

# RoboCIn IA Description Paper

José Nilton de Oliveira Lima Júnior<sup>1</sup>, Pedro Henrique Magalhães Braga<sup>1</sup>,  
Renê de Azevedo Delgado<sup>1</sup>, Vinicius Bezerra Silva<sup>1</sup>,  
Hansenclever de Franca Bassani<sup>1</sup>, Edna Natividade da Silva Barros<sup>1</sup>

**Resumo**—Este *Team Description Paper* (TDP) apresenta a proposta desenvolvida por RoboCIn IA do Centro de Informática da UFPE para participar na categoria IEEE *Very Small Size Soccer* (VSS). Este TDP descreve os módulos utilizados no controle e planejamento das ações dos robôs, assim como suas especificações mecânicas e eletrônicas. Nesta equipe todas as tomadas de decisões dos robôs foram aprendidas através de algoritmos de aprendizagem de máquinas que visam criar estratégias de jogo de forma cooperativa entre os agentes.

## I. INTRODUÇÃO

A competição da categoria VSS exige de cada equipe 4 módulos principais: detecção de objetos, planejamento de ações, agente físico e comunicação. O módulo de detecção de objetos deve fornecer as informações do estado do ambiente como entrada para o sistema. Este módulo foi implementado através de técnicas clássicas de visão computacional para extrair informações úteis como posição e velocidade dos agentes e da bola. Técnicas estado da arte de aprendizagem semi-supervisionada foram utilizadas para o propósito de planejamento de ações. Cada agente do nosso time foi planejado para ser o mais modular e de fácil manutenção possível. A comunicação foi desenvolvida sobre a tecnologia ZigBee e NRF. A equipe ainda utilizou de um simulador da categoria VSS para treinar sua estratégia de maneira rápida. Esta equipe contou com o apoio da equipe principal do RoboCIn para o desenvolvimento dos módulos do sistema, nossos robôs também compartilham parte da arquitetura da equipe principal.

Este trabalho se divide da seguinte forma: Seção II fala sobre o sistema de visão e as técnicas escolhidas para identificar os objetos no ambiente, seção III apresenta como a estratégia de jogo foi criada, seção IV explica as características mecânicas dos agentes, seção V descreve a eletrônica dos agentes e seção VI apresenta as conclusões do nosso projeto.

## II. SISTEMA DE VISÃO

O sistema de visão computacional é responsável por extrair as posições dos robôs em campo e da bola através de uma imagem capturada de uma câmera posicionada a 2 metros de altura sobre o campo. O time RoboCIn IA utiliza um software desenvolvido em C++ utilizando as bibliotecas *Qt* e *OpenCV* 2.4.13. Para que o sistema de aprendizagem supervisionada tenha bom desempenho, é necessário que o sistema de visão seja estável e apresente pouco atraso entre as informações extraídas e a situação atual do campo. Para satisfazer tais

requisitos, é utilizada a técnica de segmentação por cores como a base do sistema de visão em discussão. A segmentação por cores apresenta como características o baixo tempo de processamento e robustez a ruído gaussiano na imagem, características que atendem aos requisitos discutidos.

A técnica de segmentação por cores se baseia na classificação de cada *pixel* da imagem em uma classe, onde a pertinência de um *pixel* em uma classe é satisfeita se os valores dos canais do *pixel* se encontram dentro de um intervalo pré-definido. Um fator determinante para uma boa performance da segmentação é a escolha de um espaço de cores que facilite a definição desses intervalos. Dessa forma, o espaço de cores YUV se apresenta como um ótimo facilitador da segmentação, pois mantém as informações acerca da luminosidade em apenas um canal, tornando o sistema mais robusto à luminosidade. A Figura 1 apresenta a definição de um intervalo no espaço YUV.

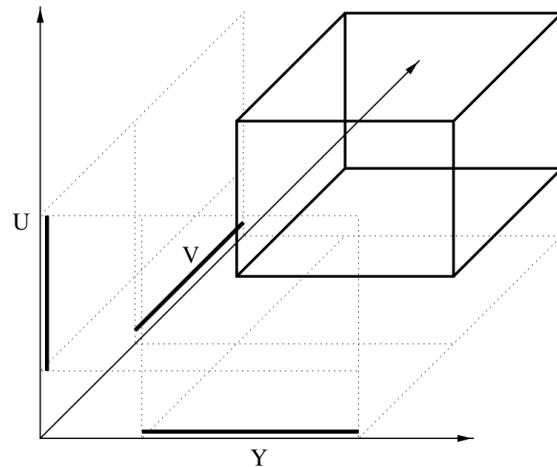


Figura 1. Intervalos de segmentação no espaço YUV.

Para otimizar o tempo de processamento, o software do RoboCIn IA também se utiliza do fato de que a classificação de um *pixel* é estática, isto é, uma tripla (x,y,z) de valores dos canais de um *pixel* sempre serão classificados na mesma classe caso os valores de x, y e z não mudem. Dessa forma, o software do RoboCIn IA calcula previamente as classes de todos os valores de *pixels* possíveis previamente, guardando tais valores em uma tabela. Essa tabela é então consultada em

<sup>1</sup>Todos os autores estão no RoboCIn no Centro de Informática, Universidade Federal de Pernambuco, Brazil [robocin@cin.ufpe.br](mailto:robocin@cin.ufpe.br)

tempo de execução para classificar os *pixels* da imagem. A utilização de uma tabela de consulta otimiza o processamento pois elimina a necessidade de cálculos complexos em tempo de execução.

Por fim, para extrair as posições da imagem com os *pixels* classificados, é utilizada a técnica de crescimento de regiões, onde *pixels* vizinhos com a mesma classe são agrupados e o centro da região é calculado através da média das posições dos pixels de um mesmo grupo. Dessa forma, as posições de cada robô são calculadas através da média entre as posições de duas regiões de cores que compõem sua etiqueta. Essas informações são então passadas para o módulo de estratégia, que utilizará as posições para tomar as decisões de jogo.

### III. ESTRATÉGIA

O módulo de estratégia é responsável pelo planejamento e tomada de decisão dos robôs, o que envolve a escolha das melhores ações e dos caminhos à serem seguidos. Para tal, utiliza-se como fonte de informação a posição, orientação e velocidade dos robôs e da bola presentes em campo. Essas informações correspondem ao resultado do processamento realizado pelo módulo de visão. É importante mencionar que o módulo de estratégia precisa levar em conta não só a dinâmica e disposição dos robôs e da bola, mas também o tempo necessário para que uma ação seja tomada.

Nossa equipe se propõe a utilizar ferramentas de aprendizagem de máquina para a realização das tarefas supracitadas. Dessa forma, resolveu-se utilizar aprendizagem semi-supervisionada e redes neurais profundas como aproximadores de função. Estas técnicas tem apresentado resultados que ultrapassam até mesmo a eficiência de humanos em determinados ambientes. VSS é um desafio complexo para técnicas de aprendizagem, o jogo possui muitos estados e ações possíveis em cada instante. O VSS também exige cooperação de agentes independentes no campo e antecipação das ações dos oponentes. Por isso, a criação de estratégias através de aprendizagem semi-supervisionada profunda mostra-se uma tarefa desafiadora, mas esse tipo de técnica tem o potencial para produzir resultados únicos. Resultados estes melhores que os obtidos por humanos no mesmo ambiente.

No decorrer desta seção falaremos um pouco em como adaptamos as técnicas mencionadas para o contexto da competição.

#### A. APRENDIZAGEM SEMI-SUPERVISIONADA PROFUNDA

Aprendizagem semi-supervisionada depende diretamente da interação de uma agente com seu ambiente e da percepção desse agente sobre suas ações. Uma transição de estados no ambiente pode ser modelada como um processo de decisão markoviano (MDP), Figura 2. Dado que o agente se encontra em um estado qualquer, ao aplicar uma ação escolhida tem-se uma probabilidade de ir para um novo estado do sistema, recebendo por essa ação uma nova recompensa do ambiente. O objetivo do agente inteligente apresentado aqui é modelar o processo de decisão markoviano e decidir uma política de ações com maior probabilidade de garantir reforços positivos.

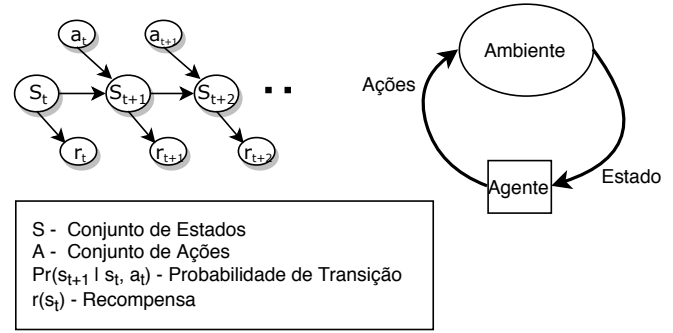


Figura 2. Processo de decisão markoviano.

As técnicas de aprendizagem semi-supervisionada começaram a ganhar grande vigor quando usadas em conjunto com redes neurais profundas. As redes neurais profundas servem como aproximadores de função nas diversas técnicas usadas. Nos últimos 10 anos redes neurais profundas tem tido resultados muito bons na área de visão computacional, principalmente em aprendizagem supervisionada, devido à sua capacidade de criar representações internas dos dados recebido que facilitam a tarefa final da rede. Essas representações criadas pela rede neural profunda vem mostraram-se, posteriormente, efetivas para aprendizagem semi-supervisionada em ambientes simples como jogos de Atari. Recentemente, estas técnicas foram sendo rebuscadas para ambientes cada vez mais dinâmicos e diversos, desde jogos como Quake, até aplicações em robótica, e no mercado financeiro. Em um dos mais recentes artigos, Jaderberg em [12] consegue treinar um agente para jogar Quake III, um jogo tridimensional de caça-bandeira. O desempenho do agente superou a performance de humanos neste ambiente.

Mnih, Volodymyr, et al. em [13] usa a arquitetura mostrada na Figura 3 para aprender diversos tipos de jogos de Atari a partir dos *pixels* da imagem. Nos baseamos nesse e em outros modelos de arquiteturas recorrentes para criar uma arquitetura que pudesse ser usada em tempo real e que fosse robusta o suficiente para aprender a complexidade da competição.

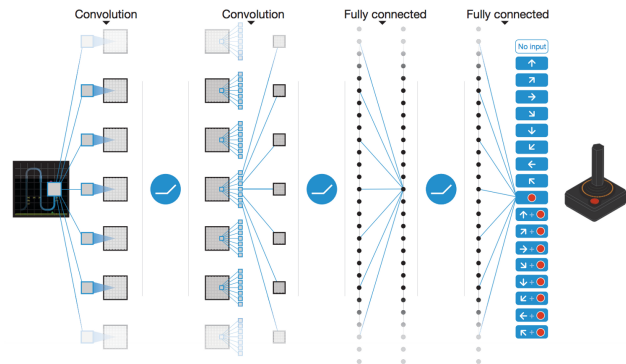


Figura 3. Arquitetura da Rede Neural utilizada em [13] para jogos de Atari.

## B. SIMULADOR

Modelos de aprendizagem profunda normalmente necessitam de muitas amostras para convergir para uma boa solução. Em aprendizagem semi-supervisionada esse problema é ainda mais grave, as técnicas por muitas vezes se mostram instáveis e precisam de muitas execuções no ambiente para criar um mapeamento confiável entre espaço de estados e espaço de ações. Devido a essa limitação, torna-se necessário para o treinamento dessas técnicas um ambiente simulado do jogo. Por isso, durante o treinamento utilizamos o simulador VSS-SDK [10]. Este simulador usa a engine de simulação física *Bullet* [11]. Embora esse simulador não seja fiel a realidade devido a simplificações na simulação física, esta característica faz com que possamos executar a simulação muito mais rapidamente, o que é de extrema importância para o sucesso do treinamento.

O simulador funciona recebendo comandos em certos intervalos de tempo do módulo de planejamento que executa independentemente. Essa comunicação é feita através de *sockets*. Para assegurarmos que o sistema simulado vá ser uma cópia fiel do sistema real, nós tivemos que certificar que o simulador receba uma taxa de comandos por volta de 4hz, que foi a taxa média percebida em cenário real de jogo. Com as alterações feitas podemos gerar simulações bem controladas até 10x mais rápidas do que execuções em tempo real.

## C. ESCOLHA DA AÇÃO

A qualquer instante do jogo a rede neural recebe do módulo de visão o estado dos agentes e da bola no campo e deve decidir qual o comportamento de cada robô que produzirá um melhor resultado para o time no futuro. Este trabalho procurou introduzir o mínimo de viés possível na rede quanto a tomada de decisões. O propósito é que a rede com o menor número possível de pistas externas encontre uma boa solução para cada situação. Embora a ausência desse bias nos produza uma estratégia livre de interferências humanas, que podem ser prejudiciais para a solução ótima, menor interferência também torna o aprendizado da estratégia lento.

A nosso módulo de planejamento considera a posição e deslocamento da bola em relação aos gols como indicativo de quão bom está a sua situação no jogo, sendo um gol o momento de melhor ou pior retorno para a rede. Para este trabalho, escolhemos fazer um controle de mais baixo nível dos agentes, aplicando diretamente ou suas posições desejadas ou suas velocidades. Assim o sistema deve aprender a planejar e controlar os agentes na cena.

Como parte da estratégia, um robô do jogo foi treinado para ser responsável em evitar retornos negativos para a rede, ou seja, sua prioridade é ser um defensor ou goleiro, devendo enviar a bola para o lado oposto do campo, os demais agentes são treinados para cooperar e escolher os melhores posicionamentos para que o sistema obtenha recompensas mais positivas, procurando sempre fazer um gol. As ações escolhidas pelo sistema são subótimas do contexto da percepção humana, mas criam comportamentos que atingem o objetivo do jogo de fazer e evitar um gol.

## IV. MECÂNICA

A mecânica foi projetada baseada nos trabalhos da categoria VSS da equipe RobôCIn. Então, foi incorporado a arquitetura modular da mecânica do robô pensando na facilidade no momento da montagem e manutenção. Para garantir esta modularidade, o robô foi todo construído utilizando-se tecnologia de impressão 3D. A Figura 4 apresenta a estrutura interna do robô construída para competição. Para o projeto de todas as peças usadas, utilizou-se a ferramenta Solid Works 3D CAD [5], uma ferramenta que permite a modelagem e exportação para impressão 3D. A estrutura é dividida em três módulos gerais:

- 1) Bateria, que se conecta ao módulo de motores por meio de encaixe;
- 2) Motores, que se conectam na placa de circuito através de parafusos;
- 3) Capa, que é encaixada na placa de circuito impresso que fica na parte inferior do robô.

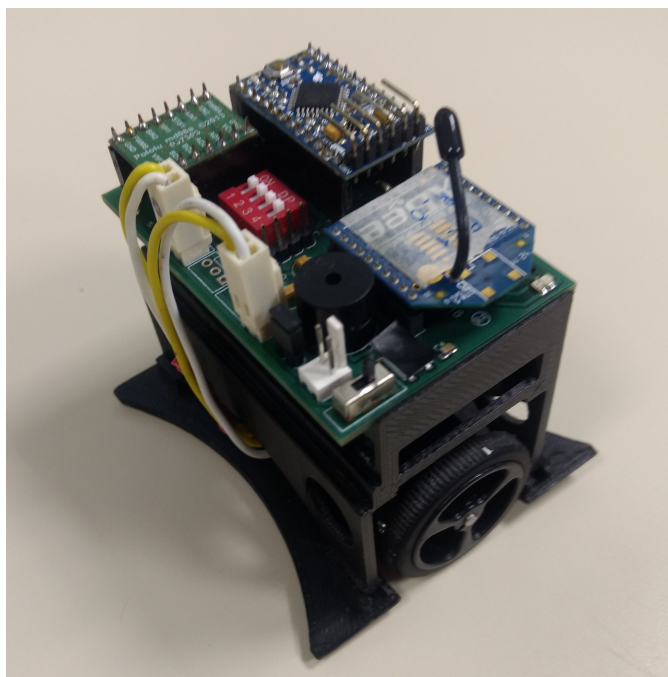


Figura 4. Estrutura interna do robô, seus módulos estruturais e eletrônicos.

Para compôr o robô dois micro motores com redução integrada de 30:1 foram utilizados na montagem. Tais motores conseguem atingir 1000 RPM com torque de 0,45 Kg-cm. O robô utiliza duas rodas de 32mm de diâmetro para sua locomoção.

## V. ELETRÔNICA

O robô foi implementado em um Arduino Pro Mini de 16 MHz que controla todas as suas operações locais. Esta placa foi escolhida por ser de fácil acesso para desenvolvimento, ter boa disponibilidade de módulos prontos utilizados no controle de outras funções do robô e, por seu tamanho reduzido, ser fácil de ser embarcada no robô. O controle de potência dos

motores é feito através do driver TB6612FNG. Este módulo driver ponte H duplo 1A para motor DC e motor de passo feito para arduino pode controlar até 2 motores DC (motor de corrente contínua) com uma corrente constante de 1.2A (3.2A de pico).

Cada robô é composto de uma IMU (*Inertial Measurement Unit*), que contem sensores importantes para determinar-se o estado do robô no ambiente, como: acelerômetro, magnetômetro e um giroscópio. Tais informações são essenciais para o planejamento de ações do time. A transmissão dessas informações é feita ao computador através de módulo de interface *Wireless* do Arduino. Mais aspectos dessa comunicação são descritos na seção COMUNICAÇÃO deste documento.

Cada robô possui dois *encoders* desenvolvidos pela equipe parceira RoboCin acoplados às rodas, que, através de suas aferições, possibilitam um controle preciso por parte do Arduino das velocidades em cada roda do robô. Tal controle é feito por um controlador PID implementado no Arduino, cujos parâmetros foram empiricamente ajustados para uma maior estabilidade das velocidades. Além disso, planeja-se ainda incluir sensores ópticos, como os utilizados em mouses, que, juntamente com as informações da IMU, melhorarão as medições referentes a posições e orientações dos robôs e seus derivativos.

O circuito utilizado em cada robô foi impressor em placa de dupla camada utilizando a prototipadora LPKF ProtoMat S63. O desenho do circuito foi feito utilizando o Eagle [6], software especializado em produzir placas de circuito impresso. A Figura 5 mostra o resultado final do circuito implementado.

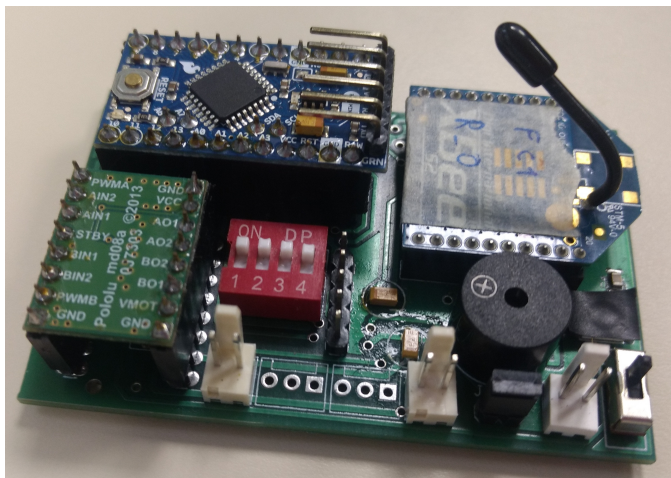


Figura 5. Circuito elétrico montado projetado através do Eagle Circuit CAD.

## VI. COMUNICAÇÃO

Na etapa de comunicação, onde ocorre a troca de mensagens entre o computador e os robôs, foi utilizada a tecnologia ZigBee, mais especificamente o xBee Series 2. Assim, torna-se possível enviar pelo computador informações como velocidade dos motores, além de receber dados como o nível de bateria de cada robô.

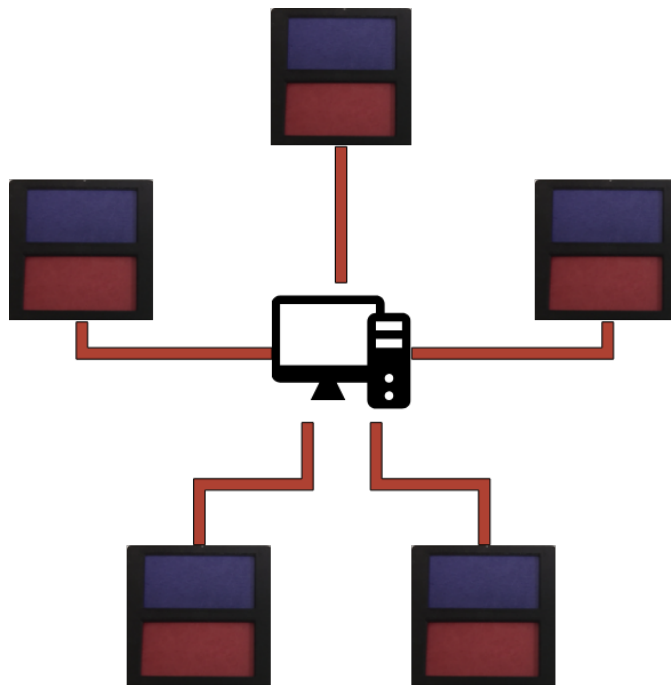


Figura 6. Topologia de rede estrela. Cinco robôs são diretamente conectados ao master, representado pelo computador (no centro).

Para a realização da comunicação, foi definido um protocolo para a mensagem *broadcast* enviada do computador para o robô. Esse protocolo é dividido em seções de bits para conter informações do ID do robô de destino, a velocidade de cada motor, e se é preciso que o robô retorne o nível de bateria.

Contudo, uma equipe parceira do RoboCin já havia tido problemas de interferência com o rádio xBee durante a competição Very Small Size de 2016 e, conseqüentemente, houveram falhas na comunicação com todos robôs. Isto acontece porque o xBee Series 2 utiliza um protocolo que busca a entrega das mensagens, ou seja, caso algum robô não seja encontrado na rede, todos os rádios da rede ecoavam a mensagem para tentar achá-lo, e, portanto, com um robô perdido, todos os outros ficavam ocupados ecoando a mensagem, mas sem obter sucesso. Logo, com todos os rádios ocupados, os robôs deixavam de receber os seus próprios pacotes que continham os dados para navegação. Reconhecendo que a comunicação é crucial para o funcionamento do sistema, testes em outras tecnologias foram realizados pela equipe. Assim, surgiu então, uma adaptação no circuito para permitir o uso tanto do xBee quanto do módulo nRF24L01+. Desta maneira, existe uma tecnologia de *backup* que permite se comunicar com até 5 robôs numa topologia de rede em estrela (Figura 6), que também é mais flexível que o xBee na implementação do protocolo de comunicação.

## VII. CONCLUSÃO

Este *Team Description Paper* (TDP) descreve o projeto e a arquitetura dos módulos contruídos pela equipe para participar da categoria *Very Small Size Soccer League*. Nós descrevemos

neste TDP o módulo de detecção de objetos para localizar a posição dos agentes e da bola no campo, um módulo de planejamento que gera estratégias e controla os agentes em campo, a arquitetura mecânica, eletrônica e de comunicação que constituem cada agente.

## AGRADECIMENTOS

A equipe gostaria de agradecer o Centro de Informática da UFPE e a FACEPE pelo apoio financeiro e de recursos durante todo o processo do projeto. Também gostaríamos de agradecer à todo apoio dado pelos professores Edna Barros e Hansenclever Bassani.

## REFERÊNCIAS

- [1] OpenCV. Disponível em: <http://opencv.org/>. Acessado por último em 20 de junho de 2016.
- [2] Sommerville, Ian. Engenharia de Software, 9ª edição. Pearson Education.
- [3] Discrete YUV Look-Up Tables for Fast Colour Segmentation for Robotic Applications
- [4] Hartigan, J., & Wong, M. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1), 100-108. doi:1. Retrieved from <http://www.jstor.org/stable/2346830>
- [5] CAD Solid Works. Disponível em: <http://www.solidworks.com/>. Acessado por último em 20 de junho de 2016.
- [6] CAD Eagle. Disponível em: <http://www.cadsoftusa.com/>. Acessado por último em 20 de junho de 2016.
- [7] Roland Siegwart and Illah Reza Nourbakhsh and Davide Scaramuzza *Introduction to Autonomous Mobile Robots*. The MIT Press, 2ª edição 2011.
- [8] Seesink, R. A. (2003). Artificial Intelligence in multi-agent robot soccer domain (Doctoral dissertation, Masters Thesis, University of Twente).
- [9] Seesink, R. A. (2003). Artificial Intelligence in multi-agent robot soccer domain (Doctoral dissertation, Masters Thesis, University of Twente).
- [10] VSS-SDK Simulator. Disponível em: <https://github.com/VSS-SDK/VSS-Simulator>. Acessado por último em 04 de julho de 2018.
- [11] Bullet physics engine. Disponível em: <https://github.com/bulletphysics/bullet3>. Acessado por último em 04 de julho de 2018.
- [12] Jaderberg, M., Czarnecki, W. M., Dunning, I., et al. 2018, arXiv:1807.01281
- [13] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529.