RoboCIn IA Description Paper

José Nilton de Oliveira Lima Júnior[1], Heitor Rapela de Medeiros [1],
Pedro Henrique Magalhães Braga[1], Reniê de Azevedo Delgado [1],
Hansenclever de Franca Bassani[1], Edna Natividade da Silva Barros[1]

***Resumo*—Este *Team Description Paper* (TDP) aprensenta 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 profunda por reforço que visam criar estratégias de jogo de forma cooperativa entres 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 por reforço profundas 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 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 base para o sistema de visão em discussão. A segmentação por cores apresenta como características o baixo tempo de processamento e robustez a ruído gaussiano na imagem, atendendo 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.

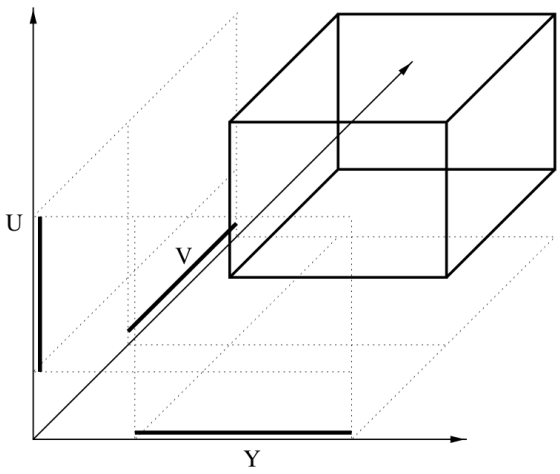

Figura 1. Intervalos de segmentação no espaço YUV.

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

[1]Todos os autores estão no RobôCIn no Centro de Informática, Universidade Federal de Pernambuco, Brazil robocin@cin.ufpe.br

tais valores em uma tabela. Essa tabela é então consultada em 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 real.

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, foi desenvolvido um módulo de estratégia baseado em aprendizado por reforço profundo. Estas técnicas tem apresentado resultados que ultrapassam a performance de humanos em determinados ambientes, como Go e Xadrez [13], [14], porém ainda não existem aplicações dessas técnicas no domínimo do futebol de robôs.

VSS é um desafio complexo para técnicas de aprendizagem, pois o jogo possui uma grande quantidade de estados e ações possíveis por instante de tempo. O VSS também exige cooperação de agentes independentes no campo e antecipação das ações dos oponentes num ambiente altamente dinâmico. Por isso, a criação de estratégias através de aprendizagem por reforço profunda mostra-se uma tarefa desafiadora, mas com potencial para produzir resultados únicos, pois permite uma grande abstração de comportamentos complexos do futebol de robôs que necessitariam ser programados diretamente nos robôs em alto nível.

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

A. APRENDIZAGEM POR REFORÇO PROFUNDA

Aprendizagem por reforço lida com o treinamento de um agente imerso em um ambiente que aprende através do método de tentativa e erro. O ambiente fornece ao agente informações sobre o estado que se encontra e o agente toma ações no ambiente de modo a modificar seu estado. Ao executar uma ação no ambiente, o agente também recebe um valor de recompensa, que pode ser positivo ou negativo e que diz ao a gente o quão bom sua ação foi para o sucesso da tarefa. Dessa forma, a tarefa do agente de aprendizado por reforço é maximizar a recompensa cumulativa no ambiente o quão é treinado.

A 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 pode ser definido formalmente como modelar o processo de decisão markoviano e decidir uma política de ações com maior probabilidade de garantir reforços positivos.

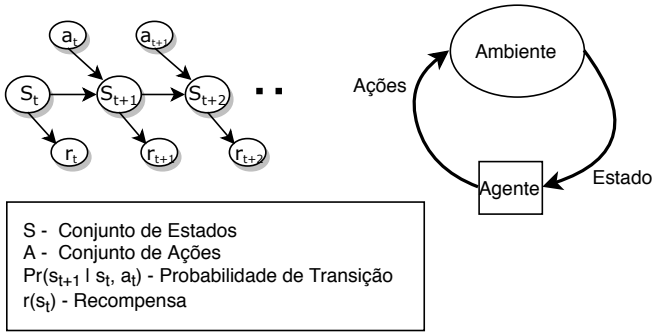

Figura 2. Processo de decisão makoviano.

O algoritmo *Deep Q Network*(DQN) [15] foi escolhido para treinar 3 agentes de aprendizado por reforço, um para cada robô. O algoritmo DQN utiliza uma rede neural para estimar os Q-valores, isto é, o quão bom é executar uma ação num determinado instante. De acordo com as interações feitas com o ambiente, o agente vai aprendendo que ações em que estados levam a recompensas boas. A Figura 3 mostra um exemplo de uma arquitetura DQN que aprende dos pixels da imagem para tomar ações em jogos de Atari3. Dessa forma, é escolhida a a tomada de ação com maior valor no ambiente.

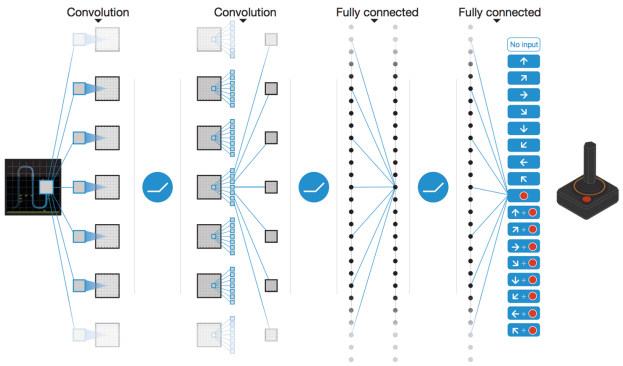

Figura 3. Arquitetura da Rede Neural utilizada em [12] para jogos de Atari.

Modelos de aprendizagem profunda necessitam de muitas amostras para convergir para uma boa solução num ambiente

com alta dimensionalidade. 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 [9]. Este simulador usa a engine de simulação física *Bullet* [10].

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.

Nosso agente DQN tem como entrada as posições, orientações e velocidade de todos os robôs em campo, além da posição e velocidade da bola. O agente pode tomar uma das 5 ações descritas na Tabela I. Essas ações modificam um alvo, que aponta para qual posição o robo deve ir no momento. Após essa modificação, é utilizado um controlador PID para levar o robô até a posição desejada. O funcionamento do agente DQN está ilustrado na Figura 4. Para possibilitar o aprendizado, cada agente aprende através de recompensas que estimam a distancia da bola para os gols e a distância do agente para a bola.

Ações	Descrição
0	Matenha a posição do alvo
1	Rotacione o alvo 15 graus no sentido anti-horário
2	Rotacione 15 graus no sentido horário
3	Avançe o alvo 12 cm
4	Recue o alvo 12 cm

Tabela I
AÇÕES DO AGENTE DQN

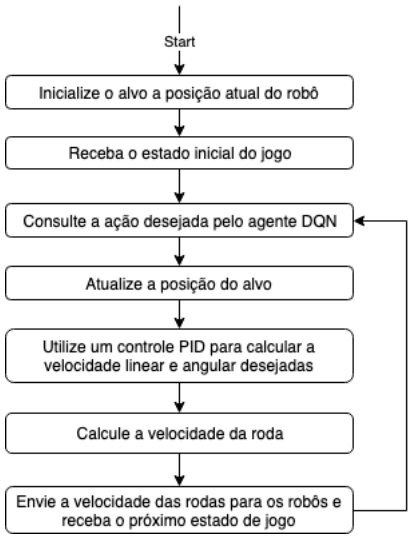

Figura 4. Fluxo de funcionamento do agente DQN

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 atigem 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 5 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 dois módulos gerais:

1) Chassi do robô, que fornece suporte estrutural as peças do robô;
2) Capa, que é encaixada na parte superior do robô para proteger a placa de circuito.

Figura 5. Estrutura do robô, seus módulos estruturais e eletrônicos.

Para compôr o robô, dois micro motores com redução integrada de 50:1 foram utilizados na montagem. Tais motores conseguem atingir 600 RPM com torque de 0,86 Kg-cm. O robô utiliza duas rodas de 50mm 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 continua) 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 VI.

Cada robô possui dois sensores ópticos, como os utilizados em mouses, que, juntamente com as informações da IMU, melhoram as medições referentes a posições e orientações dos robôs e seus derivativos.

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

Figura 6. 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.

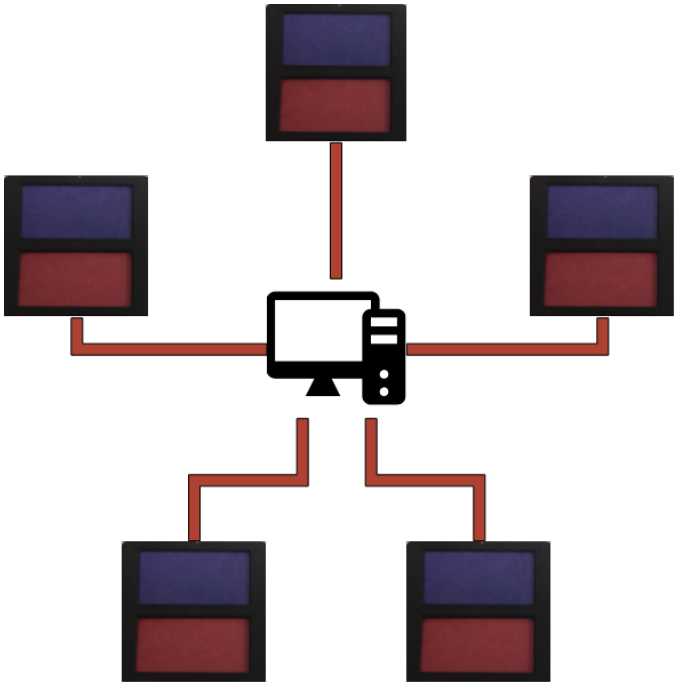

Figura 7. Topologia de rede estrela. Cinco robôs são diretamentes conectados ao master, representado pelo computador (no centro).

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, consequentemente, houveram falhas na comunicação com todos robôs. Isto acontece porque o xBee Series 2 utiliza um protocolo que busca uma entrega confiável 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 prórpios 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 7), 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] VSS-SDK Simulator. Disponível em: https://github.com/VSS-SDK/VSS-Simulator. Acessado por último em 04 de julho de 2018.
[10] Bullet physics engine. Disponível em: https://github.com/bulletphysics/bullet3. Acessado por último em 04 de julho de 2018.
[11] Jaderberg, M., Czarnecki, W. M., Dunning, I., et al. 2018, arXiv:1807.01281
[12] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning."Nature 518.7540 (2015): 529.
[13] Silver, David, et al. "Mastering the game of go without human knowledge."Nature 550.7676 (2017): 354.
[14] Campbell, M., Hoane, A. & Hsu, F.-h. "Deep Blue". Artificial Intelligence 134, 57-83 (2002).
[15] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning."arXiv preprint arXiv:1312.5602 (2013).