RoboIME: From the top of Latin America to RoboCup 2020

Lucas G. Corrêa, Carla S. Cosenza, Gabriel B. da Conceição, Felipe W. V. da Silva, Antônio S. G. Pereira, Gabriel T. Pinheiro, Gustavo A. Testoni, Davi H. M. Pontes, Daniel S. C. Bello, Ana C. A. Monteiro, José L. de O. Schramm, Gustavo C. K. Couto, Leonardo G. Gonçalves, Lucas B. Germano, Luiz R. L. Rodrigues, Guilherme M. Gonçalves, Vinicius de F. L. Moraes, Gabriel H. M. Silva, Mayara R. Mendonça, Ana L. B. da Silva, Matheus Bozza, Luis D. P. de Farias, João G. O. C. de Melo, Nicolas S. M. M. de Oliveira and Paulo F. F. Rosa

Instituto Militar de Engenharia, Rio de Janeiro, Brasil

rpaulo@ime.eb.br
http://roboime.com.br

Abstract. This paper describes the electronic, mechanical and software designs developed by the RoboIME Team in order to join the RoboCup 2020. The overall concepts are in agreement with the rules of Small Size League 2020. This is the seventh time RoboIME participates in the RoboCup.

1 Introduction

RoboIME is a Small-Size League team from the Instituto Militar de Engenharia, IME - Brazil, and this is the 15th time the team participates in a competition. The team already have gotten good results: (i) first place in the Latin American Robotics Competition 2017 (LARC 17); and (ii), six second places in RoboCup Brazil Open 2011, LARC 2012, RoboCup 2018 division B, LARC 2018, RoboCup 2019 division B and LARC 2019.

All students that work in the SSL project are members of the Laboratory of Robotics and Computational Intelligence at IME. Team's previous works were used as reference [3] , as well as the help from former members of the team as consultants and tutors.

This article describes the team's general information and improvement in the most recent semester, since our previous TDP for RoboCup 2019 has detailed explanations on our previous changes. This article is organized as follows: software in section 2, embedded eletronics in section 3 and mechanical design in section 4. Conclusions and future works are discussed in section 5.

2 Software Project

This paper reports the main improvements and changes since 2019 RoboCup project.

The main focus of the project was to rebuild all artificial intelligence with a hierarchical architecture, based on the STP(Skills, Tactics and Plays) structure, aligned with object-orientated programming. Hence, few changes in robot's behavior were implemented in order to obtain a functional and organized project. Our architecture implementation was based on the following references: [1] and [2] .

In addition, it were done two main implementations in vision treatment. The first is the treatment of false balls in vision, for example people in the field and cameras' calibration imperfections. The second is about merging duplicate information of any element (ball or robot) due to cameras intersection regions in field.

2.1 New Software Architecture

It was necessary to transform and innovate the form of treating the intelligence programming at RoboIME, so in these terms, there were great structural changes in the code, since it was reformulated for object-oriented programming. In this aspect, it was possible to modify the way of approaching the artificial intelligence, following the STP architecture.

The behavior of the robot is divided in a hierarchical form. In a summarized way, it can be said that: Skills are the pillar of the architecture since they are the minimum robot actions. Skills are selected by a State Machine that gives the robots orders through a set of skills. This State Machine has the name Tactic. In this sense, a sequence of Tactics controls the robot's behavior in each determined Play. Besides, Play is the architecture's component that dictates, through a sequence of Tactics for each robot, the team behaviour at any given moment, such as a pass state or a situation to steal the ball with opponent.

The main STP execution algorithm can be seen in figure 1

Process STP Execution
1. CaptureSensors()
2. RunPerception()
3. UpdateWorldModel()
4. $P \leftarrow$ ExecutePlayEngine()
5. **for each robot** $i \in \{1, \ldots, N\}$
6. $(T_i, TParams_i) \leftarrow$ GetTactic(P, i)
7. $(SSM_i, SParams_i) \leftarrow$ ExecuteTactic(T_i, $TParams_i$)
8. **if** NewTactic(T_i) **then**
9. $S_i \leftarrow SSM_i(0)$
10. $(command_i, S_i') \leftarrow$ ExecuteStateMachine(SSM_i, S_i, $SParams_i$)
11. $robot_command_i \leftarrow$ ExecuteRobotControl($command_i$)
12. SendCommand(i, $robot_command_i$)

Fig. 1: STP execution algorithm [1]

2.1.1 Plays are the highest level element in the STP architecture, they define the strategic behaviour of team. Different from the architecture implemented in the common STP, as specified in [1] and [2] , in which it was used exactly one play for the entire team, the STP version implemented in this project uses three simultaneous Plays, for the Keeper Team, the Offensive Team and Defensive Team. They both work independently to gain more liberty of action during the game. From the programming point of view, each Play is a class inherited from a parent class named Play, and each Play has two methods: one to check the finish conditions and another to check the begin conditions.

Finish conditions specify when the Play's execution should stop; begin conditions stipulate the requirements that the game must satisfy in order to execute the Play. Both of them are based on the referee command, ball's and robots' position.

A certain state can satisfy the begin conditions of multiple plays or there may exist a situation in which two or more plays are equally likely. To solve this problem, a score system was created to choose among these plays, that will be explained in the next topic.

The Artificial Intelligence in code is divided in three modules, as seen in figure 2. Each subteam (Keeper, Defensive and Offensive) has its Artificial Intelligence in parallel.

Fig. 2: Artificial Intellignece Modules

2.1.1.1 Choose Play is the first module of the Play selection system, and it's responsible for switching among Plays if the game situation requires.

The first step is to verify if the current Play's finish condition is satisfied; if not, the current play maintains its execution. After the finish condition occurs, the second step is to select every play that has the begin conditions satisfied by the current state of the game. The third and final step is choose the play with higher score in the score system.

The score system was created to choose among various plays with begin conditions satisfied and it was based on [2] . Each Play has initial score hardcoded, forming a priority list. The goal is that during the game, as this plays succeed or fail, points are added or subtracted to update the list and modify the priority among plays.

The diagram of the process can be seen in figure 3

2.1.1.2 Pick Robots is the module responsible for, after the Play has already been chosen, coordinating the process of choose of robots from all robots of entire team to the sub-teams, and inside each sub-team, associating the robots to roles.

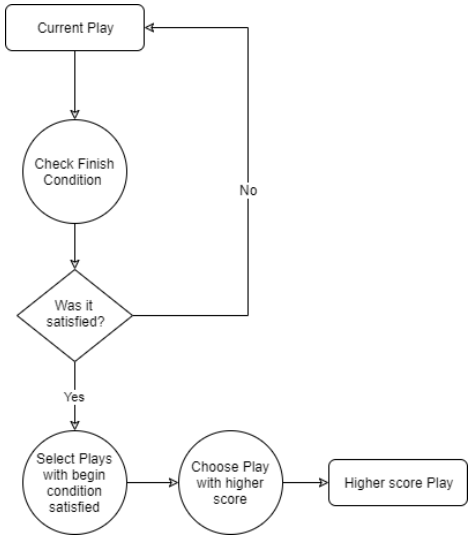

Fig. 3: Choose Play Module Diagram

Initially, in a previous part of the code, all robots of our team are sorted in a vector. At this point, the robot's position and the ball's parameters are used to determine which of them is more favourable to be in a offensive posture. After that, the robot assigned to be the keeper is placed in the top of the vector and the rest of all team are allocated at end of the vector. This vector is shared to the sub-teams using parameter passing by reference. So, in parallel, each sub-team chooses their robots.

Keeper Team choose the first robot in vector. The Offensive Team chooses the necessary quantity from the top of the vector, excluding the robot chosen by Keeper Team; the quantity of robots to Offensive Team depends on which Play is in execution and it is a attribute of each one. The Defensive Team chooses all robots of the vector excluding that chosen by Keeper and Offensive Teams. The information about which robots each sub-team chose is passed by reference to all three sub-teams. How the process described is done in parallel, it may happen, for example, that when the Defensive Team will to choose its robots, the Offensive Team don't chose yet, so Defensive Team will to choose incorrectly, but it will occur just few iterations until Offensive Team chooses its robots.

After, each sub-team associates each respective robot to a role. A role consists of behaviors for the robot to perform in sequence, which are the Tactics, and they will simply be executed until one of the finish conditions apply.

2.1.1.3 Execute Play is the module responsible, after a Play has been chosen and every robot has already been associated to a role, for proper executing the Play's content and assigning content into robot behavior. The module starts to execute each Tactic of the roles.

The STP architecture requires that all robots advance to its next Tactic simultaneously. As an example, consider the Pass Play: one of the robots positions itself to perform the pass and the another one position itself to receive the pass. When these robots have already positioned themselves, both advance to its respective next Tactic. So, the first robot executes the pass while the another one makes adjustments to its position to ensure that it is in the ball's direction. When the receiver robot get the ball, both advance to its respective next Tactic. So, the first robot position itself in strategic position to follow the attack when the another one shoots the ball in order to try make a goal.

2.1.2 Tactics are a Skill State Machines. As said in Plays subsection, when a Play begins, each robot receives a sequence of Tactics (Role).

For example, in Pass Play, one of the robots has a sequence with two Tactics: WaitPass (Tactic responsible for positioning the robot to receive the ball) and Shoot (Tactic responsible for shooting the ball towards the goal after having the ball under control).

Meanwhile, the other robot has a sequence with two tactics: KickToReceiver (Tactic responsible for performing the pass to the receptor robot) and FollowAttacker (Tactic responsible for positioning the robot, after it has performed the pass, in strategic location near the robot with Shoot Tactic)

In this Play, a third robot has a sequence with one Tactic: PassObserver (Tactic responsible for positioning this robot in a strategic location to attract enemy robots in order to reduce number of enemies near the robot that will receive the ball and try to score the goal).

From the programming point of view, each Tactic is a class with just one method whose objective is to execute the Skill State Machine. Then, in the Tactic method there isn't any logic of the robot's decision or action explicitly, there is only its Skill State Machine operation: state execution call and states transition logic.

It is worth mentioning that all Tactics are free, that is, a Tactic can be inserted in various sequences of Tactics (Roles).

2.1.2.1 Tactics Management is an important point in this Structure. A robot just advances to the next Tactic in its sequence if all other robots of the current Play can too. This is a manner to coordinate all robots' actions in the Play's execution.

To do this, after each iteration of each robot Tactic, it informs the number of Tactics in its sequence that it is enabled to execute. This number can be i (number of current Tactic, that is, current Tactic is not finished yet), $i + 1$ (number of next Tactic, that is, current Tactic is already finished) or -1 (the current Tactic is already finished and it is the last Tactic in sequence of this robot).

By the numbers distinct from -1 informed by each robot, it is caught the smallest and this number is the Tactic index for the next code iteration for all robots. In the first iteration, all robots are initialized in Tactic 0.

For each robot, if the Tactic index informed to current iteration is bigger than the last Tactic index, this robot executes its last Tactic. This occurs when two or more robots have different number of Tactics, then if a robot has less Tactis than other, this robot continues executing its last Tactic while the other will advancing in its Tactic sequence.

2.1.3 Skills are the robot's actions, they are the component of Structure that effectively performs calculations and logic in order to set the robot destination pose and determines when the robot will activate the kick or dribbler.

Some Skills are joined to build Skill State Machine (Tactic). To clarify, consider the Duelist Play (a Defensive Play in which a defense robot goes against ball to put it away). In this play, one robot receives a sequence with one Tactic: Shoot. This Tactic has two states: KickToGoal (robot go to the ball and positioned to kick in a calculated point of goal) and StealBall (robot goes to the ball with kick activated but its positioning is calculated in order to block a possible enemy kick). The choice of these states depends on the distances of the enemy and ally to the ball.

From the programming point of view, each Skill is a class with a main method that generates the robot destination pose and possible auxiliaries methods inside this main method to subdivide the logic and calculations.

2.2 Treatment of possible balls in vision

2.2.1 Why is ball treatment necessary? During the game, there are some calibration mistakes, for instance, something that is not a ball is understood as a ball due identification of orange color. Hence, a person near the eld or even a robot itself can be mistaken for the ball due to cameras' calibration imperfections. Other motivation is to avoid that other balls that appear in field during a test confuse the robot.

Therefore in the game the code must prevent that the robots stop or be confused due to the appearance of false balls in vision. The diagram of the process can be seen in figure 4.

The diagram follows a logical sequence, since if there is no ball in the last frame, any current ball in the field will is possible, but if there is ball in the last frame, the code will create a region in which the current ball can be possible, so balls within this region are possible and if there is no ball in the current time or there is no ball within this region, the lifetime will be activate.

2.2.2 Lifetime was created to prevent oscillations in the existence of the ball in code, since there are moments that due to the bad camera calibration the ball disappears. So, the lifetime shows the most probable position of the ball using Kalman Filter. So it keep the imaginary ball if threre doesn't exist a possible ball for a small predetermined time when it will eliminate this ball so that any ball in the vision becomes possible, if in that time a possible ball appears, the lifetime will be disabled.

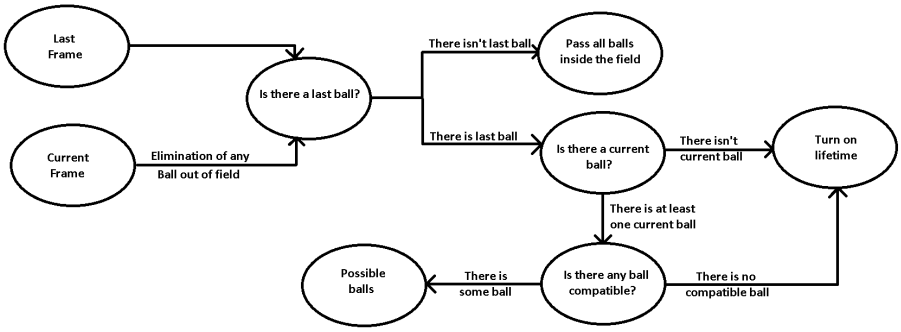

Fig. 4: Treatment of possible balls diagram

2.2.3 Possible balls are the balls received from vision that the code considers real balls. They are chosen by the zone which the old ball could reach. In this calculation, the position of the last ball, the maximum ball velocity and the average code processing (frames per second), that offer the time between frames are used to determine the zone. So if the balls received by vision in current frame are within the zone they will be considered possible.

2.3 Choice of cameras

In order to improve the position's precision of the robots and the ball on the field, a layer of camera filtering was implemented, in the form of a choice of cameras.

Considering a field with 4 cameras, numbered from 0 to 3 anticlockwise from top left, we partitioned the field into the regions as demonstrated in 5.

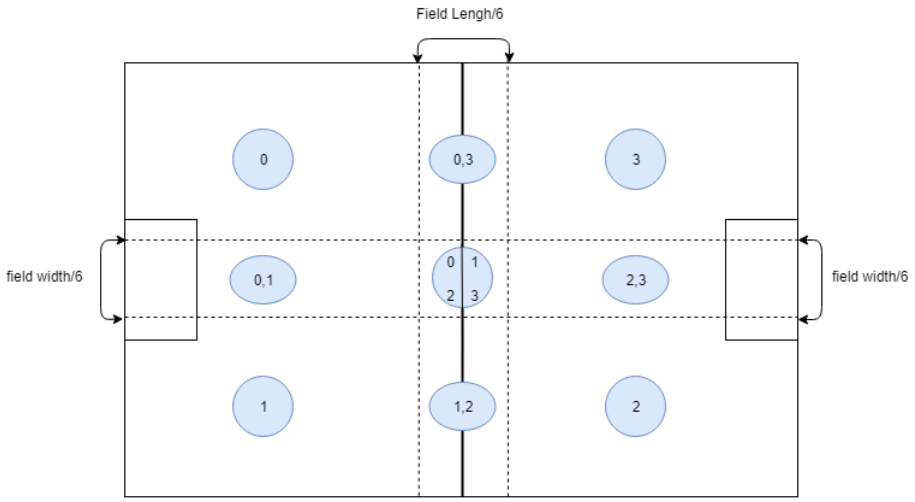

Fig. 5: Camera's choice field division

Depending on the region of the field that the object in analysis is, only the cameras depicted in the blue circles will be considered to calculate its position.

Also, when more than one camera is considered, the average position is calculated, weighted by the inverse of the distance between the camera and its image of the object in analysis.

Furthermore, if none of the selected cameras have vision of the object, but other cameras do, then the average between all the cameras with the objects vision the position to be considered instead.

It is worth mentioning that this camera configuration around the field was used because this improvement of code was done for LARC2019. So, to RoboCup2020, the calculation will be done considering the respective camera configuration around the field.

2.4 2nd Striker positioning

During the offensive plays of Direct Kick and Indirect Kick, there are three robots on the Offensive Team. The first one is named Attacker, that is the robot that executes the shoot or pass. The other robots are named Striker and 2nd Striker, that are the robots that position themselves to be an option of pass receiver. After the robots position themselves, it is done an analyze of the situation, and it is chosen which robot, Striker or 2nd Striker, will receive the pass or opt to a direct shoot to goal.

This topic is responsible for explaining the new 2nd Striker's positioning algorithm.

Its position is defined by searching for the biggest gap in the enemy defense, without going further, in order to, if there is a counter-attack, come back and help the ally defensive team.

To find the biggest gap, first of all, the offensive field is divided in three zones (figure 6), in order to restrict the 2nd Striker's advance. The alpha angle can be changed on demand, but it was decided to define, at first, the line going from the center of the enemy goal to the point P (L/6, -h/2), where L is the field length and h is the field height.

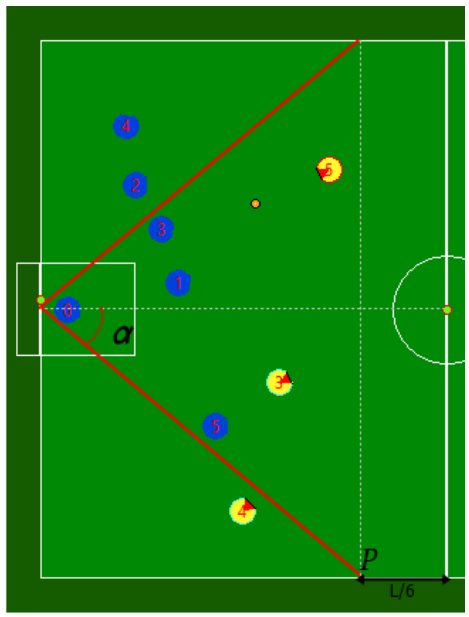

Fig. 6: The three zones of the offensive field

Subsequently, the value of the angle between the adjacent lines that go from the center of the goal to the enemy robots is calculated (ignoring the enemy goalkeeper)(figure 7). If all the enemy robots are inside the zone determined by the red lines showed in the previously figure (figure 6) those lines will be taken into account in the calculation of the angles.

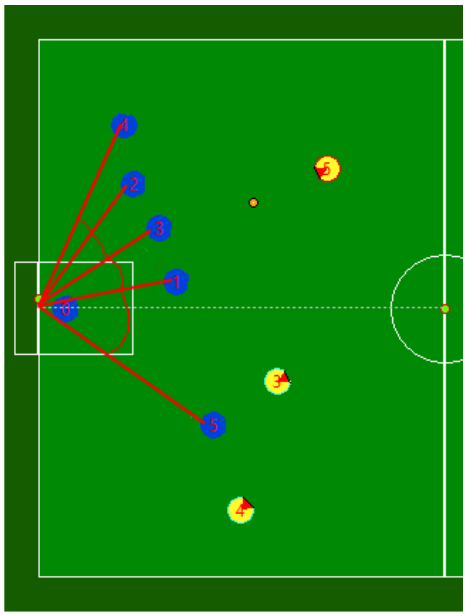

Fig. 7: Lines from the center of the goal to the enemy robots, creating the angles of the gaps

After calculating those angles, it is chosen the biggest gap, in other words, the one that has the biggest angle. Then, it's calculated the incentre of the triangle defined by the lines that go from the center of the goal to the robots that define that gap, and the line that goes from the center of the field to the point P (L/6, -h/2), or P' (L/6, h/2), depending on the position of those robots (figure 8).

If the incentre of that triangle is outside the mid zone (figure 6) that gap is not considered in the search for the best gap to place the 2nd Striker. Subsequently, it is calculated the angle between the ally robots near the edges of the mid zone (the ones that have the higher and the lower y) and those edges (figure 9).

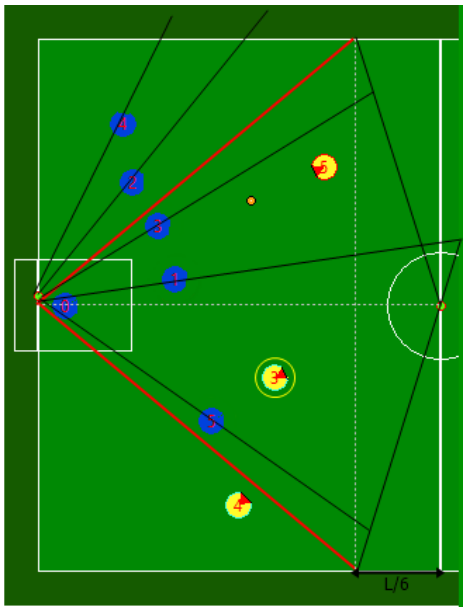

Fig. 8: Calculating of the incentres of the triangles

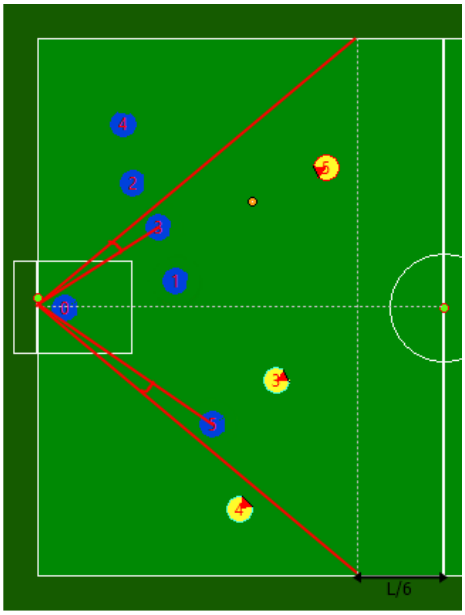

Fig. 9: Angle between the ally robots near the edges of the mid zone and the actual edges

Then, from those gaps whose incentre is inside the mid zone and the two gaps of the edges, shown before, it is chosen the biggest and it is calculated the associated incentre, which will be the 2nd striker position.

However, the 2nd Striker can't be too close to the Striker. In that case, if the 2nd Striker is at a distance less than 1/10 of the diagonal of the field from the Striker, the first one moves towards the center of the filed, until the distance pre-established is respected.

Another situation that must be avoided is if the 2nd Striker is in the front of the ball, getting in the way of the direct kick. Then, if it is at a distance less than 250 millimeters from the line that goes from the ball to the center of the goal, the robot moves towards the center of the field, following the line perpendicular to the previous one.

2.5 PassObserver

PassObserver is the name of the role given to the robot that was one of two possible pass receivers but, after having started a pass, will not to receive the ball. It has as purpose the distraction of the enemy team, trying to attract markup to itself in order to facilitate the execution of the Play, which is composed by pass followed by shoot.

During this Play, one robot will accomplish the pass (PassKicker) and other will be responsible for receiving the pass (PassReceiver) and the third robot of

Offensive Team is called PassObserver and the last will be used to maximize the chances of the pass be succeeded.

For this role to work properly, it is need to consider some aspects: the place where the PassObserver will be positioned needs to be near a place where the probability to find the enemy's defense is very high, the direction of the shoot to goal must be known and the robots position need to be considered. These things are used to minimize the chances of the team itself getting in their own way, this could cause a deviation to the ball course which have lead to an ineffective play.

Based on what was said in the last paragraph, it are studied two different cases: the first case occurs when the PassObserver is positioned in a place which will not cross the ball's pass course, the second case is the complementary of the first one.

– **First Case**
  To help the understanding of this explanation we will use these abbreviations to define vectors and points on the field (figure 10).
  PK - PassKicker
  PR - PassReceiver
  PO - PassObserver
  G - Goal Center

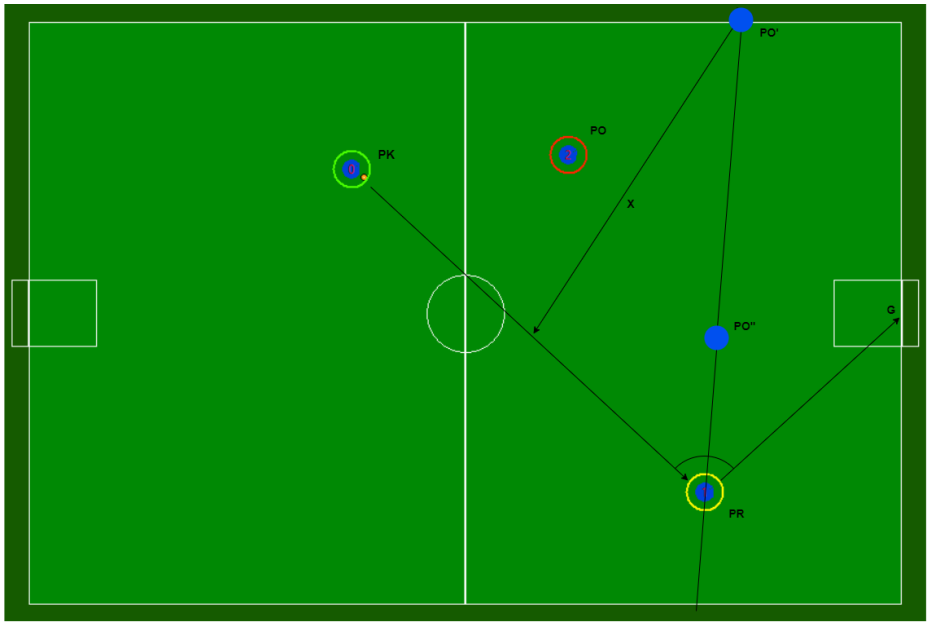

Fig. 10: Situation in which the first case is used

In this case, it can be seen that the PO is positioned above the PKPR vector because of this it will be consider in the first case - because it will not interfere in the pass between PK and PO. The new position of PO (PO') must be on the line defined by the vector positioned on PR with direction on the angle bisector between PKPR and PRG - this line is defined based on the positions of PK, PR and G. The distance between PO and PR is defined as the same distance between PK and PR but some parameters must be checked before that. The parameters and what is need to do are defined here:
- If PO' is positioned outside of the field:

The position PO' will change for the new position PO" which is based on the previous position. PO" is defined by the point PR and the vector $\frac{PRPO}{3}$.
- If PO' is positioned inside of the goal area:

The position of PO' will change for the new position PO" which is based on a vector X previously calculated. The new position is calculated as the result of the vector from the position PO' added with the X vector.

– **Second Case**
To help the understanding of this explanation, it will be used these abbreviations to define vectors and points on the field (figure 11).
PK - PassKicker
PR - PassReceiver
PO - PassObserver
DW - Defense Width
FW - Field Width

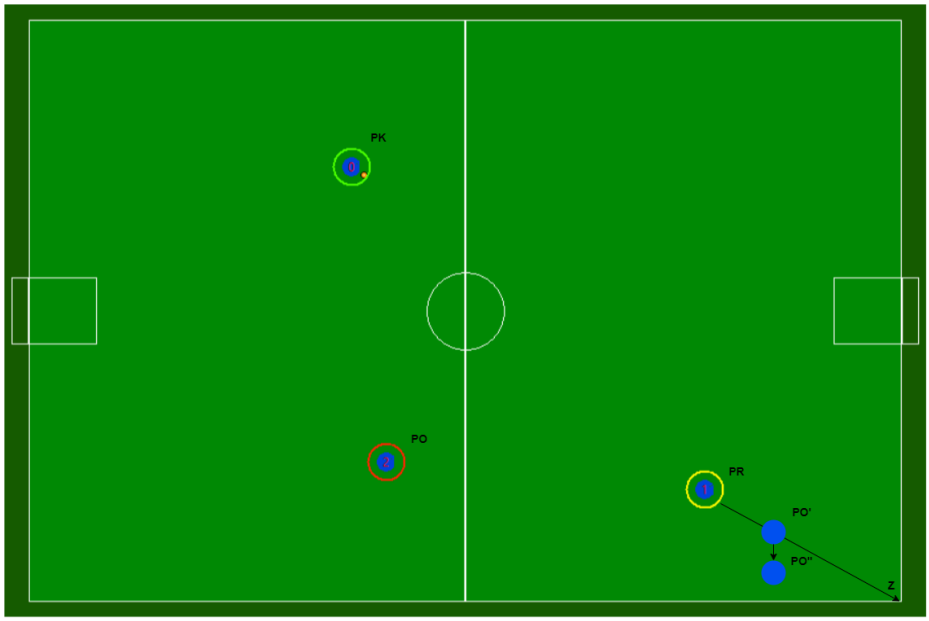

Fig. 11: Situation in which the second case is used

In this case, the PO is positioned below the PKPR vector. So, it will be consider the second case - because it could interfere on the play if it was defined as the first case. The new position of PO (PO') must be defined based on the positions of PR and the corner of the field that is closest to the PR location (Z). The PO must be positioned on the line formed between PR and Z, the distance to Z is calculated as 3*DW divided by the cosine of the module of vector PRZ (PO') to this need to sum on the y coordinate the value of $\frac{FW}{3}$ (PO").

3 Electronics Project

For the RoboCup 2020, the project comes with changes in board design and firmware. The changes in the design were made with the objective of making

the robot more robust. The additions and modifications made to the firmware focused in fixing control issues and creating a new telemetry system that allows receiving data from the robot. One final upgrade was the replacement of older parts such as motors witch due to different times of use were showing different outcomes to the same input.

In the 2018 project, major changes were made focusing the robot's preexisting features more reliable. Despite of that, new additions and lots of improvements were made in the project, such as modularization and changes in the power supply. In the final product, some problems carried from the 2016 model and a few related to the 2018 edition were observed. Fixing those problems and adding new features motivated the creation of the 2020 model.

3.1 Firmware

The firmware was the part that changed the most. The changes focused the modification of the control routine to optimize the execution of commands and addition of a new telemetry system that allows checking robot status and sensors readings without the need to opening the robot.

3.1.1 Robot Communication Changes in the communication were the fundamentals for the creation of the telemetry system. The major changes happened in the timings to avoid loss of packages and enabling backward communication to send data of robot status and readings of the sensors. The feedback packages include battery charge, wheels speed, ball possession, kicker charge, gyroscope readings and current measurements from critical parts of the circuit. Those changes were made looking for fixing package loss issues, making debugging easier and making the AI work better with the extra data.

3.2 Control

Control was one of the biggest issues observed in latter versions, therefore big changes were made in the way calculations, data filters and signal generations are made in the firmware. Those modifications focused in making the movement more stable and the responses to errors and trajectory corrections faster. Related to controlling issues, the motor themselves were a problem in older projects, since they were bought in different years and used for different times, they presented different responses to the same input. Therefore a measure taken to prevent those kind of issues was the acquisition of all the motors from the same batch to guarantee a similar output to the same entry signal.

3.3 Board Designs

Few modifications were made in the 2020 project, the main ones being the addition of adjustable resistors to regulate the ball possession triggers, changes to

the IMU trails, since the model used in 2018 version had it's production discontinued, addition of fuses and optical connectors to the kicker charger to make the charger circuit more robust and isolating the motherboard from the kicker module discharges and addition of tank capacitors to the communication module to avoid problems due to transient state when the dribble was connected.

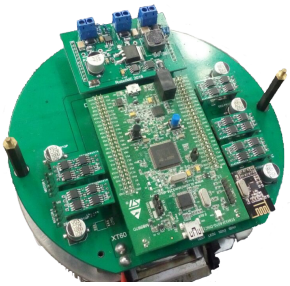

Fig. 12: Picture showing all boards: the kicker module at the top, the stamp module at the center, five motor modules at the sides and one communication module at the corner. Beneath all, the main board.

The motor module was changed to be able to deliver the maximum power that the motors may demand, maintaining the same pinout as it's older version to keep the modularity of the project.

3.3.1 Stamp module The module is the STM32F407 – Discovery, a development kit that includes an Arm Cortex M4 and other peripherals sensors, USB plugs, debugging LEDs, push buttons, motion sensors and others. This module is responsible for performing calculations and coordinating the signals sent and received from the different parts of the robot, like motor drivers, kicker module, communication board and sensors.

3.3.2 Main Board The Main Board, figure 12, provides physical support to the modules and the connections between them and the robot's actuators, sensors and power supply. Most of the main board is composed of simple routes and planes that make these connections. It also implements some important circuits, such as the currents that flow to the motors and the side circuitry for CI's. Further changes were made to improve the 2018 model. An important one was the thickening of the tracks and the addition of tank capacitors which was implemented after a power loss test and powering down of the communication module when the dribbler was powered. This delivers more current for the motors, since track's resistance is decreased. Also, the capacitor can sustain the power to the communication module as the dribbler runs, avoiding problems due to the transient state. Likewise, the MPU-9250 was replaced, since the manufacturing of

the model was discontinued. Another change was the addition of adjustable resistors to set the trigger of the ball possession sensors. Other additions were fuses and optical connectors to protect the main board circuitry from the kicker discharges.

3.3.3 Kicker module In the 2020 model, the kicker board circuit has been improved, to become more robust and safer. The circuit is controlled by the LT3750 IC, which charges a 2200uF capacitor up to 190V in less than 5 seconds. As the board takes approximately 5 seconds to charge up the capacitor, the firmware is programmed to send a charge signal in 6 seconds interval. It also sends the step inputs after a kick is executed. Also a resetable fuse was added to protect the charging circuit and a optical connector to send the charging signal, in order to isolate discharge circuity from the main board, therefore protecting the other circuits.

Fig. 13: Kicker module

4 Mechanical Project

The mechanical project suffered some changes compared to the previous year's version. Currently, the team is focusing on improving not only the robot's efficiency and robustness, but also the ease of maintenance. With this in mind, RoboIME is constantly seeking new solutions for the project and through much research and information exchange with other teams some changes were made in the mechanical project. Below, are described the developments and the planning for RoboCup 2020.

4.1 Omni Wheels

The robot's omni wheels have been completely modified. The main objectives of this change were to smooth the robot's movements and use a commercial external

gear for the transmission instead of a internal gear. The omni now features two layers of 10 small wheels each (see figure 14). The use of a gear with half the number of teeth and the reduction of the diameter allowed the robot to go faster. The lost of torque was compensated with the effort of making lighter parts.

Fig. 14: Omni Wheel's isometric view

In addition, the main bodies of the omni are now printed, in contrast with the previous version, which were machined in aluminum. That reduction of mechanical resistance was compensated by using a PETG filament reinforced with carbon fiber. The wheels are now much lighter and inexpensive.

Futhermore, the motor supports are not symmetrical to the omnis anymore. This change allowed the motors to be lower and consequently making the supports smaller and creating more room for the kick system (See figure 15)

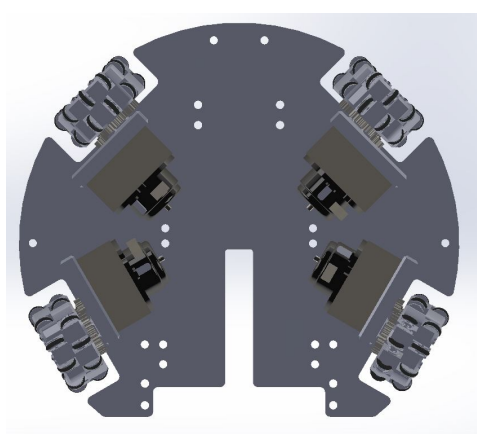

Fig. 15: Omni Wheel's top view

4.2 The Kick System:

- Low Kick: The low kick was designed to pass over the high kick plate, in order to optimize the useful space inside the robot. This system consists of a cylindrical solenoid, a piston which has the same geometry, a spring on one of the edges of the piston and the low kick plate on the other edge (figure 16).

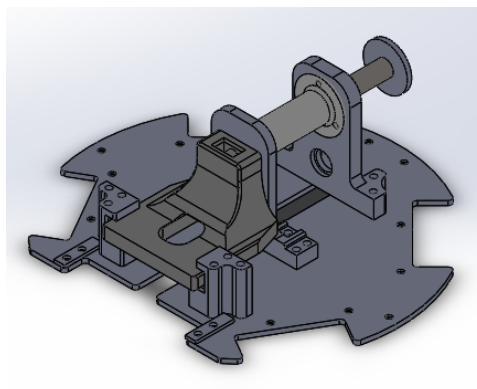

Fig. 16: Low Kick System

  While a current is passing through the solenoid, the piston goes ahead and then the low kick plate hits the ball. This movement stretches a spring and when the current finishes, the spring pushes the piston back again.

- Guide System to Low Kick: There are two guides, on the sides of the low kick plate (figure 16)that aim to help the kick plate go forward as straight as it can go and make the kick more accurate.

- High Kick: The high kick system has the same activation as the low kick system, but with some differences. The high kick was created to use the lowest part of the robot and its kick plate was designed to pass under the low kick one (figure 17).

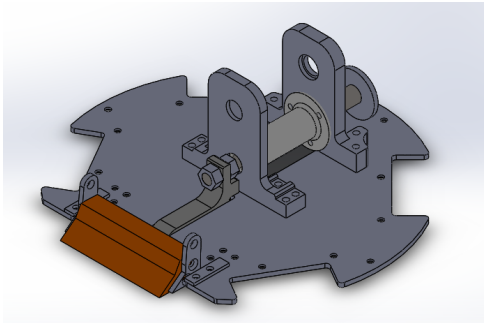

Fig. 17: High Kick System

The piston moves and hits a little plate, which has a ramp on its front. This ramp was designed to cause the ball to move upward.
The two systems have the same activation and piston geometry in order to make the analysis and the modeling simpler.

4.3 Battery Holder

The battery holder consists of a mechanism which can hold the battery in a specific place. Besides that, it needs to be easy to handle, that means it has to be easy to be placed and removed.

In the modeling, it was kept the best geometry to permit accessibility and not let the battery free to move when the robot moves around. This project so simple,it is basically made of a battery compartment and two elastic bands that keep it pressed and prevent it from leaving the holder. Therefore, it helps to maintain the battery at the desired position (See figure 18).

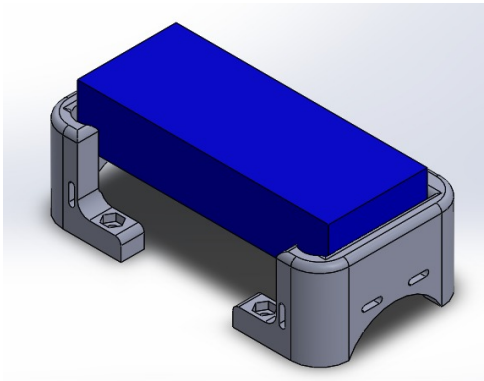

Fig. 18: Battery Holder System

4.4 The roller

The roller of the dribbler was made using a 3D printed mold and silicon with a catalyst. Many different molds were made in order to test which was the best format. The ideal roller would be able to guide the ball properly to its center. The current was designed having in mind that the helix pitch has to be smaller than the contact surface of the ball with the roller. In addition, there is also a space, in the middle of the roller, so that the ball is in equilibrium. Hence, the roller is expected to guide the ball satisfactorily towards the center. This will be confirmed through tests that will be done until RoboCup 2020.

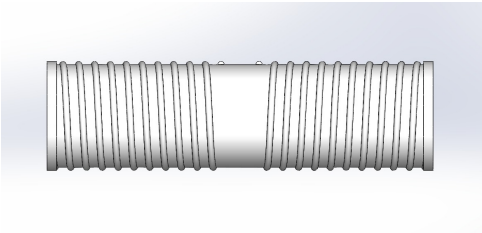

Fig. 19: The roller

4.5 Planning for RoboCup 2020

For the Robocup 2020, new robots are going to be assembled without reusing parts from the robots that are already assembled. However, for the current RoboCup, some robot's parts that were made of plastic through 3D printing will be made of metal in order to increase the robustness and durability of the robots.

5 Conclusions

For this competition, the aim is to continue the progress established last year: experimenting a new approach to the software project, modularizing the electrical project and producing more reliable CADs and CAMs in the mechanical project.

5.1 Acknowledgement

This research was partially supported by the Army's Department of Science and Technology (DCT), Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro - FAPERJ(grant E-26/111.362/2012); Fábrica de Material de Comunicação e Eletrônica (FMCE/IMBEL) and the IME alumni association. Special thanks to all former members of RoboIME. Without their support, this team would not be here.

References

1. M. Bowling M. Veloso B. Browning, J. Bruce. Stp: skills, tactics, and plays for multi-robot control in adversarial environments. pages 1–20. Available at http://papersdb.cs.ualberta.ca/~papersdb/uploaded_files/556/paper_p33.pdf.
2. Florian Schwanz Gunther Berthold Malte Mauelshagen Tobias Kessler Christian Konig, Daniel Waigand. Tigers mannheim, atificial inteligence. pages 10–17. Available at https://tigers-mannheim.de/download/papers/2011-AI-Structure_Koenig.pdf.
3. Carla S. Cosenza Gustavo C. K. Couto Luciano de S. Barreira Luis D. P. Farias Luis R. L. Rodrigues Jan L. L. Segre Matheus C. Castro Nicolas S. M. M. de Oliveira Onias C. B. Silveira Renan P. de Souza Victor Bramigk Yugo Nihari Paulo F. F. Rosa. Roboime: on the road to robocup 2019. pages 1–12. Available at https://github.com/roboime/roboime-tdp.