Deep Neural Network algorithm to control a curved kicking mechanism in RoboCup Small Size League

Francisco Arthur Bonfim Azevedo[1] and Guilherme Pinheiro Cordeiro Leão[2]

***Abstract*— Small Size League (SSL) is a robotics soccer league at the Robotics World Cup RoboCup. In this league, the proposition of a mechanism capable of performing unpredictable kicks and passes inspired works of either of the mechanism and algorithms to try to extract the best from it. The present work is an improvement on a previously proposed one that aims to perform a real-time inversion of the non-linear ODE that models the ball's trajectory to find the parameters to hit a target with the curved kick mechanism, by adding new concepts to the problem and by using a Deep Neural Network. New strategies are added to improve the efficiency of the algorithm and to make it easier to be integrated with other algorithms on the robot and also make it capable of finding a trajectory that avoids enemies on the field.**

I. INTRODUCTION

The Small Size League is a robot soccer competition of omnidirectional robots with usually 4 wheels with several rollers to allow movements in any direction of a 2D plane. The teams compete in matches of a maximum of 6 robots for the Division B or 8 robots for the Division A [1]. The vision system is based in cameras positioned on the ceiling of the field and image processing algorithms capable of identifying each robot of each team by a color pattern on the top of the robots pre-defined by the competition. the pose of the robot is also defined by the cameras. The algorithms more computationally consuming are executed in a central computer that communicates with robots by radio. The robots are usually equipped with two kicking mechanisms, a chip kicker for higher kicks and a kicker for low kicks, and a dribbler mechanism, that fixes the ball against the robot using the rotation of a roller. The kicker and chipper mechanisms allow straight passes and kicks.

Some teams of the Small Size League of the RoboCup competition have been experimenting with curved kick mechanisms [2], [3], [4], by the combination of the kicker and another mechanism able to curve the speed vector applied in the ball during the kick. The prediction of the ball trajectory after a kick is relatively easy, because it follows a straight line, making it easy to intercept by the enemy team, even before the kick because of the direction of the robot received by the vision algorithm. The addition of an angled kick makes these prediction algorithms to return wrong values and makes this type of kick harder to predict, even more because the angle of the kick is defined by the team that developed the mechanism.

The present work aims to improve the algorithm proposed in [4], that develops a Shallow Neural Network to the Multiple Angles Kicker proposed by SSL ITAndroids team. The referenced work proposed a system integrated with the kicker of the robot, based on the one from Op-AmP team [2]. The work from ITAndroids also discusses the applications of the system and proposes three types of kicks possible because of the device: an angled kick, by simply turning the kicker and kicking the ball, a curved pass, by turning the kicker and applying a rotation to the ball with the dribbler, and the most impressive one, the curved kick with the same strategy of the curved pass.

The angled kick makes a quick shot with an angulation with the robot orientation and the ball follows a straight but hard-to-predict trajectory. The curved pass allows the ball to make a curved trajectory that confuses the ball interception trajectory because of its non-straight nature and aims in an ally. The curved kick is the same process as the curved pass but aims directly at the goal and allows impressive strategies like the Olympic goal.

The trajectory of the ball can be computed by numeric integration and by knowing some parameters, named the kicking parameters: the initial linear speed applied to the ball by the kicker, the direction of the velocity, and the angular speed imposed by the dribbler. But, the decision-making algorithms need the opposite of this, it is more useful if the kicking parameters can be calculated based on a position of the field, known as target, that the algorithm wants to hit with the ball. This inversion of the problem is much harder, especially in real-time calculation than in the original formulation, because of the nonlinear nature of the ODE that models the trajectory of the ball. This is the main motivation of the proposition of a neural network to decide the kicking parameters.

In a straightforward explanation, the ODE is used to generate a large amount of data computed with a set of kicking parameters, this data is used to train the neural network, which then predicts the kicking parameters that can hit a target with a certain precision.

The present work refines the equations that model the movement of the ball, changes the net from a Shallow Neural Network (SNN) to a Deep Neural Network, and presents a new formulation of the field to avoid observed problems in the former formulation of the net.

The ODE, simulations and validation of the algorithm were developed in Matlab and the neural network was

[1]Francisco Arthur Bonfim Azevedo is an Aeronautical Engineer graduated at Aeronautics Institute of Technology (ITA) and master's student of Aeronautical and Mechanical engineering at ITA, arthurazevedo41@gmail.com

[2]Guilherme Pinheiro Cordeiro Leão is an Electronic Engineer graduated at Aeronautics Institute of Technology (ITA) and master's student of Electronic and Computer engineering at ITA, guipcleao@gmail.com

developed in Python 3.7.

The remaining of this paper is organized as follows: Section II models the ODE of the ball's trajectory, Section III prepare the dataset in III-A and proposed a Deep Neural Network to solve the problem of the inversion of the ODE in III-B, Section IV shows the validation of the neural network, Section V shows simulation results of the network, Section VI proposes the adversary avoidance algorithm and finally Section VII concludes the work and propose future works to improve the results presented here.

II. Model of the Ball Trajectory

As discussed in [4], [5], when a ball is propelled on a surface, a phenomenon named rolling friction may be observed. This is because of the ball not being perfectly spherical and because when the ball is kicked it starts slipping, because of the high variation of its linear momentum in a small time instant, and then it starts to roll when part of the kinetic energy is transformed in rotational energy and part being lost due to friction. Finally, it stops when all the kinetic energy is lost.

As calculated in [5], when the ball speed hit around 0.5805 of its initial value when kicked the ball stops slipping and begins to purely roll.

Now, the field is modelled. The reference coordinate system is positioned in the centre of the field to agree with the centre used on the other algorithms of the robot, The reference frame may be seen in Fig 1.

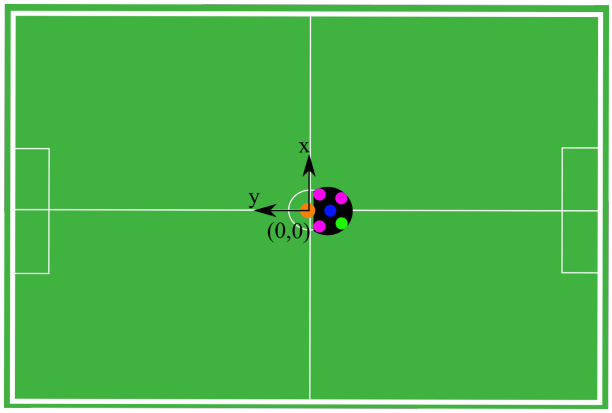
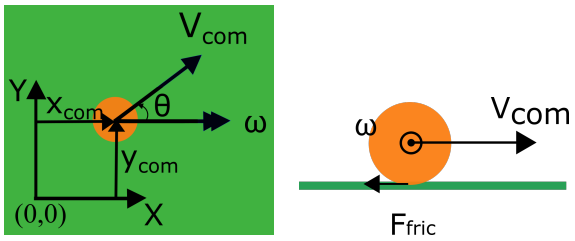


Fig. 1: Reference coordinate system of the field.

The vectors on the ball may be seen in Fig. 2a for the superior view and 2b for the lateral view.



(a) . (b) .

Fig. 2: Kinematic and dynamic of the movement of the ball.

Now, for the manipulation of the equations of the problem, $\vec{V}_{COM}$ is the center of mass velocity, θ is the angle between the the $\vec{V}_{COM}$ and the x axis of the reference system, $\vec{\omega}$ is the angular velocity of the ball, $\vec{V}_{CP}$ is the velocity on the point of contact of the ball with the ground, r is the ball radius, $\vec{F}_{fric}$ is the friction force, μ_s is the coefficient of friction of the ball with the carpet during the slipping and μ_r during the rolling, m is the mass of the ball, g is the acceleration of gravity, $\vec{T}_{F_{fric}}$ is the torque caused by the friction force, I is the moment of inertia of the golf ball, x is the position of the ball in its trajectory, V_0 is the initial speed of the ball, $\vec{\omega}_0$ is the initial angular velocity of the ball on the reference system of the field, (X_0, Y_0) is the starting point of the ball trajectory, $\vec{V}_{kick}$ was the velocity in which the ball was kicked and $\vec{\omega}_{dribbler}$ the angular speed applied by the dribbler.

The ball linear and angular velocity and the velocity on the contact point between he ball and the ground are:

$$\vec{V}_{COM} = V_{COM}(cos\theta\hat{i} + sin\theta\hat{j}) = \dot{x}\hat{i} + \dot{y}\hat{j} \tag{1}$$

$$\vec{\omega} = \omega_x\hat{i} + \omega_y\hat{j} \tag{2}$$

$$\vec{V}_{CP} = \vec{V}_{COM} + \vec{\omega} \times (-r\hat{k}) \tag{3}$$

Firstly, the movement of the ball during the slipping is modelled.

The friction force applied on the ball is:

$$\vec{F}_{fric} = \mu_s mg \left(\frac{-\vec{V}_{CP}}{\left\| \vec{V}_{CP} \right\|} \right) \tag{4}$$

During the slipping, the rotation of the ball is near zero, so the velocity on the centre of mass is approximately the same as the contact point.

The torque generated by the friction force is expressed by:

$$\vec{T}_{F_{fric}} = \vec{F}_{fric} \times (-r\hat{k}) \tag{5}$$

By the definition of torque:

$$\vec{T}_{F_{fric}} = I\frac{d\vec{\omega}}{dt} \tag{6}$$

From (1),(3), (4), (5), (6), the ODE that defines the movement of the ball during the slipping is:

$$I\left(\dot{\omega}_x\hat{i} + \dot{\omega}_y\hat{j}\right) = \mu_g mgr\frac{\dot{y}}{\sqrt{\dot{x}^2+\dot{y}^2}}\hat{i} - \mu_g mgr\frac{\dot{x}}{\sqrt{\dot{x}^2+\dot{y}^2}}\hat{j} \tag{7}$$

When the magnitude of the ball velocity hits 0.5805 of the initial magnitude, the ball starts rolling and V_{CP} is no more the same as the velocity of the centre of mass.

The friction and the torque defined in the rolling movement are defined by:

$$\vec{F}_{fric} = \mu_s mg \left(\frac{-\vec{V}_{CP}}{\left\| \vec{V}_{CP} \right\|} \right) \tag{8}$$

$$\vec{T}_{F_{fric}} = I \frac{d\vec{\omega}}{dt} \tag{9}$$

From (1), (2), (3), (5), (8), (9) the ODE that models the movement of the ball is:

$$\begin{aligned} I \left(\dot{\omega}_x \hat{i} + \dot{\omega}_y \hat{j} \right) = \\ \mu_r mgr \frac{(\dot{y} + r\omega_y)}{\sqrt{(\dot{x} + r\omega_y)^2 + (\dot{y} - r\omega_x)^2}} \hat{i} \\ - \mu_r mgr \frac{(\dot{x} - r\omega_x)}{\sqrt{(\dot{x} + r\omega_y)^2 + (\dot{y} - r\omega_x)^2}} \hat{j} \end{aligned} \tag{10}$$

When the ball hits 0.02 m/s [5], the ball suddenly stops.

The moment of inertia of the golf ball used should be measured for better precision, but it may be approximated by $\frac{2}{5}mg$, which is the theoretical value of the sphere. The coefficients of friction must be empirically estimated, and this is recommended for each different field because the properties of the carpet may change and this may affect the precision.

As it may be verified, in the equations, the position of the robot is not taken into account. This was a strategy to reduce the training dataset and make the prediction easier with fewer states on the field. This is discussed in Section III-A. The process is a simple rotation and translation to align the robot with the axis in the centre of the field, then the ODE is solved with `ode45` of Matlab. The process is better illustrated in Fig. 3. Also, it is important to verify that the angle α includes both the orientation of the robot and the angle that the mechanism applies to the ball during the angled kick and the angle between the x-axis, the default angle for rotation, is $\theta = \pi/2 - \alpha$.

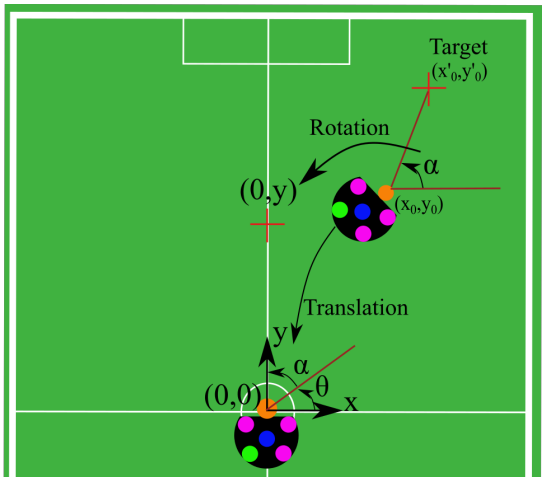


Fig. 3: Illustration of the rotation and translation process during the ODE solution.

Because of these transformations, the initial conditions of the equation are

$$\vec{V}_0 = \vec{V}_{kick} \tag{11}$$

$$\vec{\omega}_0 = \vec{\omega}_{dribbler} \tag{12}$$

$$X_0 = 0,\ Y_0 = 0 \tag{13}$$

Also because of the rotation, one can notice that the angle θ is fixed as the angulation of the speed applied to the ball, taking into account only the angulation applied by the curved kick mechanism, and not the robot rotation.

III. Neural Network

A. Dataset generation

The generation of the dataset is based on a large amount of data generated from solutions from the ODE, but if it was taken into account the position and rotation of the robot in the field, a really large amount of cases would be necessary to represent each state of the robot. To avoid this, a model was proposed.

Firstly, a target was defined to represent the point where the decision-making algorithm is aiming for a specific action, which was named as target. Then, the robot is positioned in the centre of the field aligned with the axis as in Fig. 1. Then, the target is defined as the point where the kicked ball crosses the "y" axis, giving a point of the form $(0, y)$, so only the y_{target}, or only y_t, is really important to be kept. The robot may also be rotated around the z-axis, but with the position fixed. Any other position of the robot may be obtained by a simple translation and any other target may be obtained by rotating and translating the target point.

Also, testing the results obtained in [4], it was verified some characteristics of the problem: it is not at all points of the field that the mechanism can hit, when the ball is too close or too far, maybe there is not a solution inside the tolerance predefined, because of this it is suggested to define an effectiveness area, but in order to do this, it is highly recommended to first make tests with the real robot to better tune the friction coefficients and match and validate the model with the real movement of the ball. As the mechanism is still being projected, this study was not yet made but is a very important future work. Another characteristic noticed is that the ODE presented in Section II generates the same solution for different sets of kicking parameters, so the problem does not have an exact inversion and this heavily impacts the neural network, which will no "mimics" the inversion of the problem, it will "chose" a solution between the sets of parameters that hit a defined target. An illustration of the multiple solutions problem may be seen in Fig. 4.

Here, a strategy is proposed aiming at better integration with the other algorithms of the robot in the future. A discretization of the field is proposed to make easier the integration with the vision algorithm in the future and make the problem closer to other problems solved by Reinforcement

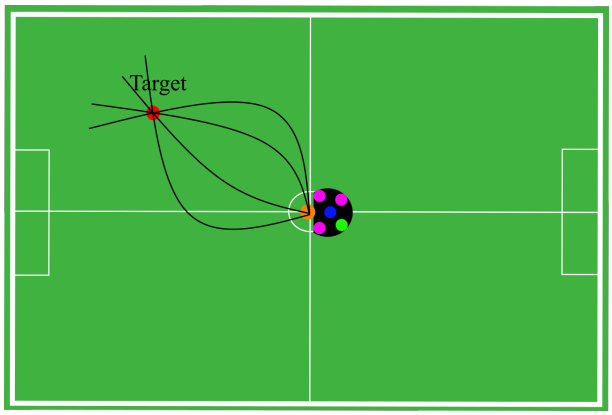


Fig. 4: Representation of sets of kicking parameters that hit the same target.

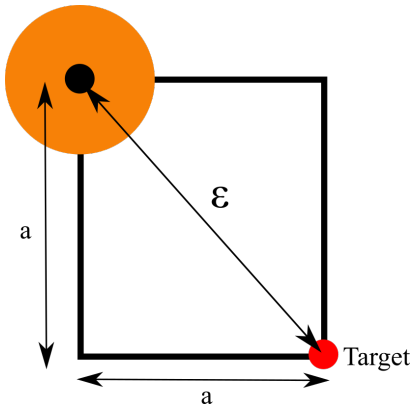


Fig. 5: Representation of a discrete unit, also know as cell, in the discrete field.

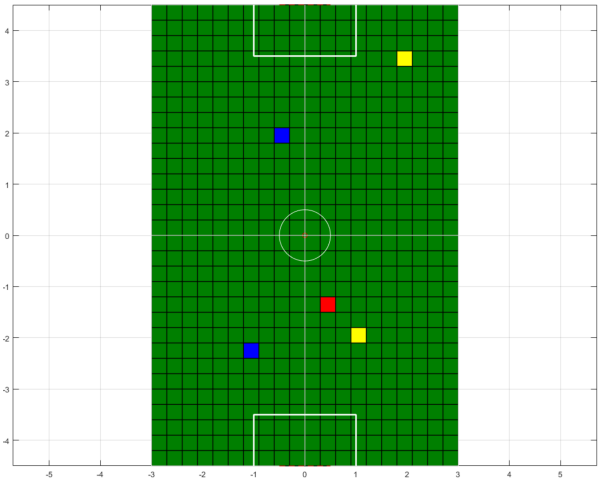


Fig. 6: Discrete field with a tolerance of 0.6 m and a cell of 0.5477 m of side. Blue represents the allies, yellow the adversaries and red the ball.

Learning, as AlphaGo [6], for a future improvement of the algorithm.

The idea is to define cells that represent the group of points inside of it. Intuitively this represents a loss of precision, but it combines sets of solutions that hit a defined target with the predefined tolerance. With this, it's easier to ask the net for sets of kicking parameters that can hit a defined target. To illustrate this, imagine that one wants to hit the target (0, 2.6) m on the field, and the net will try to interpolate a set of solutions that hit (0, 2.5) m and (0, 2.7) m, but they both are solutions inside a precision of 0.3 m for example. Combining the three points inside the same cell and asking the network for three solutions will let the other algorithms to decide which one is the best, for example, if there is an adversary in (0, 2.5), the (0, 2.7) will be a better solution than the others. It also makes it easier to combine with the vision algorithm that can mark allies and adversaries on the field for the future feature of the network to generate a set of solutions that can avoid these obstacles.

To define the dimension of the cells, it is used the tolerance region to consider a point hit. Inside the square cell, the two points with the greatest distance between them are the two vertices with a diagonal between them, and also must be the predefined tolerance to assure the precision of the prediction. Because of this, the side of the square must be $\sqrt{2\varepsilon}/2$. This may be illustrated by Fig. 5. The discrete field may be seen in Fig. 6 with the tolerance of 0.6 m for a better visualization.

To generate the data, it is fixed the position of the robot in (0,0) and the target in the y-axis just to set the rotation of the solution. Then, a large number of combinations of the velocity, the angular speed and the rotation of the robot are used to solve the ODE and the point at which the ball hits the y-axis is kept. If the ball does not hit the axis, it means that it exits the field before a considerable curve, so it is discarded. The data set was generated distributing the V_{kick} between 0.5 m/s and 8 m/s with a step of 0.3, the $\omega_{dribbler}$ between 0 rad/s and 12000 rad/s with a step of 200 rpm and θ between 0° and 360° with a step of 5°. The total of valid combinations used for the dataset was 34548. Then, the point at which the ball hits the y-axis is passed by the discretization process and only the Y is kept because on the y-axis the x is zero.

To tackle the "multi-solution" nature of this problem, a strategy has been used, called "tactic output": given all kicking parameters that hit the desired target, they were ranked taking into account a cost function. This cost function was defined to prioritize the solutions where the kicking velocity ($\vec{V}_{kick}$), the angular speed of the dribbler ($\vec{\omega}_{dribbler}$) and the rotation of the kicking mechanism (θ) were high. The idea is that, during a game, a ball with a higher velocity is harder to intercept and hits the target in a lower time, a ball with a higher rotation is more unpredictable and the robot with a higher rotation confuses more the interception algorithm of other teams. The proposed cost function was:

$$J_{ranking} = a \cdot \frac{|\vec{V}_{kick}|}{|\vec{V}_{kick}^{max}|} + b \cdot \frac{|\vec{\omega}_{dribbler}|}{|\vec{\omega}_{dribbler}^{max}|} + c \cdot \frac{|\theta|}{|\theta^{max}|} \quad (14)$$

In equation 14, $\vec{V}_{kick}^{max}$ and $\vec{\omega}_{dribbler}^{max}$ are, respectively, the kicking velocity and the angular speed of the dribbler with

the highest magnitude from the dataset, and θ^{max} is the highest rotation of the kicking mechanism from the dataset. The hyperparameters a, b and c must be tuned in simulations of games and in real life to see what affects more the tactic of the team, but here, it was used a = 123, b = 21 and c = 10, because ideally, the velocity of the ball is tactically stronger, and then the angular speed of the ball, and then the rotation of the robot, but this does not necessarily represent the reality. Also, the higher the speed, it takes more time to curve the trajectory of the ball, because it takes more time to lose enough kinetic energy to start the rolling movement, and eventually, the ball exits the field, so hardly the therm $\frac{|\vec{V}_{kick}|}{|\vec{V}_{kick}^{max}|}$ reaches 1, so it needs a higher hyperparameter to become more effective than the rotation speed of the ball, that is beneficial to the curve of the trajectory. The tactic parameter may have another cost function defined by the experience of the team, the essence of this parameter is to give higher values to a solution that is more efficient during a real game.

In light of that, each group of kicking parameters that results in a trajectory that hits the target was then ranked from higher cost function (with ranking 1) to lower cost function (with ranking N, where N is the number of kicking parameters of the dataset that makes the ball hit the target) in a called tactic ranking step. The ranking was taken as an output of the network and added to the training dataset. During the prediction step, if one asks for a set of kicking parameters with tactic rank 1, the net will return the parameters with the better tactic ranking that hits the target.

This strategy is an advantage of the discretization of the field because it groups a set of admissible solutions in a given tolerance, and the team may indirectly choose which one is the best for the team's tactic.

Finally, the training dataset matrix is composed by three columns of inputs, V_{kick}, $\omega_{dribbler}$, θ and two columns of expected outputs, the Y of the discrete field where the ball hits the y-axis and the tactic ranking parameter.

B. Structure of the network

To choose what neural network would be the most appropriate to solve the problem, we first considered the kind of problem that we want to solve: to establish a correlation between two groups of parameters, $(V_{kick}, \omega_{dribbler}, \theta)$ and $(y_{target}, ranking)$, being y_{target} the translated and rotated coordinate of the target that we want to hit with the ball. Since this is a simple supervised learning problem, an MLP network with a few layers should be enough to solve it. We studied some deep neural networks with different numbers of hidden layers, different numbers of neurons per layer and compared behaviour for some types of activation functions. The studied networks are shown in Table I. It is worth mentioning that the accuracy of the network was computed via simulation as will be discussed in Section V.

The generated dataset was divided into two groups, training and validation, with a proportion of 80:20, i.e. 80% used for training and 20% used for validation.

TABLE I: The different types of Deep Neural Network studied for the curved kicking problem.

No. of layers	No. of neurons per layer	Activation functions	MSE	Accuracy
3	75 (Layer 1) 30 (Layer 2) 3 (Layer 3)	LeakyReLU LeakyReLU Linear	27447	23%
3	75 (Layer 1) 30 (Layer 2) 3 (Layer 3)	Sigmoid Sigmoid Linear	30091	42%
10	75 (Layer 1) 30 (Layer 2) 40 (Layer 3) 50 (Layer 4) 40 (Layer 5) 30 (Layer 6) 20 (Layer 7) 20 (Layer 8) 20 (Layer 9) 3 (Layer 10)	LeakyReLU LeakyReLU LeakyReLU LeakyReLU LeakyReLU LeakyReLU LeakyReLU LeakyReLU LeakyReLU Linear	25161	90%

In order to avoid overfitting, it was decided to apply L2 regularization during training [7], which lowers the complexity of a neural network model during training. Also, the training was performed in batches of 50 during 100 epochs. The Adam optimizer was also applied with a learning rate of 0.001. The training and an initial validation were both made in python using Keras version 2.8.0 [8].

In light of the results shown in Table I, the third neural network was the one proposed for the following analysis, given the smaller MSE and higher accuracy. Its structure is shown in Figure 7.

IV. Validation of the network

A dataset of 34541 solutions for the curved kick equation was used, being 80% to train it and 20% to validate. The Optimization algorithm used was Adam with L2 regularization to avoid overfitting. The cost function used was the MSE, and all activation functions of the proposed network were LeakyReLU with $\alpha = 0.01$, except for the output layer, which was chosen to have a linear activation function. The neural network's performance during training may be seen in Figure 8. The final MSE and the simulation accuracy (based on the approach that will be discussed in Section V) are shown in Table I.

V. Simulation results

A set of 10.000 simulations were made to also validate the performance of the network. In each simulation, the error of the network, defined as the minimum distance between the ball trajectory and the target, was computed. if this error was less than the tolerance requirement, the target was considered hit. Here we are assuming that if e.g., in a pass, a small error is made, the team's robot that will receive the ball is

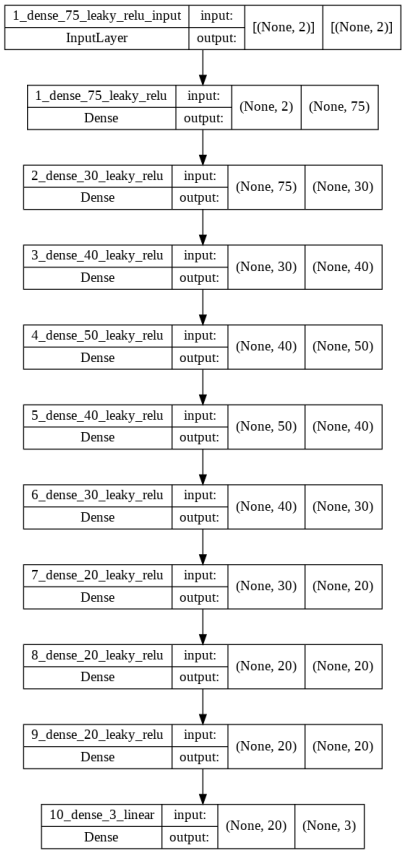


Fig. 7: Final proposed neural network.

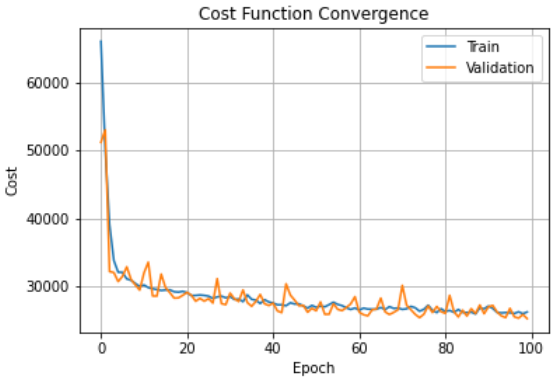


Fig. 8: Performance of the proposed neural network.

still capable of quickly adjusting himself to catch the ball. During our simulations, we defined the error requirement to be 0.6m, and the position of the robot and the target were both randomly generated. This performance validation was made in MATLAB and the results are shown in Figure 9. Figure 10 graphically illustrates one of the simulated results, where the SSL robot is the blue circular object on the top half of the field, the target is the circular blue mark in the bottom half of the field, and the simulated trajectory of the ball is the curved line in green. Furthermore, the dashed circle shows the limits of the tolerance error used to consider whether the ball was hit or not during the simulation.

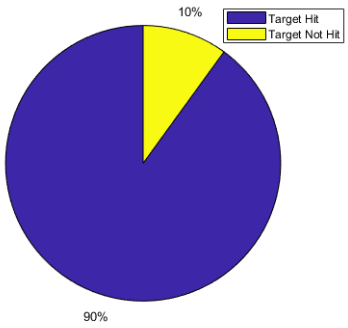


Fig. 9: Performance of the proposed neural network during simulations made in MATLAB.

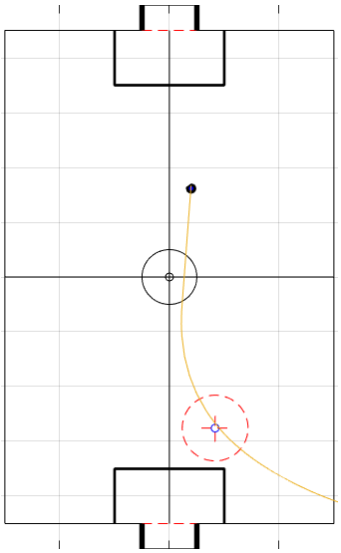


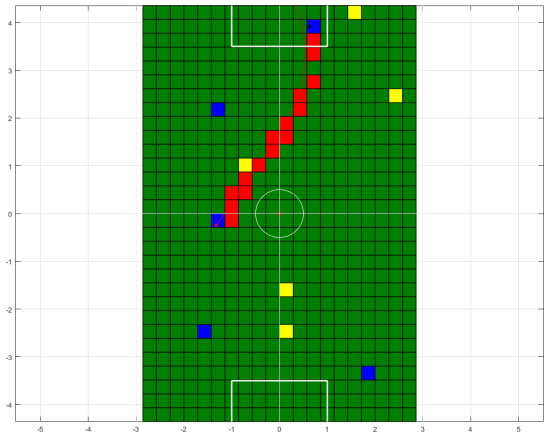
Fig. 10: Ball trajectory simulated using the proposed network.

VI. Adversary Avoidance Algorithm

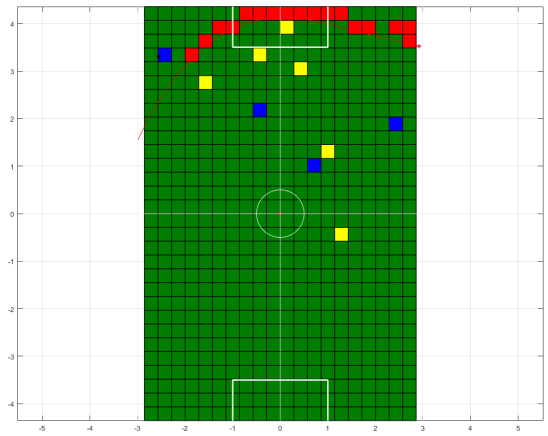
Using the concepts of the discrete field and the tactics ranking, an adversary avoidance algorithm was proposed. The main idea is to combine the network with decision-making and vision algorithms, so the version shown in this section is just a draft to propose the algorithm.

Using the matrix that keeps the cells of the discrete field, the allies and the adversaries are positioned, then, 2 allied robots are selected as initial position of the ball and the other as target, supposing a pass situation. The idea is to predict the path of the ball with the network and tactic rank 1, populate the matrix of the field and then verified if an enemy is hit. If an enemy is hit, ask for a prediction with tactic rank 2, the second best solution, and verify again. This is made until a maximum number of iterations is made, and if it does not find a solution, it may look for another ally to pass or walk with

the ball for example. Two simulations with this algorithm are shown in Fig. 11. In that figure, the tolerance was 0.6 m, the blue squares are the allied robots, the yellow ones are the adversaries, the red squares represent the movement of the ball, the red line is for better visualization of the true trajectory, with a red dot representing the start and black point the target. In Fig. 11b, the kicking robot may not be seen because it is on the border of the field, which is a problem of the discretization algorithm. Also in Fig. 11b the ball is a cell away from the represented robot, but the algorithm represented a hit because the visualization of the field is not perfect. These are two problems with the discretization and two opportunities for improvements in the method.



(a) First case of adversary avoidance algorithm with tactic rank 2.



(b) Second case of adversary avoidance algorithm with tactic rank 4.

Fig. 11: Simulations with the adversary avoidance algorithm.

VII. Conclusion and Future works

In the present work, the equation of the rolling ball in the field is presented and used to propose a way of hit a target with a pre-defined tolerance using a Deep Neural Network. This is an improvement on a previously proposed algorithm presented in [4], with the addition of techniques to better work with the multi solutions of the inverse of the ball's trajectory equation and an algorithm to avoid adversaries when kicking the ball.

The model of the ball's trajectory followed what was expected in previous works [5], but some parameters need to be calibrated with a real field during the competition. Then, the solution was used to generate a dataset in the discrete field, which was a proposition that permitted to choose between the solutions in the pre-defined tolerance using the tactic ranking parameter. A large data was generated using the model of the trajectory and this was sent to a Deep Neural Network, that worked as the inversion of the ball's trajectory equation. Then, the results were obtained with a 90% precision. With the trained network, an algorithm that, by trying various tactic ranking parameters, aims to find the trajectory that avoids adversaries on the field was proposed. The precision still needs to be improved using the presented formulation, but the problem was successfully solved and new features were added to it.

In future works, the main priority is to increase the percentage of target hits with lesser tolerance than the presented model. Also, tests of the algorithm implemented in the robot with the mechanism and on the real field are intended. Validation of the model of the trajectory and of the network is essential to validate the power of the presented strategy. Also, another improvement after the validation of the work is to evolve the algorithm into a Reinforced Learning that may study a large number of states on the field and learn how to kick the ball in each state.

References

[1] Small Size League Technical Committee, "Laws of the RoboCup Small Size League 2018," *RoboCup 2018*, 2018.

[2] T. Yoshimoto, T. Horii, S. Mizutani, Y. Iwauchi, Y. Yamada, K. Baba, and S. Zenji, "OP-AmP 2017 Team Description Paper," Tech. Rep., 2017.

[3] G. Oliveira, V. Alves, D. Pilotto, L. Silva, J. Drapella, D. Vassoler, I. Alves, R. Bezerra, F. Darsono, W. Motta, M. Daltro, G. Garcia, L. Costa, L. Pinto, G. Pauli, M. Laureano, C. Cadamuro, A. Santos, R. Bianchi, P. Junior, and F. Tonidandel, "RoboFEI 2018 Team Description Paper," Tech. Rep., 2018.

[4] A. Azevedo, B. Mirahy, E. Moreira, J. Nascimento, M. Santos, M. Marcos, R. Lima, R. Linhares, and V. Gabriel, "ITAndroids Small Size League Team Report 2018," 2018.

[5] R. Rojas and M. Simon, "LIKE A ROLLING BALL," Tech. Rep., 2015.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] S. Gupta, R. Gupta, M. Ojha, and K. P. Singh, "A comparative analysis of various regularization techniques to solve overfitting problem in artificial neural network," in *Data Science and Analytics*, B. Panda, S. Sharma, and N. R. Roy, Eds. Singapore: Springer Singapore, 2018, pp. 363–371.

[8] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras