RobôCIn Extended Team Description Paper for RoboCup 2022

Cecília Silva, Charles Alves, Edgleyson Silva, Felipe Martins, Lucas Cavalcanti, Lucas Maciel, Matheus Vinícius, João Guilherme Monteiro, João Pedro Moura, José Victor Cruz, Pedro Henrique Santana, Rebecca Sousa, Riei Rodrigues, Roberto Fernandes, Ryan Morais, Victor Araújo, Washington Silva, and Edna Barros

Centro de Informática, Universidade Federal de Pernambuco.
Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife - Pernambuco, Brazil.
robocin@cin.ufpe.br
https://robocin.com.br/

Abstract. RobôCIn has been participating in RoboCup Small Size League since 2019 and achieved its best result in 2021. This paper presents our new robot version intending to attend the Small Size League (SSL) in RoboCup 2022 in Bangkok, Thailand. This paper aims to share some of the academic researches that our team developed over the past year. Our team has successfully published five articles at two high-impact conferences: the 24th RoboCup International Symposium and the 18th IEEE Latin American Robotics Symposium (LARS 2021). We also continue to share our improvements in software development, electronics, and mechanical systems. We are making a unified software to support all football categories that our team participates in, where we can share our algorithms, filters, and strategies among our subteams.

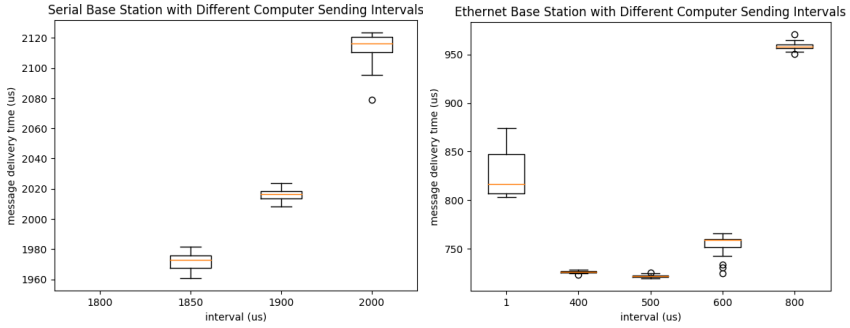
Keywords: RobôCIn · RoboCup 2022 · Robotics · Small Size League

1 Optimized Telemetry Network

The improvements in our base station after the RoboCup 2019 increased our communication system's reliability. However, it was a weak point during that competition with frequent lost connections, requiring human intervention to re-establish the connection. The work entitled *Optimized Wireless Control and Telemetry Network for Mobile Soccer Robots* [3] aims to contribute to the league, presenting our solution helping the teams to improve the quality of their communication. We present an architecture based on telemetry that sends and receives information from the robots with low latency and low packet loss rate.

This work studied the leading technologies associated with the Wireless Networked Control Systems (WNCS). The proposed architecture to control and monitor the telemetry of multiple mobile robots involves optimizing the current communication system between CPU, Base Station, and Robot to decrease the delay in sending and receiving messages, comparing the results with a previous

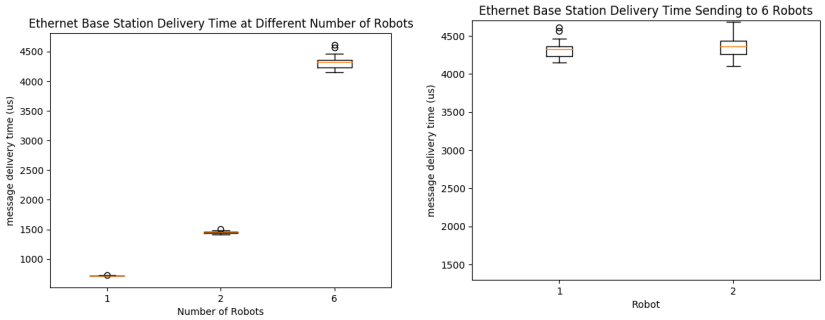
serial implementation. Figure 1 presents the communication system between the base station and CPU using Serial communication (Figure 1a), which presented reliability issues during competitions in 2019, and the optimized model using Ethernet communication proposed (Figure 1b). Validation tests include different intervals between messages.



(a) Using Serial station (b) Using Ethernet station

Fig. 1. Reception delay for 25 tests at each different sending interval, using Serial(a) or Ethernet(b) base station transmitting computer messages at the configured sending intervals[3].

Additional tests evaluate whether the implementation of telemetry, i.e., receiving information from the robot during the matches, affects the real-time control of the robots for a team in any way. For example, figure 3 shows the results of message delivery time tests for different numbers of connected robots.



(a) Delivery time with different number of nodes at Ethernet base station

(b) Delivery time at 6 robots network with Ethernet base-station from two different robots

Fig. 2. Delivery time analysis between network size (1, 2 and 6 robots) [3].

The results show that it is possible to establish telemetry communication for a team of robots with a reasonable delivery rate and without significant packet losses. The team also wants to test communication modules using different technologies in the future.

2 Dataset and Benchmark for RoboCup SSL

In parallel to the software development, the team started to take its first steps towards a fully autonomous robot in upcoming competitions. The kickoff for this challenge was object recognition through an embedded computer vision system. In this context, the work entitled *Dataset and Benchmarking of Real-Time Embedded Object Detection for RoboCup SSL* [5] aims to contribute to the development of embedded image processing and has two stages. The first stage elaborates a dataset labeled with images of the category robots, balls, and goals. The second stage uses machine learning models to evaluate the dataset on compact hardware to enable its use within the category rules.

The construction of the dataset is composed of public images of the category available in repositories on the internet, taken by various teams, and new images taken in a field in the lab to increment the dataset. The images were resized to a standard resolution of 224 x 224 pixels. The dataset has different robot and field types and different light conditions. Figure 3 below shows samples of the dataset.

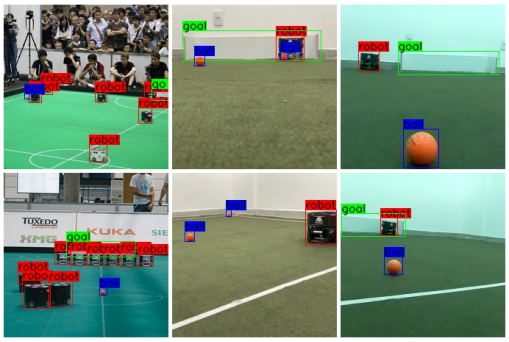


Fig. 3. Sample images from the dataset [5].

A Raspberry Pi 4B board, a Google Coral Edge TPU accelerator, and a Raspberry Pi Camera V2 were used to perform experiments with different models. We used the Transfer Learning technique to accelerate the experiments, using models pre-trained with other datasets and taking advantage of the low-level features learned. We evaluated the dataset using state-of-the-art detection models, such as MobileNet SSD v1, MobileNet SSD v2, MobileDet, and YOLO v4 Tiny. We evaluated the models for Average Precision (AP) and Average Recall

(AR) metrics, with different Intersection over Union (IoU) threshold values and different object sizes in the images (Small, Medium, and Large). The following Table 1 shows the AP results for the four methods separated by IoU threshold and object size. Moreover, Table 2 shows the RA results separated by maximum detection per image and detected object size.

Table 1. AP for each model by IoU threshold, in AP_{50} the threshold used is 0.5 and AP_{75} is 0.75, and detected object area, where AP_S, AP_M, and AP_L stands for the result by each detection size, Small, Medium or Large [5].

Method	**AP**	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MobileNet SSD v1	**44.88%**	68.81%	47.51%	26.83%	68.54%	**89.62%**
MobileNet SSD v2	43.42%	**74.41%**	44.83%	23.06%	62.55%	82.93%
MobileDet	35.96%	64.95%	36.62%	16.63%	60.48%	82.97%
YOLOv4 Tiny	42.17%	62.24%	**54.09%**	**27.34%**	**69.05%**	58.84%

Table 2. AR for each model by maximum detection per image, AR_1 for at most 1 object per image and AR_{10} for 10 objects per image, and detected object area, where AR_S, AR_M, and AR_L stands for the result by each detection size, Small, Medium or Large [5].

Method	AR_1	AR_{10}	AR_S	AR_M	AR_L
MobileNet SSD v1	**37.75%**	**62.87%**	**40.62%**	**82.18%**	**91.00%**
MobileNet SSD v2	34.76%	50.60%	29.28%	66.21%	87.00%
MobileDet	30.62%	43.66%	22.96%	65.41%	85.00%
YOLOv4 Tiny	36.72%	45.36%	30.41%	74.16%	64.00%

3 RSoccer: A Framework for studying Reinforcement Learning

Since 2019 the team has been developing Reinforcement Learning (RL) techniques applied to the IEEE Very Small Size Soccer (VSSS) competition. With the positive results in the VSSS league using the techniques implemented in both simulated and real-world environments, the team also decided to expand the studies to the SSL category. The paper entitled *rSoccer: A Framework for Studying Reinforcement Learning in Small and Very Small Size Robot Soccer*[6] presents a framework for developing robot soccer environments for RL. The proposed framework allows multiagent activities to perform tasks in cooperative and competitive scenarios.

The team adapted the open-source grSim simulator for RL-focused training conditions. Figure 4 below shows the architecture of the proposed rSoccer framework.

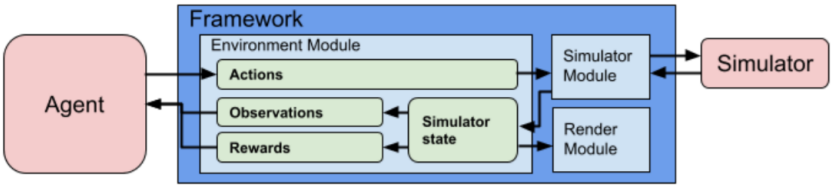
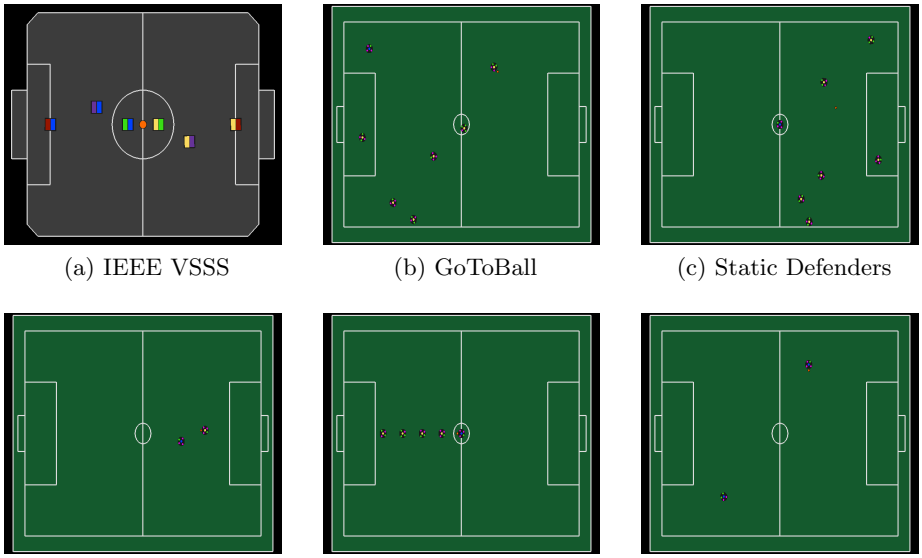


Fig. 4. rSoccer framework architecture [6].

The environments were built to receive an action from the agent, which communicates with other modules and returns new observations and rewards to the agent. The simulation module is responsible for providing a view of the working environment.

The framework has six environments developed for training and benchmarking involving robot soccer (Figure 5):



(a) IEEE VSSS (b) GoToBall (c) Static Defenders

(d) Contested Possession (e) Dribbling (f) Pass Endurance

Fig. 5. Initial states of the proposed benchmark environments [6].

– **IEEE VSSS.** In this environment, it is possible to train with single-agent mode, where only one robot learns a policy, and multi-agent, where robots share the learned policy (Figure 5a).

- **GoToBall.** Environment to teach the robot to reach the ball at a certain angle.
- **Static Defenders.** This environment is based on the Hardware Challenge (HC) of the RoboCup SSL category in 2021, where robots and the ball are randomly positioned, and the agent is trained to score a goal (Figure 5c).
- **Contested Possession.** Another environment based on the RoboCup SSL 2021 HC where the ball is placed in an opponent's dribbler, and the agent must be able to capture the ball and score a goal (Figure 5d).
- **Dribbling.** Also based on the same HC. The agent starts the challenge with the ball in the dribbler and must dodge four robots spaced in a row while keeping the ball in its dribbler (Figure 5e).
- **Pass Endurance.** The last environment based on HC, Environment for training the exchange of passes between robots of the same team (Figure 5f).

4 Telemetry-Based PI Tuning

One of the main challenges in the SSL category is the control of robots on the field, which is essential for good performance during games. With that in mind, our team looked for techniques to improve low-level control of motors. The work entitled *A Telemetry-based PI Tuning Strategy for Low-level Control of an Omnidirectional Mobile Robot* [1] proposes a way to perform the tuning of the low-level controller using the basestation, which in addition to sending the software input signals, it also collects real-time robot data on the football field.

Figure 6 below shows an overview of the proposed system. A computer connects to the basestation to send the control signals to the robot using the approach proposed in [3]. This approach reduces the packet loss rate and the delay between messages. The basestation processes the received data and sends it in a packet with a predefined message type for radio transmission to the target robots. Then, the message is decoded, processed and sends the control signals to the motors. The mainboard of each robot receives feedback from the four motors through the encoder's packages and sends the messages back to the base station in the telemetry message format. The messages are decoded and transmitted to the computer to assess performance and suggest new control parameters.

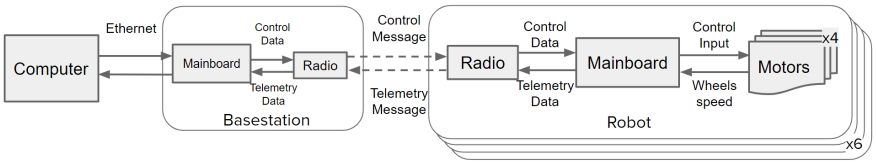


Fig. 6. Overview of the proposed PI tuning system [1].

The telemetry architecture is similar to the team's current system, which controls our robots during matches. The main difference is that we receive motor data as feedback instead of robot information such as battery status, capacitor charge or presence of the ball on the infrared sensor.

The proposed strategy uses the System Identification Toolbox from MATLAB integrated into the tuning algorithm to adjust the parameters in real-time. MATLAB processes the data and returns suitable control constants for the operation of the motors according to the reference parameters. Figure 7 below shows the result of the proposed method when compared to a robust approach, which cannot be executed in real-time and requires more in-depth knowledge of control systems for its implementation.

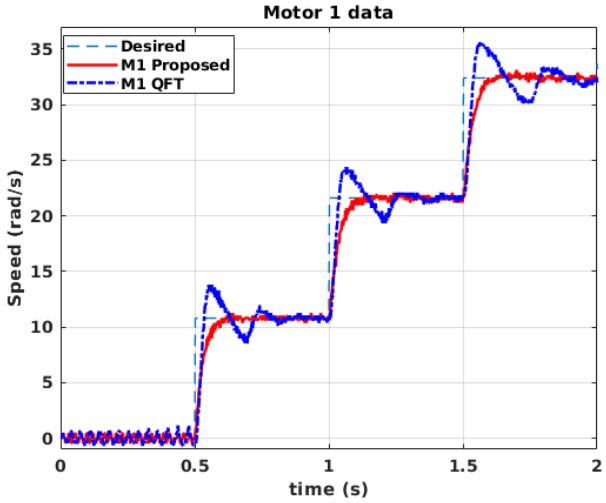


Fig. 7. Motor response to a reference speed using a robust QFT approach and the proposed telemetry-based approach [1].

Ongoing works use the proposed telemetry-based strategy to collect data for training Reinforcement Learning algorithms, genetic algorithms, and wheel failure prediction techniques.

5 Dribbling: mechanical and dynamic analysis

In the Small Size League, a good dribbler design gives a significant in-game advantage to the teams. Therefore, during the last two years, we conducted a complete study of the mechanics and dynamics of the dribbler, and we developed a functional and reliable dribbler since our team joined the league. The goal of this study entitled *Mechanical and Dynamic Analysis for Design and Development RoboCup SSL Dribbling Mechanism* [2] was to analyze all the forces

acting on the dribbler. Additionally, we performed an energy balance through a dynamic simulation to calculate how much energy a dribbler mechanism must absorb to receive a pass at a certain speed and keep the ball in contact with the system.

Figure 8 below shows an overview of the proposed system. We started with a 3D modeled design considering the constraints of our robot, such as available space, maximum covered area, and physical parts. Then we exported the designed CAD files to a simulation software responsible for the dynamic analysis of the system considering all the mechanical properties of each part of the dribbler. The dynamic analysis step performs the energy balance of the dribbling system, indicating the total energy that the mechanism needs to absorb to receive a pass and keep the ball under control. The third step is to perform a mechanical analysis of the mechanism, which means to survey all the forces that act on the system to design a set of springs capable of absorbing the impact of a pass and keeping the ball spinning in the dribbler. Finally, we have the validation stage in the real-world environment, where the designed models are made and tested for validation in a competition environment.

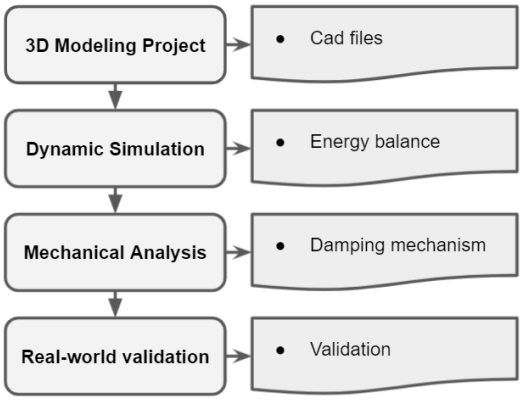


Fig. 8. Overview of the proposed mechanical and dynamic analysis method [2].

As one of the main contributions of this work, we highlight the two tests performed in the dynamic analysis stage. The first test refers to receiving the pass, where the vertical springs receive the most significant effort. The second test considers the ball already attached to the dribbler. The goal here is to maintain only the rotational kinetic energy and eliminate the translational kinetic energy of the system, preventing the ball from escaping. We also considered friction at all stages of the analysis. Figure 9 below shows a free-body diagram with a simplified dribbler structure and all the forces acting on the system.

Some of the results include the calculation of the elastic constant of the springs to absorb the indicated energy. However, the calculation of the exact di-

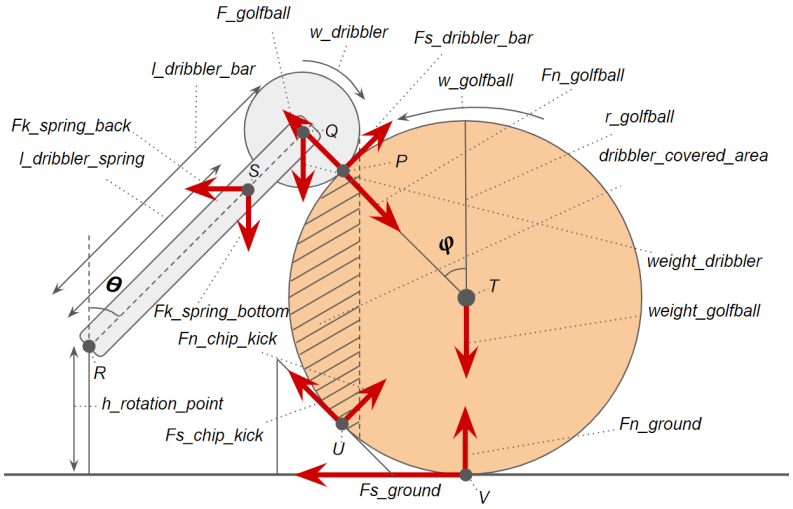


Fig. 9. Free-body diagram of forces acting on a simplified dribbler mechanism [2].

mensions of the springs depends on the space available on the robot. Our system has limited space for damping. Thus, we replaced the springs with Ethylene-vinyl acetate (EVA) copolymer pieces. In practical tests, the system successfully absorbed passes of 4.5 m/s. With the ball attached to the dribbler, we moved along the x-axis with speed from -0.5 to 1.5 m/s. On the y-axis, we moved from -0.5 to 0.5 m/s. The angular speed achieved with the robot rotating on its axis was from -6.5 to 6.5 rad/s.

With the improvements presented in this work, it was possible to correctly validate the ball placement algorithm to position the ball within time. With that, the team managed to stand out by winning first place in the RoboCup 2021 Ball Placement Challenge.

6 Software Unification

At RoboCup, we have a range of robot soccer leagues such as Humanoid, Standard Platform, Middle Size, Small Size, and Simulation. Each category has its particularities and challenges stated in its rules, using different hardware and environment. Even with different approaches, these leagues share a common goal of making robots play soccer cooperatively, sharing their solutions and codes. However, the teams usually develop state-of-the-art algorithms to their category with a separate infrastructure. As we participate in two similar soccer categories, we would like to exchange code pieces and avoid the amount of duplicated workload.

In addition to participating in RoboCup SSL, we also participate in another soccer category called IEEE Very Small Size Soccer (VSSS). This category is

considered an entry-level to SSL. However, it has several similarities, especially in terms of systems architecture, such as a camera at the top, the communication system, the use of a golfball, and algorithms.

As team members moved from one category to another over the years, we felt that the learning curve for integrating the members was very long. We have already solved some problems in one category but not in another. Simply importing these solutions can be a complicated process even with both software using C++ and the Qt framework. Even with similar proposals, the categories had a completely different ecosystem. The problem of contrasting ecosystems became visible when we tried to relate the development of the reinforcement learning team [6]. Initially, this work focused on VSSS, with the advances along the years, such as developing a cognition system completely in reinforcement for VSSS. Then, with the good results of this approach, we started to implement reinforcement solutions in SSL. First, however, we would have to rebuild all the reinforcement learning infrastructure from scratch without a standardized codebase for both teams.

The software unification project has been one of the team's most ambitious projects in recent years. In order to build a codebase for SSL and VSSS, and support the reinforcement infrastructure, we decided to create a unified ecosystem called soccer-common. Under this unification project, each category would inherit a series of standards and base algorithms, allowing the teams to speak the same "language", develop the same architecture, and avoid duplicated work. Figure 10 presents the Soccer-common structure, which facilitates code and people sharing between leagues.

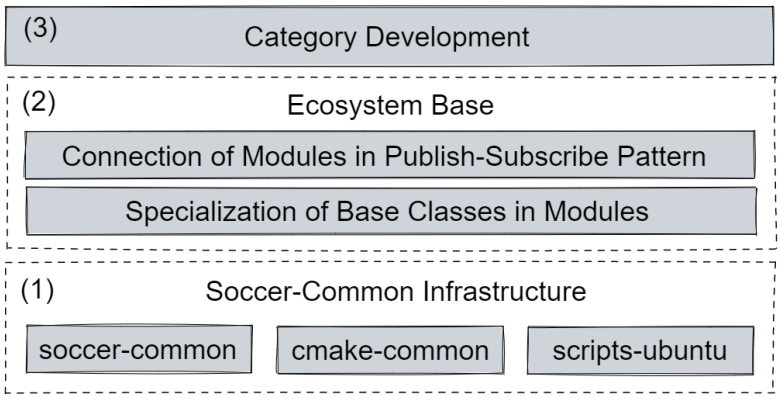


Fig. 10. Soccer-common architecture.

The proposed unification comprises a three-layer architecture:

– **Soccer-common.** It gathers all the common infrastructure with codes such as math and geometry refactored from the independent cognition software.

These codes are case standardizations to minimize verbosity. Thus, all categories now have a robust library, able to receive features specific to each category easily integrated into the common base for use in all team's leagues.
- **Ecosystem Base.** This base facilitates the use of the architecture's functionality and allows the expansion of the common base into the software of each category in a concise way. We make extensive use of the inheritance and polymorphism concepts of modern C++. We can keep our software on this infrastructure through templates and base classes, keeping everyone developing in the same pattern. In order to easily include the dependencies of each category, we started using CMake. CMake is a widely used build system that integrates most open source libraries and code. Our previous build system, QMake in Qt6, was deprecated. Now CMake is also fully integrable with several IDEs.
- **Category Development.** Finally, the top layer includes the adaptation and particularities for the different leagues. The common features can be used for both systems and reduce the development time significantly.

The demo for this architecture is available in project-unification[1], and it was used for the selective process in SSL and VSSS in the last year.

7 Works in progress

This section presents our ongoing works for this year's competition and the next ones with a long-term view aligned with the league goals.

7.1 Strategy Improvements

The SSL's Software Team is currently developing the Software Unification (Section 6) and is refactoring its cognition software within this ecosystem. The previous software version carries old patterns made when the team was inexperienced before starting in the category in 2018. With all the fixes and improvements over the years, we noticed software limitations in some aspects, such as context and information management, especially with a high learning curve for integrating new members. Thus, the code reached its limit, and its problems became more latent with a perspective of expansion and constant improvement of the software.

When we reached a stable version of mechanical and hardware in our robots, we started to verify some problematic points of our software. Unfortunately, it became hard to solve some bugs, eventual crashes, confusing/complex code to be understood, and many flags, contributing to behavioral errors introduced by the programmers.

The team started to analyze the software in-depth and noticed that most of the flags in the code focused on the referee's command processing and behavior selection according to the game state since the same function defined the behavior for all players in several game states. This function depended on many state

[1] https://github.com/robocin/project-unification

flags controlled by the referee's processing or vision information. Therefore, the decision was non-linear and had many cases and exceptions treated in conditionals, and any modification affected the game as a whole. In the new version of the software, we unified the referee's states and changes defined by the vision without using flags.

We started to build a clean code applying discretization into a state tree. This method allowed us to create small conditional and verification blocks, as well as use recent C++ features, such as std::variant[4] to match method calls of classes automatically. Each state of our game has a method responsible for defining the robots' behavior only in that state, thus avoiding the critical zone bottleneck of the previous version.

One of the main difficulties we faced during the matches was the number of fouls committed by our team, with the majority being bot crashing, which always made the team lose players and consequently lose space on the field. Some contacts are inevitable, but the number of our team's fouls could be reduced by optimizing the robot's positions. We tried to make the robot walk less on the field as the game context changes, refactoring the positioning and behavior of marking and defending. Furthermore, we developed improvements in obstacle construction by considering other robots' speed, merging overlapping obstacles, and handling particular cases like when the destination is in an obstacle.

7.2 Dribbler: centering the ball

We chose to consider the dribbler bar with a smooth surface during the analysis to simplify the simulations and decrease the system's complexity. With the positive results in terms of impact absorption and ball maintenance, the mechanic's team now gathers efforts to conduct a study on how different dribbler bar shapes and different centering profiles can impact the final result of keeping the ball centered or not. Now, the idea is to keep the dribbler structure and modify only the silicone bar. With this, the team will conduct a complete study to understand which parameters are essential for centralization and further contribute to the advancement of the category.

7.3 Minor Mechanical Improvements

As demonstrated in previous sections, the main mechanical improvement for the robot of the year 2021 was in the dribbler. However, the team is also developing minor optimizations in the kick system and the coupling between the motor and wheel.

One problem that sometimes occurs is the kick coil getting stuck and not returning to its initial state. When investigating the problem, the team discovered that the 3D printed coil deformed as the temperature increased, which led to the kicker bar not returning to the initial state after a kick. The team chose to machine the coil and perform reliability tests to see if the problem was solved. Unfortunately, due to the worsening of the pandemic, the validation tests are on hold.

We also are working on improvements in the gearing between the wheel and the motor. As shown in the 2020 TDP [7], we designed a small machined part that attaches the wheel to the robot base. This part is responsible for aligning the gearing. Over time, we notice a gap between the machined part and the 3D printed base, increasing the robot's vibration and resistance to motion. The team works to develop an alternative to the machined part involving all the motor-to-wheel connections, similar to bearing support, to improve the gearing. Figure 11 shows the concept of the solution under development.

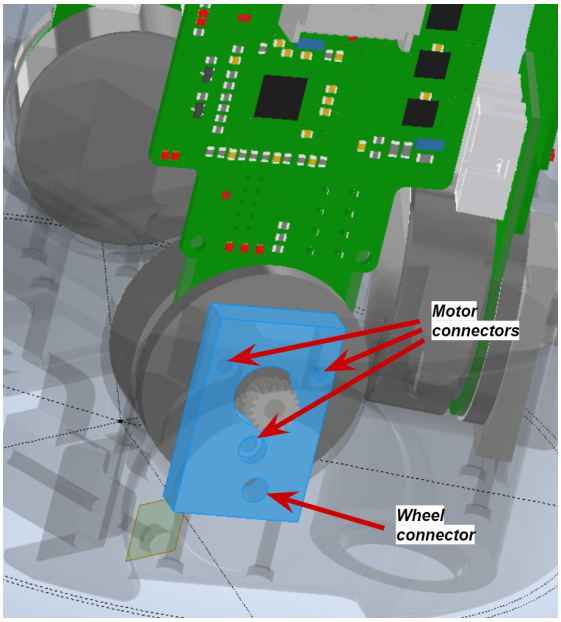


Fig. 11. 3D modelled concept to improve motor-to-wheel connections.

7.4 Ball Placement Challenge

As mentioned in Section 5, we have successfully developed a working dribbler that helped us to win the Ball Placement challenge at RoboCup 2021. However, since the dribbler was ready only a few weeks before the competition, we did not have enough time to optimize the speed and accuracy of the placement algorithm. Consequently, we lost some attempts because of a few seconds. For the challenge at RoboCup 2022, the team intends to speed up the movement and develop an approach that uses passes to complete the placement at long distances.

7.5 Vision Blackout Challenge

In 2020 the team started developing a robot for the Vision Blackout Challenge. The team found it best to develop sub-modules, integrate them and put the system to work. The work in Section 2 is one of the parts to solve the problem of how to identify objects through a front-facing camera. Other works in development for this challenge involves: adding on-board navigation to the robot, estimating the point of the object identified by the camera in the field, using sensor fusion techniques to improve navigation without feedback from external sensors, and integrating the sub-modules to build an SSL robot with a higher degree of autonomy.

We have found this as an opportunity for research, and we think this is the path the category should follow in the future. That is why we are proposing undergraduate and graduate research to develop this system, and we hope to present our solutions in competition soon.

8 Acknowledgement

First, we would like to thank our advisors and the Centro de Informática (CIn) - UFPE for all the support and knowledge during these years of project and development. Those who believed in science and respected social distancing wore a mask properly and were vaccinated during the COVID-19 pandemic. We also would like to thank all our sponsors: Maëdler, Maxon Group, and Mathworks.

References

1. Araújo, V., Martins, F., Fernandes, R., Barros, E.: Telemetry-based pi tuning strategy for low-level control ofan omnidirectional mobile robot. In: Robot World Cup. Springer (2021)
2. Araújo, V., Silva, E., Monteiro, J.G., Sousa, R., Silva, C., Santana, P.H., Barros, E.: Mechanical and dynamic analysis for design and development of a robocup ssl dribbling mechanism. In: 2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE). pp. 312–317. IEEE (2021)
3. Cavalcanti, L., Joaquim, R., Barros, E.: Optimized wireless control and telemetry network for mobile soccer robots. In: Robot World Cup. Springer (2021)
4. Cpp-Reference: std::variant, `https://en.cppreference.com/w/cpp/utility/variant`
5. Fernandes, R., Rodrigues, W., Barros, E.: Dataset and benchmarking of real-time embedded object detection for robocup ssl. In: Robot World Cup. Springer (2021)
6. Martins, F., Machado, M., Bassani, H., Braga, P., Barros, E.: rsoccer: A framework for studying reinforcement learning in small and very small size robot soccer. In: Robot World Cup. Springer (2021)
7. Silva, C., Alves, C., Martins, F., João, M., Damurie, J., Cavalcanti, L., Vinícius, M., Sousa, R., Rodrigues, R., Fernandes, R., Morais, R., Araújo, V., Silva, W., Barros, E., Bassani, H., Neto, P., Ren, T.: Robôcin 2020 team description paper (2020), roboCup Small Size League, Recife, Brazil, 2020