ITAndroids Very Small Size Soccer 3v3 Team Description Paper for LARC 2022*

Renan Ferreira[1], Marcelo Buga[1], Kenji Yamane[1], Bruno Franzini [1], Guilherme Miguel[1], Stelios Karvanis[1], Raphael Rosendo[1], Guilherme Almeida[1], Vinicius Raimundo[1], Guilherme Brasiliense[1], Guilherme Bertolino[1] and Marcos R. Omena Máximo[2]

***Abstract*— ITAndroids Very Small Size Soccer group was created in the middle of 2013 by undergraduate students at Technological Institute of Aeronautics (ITA). In recent competitions, the group has been accomplishing good results. Last year (2021) we participated in CBR and had great results, placing first in the 5v5 simulated category. In this team description paper, we detail what we did, regarding the hardware, the game strategy, and simulations.**

I. INTRODUCTION

Since 2012, some members of ITAndroids, a robotics team from Technological Institute of Aeronautics (ITA), wished to participate of a non-simulated robot soccer competition, but we had no knowledge about it.

Thus, it was clear that it would be better for the students to take part in a low-cost project in which we could learn a lot about hardware, vision processing, control and strategy algorithms. Very Small Size League meets exactly these requirements. This paper describes the evolution of the team and detail the most recent changes of the project.

Due to the COVID-19 pandemic, most of the team's development since the previous competition came in code related areas, specifically, improvements to the team's path planning by creating a new ball predictor, changes in control and development of new Behavior Trees for the game strategy.

[1]University graduate student at Technological Institute of Aeronautics, São José dos Campos, São Paulo, Brazil

[2]Doctor at Technological Institute of Aeronautics, São José dos Campos, São Paulo, Brazil maximo.marcos@gmail.com

II. THE HARDWARE

Outside of the scientific research done by one of our members in V, most of the team's hardware did not change. Therefore, this section will briefly expose some of the team's hardware and a more detailed description can be found in 2020's TDP [1].

A. The Electronics

The group's electronics project is very simple and is composed basically of a microcontroller, an H bridge, encoders for each wheel, motors and a radio.

1) The Radio Module: The nRF24L01 radio module, built by Nordic Semiconductor, carries all the communication between the computer and each robot. Each radio has one TX pipe, allowing it to send messages to one radio at time, but 6 RX pipes, making possible to the computer receive messages from all 5 robots at the same time. VSS team has five robots and one computer, so 6 radios are needed.

2) The Motors, Encoders and Motor Driver: The team's robots needs two motors (1 for each wheel). These actuators are the main responsible for the translational and rotational movements of the robot. The motors we used for this project were 2224R 006SR + IE2 512 (rising edge pulses per channel). These motors have 6V nominal voltage, with a 2048 pulses per turn magnetic encoder. The motor connector is an IDC (Insulation-displacement) connector. The main advantages of this motor are the high precision encoders, and the excellent performance for a brush DC motor.

The microcontroller can't feed the actuators alone, so it needs another subsystem to drive the motors. This subsystem is called Motor Driver (or H Bridge). Our current motor driver is DRV8872-Q1 (DRV8872-Q1 Automotive 3.6-A Brushed DC Motor Driver With Fault Reporting).

This motor driver has two operating ways with respect to current circulation inside the H Bridge: fast decay and slow decay mode. Since our motor has low inductance, and operate with a maximum 3.0A stall current, we chose to use the slow decay motor driver mode over the fast decay mode.

3) The Microcontroller: In 2018, the Nucleo F303K8 was replaced by the microcontroller itself, STM32F303RET6K8, for space saving. It was then necessary to add a clock and a reset circuit. Because of the clock circuit, it is possible to use the microcontroller at the clock frequency of 72 MHz. Since we are using the microcontroller itself, it was necessary to use an programmer device. We used the ST-LINK/V2.

4) The Power Supply: We choose a 7.4 V LiPo battery to supply power to the whole circuit. LiPo's are good for their high discharge rate, high charge capacity and light weight. We could connect the battery directly to the motors and to Nucleo. The NRF24L01 requires 3.3V with 5 V, both obtained from Nucleo regulators.

B. The Mechanics

The mechanical project can be divided in 4 main parts: the motors block, the reduction block, the circuit block and the cover. An overview of the project is show in Fig. 1.

The reduction block is composed by 9 parts, including 1 shaft, 2 wheels, 2 gears of 119 teeth, 2 bearings of 11mm x 5mm x 5mm and 2 retaining rings of 5mm. The circuit block is composed by one part that is 3D printed. The reason for this is: if it is necessary to redo the PCB, simply modify this part.

The wheel of the robot is made of aluminum, similar to the other parts. The tire was made of Poly 74-20, a Polyurethane Rubber. To manufacture them, some 3D printed molds were used.

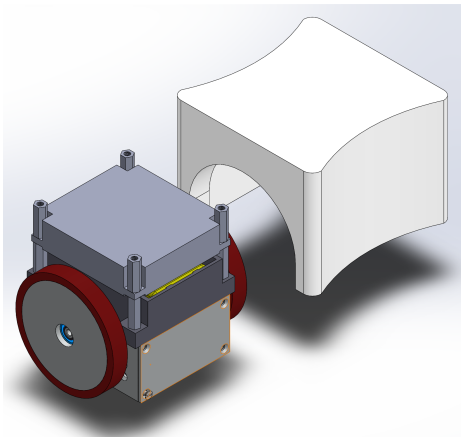

Fig. 1: Overview of the mechanics.

III. The Game Strategy

We adopted a structure to the game strategy that consists of a Finite State Machine which alternates between Coaches, i.e behavior Trees [2]. This structure was chosen because of its modularity and relative practicality in changing roles among players. As it was shown in practice, this structure is better than a pure state machine for developing the high level of the strategy, even though it gets more complicated with the lowest level details.

A. Behavior Tree VSS implementation

For using the behavior tree in the ITAndroids VSS code, there was first made a division of the strategy in levels. The lowest level is the behavior itself, and just above that is the behavior of one single player. Above that we have a coach, which chooses what player behavior goes to each player in the game. But we can also have many coaches: one for aggressive defending, other for having a better attack, etc.

Since past CBR, our team focused in rebuild the current defense behavior tree in order to improve it as well as developing and improving cooperative behavior in penalty, free ball and goal kick to work well in simulated matches and real world matches.

B. Path planning

After the best destination is determined through the strategy, the path used to get there is also needed

to be constructed, which is done using a unitary vector field, *Univector Field Navigation*, where only the desired angle is controlled. This field is constructed by setting attractive ”forces“ relative to the desired destination and repulsive ”forces“ relative to the obstacles in the way. It generates a field such as the one showed in Figure 2, where a path can be determined.

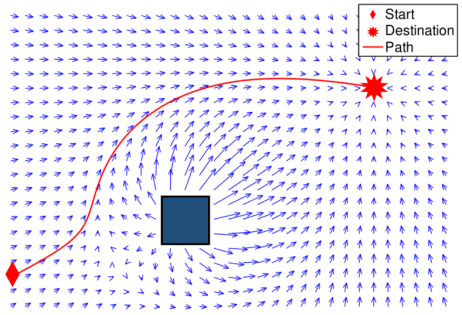

Fig. 2: Path determined through the univector field.

To calculate the trajectory, the potential field estimates the movement of the robot as circumferences arcs. Due to its nature of being a vector field, it is extremely useful to have a tool to help us visualize what was happening with the vectors while changing some constants. The GUI and the simulator have this feature implemented to help to debug and understand what is happening to the player

Since 2018, our team used a ball prediction algorithm that has not been working well nowadays. This year (2022) we implemented a new algorithm to predict the ball position. In this way, it uses the Euclidean distance between the ball to the robot at each moment as a heuristic to estimate where will be the ball when it meets the robot. Knowing the robot's speed profile and the ball speed we can approximate the robot trajectory as a line and find with a small error the desired shoot coordinate for the robot.

The algorithm iterates over the ball direction each step until a threshold between the absolute difference between the robot's time to get to the future ball prediction and the time to ball get there is small enough.

IV. Simulator

The team's simulator uses the model of our robots previously made and, with that, it simulates the strategy of the game using physics, graphics and rules made by our team assisted by tools like Ogre, to render the players and ball, blender, to render the field and ODE to simulate the physics. It also shares the Graphical User Interface used in the main executable. To simulate the vision, the simulator uses a fake webcam and it even simulates the camera delay. Last year, the team's focus in regards to simulation was in adapting it for the virtual competitions.

In the new virtual model for competitions created due to the pandemic, an entirely different simulator is used as ambient, the FIRASim. Nevertheless, the ITAndroids simulator proved useful because the univector optimization tool was based on it, and it provided an environment which can be greatly accelerated and easily rebootable, ideal for an optimization. Having this in mind, changes were made to the ITAndroids physics and judge, to make them look more similar to their FIRASim correspondents. These changes are listed below:

- Removing the artificial noise and delay
- Using the same physical robot model that is used in FIRASim
- A more conservative judge, with the same rules applied to the competition's judge

Also, the communication with FIRASim was created similarly to the already existent communication with the ITAndroids simulator, so that it facilitates integration. The library used was also ZeroMQ, which supports asynchronous UDP communication as well.

The competition judge, named VSSReferee, was yet another external application which needed to be communicated with by our project. Its interactions with our project are summarized by basically it sending a game status packet containing information regarding whether there is a foul or a change in the game state, and our project sending a response to it with a packet that may contain repositioning information, corresponding to the possible foul, if needed. The library used was also ZeroMQ, allied

with protocol buffers, also using the asynchronous UDP communication.

Since the VSSReferee only sent messages when events occurred, a second statistics were needed to be created, within the agent binary, to determine overall match status. This would then be possibly useful to program some high-level decision making, such as changing to a more defensive strategy in case our side is losing.

Another important aspect of the VSSReferee is that, just like it happens on real world matches, the teams have some degrees of liberty to position players on fouls. This positioning can sometimes be crucial in many instances of the game, therefore a positioning tool, made in Qt, was refactored and improved, in order to optimize manual repositioning between competition matches, for both the ITAndroids Simulator and the FIRASim simulator, with its VSSReferee judge.

V. Scientific Research

A. Hardware

1) Scientific Research on the Overheating of the Motors:

A problem faced with the chosen motors is their eventual overheating. The high performance demanded from the motors in addition to a structure projected without considering the heat dissipation causes the electric motor's inner temperature to rise above the tolerated limit. This problem motivated scientific research to obtain a thermal model of the robot's structure and successfully predict at which point the motors could be near to overheat.

To model the robot's structure, the Lumped Capacitance Method was used. This method consists in modeling the heat transfer as thermal resistances, which can be of both conduction or convection. Their expressions are given in the following equations:

$$R_{cond} = \frac{L}{kA} \ , \ R_{conv} = \frac{1}{hA} \qquad (1)$$

in which for the conduction resistance L is the thickness of the material, A is its surface area and k is a constant associated with the material. For the convection resistance A is the surface area and h is a constant associated to the fluid flow. Figure 3 shows the obtained thermal circuit for the robot structure.

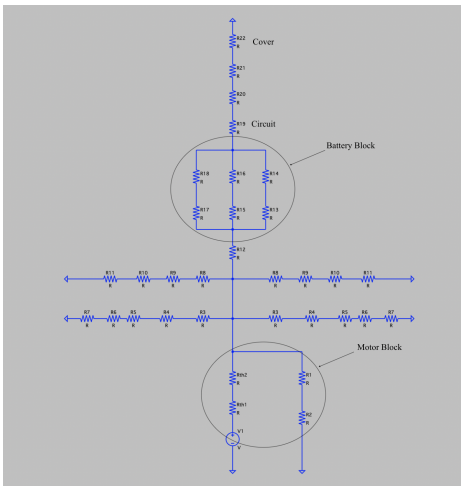

Fig. 3: Thermal circuit obtained with the Lumped Capacitance Method.

The obtained model was simulated for different current values to evaluate the motors temperature variation. Figure 4 shows this variation for different current values.

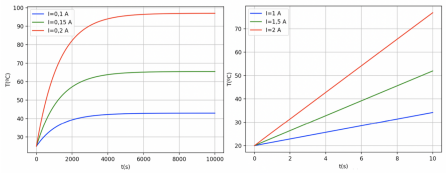

Fig. 4: Heating curves for different current values. For the left graph, smaller values were applied in a long time interval. For the right graph, bigger values were applied in a smaller interval.

This simulations were useful to evaluate possible solutions to the overheating problem, which can be implemented in the future.

B. Vision

1) Scientific research on light change compensation: The color calibrator module suffers a lot when it comes to light variations. Often present

in competitions, illumination altering affects how the computer perceives each color, and therefore results in segmented pixels loss. Without as many segmented pixels as before, the blobs can't be accurately computed, which implies in estimation errors and even position loss.

There are algorithms that aim to tackle the illumination variation problem. [4] introduces the *Center Surround Retinex* method, which is based on the following illumination model.

$$f_i(x, y) = l_i(x, y)r_i(x, y), \tag{2}$$

where f is the value found on channel i, l is luminance and r is reflectance, relative to the pixel located on the cartesian coordinates x e y. [5], in turn, attempts to estimate the global illumination variation with unsupervised learning using as prior the calibration made manually.

Two methods based on these two articles were implemented for ITAndroids' use case on the very small size soccer code. Both make use of the hsv space, because as [6] reasoned, the V channel in this space is predominant in the illumination effects, which may simplify the algorithms complexity.

In the Retinex based method, the same estimate proposed by [4] for the luminance of a pixel was used: a low-pass filter applied to the neighborhood of such pixels. A fifth order gaussian kernel was chosen in the implementation made for this team. The reflectance value was normalized with the hiperbolic tangent, followed by multiplication by 255. In addition, this reduction process was made only in the V channel, inspired by the argumentation provided by [6].

$$\hat{f}_v(x, y) = 255 \tanh \frac{f_v(x, y)}{f_v(x, y) * G}, \tag{3}$$

where G is the kernel to be applied in the neighborhood of the pixel located in x, y, and * corresponds to the convolution operator.

For the KMeans based method, The resulting configuration of the segmented HSV color space are used as initial clusters, a *cluster* for each color. Given the volume of (H, S, V) triples for which a color Y has been attributed, the *cluster* Y is computed as this volume's centroid. Then, these *clusters* are optimized using points acquired through samples of the camera's vision. These samples are chosen as circles centered on the biggest blobs of each color. Then, with a illumination change, the global change on V is computed through the distance that each cluster has made. This value is then subtracted for every pixel in the frame.

To compare the effectiveness of each method, the pixel loss experimented by the color calibration tool was used as base test. Fig. 5 illustrates this.

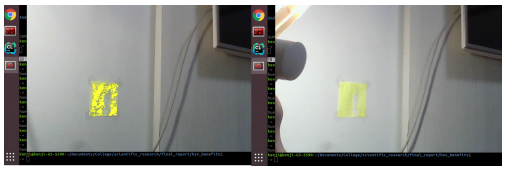

Fig. 5: Segmented *pixels* loss phenomenon.

This procedure has been simulated on three distinct scenarios: no compensation method, with the *Retinex* based method and and the *KMeans* based method, computing for each the segmented pixels quantity for each frame. The result is show Fig. 6.

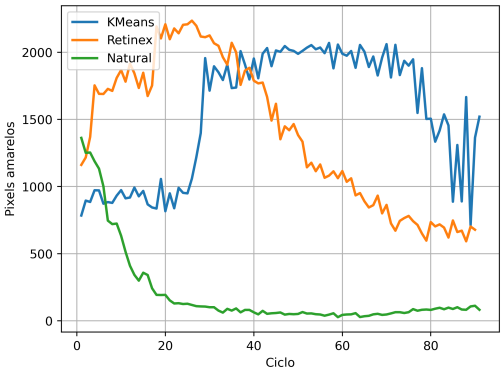

Fig. 6: *Pixels* drop on the three situations.

Both implemented methods showed a positive effect in front of the illumination variation effect. However, time should be considered as well, especially considering the VSS context. Computing

the mean time of each method with a Monte Carlo simulation over a static fixed size dummy frame, a value of $0{,}169 \pm 0{,}001$ s has been found for the *Retinex* based method, and $0{,}016414 \pm 0{,}000001$ s for the *KMeans* based method. These results indicate the latter is the more appropriate to be developed and used in real VSS matches.

ACKNOWLEDGMENT

We thanks this work to Technological Institute of Aeronautics, ITAEx, Altium, Intel, JetBrains, Mathworks, Metinjo, Micropress, Polimold, Poupex, Rapid, MATLAB, SolidWorks, STMicroeletronics and Virtual.PYXIS, CENIC, FIELD PRO.

REFERENCES

[1] ITAndroids Very Small Size Soccer Team Description Paper for LARC 2020. 2020.

[2] COLLEDANCHISE, Michele. Behavior Tree In Robotics. Doctoral Thesis - KTH Royal Institute of Technology. www.diva-portal.org/smash/get/diva2:1078940/FULLTEXT01.pdf. [Online; accessed Dec 08th, 2018]. 2017.

[3] Delaunay Triangulations. https://www.ti.inf.ethz.ch/ew/Lehre/CG13/lecture/Chapter%206.pdf. [Online, accessed Mar 07th, 2019]. 2018.

[4] JOBSON, Daniel J.; RAHMAN, Zia-ur; WOODELL, Glenn A. Properties and performance of a center/surround retinex. IEEE transactions on image processing, v. 6, n. 3, p. 451-462, 1997.

[5] PAULL, Liam et al. Duckietown: an open, inexpensive and flexible platform for autonomy education and research. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017. p. 1497-1504.

[6] CHOI, Doo Hyun et al. Color image enhancement using single-scale retinex based on an improved image formation model. In: 2008 16th European Signal Processing Conference. IEEE, 2008. p. 1-5.