

# Very Small Size Soccer (VSSS) RobotBulls Time

Alexandre B. Lugli<sup>1</sup>, Davi Rosim<sup>3</sup>, Edmundo Henrique de P. Silva<sup>4</sup>, Egídio R. Neto<sup>2</sup>, João Henrique S. Delfino<sup>5</sup>, João Roberto T. dos Santos<sup>6</sup>, Rodrigo O. Salles<sup>7</sup>, Wanderson E. Saldanha<sup>8</sup>

**Abstract**— O trabalho desenvolvido pela equipe RobotBulls, do Instituto Nacional de Telecomunicações, na categoria *IEEE Very Small Size Soccer*, traz informações sobre implementações realizadas para evoluir e melhorar o desempenho dos robôs que participarão da competição. Apresenta-se um novo método para processar as informações e tomar as decisões de maneira mais eficiente. O intuito do documento é compartilhar as informações e conhecimento adquirido, visando alcançar as equipes que desejam melhorar sua performance ou ingressar na categoria.

**Palavras-chave:** Behaviour Tree, Comunicação, Inteligência Artificial, Robótica, Software.

## I. INTRODUÇÃO

Os integrantes da categoria *IEEE Very Small Size Soccer* (VSSS) [1], vêm por meio deste estudo, incentivar pesquisas nas áreas de robótica e inteligência artificial, já que estas possuem um alto nível de complexidade. Isso ocorre por conta de muitos conceitos envolvendo *hardware*, *software* e *firmware*. Nos modelos de competições adotados atualmente, implementam-se conceitos de redes de computadores em cima do que já estava sendo utilizado.

O projeto foi desenvolvido pela equipe de robótica e inteligência artificial do Instituto Nacional de Telecomunicações - INATEL, a RobotBulls, logo evidenciada na Figura 1. Nessa categoria existem, atualmente, cinco alunos pesquisadores de diferentes cursos da graduação e três professores orientadores que guiam os trabalhos da equipe.

A categoria VSSS da equipe RobotBulls, teve como primeira participação oficial o *LARC* (Competição Latino-Americana de Robótica - *Latin American Robotics Competition*) de 2015 que aconteceu na cidade de Uberlândia/MG. Desde então os trabalhos da equipe foram crescendo e fizeram com que fosse reconhecidos como vice-campeões e depois como campeões, sempre se mantendo no pódio

<sup>1</sup>A.B.Lugli, INATEL, Santa Rita do Sapucaí, CEP 37540-000 Brasil  
Telefone: (35) 99969-9571; e-mail: baratella@inatele.br)

<sup>3</sup>D.Rosim, INATEL, Santa Rita do Sapucaí, CEP 37540-000 Brasil  
Telefone: (35) 99106-9046; davi.rosim@ges.inatel.br)

<sup>4</sup>E.H.de P.Silva, INATEL, Santa Rita do Sapucaí, CEP 37540-000 Brasil  
(Telefone: (35) 99741-8810; edmundo.henrique@geb.inatel.br)

<sup>2</sup>E.R.Neto, INATEL, Santa Rita do Sapucaí, CEP 37540-000 Brasil  
(Telefone: (35) 99949-0681; egidio.neto@inatele.br)

<sup>5</sup>J.H.S.Delfino, INATEL, Santa Rita do Sapucaí, CEP 37540-000 Brasil  
Telefone: (35) 99990-1230; joao.h@gec.inatel.br)

<sup>6</sup>J.R.T.dos Santos, INATEL, Santa Rita do Sapucaí, CEP 37540-000 Brasil  
CEP 37540-000 Brasil Telefone: (35) 99749-0015;  
joao.roberto@gea.inatel.br)

<sup>7</sup>R.O.Salles, INATEL, Santa Rita do Sapucaí, CEP 37540-000 Brasil  
Telefone: (35) 99202-7336; rodrigo.salles@gea.inatel.br)

<sup>8</sup>W.E.Saldanha INATEL, Santa Rita do Sapucaí, CEP 37540-000 Brasil  
Telefone: (35) 99808-6430; wsaldanha@inatele.br)



Fig. 1. Logo da equipe RobotBulls

desde 2016 até 2019. No último ano, em 2020, a competição ocorreu de uma forma inédita por conta da crise do Covid-19, que foi de maneira remota. Sendo assim, utilizou-se um simulador para a disputa das partidas entre as equipes.

A pesquisa realizada pela equipe se encontra detalhada nos seguintes capítulos. No capítulo II é descrito o *hardware* utilizado pela equipe na categoria. O capítulo III é atribuído à Visão Computacional, a qual é responsável pela identificação, localização e tratamento da imagem capturada pela câmera. O capítulo IV aborda o desenvolvimento do *software* e as táticas são apresentadas em um sistema de testes e tomadas de decisões impostas pela equipe, onde a inteligência artificial atua, permitindo a navegação dos robôs em campo. No capítulo V é apresentado o *software FIRASim*, que é utilizado para a simulação, testes e também é o *software* a ser utilizado na competição deste ano. No capítulo VI são vistos os testes e resultados que a equipe obteve ao longo do desenvolvimento do projeto. No capítulo VII, conclui-se os estudos do time.

## II. DESENVOLVIMENTO DE HARDWARE

### A. Eletrônica

A eletrônica básica do robô é toda desenvolvida pela equipe. Vale ressaltar os seguintes componentes e circuitos necessários para um bom funcionamento do robô desenvolvido:

1) *Step-Down*: Conversor de Tensão *DC-DC*, que tem como função converter um nível de tensão inicial em outro nível secundário, ou seja, regular a tensão de saída em relação a tensão de entrada. Faz-se necessário o uso de tal regulador para alimentar os drivers dos motores, cuja tensão de alimentação é de 6V.

2) *Microcontrolador*: STM32G491RE[2] baseado no alto desempenho Arm® Cortex®-M4 32-núcleo RISC de bits.

Operando em uma frequência de até 170 MHz. Implementando um temporizador de 32 bits, três temporizadores *PWM* (*pulse width modulation*) de 16 bits dedicados ao motor sendo este alimentado pelo regularizador de tensão de 3.3V.

3) *Ponte-H*: Dois drivers DRV-8874 um para cada motor. Driver de ponte H de 40 V, 6 A com *feedback* de detecção de corrente integrado, trabalhando com *PWM*. Implementa a detecção e regulação da corrente, assim eliminando a necessidade de um resistor de potência, economizando área da placa e custo do sistema.

4) *Encoder*: O ENX10 EASY 1024IMP[3], converte os movimentos dos motores em pulso elétricos (1024 pulsos por volta), ou seja, informa a frequência de pulsos ao controlador programável que, por sua vez, consegue calcular a velocidade e a direção da rotação do motor, que também é alimentado por um regularizador de tensão de 5V.

5) *Motores*: Dois motores Maxon Dc-max 22S GB[4], estes motores possuem tensão nominal de 6V e corrente máxima de operação de 5.39 A. As vantagens desses motores são o grande torque e a capacidade da acoplagem de *encoders* de alta precisão que auxiliam no excelente desempenho do motor.

6) *Reguladores de tensão*: Um de 5V e outro de 3.3V. O primeiro é usado na normalização da tensão que chega no encoder e o segundo serve de auxílio da diferença de potencial que chega no Microcontrolador.

#### B. Mecânica

1) *Chassi*: O chassi foi elaborado para que o robô tenha maior proximidade com o solo, tendo seu maior peso (motores) concentrado na parte inferior, ou seja, sua base (densa), assim reduzindo seu centro de gravidade ao máximo, garantindo sua estabilidade. O projeto foi desenvolvido no *SolidWorks* através desse instrumento foi possível realizar testes com as dimensões da estrutura, motores, *encoder*, bateria e placa. O material utilizado na impressão 3D do chassi foi o PETG-XT que apresenta resistência a colisões entre os robôs, além de não sofrer mudança na cor e deformação. A Figura 2 apresenta o modelo do chassi da nova mecânica desenvolvida pela equipe.

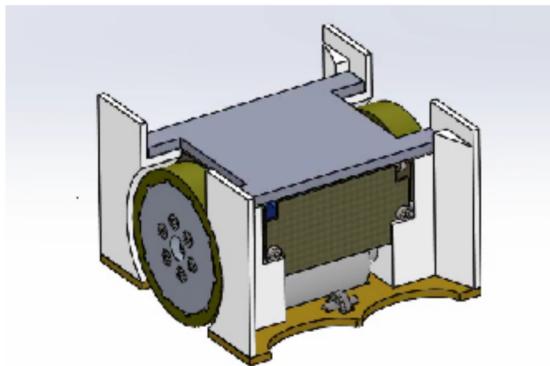


Fig. 2. Chassi do projeto

2) *Rodas*: Foram planejadas para evitar a derrapagem nas curvas mais bruscas buscando aumentar a velocidade sem perder exatidão. O material designado foi o poliuretano líquido (poly 74-20)[5] que ao endurecer e tomar forma no molde da roda, este adere ao campo, evitando giros em falso do motor.

### III. VISÃO COMPUTACIONAL

As capturas de imagens durante a partida, feitas pela câmera, são respectivamente tratadas e identificadas através das formas descritas abaixo.

#### A. Câmera

A câmera ELP-USBFHD01M-MFV, a qual é utilizada, possui uma taxa de quadros de 120 fps(quadros por segundo/*frames per second*) com uma resolução de 480 pixels. Há também um conector de entrada 2.0. Juntamente, foi utilizado uma lente de 2.8-12mm varifocal.

#### B. Transformação Linear 2D

A transformação da imagem captada pela câmera para o plano bidimensional, é a parte mais custosa computacionalmente dentro deste módulo. Esta, é de exímia importância, já que aumenta significativamente a precisão dos objetos identificados na imagem, há também a possível remoção de distorções na lente da câmera, selecionando a área útil. Uma vez restringida esta área para identificação, é possível tratar a imagem da melhor forma. Um exemplo dessa transformação pode ser visualizado na Figura 3.

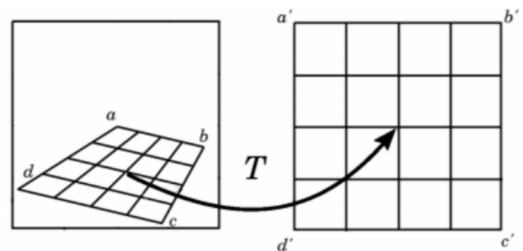


Fig. 3. Método de conversão de 3D para 2D implementado

#### C. Filtro de ruídos, conversão HSV e identificação

A imagem chega crua e após sua filtragem, é feita a conversão de *RGB* (vermelho, verde e azul - *red, green and blue*) para o *HSV*(matriz, saturação e valor - *hue, saturation and value*), com objetivo de identificar o centróide dos objetos e juntá-los. Visto isso, calcula-se a distância euclidiana entre cada um dos pontos centrais de cada região, possibilitando encontrar o centro, aproximado, do robô. Ainda assim, também pode-se determinar a orientação espacial do robô da equipe, uma vez que o sentido de orientação do robô parte do ponto diametralmente referido, cuja *tag* identifica o time. Conforme demonstrado na Figura 4, é exibido o padrão de identificação adotado pela equipe, bem como seus centros e pontos de identificação, além dos vetores de orientação importantes para a navegação.

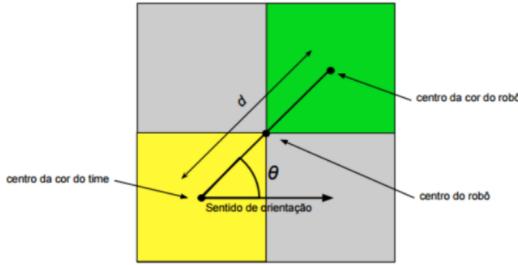


Fig. 4. Sistema de cores para identificação e orientação dos robôs

Para fazer a calibração das cores utiliza-se a *feature* apresentada na Figura 5. Com a utilização dessa interface, é possível fazer a visualização do campo e a calibração das cores em função da incidência de luz, já que em determinados pontos uma cor pode sofrer com perdas ou ganhos na saturação devido à iluminação do local, sendo um problema. Subsequentemente, alterar a intensidade de uma cor acarreta a mudança do comportamento dos robôs.

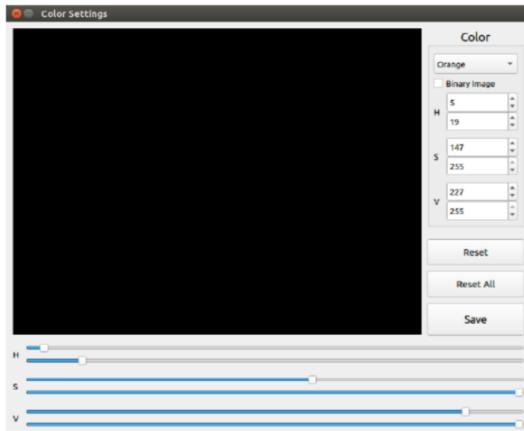


Fig. 5. Ajuste das cores para HSV

Com o intuito de corrigir isto, antecipadamente às partidas, é sempre feito a calibração das cores em todas as posições do campo em que é distinguido uma mudança de luminosidade. Para assim, reduzir os riscos de perder-se a identificação de uma cor no meio do jogo.

#### D. Mapeamento

Com o intuito de aumentar a precisão da área do campo, foi desenvolvido um programa receptor das imagens da câmera e com a captação é feita a visualização para o usuário das partes do campo, nas quais estão sendo utilizadas, há também como redimensionar o campo através do *software*.

Este é um programa desenvolvido para atender as necessidades evidentes, relacionados às linhas do campo e posicionamento das câmeras, já que durante a competição o tratamento das imagens é diretamente influenciado pela fixação da câmera, e em um efeito cascata, o comportamento dos robôs também são afetados. A partir da tela apresentada na Figura 6, pode-se verificar todos os elementos no campo,

além de observar se estes não estão sendo identificado pela visão.



Fig. 6. Mapeamento do campo através do programa

## IV. DESENVOLVIMENTO DO SOFTWARE

A fim de realizar uma melhor tomada de decisão com relação aos comportamentos de cada robô, inúmeros fatores são levados em consideração. Portanto, o conceito de inteligência artificial se aplica a este contexto, visto que as ações são executadas de modo a obter um bom desempenho defensivo e ofensivo dentro da partida.

### A. Estratégia

A estratégia adotada pela equipe parte do princípio que o time consiga anular as principais características ofensivas do time adversário. Dessa forma, dada a condição de jogo, é feita uma análise sobre o comportamento adotado pelos robôs adversários a fim de definir a melhor formação a ser adotada. Tal análise leva em consideração inúmeros fatores, os quais são: placar da partida, posicionamento do time adversário, número de robôs atacantes, entre outros. Ao realizar um estudo acerca de cada equipe, evita-se ser surpreendido em um início de partida.

### B. Máquina de estado

A definição da estratégia e, consequentemente, a função de cada jogador é definida por meio de uma máquina de estados[6]. Dessa forma, essa estrutura funciona como um técnico. A partir da posição da bola, a qual é tomada como referência para definição da estratégia, permite-se definir o papel de cada robô. A zona do campo na qual se encontra a bola é de suma importância nessa decisão. Sendo assim, a equipe busca métodos mais eficientes para a realização de tal análise de modo que o desempenho do time melhore. O principal fator a ser analisado é buscar um bom desempenho defensivo, a fim de não ser surpreendido em jogadas de contra-ataque.

### C. Behaviour Tree

A árvore de comportamentos [7] vem sendo implementada pela equipe a fim de substituir a máquina de estados que atua no comportamento dos robôs. Isso possibilita uma maior versatilidade na mudança de funções entre os jogadores e

uma maior modularidade do *software* para a implementação de novos comportamentos para cada agente (robô). A Figura 7 apresenta uma estrutura de dado do tipo árvore:

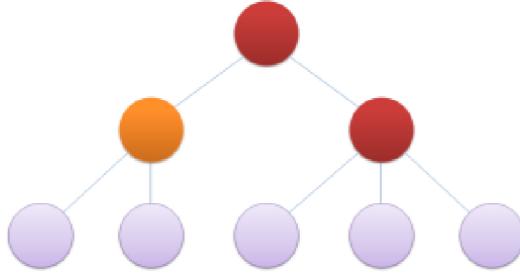


Fig. 7. *Behaviour Tree* - Árvore de Comportamento

Tal estrutura do tipo árvore é composta por nós, os quais podem ser dos seguintes tipos:

- *Nó Seletor*: O nó seletor é utilizado quando deseja-se que apenas um comportamento do nó filho seja executado. A estrutura é semelhante ao nó de sequência, mas uma vez que um nó filho (de qualquer tipo) retorna *SUCCESS*, nenhum nó irmão será executado. Porém, caso algum nó retorne *FAILURE*, o próximo nó é selecionado e assim por diante, até que algum retorne *SUCCESS*.
- *Nó de Sequência*: O nó de sequência é utilizado quando deseja-se que vários comportamentos sejam assumidos em sequência. Dessa forma, quando o primeiro nó filho retorna *SUCCESS*, o nó de sequência executará o filho seguinte. Caso algum nó filho retorne *FAILURE*, o nó de sequência retorna *FAILURE* para o nível acima. Sendo assim, este tipo de nó só retorna *SUCCESS*, se todos os seus filhos retornarem *SUCCESS*.
- *Nó Decorador*: Tal nó possui apenas um único filho e pode controlar o número de vezes que um nó filho será executado. Além disso, este nó pode alterar o que o nó filho retorna, baseado no que foi definido pelo programador da estrutura.
- *Nó Condicionador*: Este nó possibilita verificar situações específicas para que um determinado comportamento seja assumido.

A partir disso, o comportamento dos robôs são assumidos com maior exatidão, pois as ações são executadas em sequência ou abortadas caso algo de errado. Em virtude de uma má execução da ação, o agente sempre executa um comportamento paralelo (secundário). Além desse fator, é possível obter um maior número de jogadas sem impactar múltiplas partes de *software* da inteligência, possibilitando a implementação de jogadas ensaiadas, posicionamentos estratégicos, entre outros recursos.

#### D. Navegação

Para determinação do caminho a ser realizado por cada robô, a equipe continua utilizando o *UVF* (*Unit Vector Field*)<sup>[8]</sup>, visto que a navegação dos robôs tem o resultado próximo ao planejado no período de testes, como é demonstrado na Figura 8.

Para buscar uma melhor estabilidade e constância da trajetória a ser realizada pelo robô, utiliza-se dois tipos de controle PID (Proporcional Integral Derivativo). O primeiro é feito via *software*, utilizando a angulação do robô como erro do sistema. O segundo é feito via *firmware*, o qual utiliza a leitura do *encoder* para verificar o número de rotações do motor em um determinado período de tempo. A partir disso, é possível verificar se está rotacionando com a velocidade necessária.

Ao unir o sistema *UVF* ao controle PID, o comportamento dos robôs no campo fica mais fluido, permitindo que estes desviam de obstáculos por meio do *UVF*, atinjam o ponto futuro com maior precisão e naveguem mais rapidamente por meio do controle PID.

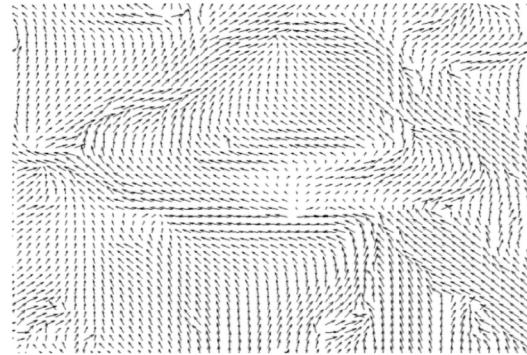


Fig. 8. Sistema *UVF* plotado na tela

## V. SIMULADOR

A fim de obter melhores resultados nos testes reais da categoria, a equipe utiliza o simulador *FIRASim*<sup>[9]</sup>. Dessa forma, possibilita-se a implementação de novas estratégias no ambiente virtual e, depois, aplicá-las em um ambiente real de VSSS. Para o funcionamento das rotinas de teste, bem como a análise sobre as situações de jogo, são necessários dois *softwares*.

#### A. FIRASim

O simulador é um *software* desenvolvido pela *ParsianLab*, o qual permite simular uma partida de futebol de robôs. Tal *software* faz uso de algumas bibliotecas, as quais são citadas abaixo:

- *ODE (Open Dynamics Engine)*: Tal recurso é uma biblioteca<sup>[10]</sup> *Open Source*, que é capaz de simular a física de corpos rígidos.
- *OpenGL*: É uma *API (Application Programming Interface)* <sup>[11]</sup>, que permite desenhar objetos no simulador, bem como robôs, bola, etc.
- *Protobuf*: Para que ocorra a comunicação entre o simulador e seus clientes, esta biblioteca<sup>[12]</sup> possibilita a serialização dos dados. Para isto, as mensagens são definidas em arquivos “.proto”, os quais são compilados, gerando as classes que permitem o acesso às informações.

- *QT*: Tal framework é *Open Source* e oferece inúmeras ferramentas no desenvolvimento da interface, entre outros recursos.

### B. Comunicação

A arquitetura de comunicação[13] segue a ideia de cliente-servidor, o qual o *FIRASim* envia pacotes multicast na rede e os clientes (times e *Referee*) captam as informações. De certa maneira, os times são considerados clientes do *Referee*, uma vez que estes dependem diretamente das situações de jogo para definir o comportamento dos agentes (robôs) em campo. Abaixo, na Figura 9 pode-se observar a estrutura de comunicação entre o Simulador, Árbitro e o time de VSSS.

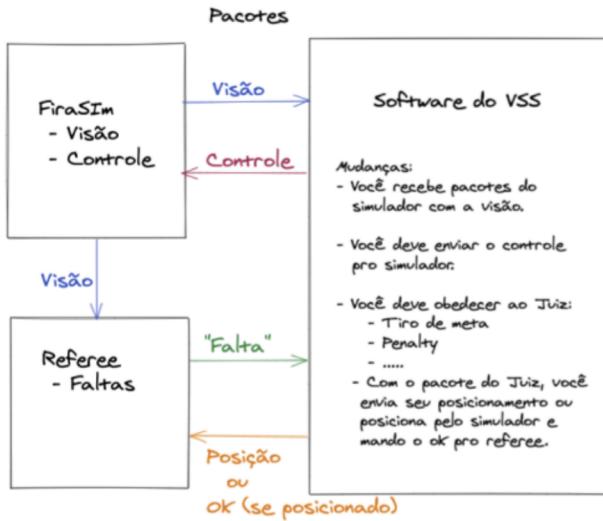


Fig. 9. Esquema de comunicação no simulador

### C. Referee

O *Referee*[14] é um cliente do *FIRASim*, o qual processa as informações de visão fornecidas pelo simulador. Dessa forma, é possível verificar as situações de jogo e possíveis faltas cometidas por ambas as equipes, além de realizar o posicionamento dos robôs. Caso alguma infração seja cometida, este envia uma mensagem para as equipes, informando a falta ocorrida. A Figura 10 apresenta o fluxo de trabalho do software do árbitro:

### VI. TESTES E RESULTADOS

Para firmar o projeto, a equipe participa de competições nas quais submete os robôs às mais diversas situações de jogo. Também são realizados experimentos no laboratório simulando o cenário dos campeonatos no intuito de verificar a exatidão dos robôs antes dos jogos oficiais, assim podendo aplicar mudanças sejam elas no código, na eletrônica ou na mecânica. Com a implementação do simulador *FIRASim* os testes desenrolam-se melhor, o código pode ser testado separadamente do *hardware*, permitindo uma maior análise de possíveis *bugs*, assim minando os erros antes do código ser testado nos robôs em campo, no laboratório ou em competições.

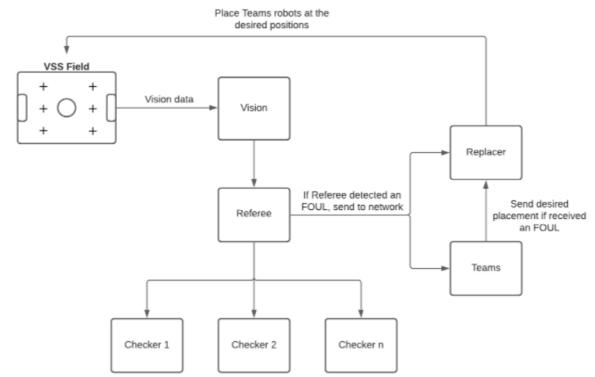


Fig. 10. Fluxo de trabalho do *Referee*

- Teste dos motores: os motores aos pares são estudados para analisar quais possuem o giro similar e serem colocados no mesmo robô com a intenção de minimizar oscilações na trajetória. O ensaio foi realizado visualmente, mas existem formas mais concretas de suceder esta verificação.
- Teste de velocidade: realizado para comparar o limite de velocidade que o robô alcança sem perder a exatidão na trajetória, habitualmente elaborado na locomoção da bolinha pelo campo e amplificando a potência nos motores.
- Teste de estratégia no simulador: realizado no simulador com intuito de criar novas táticas de modo a obter um melhor desempenho nas partidas.

Esses testes básicos asseguram a performance dos robôs em campo. Ressalta-se que, com a introdução do simulador na última competição, os testes com os robôs físicos perduram, contudo o código é testado através da simulação afirmando, em casos ideais, ainda mais o sucesso do estudo da inteligência artificial.

### VII. CONCLUSÃO

Portanto, conclui-se que o projeto tem sido aproveitado e traz resultados bastante favoráveis para o time. Desde a implementação do simulador, houve uma ampliação brusca das possibilidades, pois a elaboração de novas jogadas traz novas oportunidades à longo prazo, por poupar algum tempo e recursos que seriam utilizados na fabricação de itens da mecânica e da eletrônica. E, dessa forma, os estudos dessas partes puderam ser aprimorados com maior calma e maior eficiência. Com os resultados obtidos dos testes fica evidente que os objetivos vêm, aos poucos, sendo alcançados e os estudos aplicados foram de grande importância.

### REFERENCES

- [1] IEEE. *Very Small Size Soccer (VSSS) Rules*. URL: <https://www.cbrobotica.org/wp-content/uploads/2022/05/vssRules.pdf> (visited on 06/20/2022).

- [2] STMicroelectronics. *STM32G491RE*. URL: [https://www.st.com/content/st\\_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32g4-series/stm32g4x1/stm32g491re.html](https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32g4-series/stm32g4x1/stm32g491re.html) (visited on 06/28/2022).
- [3] Maxon Group. *Encoder ENX 10 QUAD, 1 CPT*. URL: <https://www.maxongroup.com/maxon/view/product/sensor/encoder/ENX/ENX10QUAD/ENX10QUAD01> (visited on 05/10/2022).
- [4] Maxon Group. *DC-max 22 S Ø22 mm, graphite brushes, sintered bearings*. URL: <https://www.maxongroup.com/maxon/view/product/motor/dcmotor/DC-max/DC-max22/DC-MAX22S01GBSL529> (visited on 05/10/2022).
- [5] MoldFlex. *POLY 74-20 - BORRACHA DE POLIURETANO*. URL: <https://www.moldflexmodelagem.com/poliuretano/poly-74-20> (visited on 06/02/2022).
- [6] M. Colledanchise. *Behavior Trees in Robotics and AI*. 1st. 2015.
- [7] Bazan A. *Inteligência Artificial e Behaviour Trees*. URL: <https://beattheplaguegame.wordpress.com/2012/05/25/inteligencia-artificial-e-behaviour-trees/> (visited on 05/20/2022).
- [8] Khan Academy. *Vector Fields*. URL: <https://www.khanacademy.org/math/multivariable-calculus/thinking-about-multivariable-function/ways-to-represent-multivariable-functions/a/vector-fields> (visited on 06/14/2022).
- [9] VSSSLeague. *FIRASim*. URL: <https://github.com/VSSSLeague/FIRASim> (visited on 06/29/2022).
- [10] Russ Smith. *Open Dynamics Engine*. URL: <http://www.ode.org/> (visited on 07/04/2020).
- [11] OpenGL. *OpenGL*. URL: <https://www.opengl.org/> (visited on 07/04/2021).
- [12] Google. *Protocol Buffers*. URL: <https://developers.google.com/protocol-buffers/> (visited on 06/09/2022).
- [13] Robocin. *Introdução ao FIRASim*. URL: <https://drive.google.com/file/d/14MZ13T2ztmO7dDnpCYRy4KFDlzB1HYhx/view> (visited on 07/01/2020).
- [14] Maracatronics Robotics. *VSSReferee*. URL: <https://github.com/MaracatronicsRobotics/VSSReferee> (visited on 07/13/2020).