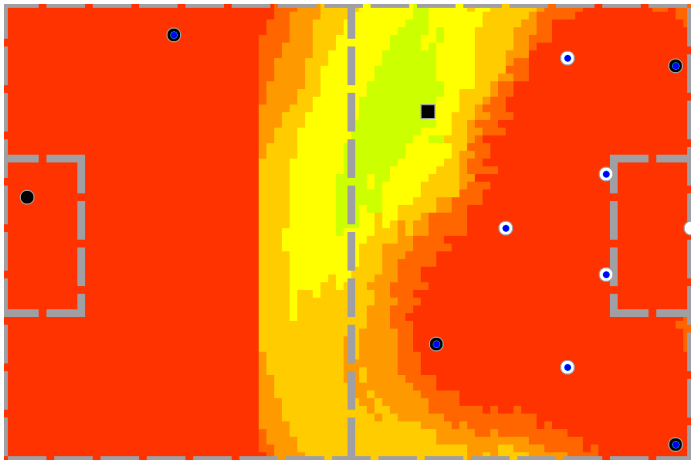

Figure 5. Mines position during normal start - situation 1

Other situations are shown on Figures 6 and 7.

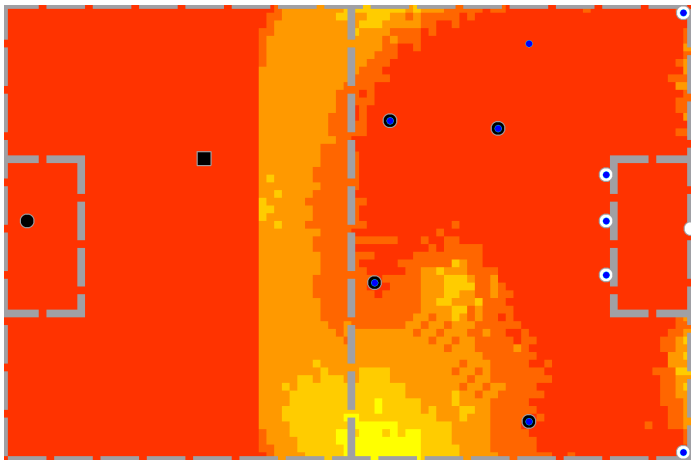

Figure 6. Mines positioning on normal start - situation 2

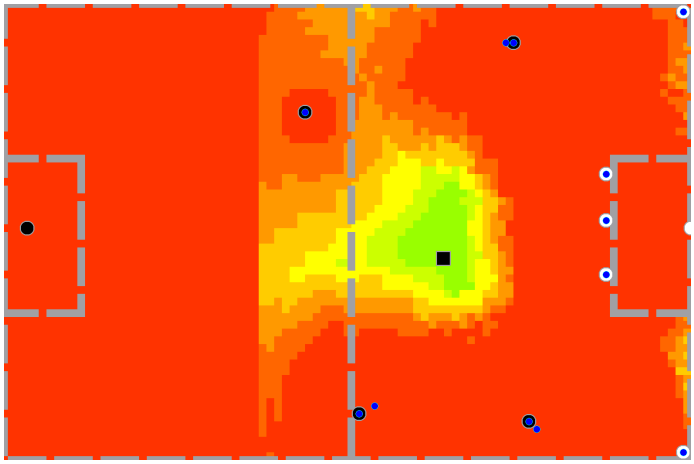

Figure 7. Mines positioning on normal start - situation 3

2.32. *Mines positioning during opponent ball placement*

During the ball placement of the opposite team, our robots must be as far as possible from the line between the designed position and the current ball position. Besides, it is preferable that our robots stay close to our defensive area, since the ball placement will most likely be proceeded with a free kick for the opponent team, hence, our robots must be prepared to defend.

For that purpose, the Mines positioning algorithm was used with no fortune points and 5 to 10 mines on the line that begins on the current ball position and ends on its designated position, and, one mine on each ally robot. Beyond that, points that are far from the defended goal are not considered.

That way, the robots tend to avoid the ball placement area and stay on the defense area, far from each other, as it can be seen on the different scenarios shown on Figures 8, 9 and 10.

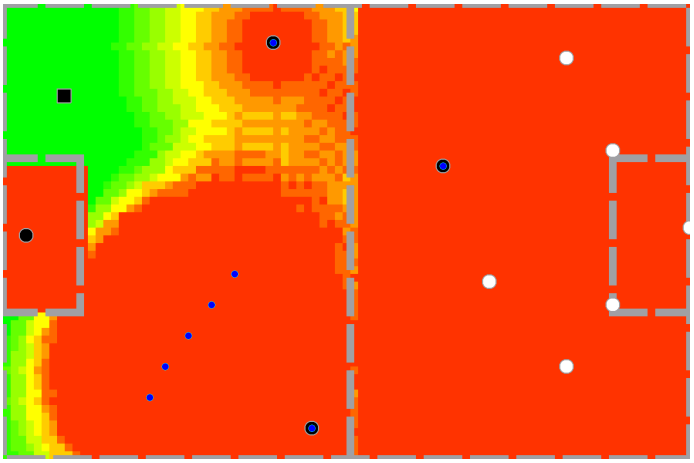

Figure 8. Mines positioning on defensive ball placement - situation 1

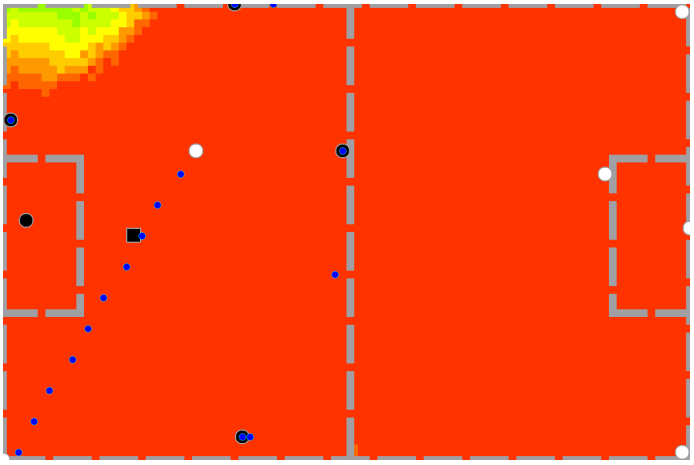

Figure 9. Blue team with mines positioning on yellow ball placement. Scenario 2

3. Acknowledgements

We would like to thank, in advance, the Small Size League Committee, for the consideration of our material. We would also like to immensely thank the staff of Centro Universitário FEI, for all the help we always received from them.

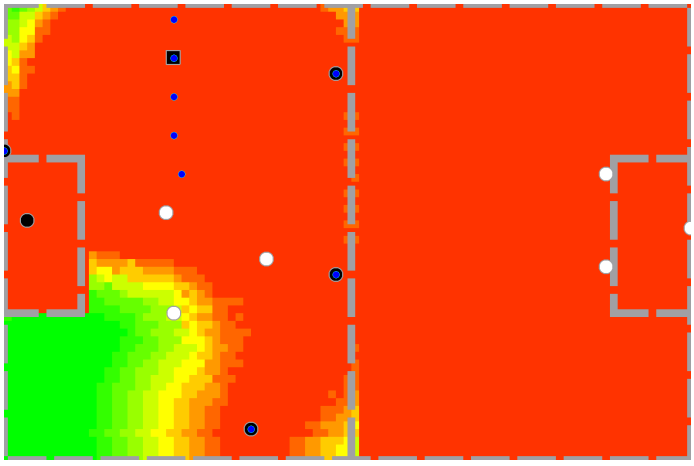

Figure 10. Blue team with mines positioning on yellow ball placement. Scenario 3

References

[1] RoboCup. Hall of fame - small size league — robocup soccer, 2021. https://ssl.robocup.org/hall-of-fame/.

[2] RoboFEI-SSL. Robofei - small size league - gitlab, 2021. https://gitlab.com/robofei/ssl.

[3] Wikipedia contributors. Decision tree — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Decision_tree&oldid=1035480568, 2021. [Online; accessed 29-July-2021].

[4] github/Bollos00. Bollos00 - databaseforkicksandpassesrobocup - github, 2020. https://github.com/Bollos00/DatabaseForKicksAndPassesRobocup.

[5] Andriy Burkov. *The Hundred-Page Machine Learning Book*. 2019.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] Wikipedia contributors. Adaboost — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=AdaBoost&oldid=1025199557, 2021. [Online; accessed 29-July-2021].

[8] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 – seamless operability between c++11 and python, 2017. https://github.com/pybind/pybind11.