ITAndroids Small Size League Team Description Paper for RoboCup 2020

Alexandre Maranhão, Ana Schuch, Arthur Azevedo, Arthur Rodrigues, Eric Lima, João Sarmento, Maihara Santos, Marcos Maximo, Matheus Sales, Matheus Dias, and Reynaldo Lima

Autonomous Computational Systems Lab (LAB-SCA)
Aeronautics Institute of Technology (ITA)
São José dos Campos, São Paulo, Brazil
{alexandremr01,aplschuch,arthurazevedo41,arthur2000jose,joaolrsarmento,
maiharagabriellisantos,mateussalesms19,reynaldo.sdlima}@gmail.com
{mmaximo,matheusmdm}@ita.br
eric.lima@ga.ita.br
http://www.itandroids.com.br

Abstract. ITAndroids is a robotics competition group associated with the Autonomous Computational Systems Lab (LAB-SCA) at Aeronautics Institute of Technology (ITA). Our Small Size League (SSL) team started its activities in late 2016. Based on many open source releases by the SSL community, especially by RoboFEI, Tigers, and Skuba, the team finished a working prototype in 2018. Then, throughout 2019, ITAndroids SSL further acquired experience and matured its project. This paper describes our recent efforts to compete in the Division B of RoboCup 2020 in Bordeaux, where we expect to participate with a full team of six robots.

1 Introduction

ITAndroids is a robotics research group at Aeronautics Institute of Technology (ITA). As required by a complete endeavor in robotics, the group is multidisciplinary and contains about 40 students from different undergraduate engineering courses. In the last 8 years, the team was awarded 42 prizes in national and latin american competitions. In Latin American Robotics Competition (LARC) 2019, acknowledged as the most important competition on intelligent robots in Latin America, ITAndroids received 7 trophies, being the team with the most number of awards in the competition. Unfortunately, being the newest category, SSL is the only team of ITAndroids which does not have an award yet, but we expect to change this soon.

This report presents ITAndroids SSL team's recent efforts in developing a Small Size team to compete in RoboCup 2020 in Bordeaux. The rest of the paper is organized as follows: Section 2 presents the electronic design. Section 3 explains the mechanical design. Section 4 describes the high-level software. Finally, Section 5 concludes and shares expectations for the future.

2 Electronic Design

Currently, two projects are being considered by the electronics team. The first one was based on the RoboFEI open source project [12]. The team uses the main and kicker printed circuit boards (PBCs) from RoboFEI's project without modifications. Over the last years, the electronics team has worked on understanding and correctly using the most important features, as well as implementing our own firmware to improve the robots' functionality. We praise the RoboFEI's boards and are deeply thankful for the help of many RoboFEI's members in understanding and debugging the boards. However, to foster our evolution in electronics, we decided to design new versions of these PCBs.

The new main board is under development and the team plans to execute firmware development and testing during the first semester of 2020. The new design combines ideas from RoboFEI, Skuba, and ITAndroids Humanoid [7]. The main changes regard stronger motors (with Maxon EC-45 50 W) and the main controller, which is now an STM32H7 microcontroller instead of a Xilinx FPGA.

2.1 Current electronic design

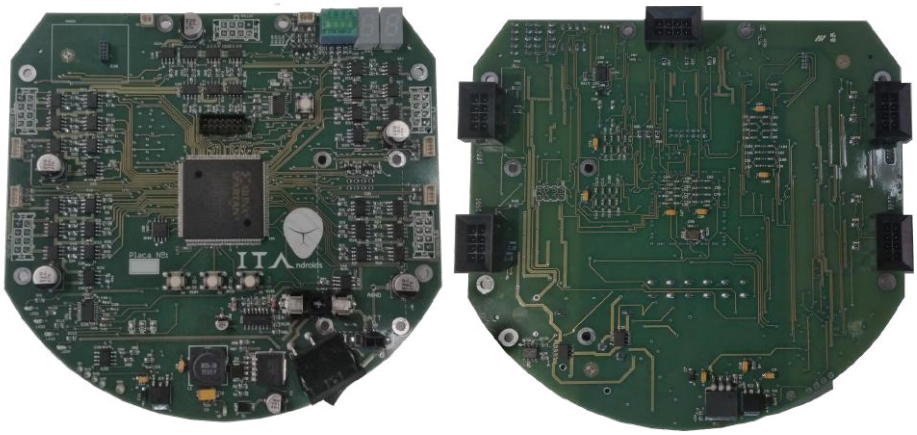
The current electronic design is comprised of two PCBs: the main and the kick board.

Main Board The main board is responsible for processing all logic functions with a Xilinx Spartan 3 FPGA as the main controller, as shown in Figure 1.

Kick Board The electronic kick system is contained in a separate circuit board opto-coupled to the main board, for safety measures since it is a high power circuit. The purpose of the kick board is to control the charging of two capacitors of 2200μF, which are connected in parallel, to a voltage of 180 V value through a DC-DC converter. If the mainboard sends a signal to execute a kick, a MOSFET will be activated and all the energy stored in the capacitors will be discharged into a solenoid. Then, the magnetic field generated by current passing through its winding moves the plunger to effectively kick.

2.2 New main board under development

In the second half of 2019, we started designing a new main board. Among the main changes are the FPGA replacement by the microcontroller STM32H742BI [17], with 480 MHz and 2 Mbytes of Flash memory. This change is due to greater familiarity with working with a microcontroller than with the FPGA in our team [7] and the high performance of that current microcontrollers. Beyond that, another major change involves changing the motors to Maxon EC-45 50 W instead of 30 W. For this change, we needed to redesign the switched-mode power supply system to support a new 4S LiPo battery of 14.8V as well as substituting



(a) (b)

Fig. 1: Main board used as the robots' low-level processing unit: (a) front and (b) back.

the motor drivers by the A3930 MOSFET Driver and the current sensor by the ACS730. Figure 2 illustrates the new main board block diagram.

Beyond these changes, some important small modifications were also done, such as the types of connectors used, as well as the removal of unused components: some buttons, the 7-segment display, the PROM, the external Flash memory, and the external ADC converter. Some features were also introduced: status LEDs to indicate if the communication between the PC and the robots is working, a push button to reset the board, add a UART for debugging and recovering log data from the Flash memory. We expect that these changes will make the board cheaper, more reliable, and easier to debug.

For a better understanding of the power distribution by the board and to prevent possible problems generated by coupling between tracks during the board routing, a power tree was elaborated, as mandated by good practices, and is illustrated in Figure 3.

For now, the schematic design is being finished. The next step will be to do the layout project and build the prototype board to test and debug problems.

3 Mechanical Design

ITAndroids SSL mechanical design now subdivides into two projects, one started in 2016 and the other in 2019. In late 2016 the team started the design that in the future would be its first functional SSL robot, called Glados. This project was based on Skuba's [16] available part drawings.

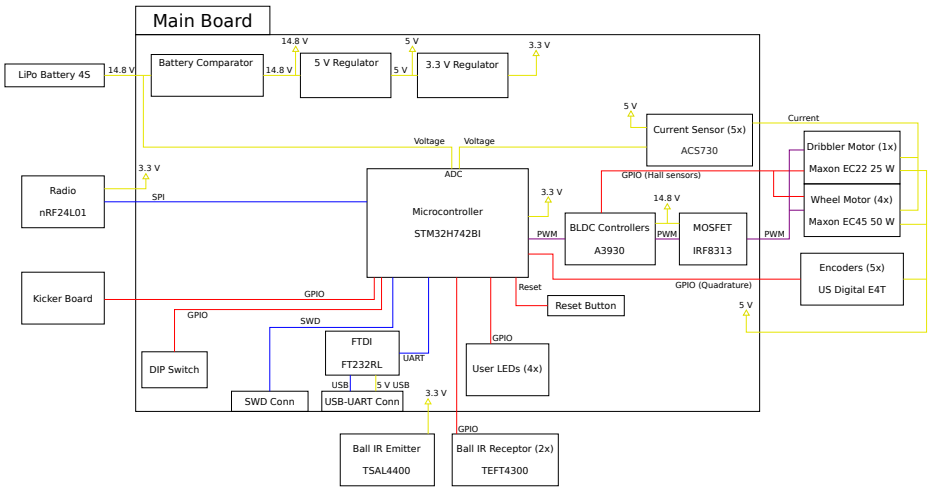


Fig. 2: Main Board block diagram.

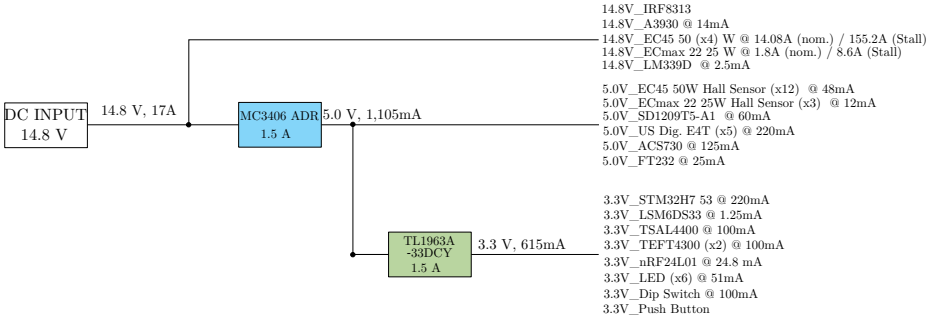


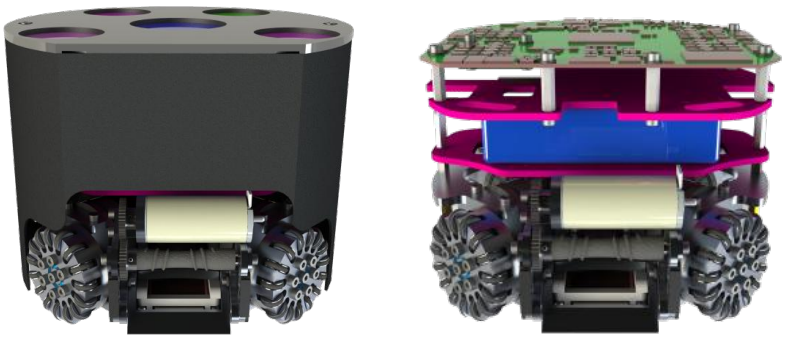
Fig. 3: Main board power tree.

The new electronic design also required a new mechanical design. The 2nd ITAndroids SSL robot is called Polimold and the main changes from Glados were in its wheels alignment (as the first project was characterized for a perpendicular alignment within its wheels), allowing a larger dribbling and kicking mechanisms, and in its wheels system, with new motors and gears. The new design is based on ideas from our previous design and Tigers Mannheim's [14] open source releases.

For the 3D model CAD, the team used Dassault Systèmes® SolidWorks for drawing and assembling the parts. The design was modified to better adapt to the team's manufacturing methods. Recently, for our experience in competitions, minor modifications were made in Glados, which will be further discussed in this paper.

3.1 1st ITAndroids SSL robot: Glados

The whole design may be seen in Figure 4. This project is currently functional and few changes have been made since 2019's ITAndroids RoboCup submission.



(a) (b)

Fig. 4: 1st ITAndroids SSL robot. (a) with cover; (b) without cover.

Encoder support In order to obtain more reliable information of the wheels' angular speeds. Previous versions had issues regarding the proximity to the motors' shafts. Reviewing informations in the product's features [5] we noticed that the minimum shaft length in order to properly use the sensor was $7mm$. In Figure 5 it's showed the final version respecting this restriction.

Dribbler and high kicking systems Dimensions from the chipper and dribbler systems have been revised due to mechanical incompatibilities. This problem was a result of an initial choice in the parts design not to include screws, which resulted in a prototype with conflicts. Changes regarding the dribbling system

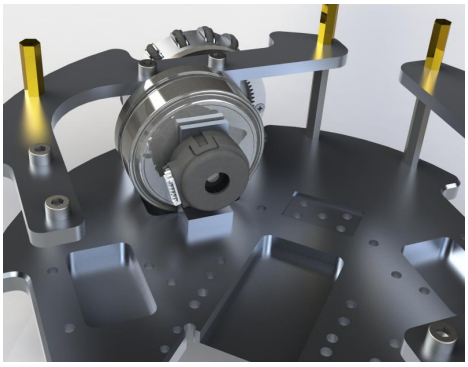


Fig. 5: Encoder support.

transmission were also implemented, also in order to better fit the parts in the robot. Final system may be seen in Figure

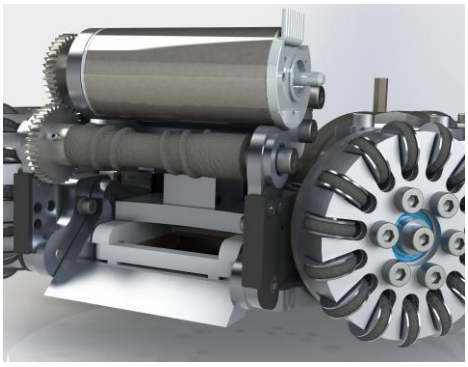


Fig. 6: Kicker and dribbler final designs for Glados.

Supports and Ground plate The main plate structure, close to the ground, has been redesigned regarding a socket space for screws. The support system has also been redesigned with fixed dimension spacers, whereas previous versions used a vertical rod and 3D-printed spacers to assemble the vertical levels.

Aside from those changes, an overview of the mechanical systems that composes both 1st and 2nd ITAndroids SSL robots may be seen in Figure 7.

3.2 2nd ITAndroids SSL robot: Polimold

The 3D model CAD was based on the Glados robot, Tigers Manhein[14, 6] open source part drawings, and RoboIME [4] part drawings provided by the team.

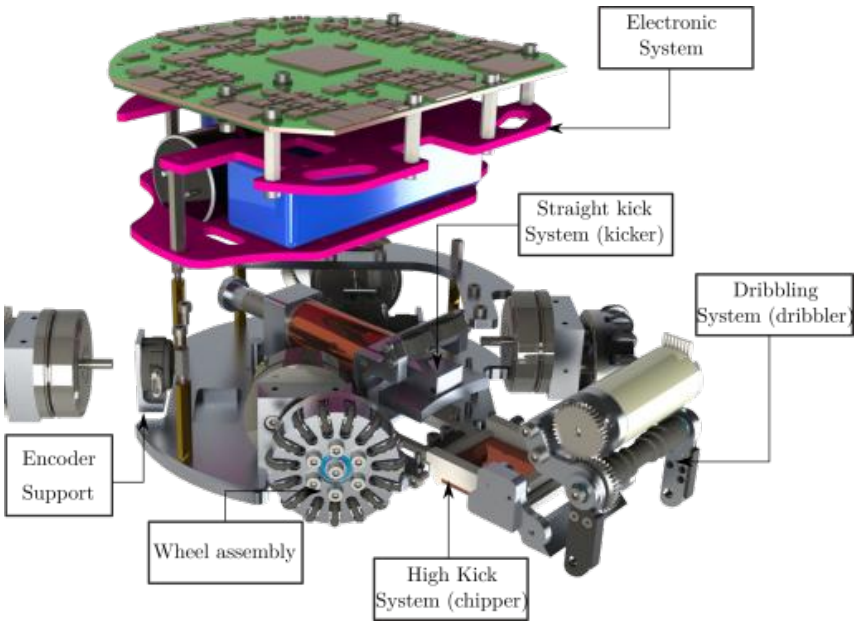
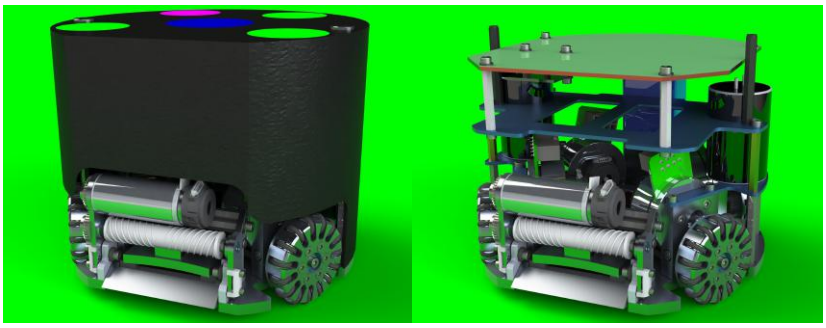


Fig. 7: ITAndroids SSL mechanical systems. An exploded view of the 1st robot (Glados) with some occult components for better visualization.

Thanks to the support of those teams, this project gained considerable maturity, resulting in its current mechanical design, which can be seen in Figure 8. This design is called Polimold. The discussion in this item may be shortened as we did not test all changes implemented and cannot safely assume they work properly.

The team used Dassault Systèmes® SolidWorks for drawing and assembling the parts. Now, ITAndroids is in the manufacturing process stage and plans to have two prototypes in early 2020 for integration tests with the electronic system. The main changes (aside from the wheels alignment and new motors) in this robot from the first are the following:

Wheels The wheel cover is thicker in this design, in order to fully hide the wheels rollers. This was motivated while studying previous ETDPs from Tigers Manhein [6], which stated the possibility of damage to the rollers as a result to its exposure. With this modification, socket holes were also added to the wheel cover to hide the screws that lock the wheel. The gear system has also been changed to regular spur gears, abandoning the internal tooth gears from Glados after difficulties in lubrication with the robot assembled and in keeping a safe distance from the motor shaft to the wheel. The design may be seen in Figure 9.



(a) (b)

Fig. 8: ITAndroids SSL second robot mechanical assembly.

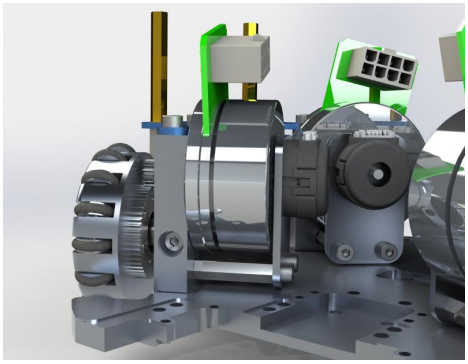


Fig. 9: Polimold moving system.

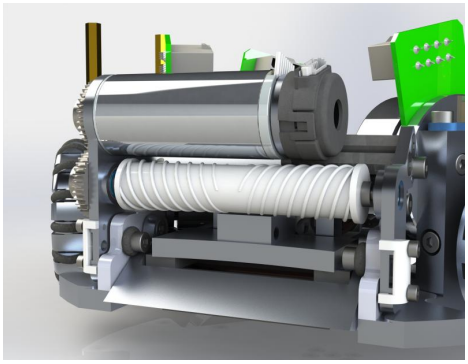


Fig. 10: Polimold dribbling and kicking systems.

Dribbling System Besides the increase in dimensions from the front view, as seen in Figure 8, we opted for a more symmetric system for catching the ball, letting the gears in the external face of the dribbler assembly. Besides, we included a support in the structure in order to use an encoder to measure the speed of the dribble motor, similar to the encoders used in the wheels. Further results on this options may be presented in future work. The design may be seen in Figure 10.

4 Software

Our code adopts a layered component-based architecture, where it is subdivided in several modular components, so each part can be isolated from the others to facilitate testing and debugging. Our team uses three open-source softwares from the SSL community: SSL-refbox and SSL-vision, respectively the shared referee and vision systems for SSL, and grSim [9], a SSL simulator with high-fidelity physics by the Parsian team.

4.1 Communication

This layer is responsible for dealing with low-level communication issues, such as sending information to the robots (simulated in grSim or real robots) and for receiving data (from grSim, SSL-Vision, and SSL-refbox). Communicating with SSL-refbox, SSL-Vision or grSim is done by UDP sockets using the Google Protocol Buffers Library for serialization. Communication with the real robots uses UART communication to direct messages to a radio station, which are then sent to the robots via radio.

4.2 Modeling

Modeling interprets the data coming from Communication and models the world state. It executes algorithms to determine where the robots are in the field. This layer creates an entity called `World Model`.

Stochastic signal processing algorithms are employed for filtering the noisy information received from the vision subsystem. Linear Kalman filters are used for ball and opponents' tracking, whereas the extended Kalman filter (EKFs) is used for our robots' localization.

4.3 Decision Making

This layer deals with high-level decision making. The implementation is based on the Behavior Tree and Coach format, which is used in other ITAndroids' projects. The Behavior Tree (BT) [11] is a mathematical model of plan execution commonly used in video games and robotics and the Coach is the agent that assigns this structure to each player depending on the current situation happening in-game and its role (attacker, defender, and goalie).

Each behavior can create requests, which are communication interfaces with the Control layer, and by knowing the state of the game at the moment the agent decides which behavior to follow. The Behavior Tree (BT) rises as an superior alternative to state machines for complex AI systems. The main difference between the two approaches is that behaviors are built around tasks in a BT, and not states as in a finite-state machine. In a BT, as shown in Figure 11, the root node branches down to more nodes, until the leaf nodes are reached. Leaf nodes effectively execute actions while intermediary ones are responsible for controlling the decision flow.

Using a BT provides a better way of designing and visualizing the IA architecture than a series of states and transition rules, that can get quite complicated to understand when the project scales up. Besides, behaviours are more modular, since they can be reused easily just by being attached to a different BT or a different node of the same BT, whereas transition rules inherent to states make harder to reuse them in different situations.

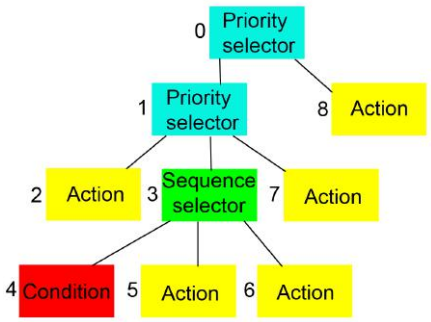


Fig. 11: Generic Behavior Tree structure, showcasing the AI modularity.

4.4 Positioning

A popular approach for player positioning in RoboCup soccer simulation leagues is to use the technique of Delaunay triangulation positioning [1], idealized by the HELIOS team. The idea consists of using human intuition to define players' positioning for a finite set of ball positions, and them generalizing the positioning for new ball positions by interpolation. The Delaunay triangulation algorithm divides the plane in a way that favors interpolation quality.

Each positioning set is stored on a text file, to be parsed by the main software, and can be created or modified using a graphical tool developed by our team (an adapted version of `fedit2`), as seen in Figure 12.

Delaunay triangulation positioning is the default defender strategy used when no threat justifies using Team Based Defense (TBD) positioning [2], developed by CMDragons. When our software detects that an opponent has a high probability

of receiving a pass, the TBD positioning is activated to reduce the chance of our team suffering a goal.

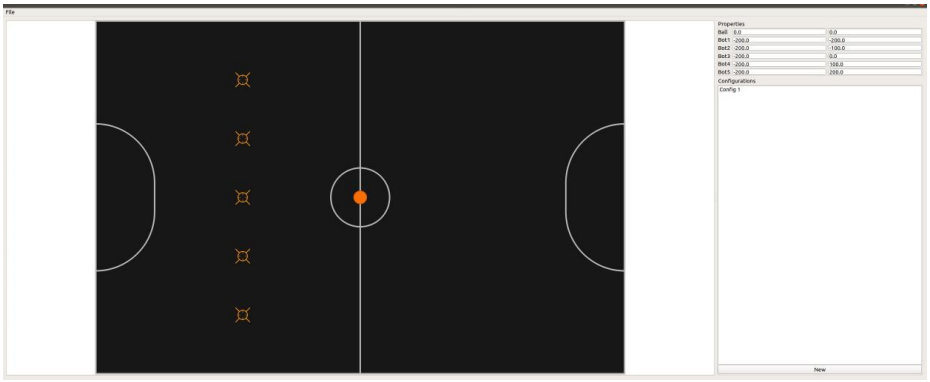


Fig. 12: Graphical tool developed by the team to create or modify key positions for Delaunay triangulation.

4.5 Path Planning

Inspired by the seminal work by Bruce and Veloso [3] and other SSL teams, we previously used a modified version of the ERRT (Extended Rapidly-Exploring Random Tree) algorithm [8]. Despite its popularity, we found many shortcomings in the algorithm, especially due to its stochastic nature. We were able to mitigate some of these effects using heuristics [10], but there was always a small fraction of paths which were unusually long or took too much time to compute.

Then, we decided to implement a path planning approach which is gaining popularity in the Standard Platform League (SPL) [13, 18], namely a A* planner on a visibility graph (VG) where obstacles are circles and the robot may move from obstacle to obstacle through their tangents, as shown in Figure 13. Our implementation was particularly influenced by Warmuth of the HULKs team [18]. The classic VG represents obstacles as polygons [15], thus planned paths have sharp turns, making it hard for a real robot to follow. On the other hand, paths planned using the circle-based VG are interesting for SSL, since they are optimal in terms of path length (ignoring the robot's dynamics) while also being smooth due to the circular obstacles. Furthermore, the shape of a typical SSL robot is similar to a circle. Finally, no randomness is used in A*, hence paths are more stable than those from sampling-based techniques.

As suggested by Warmuth [18], we build the graph as the searching goes, avoiding constructing unused parts of the graph. We also avoid planning altogether if the source and goal are visible. Furthermore, extensive code profiling and optimization were employed. Simulation scenarios based on the SSL rules

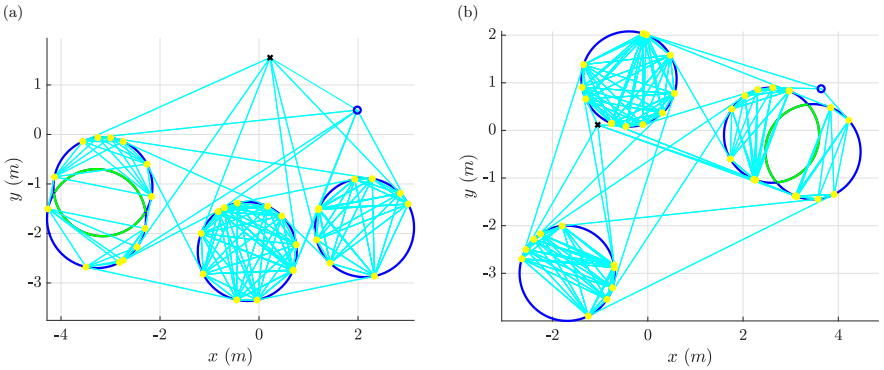


Fig. 13: Examples of visibility graphs were the obstacles are modeled as circles. Cyan lines indicate visibility. The source and goal positions are depicted as a small blue circle and a black cross, respectively.

for Division B were randomly generated to evaluate the planner. For a single robot, the mean execution time computed over 1000 plannings using A* was $7.23 \cdot 10^{-6}$ s with the worst case smaller than $5 \cdot 10^{-4}$ s (not shown in Figure 14) on an Intel Core i7 @ 2.6 GHz hexa-core notebook. This execution time is small when compared to the camera framerate. Figure 14 shows a histogram of the execution times. Examples of paths planned by our implementation are presented in Figure 13. Notice that most of the obstacles were not considered during searching, so they have not been populated. Due to the small computation burden, we expect to replan every robot path at every time step in order to mitigate the mismatch between the static obstacles used in planning and the dynamic environment of SSL.

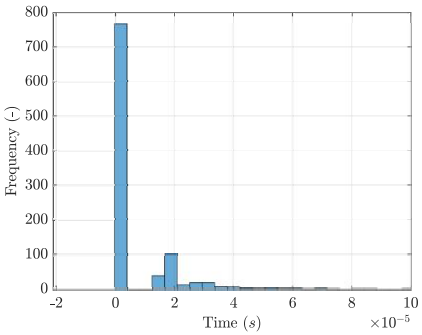


Fig. 14: Histogram of execution times for the A* circle-based visibility graph algorithm.

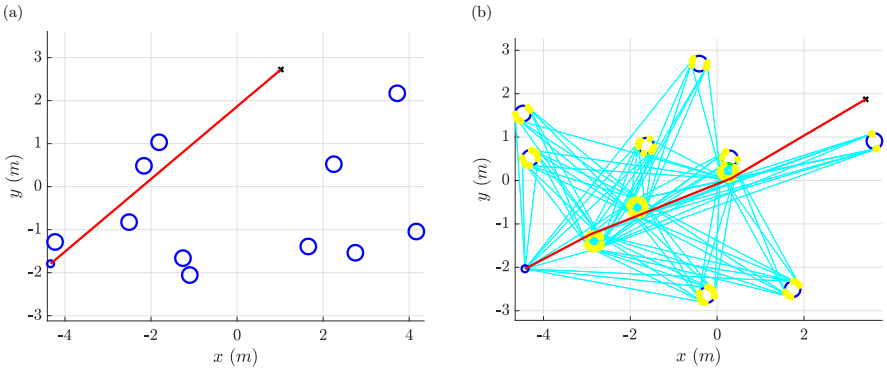


Fig. 15: Examples of plans computed by the A* circle-based visibility graphs: (a) direct visibility from source to goal, so no planning occurs (b) harder case where the robot has to evade obstacles to reach the goal.

5 Conclusion and Future Work

This paper presented the recent efforts of ITAndroids SSL team. In terms of hardware, the team has two ongoing designs. The first one was based on many open source projects, which permitted a fast evolution of the team. The second design also considers ideas from open source projects, but also from our own experience. Moreover, we have implemented a working code base, but our strategy still lacks robustness and cooperation.

Our team currently have two working robots from the first generation. Moreover, we intend to make more 4 robots using the new design in 2020, in order to participate in RoboCup with six robots, a whole SSL Division B team. Our software team is mainly occupied with refactoring our code to improve its readability and stability. Furthermore, until RoboCup, the team intends to create a new GUI to facilitate real-time monitoring, develop an attack strategy based on passes, aiming to make use of a larger number of robots in a more cooperative way, and evaluate the new path planning algorithm based on visibility graphs in real robots.

Acknowledgment

We thank ITAEx, ITA for support, our sponsors Micropress, Polimold and Rapid. We also acknowledge Mathworks (MATLAB), Atlassian (Bitbucket), JetBrains (CLion), Altium Company (Altium) and Dassault Systèmes (SolidWorks) for providing access to high quality software through academic licenses and CCM (Centro de Competência e Manufatura) for providing free aluminum to build the robots. Finally, we appreciate RoboFEI, RoboIME, Skuba, and Tigers for opening their projects, especially RoboFEI members, who have helped a lot during the development stage, sharing knowledge and designs.

References

1. Akiyama, H., Nakashima, T.: Helios base: An open source package for the robocup soccer 2d simulation. vol. 8371, pp. 528–535 (01 2014). https://doi.org/10.1007/978-3-662-44468-9_46
2. Biswas, J., Mendoza, J.P., Zhu, D., Choi, B., Klee, S., Veloso, M.: Opponent-driven planning and execution for pass, attack, and defense in a multi-robot soccer team
3. Bruce, J., Veloso, M.: Real-time randomized path planning for robot navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. vol. 3, pp. 2383–2388 vol.3 (Sep 2002). https://doi.org/10.1109/IRDS.2002.1041624
4. Cosenza, C.S., da C., G.B., Fernandez, G., Couto, G.C.K., Gonalves, L.G., Gomes, H., Germano, L., Correa, L.G., de S. Barreira, L., de Farias, L.D.P., Rodrigues, L.R.L., de Melo, J.G.O.C., Bozza, M., de Souza, M.P., de Oliveira, N.S.M.M., Silveira, O.C.B., dos Reis, R.P., de Souza, R.P., Dias, S.G.S., Nihari, Y., Rosa, P.F.F.: Roboime: From the top of latin america to robocup 2019
5. Digital, U.: E4t, miniature optical kit encoder (2020), `https://www.usdigital.com/products/encoders/incremental/kit/E4T`, [Online; accessed 6-March-2020]
6. Geiger, M., Carstensen, C., Ryll, A., Ommer, N., Engelhardt, D., Bayer, F.: Tigers mannheim extended team description for robocup 2017
7. Justa, A., Azevedo, A., Herculano, D., Vacarini, D., Tanomaru, F., Silva, I., Filho, J., Steuernagel, L., Maximo, M., Ângelo, M., Levi, M., Fonte, P., Aki, R., Tonaco, T., Araujo, V.: Itandroids humanoid team description paper for robocup 2019
8. Mirahy, B., Éric Machado, Moreira, E., Azevedo, F., da Silva, J., Luciano, J., Santos, M., Máximo, M., Lima, R., Linhares, R., Bittencourt, R., Duarte, V.: Itandroids small size soccer team description 2019
9. Monajjemi, V., Koochakzadeh, A., Ghidary, S.S.: grsim – robocup small size robot soccer simulator. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011: Robot Soccer World Cup XV. pp. 450–460. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
10. Okuyama, I.F.: Trajectory planning considering acceleration limits for an autonomous soccer player (2019)
11. Pillosu, R.: Coordinating agents with behavior trees
12. Pucci, D., Jr., F.R., Schunk, C., Braga, C.J., Torres, V., Silva, T., Amaral, V., Silva, A.D.O.D., Jr., J.A.G., Bianchi, R.A.C., Tonidandel., F.: Robofei 2014 team description paper
13. Röfer, T., Laue, T., Hasselbring, A., Richter-Klug, J., Röhrig, E.: B-human 2017 - team tactics and robot skills in the standard platform league. In: XXI, Lecture Notes in Artificial Intelligence. Springer (2017)
14. Ryll, A., Ommer, N., Geiger, M., Jauer, M., Theis, J.: Tigers mannheim team description for robocup 2014
15. Silva, C., Martins, F., Machado, J.G., Cavalcanti, L., Sousa, R., Fernandes, R., Araújo, V., Silva, V., Barros, E., Bassani, H.F., de Mattos Neto, P.S.G., Ren, T.I.: Robôcin 2019 team description paper
16. Srisabye, J., Hoonsuwan, P., Bowarnkitiwong, S., Onman, C., Wasuntapichaikul, P., Signhakarn, A., e.a.: Skuba 2009 Team Description of the World RoboCup 2008
17. ST: Stm32h742bi. `https://www.st.com/en/microcontrollers-microprocessors/stm32h742bi.html`
18. Warmuth, F.: A Graph Based Path Planning Approach for the RoboCup Standard Platform League. Tech. rep., Hamburg University of Technology (April 2019)