UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO VITOR MARIANO CORREIA

APRENDIZADO POR REFORÇO ASSISTIDO APLICADO AO CONTROLE DE ROBÔS DE ACIONAMENTO DIFERENCIAL

BAURU
Janeiro/2023

JOÃO VITOR MARIANO CORREIA

APRENDIZADO POR REFORÇO ASSISTIDO APLICADO AO CONTROLE DE ROBÔS DE ACIONAMENTO DIFERENCIAL

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. Renê Pegoraro

BAURU
Janeiro/2023

C824a

Correia, João Vitor Mariano
APRENDIZADO POR REFORÇO ASSISTIDO APLICADO AO CONTROLE DE ROBÔS DE ACIONAMENTO DIFERENCIAL / João Vitor Mariano Correia. -- Bauru, 2023
39 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru
Orientador: Renê Pegoraro

1. Robótica. 2. Aprendizado Profundo. 3. Aprendizado por Reforço. 4. Futebol de Robôs. 5. Agentes Inteligentes. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

João Vitor Mariano Correia

Aprendizado por Reforço Assistido Aplicado ao Controle de Robôs de Acionamento Diferencial

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Renê Pegoraro
Orientador
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Profª. Drª. Simone das Graças Domingues Prado
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Prof. Dr. João Paulo Papa
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Bauru, 16 de janeiro de 2023.

Dedico este trabalho para minha mãe.

Agradecimentos

Gostaria de agradecer aos meus familiares, especialmente minha mãe, que fez o possível e impossível para me proporcionar a melhor qualidade de vida possível. Aos amigos que tornaram os momentos estressantes durante os anos de graduação mais suaves e proporcionaram memórias e experiências as quais levarei para a vida toda.

À Unesp por me fornecer um ensino de qualidade, tanto durante o Ensino Médio no Colégio Técnico Industrial da Faculdade de Engenharia de Bauru quanto durante a graduação. Ao meu orientador Renê Pegoraro pelo auxílio durante o desenvolvimento deste trabalho, bem como na iniciação científica e na equipe de futebol de robôs.

Aos outros professores que lecionaram e me ensinaram a gostar cada vez mais de computação e matemática, sem o empenho destes a já sucateada universidade pública estaria em ruínas. Ao meu também orientador João Paulo Papa que possibilitou a realização de um projeto de Iniciação Científica e me fez querer continuar no mundo acadêmico.

Aos membros dos projetos de extensão os quais passei durante a graduação por proporcionarem um crescimento além do que o adquirido em sala de aula, às equipes do IEEE RAS, Biblioteca Falada, Carrossel Caipira, Centro Acadêmico Ada Lovelace e Carrossel Caipira e a equipe do Recogna.

Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less.

Marie Curie

Resumo

O presente trabalho tem como objetivo o estudo e a implementação de técnicas de aprendizado profundo aplicado ao controle de robôs de acionamento diferencial. A aplicação final destinada dos modelos apresentados são os robôs da categoria IEEE Very Small Size Soccer. Diversas são as técnicas existentes para o controle de tais agentes, as quais podem ser aplicadas em diferentes escopos de controle e navegação robótica. Este trabalho busca uma rede capaz de ser aplicada ao controle local dos agentes, para este fim foi utilizada uma arquitetura que combina uma rede ator-crítico e uma rede crítica DQN auxiliada por um controlador PID. Tal arquitetura possibilita acelerar o processo de aprendizado, uma vez que o controlador clássico pode ser utilizado nas épocas iniciais evitando que o agente faça movimentos aleatórios que pouco retorno traria à rede, e a partir do momento que a política aprendida seja eficiente na locomoção do agente, essa passe a ser utilizada. Os resultados obtidos a partir do treinamento em ambiente simulado foram comparados à implementação original e apresentadas as conclusões a partir destes.

Palavras-chave: Robótica, Aprendizado Profundo, Aprendizado por Reforço.

Abstract

This work aims to study and implement deep learning techniques applied to the control of differential drive robots. The model of control is applied in the IEEE Very Small Size Soccer league aiming to validate the code implementation. Several frameworks to control such agents exist and they can be applied to different scopes of robotic control and navigation. This work seeks a network architecture capable of being efficiently applied in the local control of such agents, for this purpose it's used a architecture of an actor-critic network and a critical DQN network aided by a PID controller. Such an architecture makes it possible to accelerate the learning process since the classic controller can be used in the initial stages, preventing the agent from making random movements that would bring little return to the network and, from the moment that the learned policy is efficient in the agent's locomotion, this starts to be applied. The results obtained from training in a simulated environment are compared to the original implementation results and the conclusions are presented from these.
Keywords: Robotics, Deep Learning, Reinforcement Learning.

Lista de figuras

Lista de abreviaturas e siglas

AFD	Autômatos Finitos Determinísticos
CNN	*Convolutional Neural Network*
DDPG	*Deep Deterministic Policy Gradient*
DQN	*Deep Q Network*
IA	Inteligência Artificial
LARC	*Latin American Robotics Competition*
MDP	*Markov Decision Process*
PID	Proporcional Integral Derivativo
ReLU	*Rectified Linear Unit*

Sumário

1 Introdução

Desde os primórdios da computação, pesquisadores da área e a população em geral se perguntam quando - e se - essas máquinas viriam a se tornar inteligentes de forma semelhante à inteligência humana. No início das pesquisas e desenvolvimento no campo de inteligência artificial, as técnicas existentes eram extremamente eficientes em resolver problemas estruturados e que podem ser formulados a partir de regras ou fórmulas matemáticas, problemas esses que são especialmente difíceis para seres humanos devido à complexidade matemática, a grande dificuldade a ser superada eram problemas com dados não estruturados que são relativamente fáceis de serem resolvidos por humanos mas difíceis de serem representados por fórmulas matemáticas ou regras, como a identificação e reconhecimento de imagens e voz por exemplo. (GOODFELLOW; BENGIO; COURVILLE, 2016).

Nas últimas décadas, técnicas de inteligência artificial se tornaram uma realidade presente no cotidiano de muitas pessoas, mesmo sem que estas pessoas se dêem conta disso, transformando diversas áreas do conhecimento e indústrias e possibilitando que computadores utilizem dados das mais diversas formas ao identificar padrões e aprender representações abstratas de alto nível inerentes a estes, como reconhecer falas ou imagens, realizar diagnósticos médicos ou automatizar rotinas de trabalho (TAI; LIU, 2016).

O campo de estudo em robótica vem se beneficiando da aplicação de tais técnicas em todos os níveis hierárquicos no planejamento e navegação de um agente, possibilitando que estes aprendam políticas de controle a partir de dados de entradas coletados por sensores, o que permite que atuem de forma totalmente autônoma e, em certo grau, inteligente. Um agente inteligente deve ser capaz de inferir conhecimento a partir da interação com o ambiente em que está inserido, melhorando a forma com que interage com o ambiente (SHABBIR; ANWER, 2018).

Uma tarefa de navegação em robótica pode ser definida como a necessidade de levar um agente de um estado inicial até um estado objetivo de forma suave e evitando colisões. Essas tarefas são especialmente desafiadoras pois requerem o reconhecimento e percepção do espaço, estimação do estado do agente, planejamento de caminho e controle. Atualmente, a maior parte dos sistemas de navegação utilizam a hierarquia clássica de navegação, com um módulo de planejamento de caminho seguido por um módulo de controle local. Apesar de ser extremamente eficiente, essa abordagem não apresenta robustez para navegação em ambientes diversos ou restritos (XIAO et al., 2020).

Uma abordagem diferente de navegação vem através do uso de redes neurais artificiais e aprendizado de máquina, com alguns trabalhos utilizando uma abordagem de aprendizado *end-to-end*, onde o modelo busca encontrar comandos de controle a partir de dados de entrada

diretos dos sensores presentes no agente, possibilitando que o mesmo se mova sem a necessidade de representações e conhecimentos humanos. O problema por trás de tal abordagem reside na grande necessidade de dados e geralmente não fornecer garantias de segurança pois funciona como uma caixa preta (XIAO et al., 2020).

Um dos principais e mais relevantes artigos e metodologias de treinamento dentro do universo de robótica e aprendizagem por reforço foi proposta por Mnih et al. (2013) onde pela primeira vez foi apresentado um modelo de aprendizado profundo capaz de aprender políticas de controle sem a necessidade de ajustes da arquitetura ou algoritmo de forma mais generalista. Para isto, é utilizada uma rede neural convolucional que utiliza de técnicas de *Q-learning* adaptadas, formulando assim as chamadas redes de *Deep Q Network* (DQN). Em cima dessa proposta, Lillicrap et al. (2015) propõe uma adaptação para que tais redes possam ser utilizadas em ambientes onde o domínio de ações possíveis é contínuo, criando um modelo ator-crítico baseado no algoritmo *deterministic policy gradient* (SILVER et al., 2014).

O presente trabalho tem por objetivo a implementação de um modelo de controle para um robô de acionamento diferencial, para isto, foi utilizada como base a arquitetura proposta por Xie et al. (2018), aliando o uso de modelos ator-crítico com técnicas de aprendizado assistido através de uma rede *Deep Deterministic Policy Gradient* (DDPG) que tem por objetivo aprender uma política de controle, um Controlador Proporcional Integral Derivativo (PID) que deve auxiliar no processo de aprendizagem do agente e acionado em determinados casos e por fim uma Rede DQN que escolhe se, para determinado estado deve ser usado a política de controle ou o controlador PID.

Na seção 2 são apresentados os conceitos por trás das técnicas de Aprendizado por Reforço, *Deep Q Learning*, *Deep Deterministic Policy Gradient*, Robôs de Acionamento Diferencial e a aplicação em competições de Futebol de Robôs. Na seção 3 é apresentada a metodologia proposta e utilizada para o trabalho e na seção 4 são discutidos os resultados obtidos.

2 Fundamentação Teórica

Este capítulo busca criar um conhecimento comum sobre os temas e problemas abordados no trabalho. Inicia-se com uma breve introdução sobre Redes Neurais Artificiais na seção 2.1, seguido por um conhecimento mais aprofundado sobre Aprendizado por Reforço na seção 2.2, na seção 2.3 discute-se a técnica de *Deep Q-learning*, seguido pela técnica de *Deep Deterministic Policy Gradient* na seção 2.4. Por último, na seção 2.5 discute-se sobre o cenário de Futebol de Robôs e a estrutura do time de futebol de robôs do Departamento de Computação da UNESP/Bauru.

2.1 Redes Neurais Artificiais

Responsáveis por grande parte do recente avanço no campo de estudo em Inteligência Artificial (IA), as redes neurais artificiais são algoritmos inspirados no funcionamento do cérebro humano e no processamento de sinais dos neurônios. São geralmente compostas por diversas camadas, sendo uma camada de entrada, uma ou mais camadas escondidas como são chamadas as camadas intermediárias, e uma camada de saída. Cada uma dessas camadas possuem diversos nós, estes são chamados neurônios e carregam uma informação chamada peso, pelo qual são multiplicados os sinais de entrada da rede (GOODFELLOW; BENGIO; COURVILLE, 2016).

Tais redes contam também com funções de ativação, que possuem como objetivo dizer se um neurônio deve ser ativado ou não, decidindo assim se a informação contida nele é relevante para o sistema ou não. Diversos tipos de função podem ser utilizados como função de ativação, sendo os mais relevantes as funções de etapa binária, função linear, sigmóide, função tanh e as ReLU's. No escopo do presente trabalho, as ReLU's se destacam quanto às demais. ReLU é uma abreviação para o termo do inglês *rectified linear unit*, que em português significa unidade linear retificadora e são definidas pela equação 2.1.

$$f(x) = max(0, x) \tag{2.1}$$

As redes neurais artificiais podem ser divididas em diversas arquiteturas onde cada uma possui uma especificidade e resolve melhor um tipo de problema. Entre as arquiteturas mais relevantes estão os perceptrons de múltiplas camadas, que são tidos como a arquitetura base para todas as outras, baseando-se no empilhamento de diversas camadas de neurônios, as redes neurais convolucionais, que possuem como especificidade o processamento e classificação de imagens ou outras tarefas de visão computacional, as redes neurais recorrentes as quais são voltadas ao processamento de sinais com características temporais e sequenciais, como o processamento de áudio e reconhecimento de linguagem natural, e também .

Olhando com cuidado para as redes neurais convolucionais, esta arquitetura possibilitou enormes avanços no processamento de imagens e outros sinais com características espaciais, sendo inspirada na organização do córtex visual humano. Esta arquitetura possibilitou uma drástica redução do número de parâmetros utilizados quando comparados com uma rede perceptron de múltiplas camadas para a identificação de imagens. Isso é feito através da aplicação da operação de convolução de matrizes entre uma matriz chamada *kernel* e a matriz de entrada. O *kernel* busca atualizar seus valores em busca de uma melhor identificação e extração das características presentes nas imagens. Este pode possuir várias dimensões, sendo os de uma dimensão (ou 1D) aqueles que se movem apenas em uma direção, obtendo como saída um vetor unidimensional, os de duas dimensões (2D) os que realizam a convolução em duas direções e obtém como saída uma matriz de duas dimensões e por fim os kernels tridimensionais que realizam a convolução em três dimensões (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.2 Aprendizado por Reforço

Enquanto analisando a inteligência humana e o processo natural de aprendizagem a primeira coisa que salta aos olhos é o aprendizado a partir da interação com o ambiente, tal tipo de aprendizado pode ser observado em todos os momentos da vida humana, desde crianças até idosos. Tomando como exemplo um bebê humano, em seu primeiro ano de vida todo o seu aprendizado ocorre através da interação com o ambiente em um processo de tentativa e erro uma vez que o mesmo não é capaz de assimilar conceitos complexos e aprender a partir de uma metodologia estruturada, mas, mesmo assim são capazes de aprender a engatinhar e consequentemente andar apenas mexendo seus braços e pernas de forma que, após algumas tentativas ele assimile os conceitos necessários e crie a coordenação motora para tal tarefa (TAI; LIU, 2016).

Problemas de aprendizado por reforço podem ser resumidos na necessidade de aprender quais ações devem ser tomadas de forma a maximizar a recompensa recebida pelo agente. São três as principais características que distinguem o Aprendizado por Reforço das técnicas de Aprendizado Supervisionado e Aprendizado Não Supervisionados: estes problemas podem ser descritos como um laço fechado uma vez que a ação tomada no instante atual pode afetar os dados de entrada futuros, o agente não sabe qual ação deve ser tomada de antemão e deve aprender isso tomando as ações e também as ações tomadas agora irão afetar não somente a recompensa atual como também as recompensas futuras (SUTTON; BARTO, 2018).

Dentro desse paradigma de aprendizagem os agentes são programados para aprender a partir de episódios de interação com o ambiente, o que é necessário fazer a fim de alcançar um determinado objetivo (SUTTON; BARTO, 2018). O objetivo aqui é minimizar uma função de custo associada a tarefa que se deseja realizar a partir da otimização da Função de Valor

ao inveś das recompensas, aumentando então as chances de sucesso ao longo prazo. Como neste tipo de técnica o agente não sabe qual ação deve ser tomada dentro do conjunto de possibilidades em um primeiro momento, a ideia é que a partir da interação com o ambiente os agentes aprendam qual a melhor ação a ser tomada em cada cenário a partir de um histórico de recompensas recebidas. Este processo de aquisição de informações geralmente não está sob total controle do agente uma vez que não é possível determinar com antecedência as consequências e resultados para todas as ações já que o modelo do processo não é preciso a tal ponto (XIAO et al., 2020).

Alguns dos conceitos mais importantes para técnicas de aprendizado por reforço são as políticas π, Funções de Valor V e o Valor Q, as quais serão explicadas a seguir. Uma política é a estratégia do agente para determinar a próxima ação a partir do estado atual, para isso, o mesmo mapeia quais ações maximizam as recompensas futuras. As Funções de Valor V dizem respeito ao retorno esperado a longo prazo em oposição à recomposição de curto prazo. Pode-se dizer que o valor V de uma política π no estado s_t é o retorno esperado a longo prazo do estado atual dentro da política atual. Em cima do conceito de Valor é definido o Valor Q, que mensura a qualidade de uma ação.

Introduzindo algumas notações matemáticas, para uma política $\pi_\psi(a|s)$ busca-se aprender o conjunto de parâmetros ψ que maximizem sua recompensa, isso é feito através da equação 2.2 onde $r(s_t, a_t)$ são as recompensas obtidas.

$$\psi^* = argmax E_\psi[\sum_t r(s_t, a_t)], \tag{2.2}$$

A Função Q por sua vez é descrita por 2.3. Para realizar o aprendizado do Valor Q é feito uso das propriedades de Markov que serão descritas mais a frente neste trabalho, com isso é possível escrevê-la com a forma de uma Equação de Bellman e utilizar os valores estimados de Q^π para melhorar nosso modelo.

$$Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} E_\pi[r(s_{t'}, a_{t'})|s_t, a_t] \tag{2.3}$$

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha\delta \tag{2.4}$$

Na equação 2.4 α é a taxa de aprendizado e δ o erro de Diferença Temporal, dado por $\delta = Y - Q^\pi(s'_t, a'_t)$, onde Y é o um valor objetivo obtido através da otimização dos valores presentes no conjunto atual Q.

Por fim, a Função de Valor é dada por 2.5

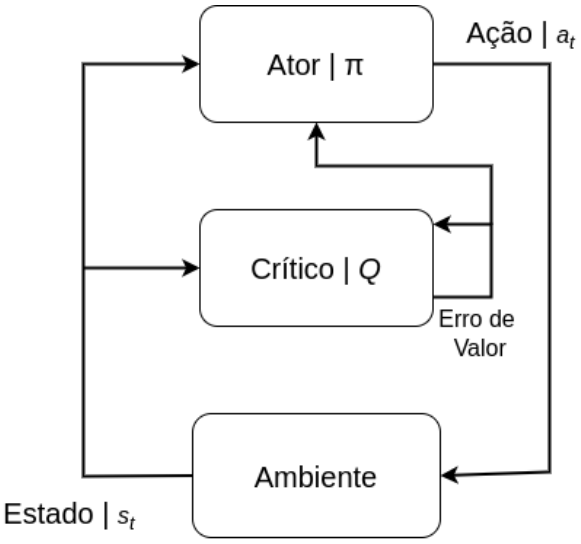
$$V^\pi(s_t) = E_{a_t \sim \pi}[Q^\pi(s'_t, a'_t)]. \tag{2.5}$$

Uma divisão clássica dos algoritmos de Aprendizado por Reforço é feita em três categorias, os baseados em função de valor, busca de política e os ator-crítico. A seguir serão discutidas as peculiaridades de cada um destes modelos.

Os métodos baseados em função de valor buscam aprender as já apresentadas Funções de Valor Q e V e, através destas, chegar a uma política de controle π. Apesar de práticos e bem difundidos, alguns riscos associados a estes modelos são: nem sempre uma função pode convergir, assim, a política derivada pode não ser a ótima, outro problema é a necessidade de ajustes de parâmetros dos modelos de forma manual para uma melhor performance do mesmo. Algoritmos baseados em modelo por sua vez buscam aprender uma representação fiel ao ambiente utilizando a partir da interação com o mesmo, possibilitando que o modelo diga uma matriz de probabilidade de transição de estados. O problema destes reside na complexidade de se gerar um modelo fiel ao ambiente, uma vez que estes geralmente são complexos e dinâmicos (ARULKUMARAN et al., 2017).

Modelos ator-crítico formam uma categoria híbrida, que faz uso tanto dos conceitos dos modelos baseados em função de valor quanto dos baseados em modelo. Para isto, a estrutura destes conta com dois módulos, uma é denomidado ator, este é responsável pela execução de uma política que se busca otimizar, o segundo módulo denominado crítico busca aproximar uma função de valor a partir das experiências do agente, este processo é ilustrado pela figura 1. Uma outra abordagem alternativa são os algoritmos livre de modelo, que em oposição aos métodos baseado em modelo não necessitam de um modelo de ambiente e são úteis quando a construção de uma representação do ambiente é desafiadora (SUTTON; BARTO, 2018).

Figura 1 – Arquitetura dos algoritmos ator-crítico



Fonte: Elaborado pelo Autor

Outra forma importante de segmentação dos algoritmos de Aprendizagem por Reforço que se encontra na literatura é a divisão entre algoritmos *on-policy* e *off-policy*. Conforme sugerido pelo nome, modelos *on-policy* necessitam seguir estritamente a política atual do

agente a fim de avaliar a mesma, assim, precisam gerar um conjunto de experiências para avaliar a si mesmo. Os algoritmos *off-policy* por sua vez podem fazer uso de experiências passadas e utilizá-las no treinamento da política atual (SUTTON; BARTO, 2018).

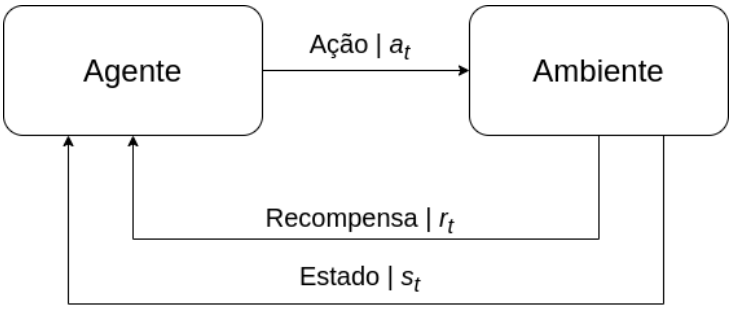
2.2.1 Processo de Decisão Markoviano

No processo de aprendizagem por reforço um agente deve fazer decisões a partir de um sinal captado do ambiente, o qual é denominado estado e é representado pela notação s_t composto por observações momentâneas do ambiente bem como observações anteriores que podem ser utilizadas para construir uma representação mais complexa do ambiente. Uma representação ideal do estado deve ser capaz de reter todas informações relevantes de forma sucinta, e quando isso é alcançado esse sinal é chamado de Propriedade de Markov (SUTTON; BARTO, 2018).

De acordo com Tai e Liu (2016) uma tarefa robótica pode ser definida formalmente como um processo de decisão de Markov, onde um agente interage com o ambiente através de observações e ações enquanto recebe sinais de recompensa por tais ações. São considerados principalmente processos de decisão de Markov episódicos, os quais possuem um estado terminal, que quando atingido finaliza o episódio atual. Tais processos podem ser definidos matematicamente como uma quíntupla ordenada $(S, A(S_t), P, R, \gamma)$.

De forma mais específica, o agente irá interagir com o ambiente a cada instante de tempo t, recebendo a recompensa e observando o estado s_t do ambiente, existente no conjunto S. A partir do ambiente observado é necessário selecionar uma ação A_t existente em $A(S_t)$ e em s_{t+1} o agente recebe a recompensa r_{t+1} conforme ilustrado na figura 2

Figura 2 – Diagrama de representação agente ambiente



Fonte: Elaborado pelo Autor

A cada instante t o agente calcula a probabilidade de uma ação ser tomada através da política π_t, denotado por $\Pi_t(a|s)$. Ao mapa formado por estas probabilidades é dado o nome de política do agente. Como é buscado maximizar a recompensa total recebida pelo agente, muitas vezes isso significa que se deve abrir mão de uma recompensa imediata mirando um ótimo global. Um dos principais desafios ao criar um modelo é formular tais sinais de recompensa

de forma que não limite o aprendizado do agente e realmente represente os objetivos que são esperados que ele alcance.

Em um MDP tem-se uma tarefa de aprendizado por reforço que satisfaz à propriedade de Markov. Essas tarefas são extremamente importantes para a área de aprendizado por reforço atual e compreende aproximadamente 90% das técnicas modernas de aprendizado por reforço. Dado qualquer estado dentro do nosso conjunto S e uma ação a possível dentro deste, a probabilidade de cada possível próximo par de ação e estado é dado por

$$p(s^{'}, r|s, a) = PrS_{t+1} = s^{'}, R_{t+1} = r|S_t = s, A_t = a, \tag{2.6}$$

A partir desta calcula-se a recompensa esperada para o próximo par de ação e estado

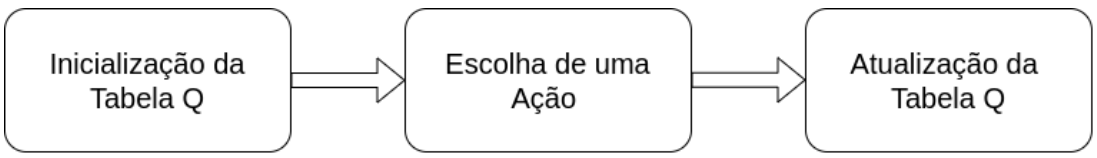
$$r(s, a) = E[R_{t+1}|S_t = s, A_t = a] = \sum_{r \in R} p(s^{'}, r|s, a). \tag{2.7}$$

2.3 *Deep Q-Learning*

O processo de *Q-learning* consiste na criação de uma tabela de valores Q a qual o agente consulta a fim de maximizar sua recompensa de longo prazo, onde a letra Q representa a qualidade de um ação para o modelo. *Q-learning* faz uso de um método *off-policy* a fim de separar a política de ação da política aprendida, assim, caso a ação selecionada não seja a ótima naquela posição o valor não será incluso na tabela Q (SUTTON; BARTO, 2018).

Um paradigma tradicional de *Q-learning* é representado pelo diagrama exposto na figura 3. Inicialmente deve-se inicializar a tabela Q, que é um mapa de conjuntos de pares de estados e ações que armazena o valor Q para cada par. A inicialização pode ser feita de diversas maneiras, a mais comum é com todos os atributos nulos, representando que o agente ainda não possui conhecimento sobre o ambiente em que está inserido (MNIH et al., 2013).

Figura 3 – Algoritmo de *Q-learning*



Fonte: Elaborado pelo Autor

Conforme o processo de aprendizagem ocorre e o agente explora os estados do ambiente, a tabela Q é atualizada e se faz necessário a preocupação com a qualidade das informações que estão sendo incorporadas a mesma. Dois termos são utilizados para explicar os desafios na fase de treinamento, onde se busca o equilíbrio entre as tarefas de *exploration* e *exploitation*. Define-se *exploration* como o processo de tentativa e erro do agente a fim de atualizar a tabela,

buscando novos estados e ações possíveis dentro to ambiente, *exploitation* por sua vez diz respeito ao processo de buscar a ação que irá trazer maior retorno entre os dados existentes na tabela, não incluindo novas ações.

A fim de equilibrar as tarefas de *exploration* e *exploitation* durante o treinamento utiliza-se a estratégia de exploração $\epsilon - greedy$. Esta consiste na estipulação de um valor ϵ que irá limitar a probabilidade de realizar uma ação fora da política atual, caso a probabilidade de retorno para determinada ação seja menor que ϵ o agente deve se manter na política conhecida, caso contrário se faz necessário explorar novos estados. A Equação de Bellman 2.8 descreve a forma de atualização do valor Q. Para o par estado-ação, ele é atualizado a partir do retorno observado, assim, após algum tempo é esperado que os valores de Q convirjam (IRODOVA; SLOAN, 2005).

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma maxQ(s',a') - Q(s,a)] \tag{2.8}$$

Apesar de muito útil, esta técnica possui como principal limitador a impossibilidade de ser utilizada em ambientes dinâmicos, onde as ações e os possíveis estados são mais numerosos. Uma alternativa de solução proposta para tal impedimento é a utilização de redes neurais profundas a fim de aproximar os valores Q, uma vez que sua importância é maior em relação aos outros valores da tabela do que seu valor exato. A partir de tal conceito surgiram as *Deep Q-Networks* (DQN) propostas por Mnih et al. (2013).

As DQN's em muito se assemelham aos algoritmos tradicionais de *Q-learning* com a diferença que, a fim de aproximar a Função Q estas redes utilizam Redes Neurais Profundas. Para isso, essas redes mapeiam os dados de entrada em pares $(ação, valorQ)$. São utilizadas duas redes distintas de mesma arquitetura porém com pesos diferentes a fim de se obter maior estabilidade durante o treinamento. Os pesos da rede principal são copiados para uma rede auxiliar chamada *target* a cada N épocas (MNIH et al., 2013).

2.4 *Deep Deterministic Policy Gradient* (DDPG)

Semelhante ao conceito apresentado de Deep-Q-Learning, as redes DDPG buscam aprender concorrentemente uma Função Q e uma política de controle. Proposta por Lillicrap et al. (2015) estas redes buscam adaptar as idéias utilizadas na criação das DQN's para espaços de ações contínuos. O trabalho é baseado no algoritmo de *deterministic policy gradient* de Silver et al. (2014) e propõe um algoritmo livre de modelo, *off-policy* e ator-crítico. Segundo os autores, o modelo é capaz de aprender boas políticas em todas as tarefas propostas que fazem usos de entradas de baixa dimensão e em vários casos consegue fazer o mesmo com entradas de alta dimensão, sempre mantendo a mesma arquitetura e parâmetros de treinamento. São utilizadas duas redes, uma rede crítica denotada por θ^Q e uma rede ator denotada por θ^π que busca ações ótimas que são utilizadas pela rede crítica. O treinamento muito se assemelha ao

utilizado nas DQN's, com a diferença de que a rede ator é atualizada utilizando uma política de gradiente definida pela equação 2.9, que mostra que pode-se chegar à política de gradiente multiplicando a derivada parcial do valor Q obtida da rede crítica em relação à ação a da rede ator com a derivada parcial da ação a em relação aos paramêtros da rede θ^{π}

$$\bigtriangledown_{\theta^{\pi}} \pi \approx E[\bigtriangledown_{\theta^{\pi}} Q(x, a|\theta^{Q})|_{x=x_t, a=\pi(x_t|\theta^{\pi})}] = E[\bigtriangledown_{a} Q(x, a|\theta^{Q})|_{x=x_t, a=\pi(x_t)} \bigtriangledown_{\theta^{\pi}} \pi(x|\theta^{\pi})|_{x=x_t}] \tag{2.9}$$

Este modelo também conta com uma memória de repetição que armazena as experiências do agente a fim de diminuir a variância durante o treinamento da rede crítica, assim, as diversas trajetórias simuladas e correlacionadas temporalmente são mantidas e aleatoriamente selecionadas durante o aprendizado para quebrar a correlação temporal dos dados de entrada. Apesar de uma lentidão causada pela adição deste passo, o agente tende a ganhar em desempenho com o emprego dessa memória.

2.5 Futebol de Robôs

O time de futebol de robôs do Departamento de Computação da UNESP Bauru, nomeado Carrossel Caipira pela sua forma de jogar semelhante ao time da seleção nacional holandesa, conhecida como carrossel holandês surgiu em 1997, sendo uma das equipes mais antigas dentro do cenário nacional de futebol de robôs.

Em sua história, a equipe sagrou-se campeã duas vezes, nos anos de 2003 e 2005 pelo VI e VII SBAI (Simpósio de Automação Inteligente), além de ter conquistado o vice-campeonato de 1998 e, com uma equipe criada em parceria com a Escola Politécnica da Universidade de São Paulo nomeada Time Guaraná, sagrou-se vice-campeão mundial.

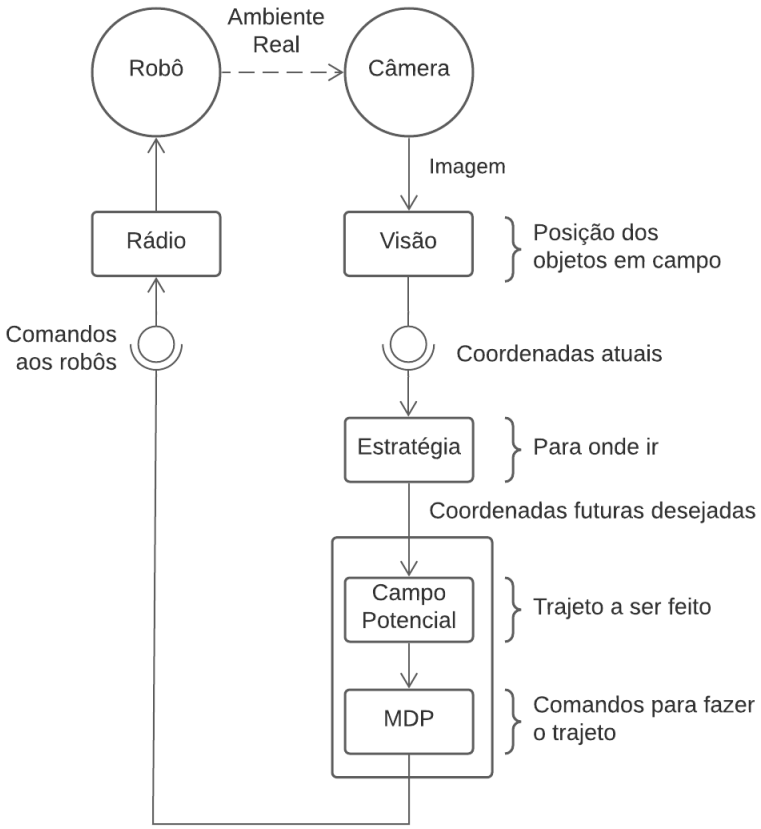
Atualmente atua na categoria IEEE Very Small Size Soccer (IEEE-VSSS), a qual conta com duas modalidades, uma com 5 robôs em cada time e outra com 3 robôs, sendo a segunda a modalidade principal até o ano de 2022. Tal modalidade acontece em um campo de 150cm x 130cm feito em uma chapa plana e rígida de madeira pintada em cor preta fosca.

Cada time conta com 3 robôs com dimensões máximas de 7,5 cm X 7,5 cm X 7,5 cm, uma câmera posicionada 2 metros acima do campo, um computador e um módulo de transmissão de comandos. Os robôs devem atuar de forma totalmente autônoma, sem intervenção humana, com o objetivo de conduzir a bola até o gol adversário, ao mesmo tempo em que defende sua própria meta.

O software do time da UNESP conta com 4 módulos conforme indicado pela figura 4, sendo eles: visão, estratégia, controle e comunicação. A câmera posicionada acima do campo funciona como um sistema global de visão, capturando imagens do campo em uma frequência de 30 quadros por segundo. Essas imagens são enviadas ao módulo de visão que possui como

objetivo a identificação das poses dos robôs e da bola. Com as poses[1] identificadas, estas são enviadas para o módulo de estratégia que as utiliza - bem como informações de poses anteriores - para definir as poses destino dos agentes em campo. O módulo de controle tem por objetivo, dada as poses atuais identificadas pela visão e as poses destino definidas pela estratégia definir os comandos que devem ser enviados ao robôs pelo módulo de comunicação.

Figura 4 – Diagrama dos módulos do time Carrossel Caipira em ambiente real

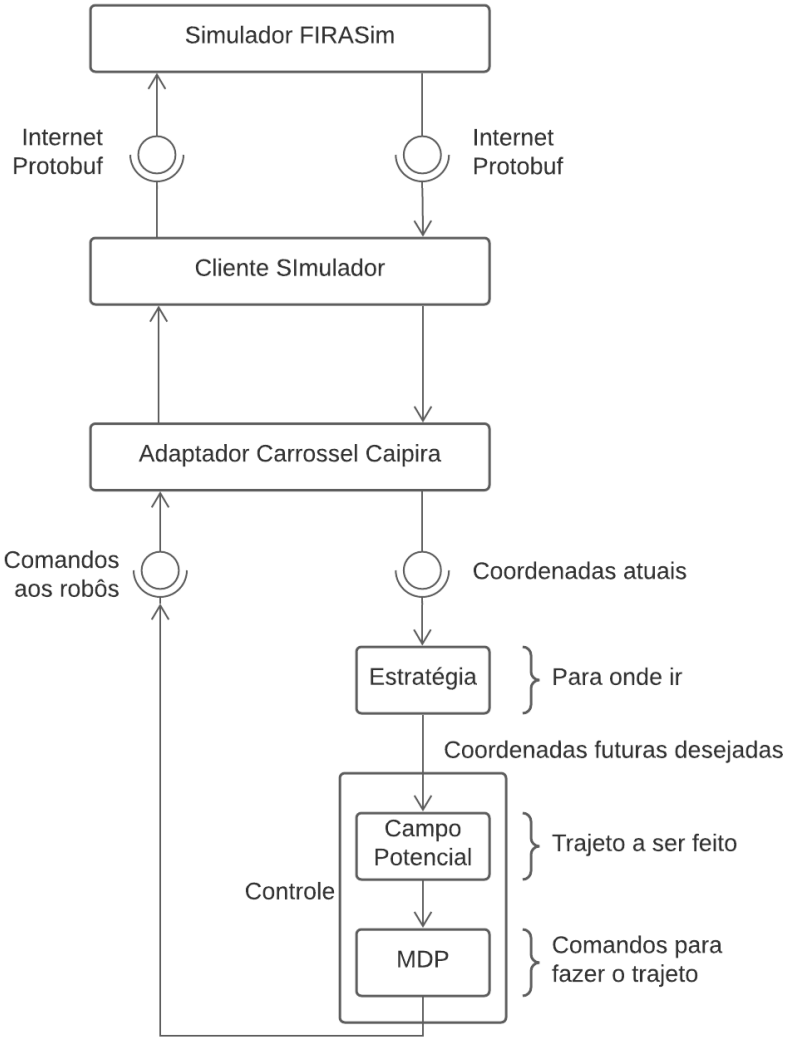


Fonte: Rossetti et al. (2022)

Com as dificuldades impostas pela pandemia de COVID-19, nos anos de 2020 e 2021 o principal campeonato da categoria, a Latin American Robotics Competition (LARC), teve seu formato alterado, passando de ocorrer em ambiente físico e passasse para um ambiente simulado, sendo realizada em formato virtual, respeitando as restrições de locomoção e distanciamento. Para que a realização dos jogos fosse possível adotou-se a utilização de um simulador chamado FIRASim. Com este, o software do time funciona como um cliente, se comunicando através da biblioteca Google Protobuf, recebendo os dados de posição dos agentes em campos - o que faz com que não seja necessário o módulo de visão - e enviando comandos para os mesmos conforme indicado na figura 5.

[1] O termo "pose"é usado para indicar a situação do robô no campo, ele inclui as coordenadas (x, y), a velocidade e a orientação de um robô.

Figura 5 – Diagrama dos módulos do time Carrossel Caipira em ambiente simulado

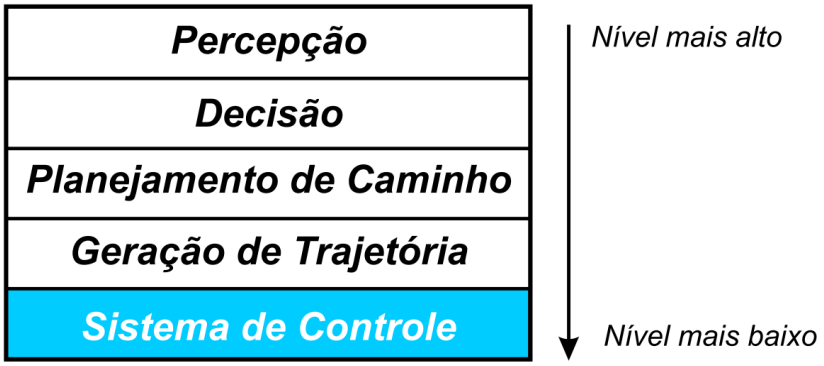


Fonte: Rossetti et al. (2022)

2.5.1 Robôs de Acionamento Diferencial

Um robô móvel são agentes que possuem hardware e software que os permitem se locomover de forma correta. Para este fim é necessário a execução de diversas tarefas a fim de trazer mais confiabilidade ao agente. A figura 6 exibe uma divisão hierárquica comumente utilizada. A percepção conta com diversos sensores que coletam informações sobre o ambiente e o agente e buscam identificar informações relevantes à tarefa utilizando algoritmos específicos. A Decisão por sua vez toma as decisões necessárias para a realização da tarefa a partir do estado obtido da percepção. O Planejamento de Caminho busca a melhor forma de levar o agente de sua pose atual à pose desejada. A Geração de Trajetória determina as restrições para a realização do caminho sugerido pelo módulo anterior. Por fim, os Sistemas de Controle garantem a execução das trajetórias, formulando os sinais de controle para os atuadores (VIEIRA, 2005).

Figura 6 – Níveis hierárquicos da arquitetura de navegação para robôs móveis



Fonte: Vieira (2005)

Robôs de Acionamento Diferencial são aqueles que utilizam rodas posicionadas em lados opostos do robô e seu controle é feito através da diferença de rotação destas. Tais agentes são ditos não-holonômicos pois não são integráveis uma vez que não podem se mover em todas as direções. O controle de tais robôs é complexo e podem ser abordados de diversas formas e com estratégias diferentes.

2.5.2 Módulo de Estratégia

Responsável por definir os objetivos dos agentes em campo, conforme anteriormente explicado, este módulo faz uso das poses atuais dos robôs e da bola bem como poses prévias, decidindo assim qual a posição ideal para que o time como um todo possa defender sua meta e pontuar marcando gols nos adversários.

As estratégias dos três agentes em campo são descritas por Autômatos Finitos Determinísticos (AFD) onde cada estado desses autômatos descreve um objetivo para os agentes. Os AFDs são classes que herdam de uma classe abstrata que representa uma máquina de estados. A transição entre os estados do autômato é feita através da observação do estado atual feita pela estratégia.

2.5.3 Módulo de Controle

A função deste módulo é, dadas as poses atual e destino, definir a trajetória a ser percorrida bem como os controles para que esta ocorra. Nas últimas competições, para traçar a trajetória é utilizado neste módulo campos potenciais, propostos por Khatib (1980) a técnica conta com a criação de forças imaginárias de atração e repulsão atuando sob os agentes, onde os campos de repulsão geralmente são criados por obstáculos no caminho entre o agente e seu objetivo, e o campo de atração é criado pela coordenada objetivo.

Um dos problemas desta abordagem é a interferência entre os campos gerados, o que pode levar a criação de ótimos locais, impossibilitando que o agente chegue ao ótimo global. A

fim de corrigir tal comportamento Connolly e Grupen (1993) propõem a utilização de funções harmônicas no cálculo do campo potencial.

Os robôs reais utilizados, bem como os simulados via FIRASim usam o sistema de tração diferencial, explicados anteriormente. Quando calculado o campo potencial e consequentemente o plano de navegação, é possível calcular a velocidade e direção a ser enviado para cada um dos motores a fim de que o robô chegue ao destino, isso é feito a partir da diferença entre o ângulo obtido do campo potencial na célula onde o agente está e o ângulo percebido do agente, com esta diferença são selecionados os comandos a partir de uma tabela.

2.5.3.1 Controlador PID

O controlador PID gera um sinal de controle que une as ações proporcional (K_p), integral (T_i) e derivativa (T_d) buscando extrair as características específicas de cada uma destas. O controlador conta com um *set point*, que é o valor de uma variável controlada que se deseja alcançar, esta pode ser uma posição, velocidade ou outros, e com o valor medido desta variável. O objetivo é levar o erro $e(t)$ definido por $e(t) = r(t) - c(t)$ a zero, onde $r(t)$ é o *set point* e $c(t)$ o valor medido. A ação proporcional do controlador K_p define o ganho do controlador em função do erro calculado, assim, a saída do controlador é dada pela equação

$$u(t) = K_p e(t) \tag{2.10}$$

A ação integral do controlador tem por função manter o sinal de saída constante quando for alcançado um sinal de erro nulo, garantindo a estabilidade do sistema. Isso é alcançado ao fazer com que o sinal de saída seja proporcional à integral do sinal de erro ao longo do tempo conforme definido pela equação 2.11.

$$u(t) = \frac{1}{T_i} \int_0^t e(t)dt \tag{2.11}$$

Por fim, a ação derivativa busca antecipar o comportamento futuro do sinal de erro baseando-se em sua taxa de variação, assim, o atraso obtido pela introdução das outras malhas de controle pode ser parcialmente compensado. A equação 2.12 define a saída do controlador em função de T_d.

$$u(t) = T_d e(t)'dt \tag{2.12}$$

O uso do controlador PID substitui a tabela atualmente utilizada além de possibilitar uma melhor adaptação à variação de cenários durante a partida, definindo comandos de acionamento de motores mais precisos.

3 Metodologia

O trabalho foi dividido em 4 momentos principais, iniciando com um levantamento da literatura existente sobre a área de atuação, seguido pela configuração do simulador e ambiente de treinamento, implementação do modelo de rede selecionado e por fim a análise dos resultados obtidos. Este capítulo busca explicar os detalhes de implementação dos modelos introduzidos anteriormente

3.1 Revisão Bibliográfica

Com a recente explosão nas pesquisas em aprendizado de máquina, diversas técnicas orientadas a dados passaram a ser utilizadas para navegação e controle de agentes autônomos, a fim de elencar aquelas que mais se destacam e se adequam ao objetivo se utilizou os trabalhos de Xiao et al. (2020) e Tai e Liu (2016) os quais elencam e comparam diversas técnicas.

Para Xiao et al. (2020) as técnicas podem ser classificadas em três categorias de acordo com o escopo do aprendizado. O primeiro escopo pode ser focado em aprender todo o esquema de navegação, em uma abordagem *end-to-end* onde não são utilizadas representações intermediárias e a partir apenas dos sinais de entrada o agente deduz uma política de movimentação. Este escopo de aprendizado é possibilitado pelo avanço das redes neurais e possui como principal vantagem a não necessidade de utilização de conhecimentos prévios e formulados à mão. Como desvantagem pode-se dizer que tais sistemas funcionam como caixas pretas, onde não se sabe ao certo como o controle está sendo formulado e quais características foram utilizadas para chegar à determinado comando.

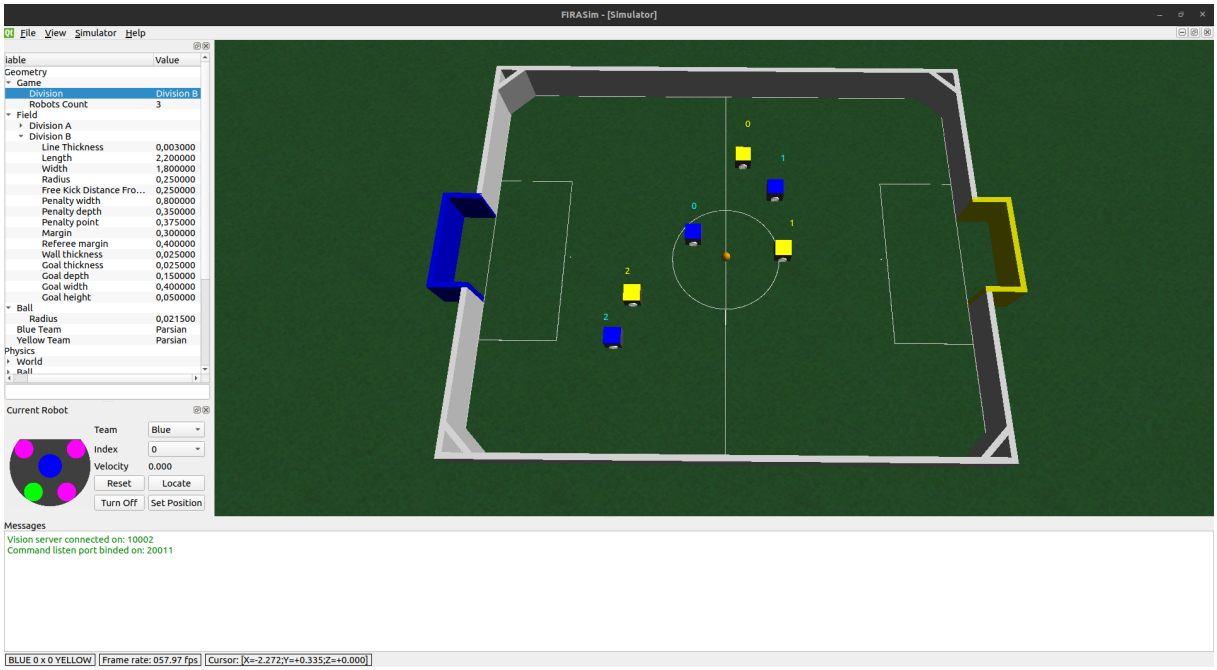
Outro escopo de aprendizado, o de subsistemas de navegação. Esta abordagem possibilita a utilização de técnicas de aprendizado de máquina juntamente com arquiteturas clássicas de controle e utilizar diferentes abordagens de acordo com o subsistema, estes são comumente divididos entre planejamento global e planejamento local sendo que as pesquisas são majoritariamente realizadas para o planejamento local. O terceiro escopo é o aprendizado de componentes individuais de controle como representações de mundo e parâmetros do *planner* de movimentação local, esta abordagem representa a menor parte da literatura disponível e muito se assemelha à abordagem clássica e hierárquica de navegação.

3.2 Simulador

O ambiente de simulação selecionado para ser utilizado foi o FIRASim (FIRASIM, 2022), simulador de código aberto e adotado como simulador oficial da categoria *IEEE Very*

Small Size Soccer. O ambiente é baseado no projeto do GRSim *grsim* e desenvolvido pela equipe ParsianLabs juntamente com colaboradores das equipes que participam da categoria. A simulação física é feita utilizando a biblioteca Open Dynamics Engine (OPEN. . . , 2022).

Figura 7 – Interface gráfica do Simulador FiraSIM



Fonte: Elaborado pelo Autor

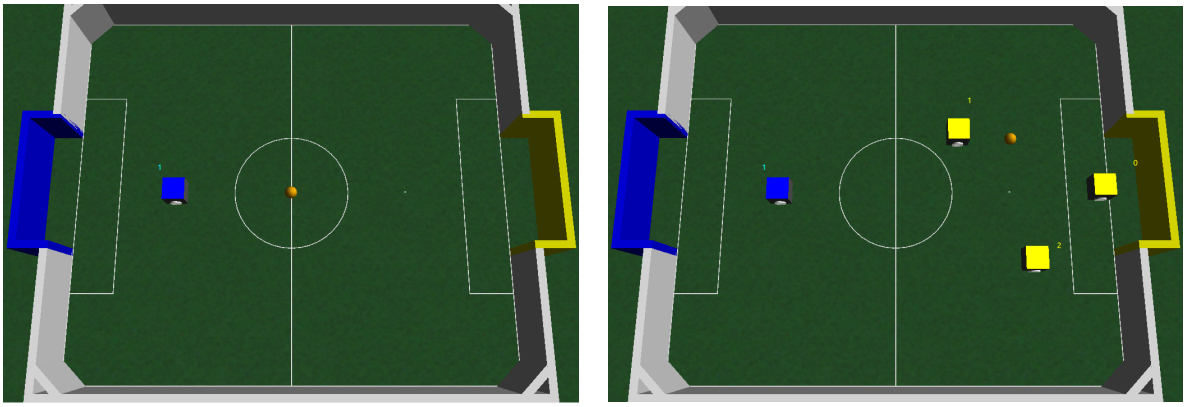
O simulador é personalizável e permite a configuração de diversos parâmetros, desde parâmetros da simulação física, alterações gráficas e também no módulo de comunicação. A comunicação com o ambiente simulado é feita através de uma arquitetura cliente-servidor, ao implementar um *socket* UDP juntamente com o Google Protobuf como protocolo de comunicação.

Foram definidos dois cenários de mundo em que o agente realizará o aprendizado, o primeiro é um mundo básico que conta somente com o próprio agente e um objetivo (bola), o segundo cenário por sua vez conta com 3 obstáculos, representados como os robôs adversários. A cada época de treinamento o agente em treinamento será posicionado no centro do campo, enquanto o objetivo e os obstáculos são posicionados randomicamente dentro do campo. A figura 8 exibe um exemplo destes dois cenários expostos.

3.3 Arquitetura do Modelo

A arquitetura selecionada para o desenvolvimento deste trabalho foi a proposta por Xie et al. (2018) e consiste na utilização de uma arquitetura de rede ator-crítico juntamente com um controlador PID a fim de acelerar e estabilizar o processo de aprendizagem por reforço. A

Figura 8 – Cenários de treinamento



(a) Mundo básico (b) Mundo complexo

Fonte: Elaborado pelo Autor

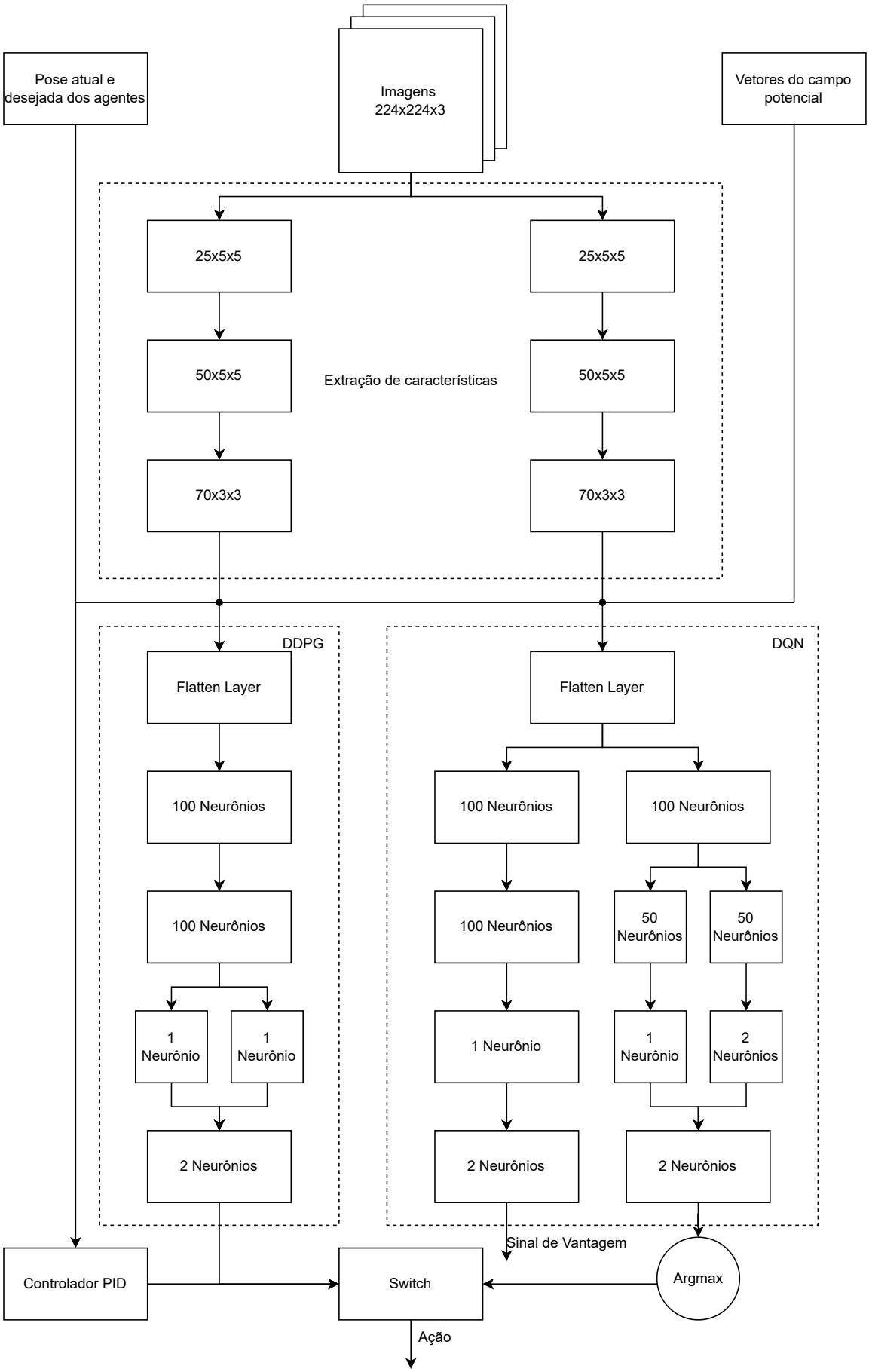
este framework foi dado o nome de *Assisted Deep Reinforcement Learning* (ADRL). O conceito por trás da proposta é intuitivo e espera que um controlador básico se comporte melhor em tarefas básicas de navegação, enquanto tarefas complexas podem ser melhor atendidas por uma política de controle.

Para julgar se deve ser utilizado os comandos de velocidade linear e angular fornecidos pela rede ator-crítico ou os comandos do controlador PID é utilizado uma rede DQN, que funciona como uma chave que opta entre os controladores buscando maximizar a recompensa obtida. Esta arquitetura diminui as chances de a política acabar em um mínimo local, uma vez que pode prevenir algumas ações aleatórias tomadas pelas redes DDPG e DQN. É esperado que em um primeiro momento o controlador PID seja selecionado com mais frequência, e assim que a rede DQN e a DDPG comecem a convergir a DDPG passa a ser selecionada com maior frequência, esperando que tanto a rede crítica e o controlador externo sejam descartados.

A figura 9 ilustra de maneira mais didática a arquitetura utilizada de forma geral. Esta pode ser dividida em três blocos principais: extração de características, rede de política e controlador assistente e por fim a rede crítica. As imagens de entrada possuem dimensão de 224 por 224 pixels e 3 canais de cores (RGB).

A primeira parte da rede é o bloco de extração de características é feita através de três camadas de convolução com kernel 2D aplicando ReLU como função de ativação. Esta parte foi adaptada da implementação original, que utiliza *laser scans* ao invés de imagens como entrada, e por esta razão utiliza redes convolucionais com kernel 1D. A partir das imagens de entrada é extraído um vetor de características que será enviado para os próximos blocos. A extração deste vetor é importante para que o modelo possa abstrair o conhecimento obtido, não ficando preso ao ambiente em que foi treinado. Na figura 9, cada um dos quadrados indica uma camada convolucional, no interior dos quadrados temos a dimensão dos filtros utilizados na camada.

Figura 9 – Arquitetura da rede utilizada



Fonte: Elaborado pelo Autor

O bloco da rede de políticas corresponde ao ator na arquitetura ator-crítico e possui como objetivo estimar os comandos de velocidade angular e linear para o agente. Este é composto por uma camada *flatten*, que possui por objetivo achatar o vetor de entrada, deixando-o apenas com uma dimensão, seguido por duas camadas totalmente conectadas de 100 neurônios com função de ativação ReLU, dois neurônios, um para estimar a velocidade angular que utiliza a função de ativação tangente hiperbólica e outro para estimar a velocidade linear que utiliza a função de ativação sigmóide. Por fim, existe uma camada com dois neurônios que recebe os valores de velocidade estimados e retorna o comando de ativação de motores.

A rede crítica DQN é dividida em duas ramificações, uma atua como a parte crítico, validando as ações geradas pela política de controle e gerando gradientes de política e o outro ramo busca acelerar o processo de treinamento e diminuir o erro de superestimação dos valores de controle ao oscilar os comandos enviados para o agente entre o comando da política de controle ou do controlador PID. O ramo crítico é similar a implementação original das redes DDPG, com a diferença que é gerado um sinal de vantagem A^{π} a partir do valor Q, isso é obtido ao inserir dois ramos na rede crítica onde um dos ramos irá aprender concorrentemente o valor V dos estados. Assim, o sinal de vantagem é obtido pela equação $A^{\pi}(x,a) = Q^{\pi}(x,a) - V^{\pi}(x,a)$. Estimar um sinal de vantagem ao invés do valor Q reduz a variação no sinal, o que se torna útil em ambientes dinâmicos e de rápida variação. O ramo da rede DQN por sua vez estima um valor Q que será utilizado para oscilar o uso dos comandos de controle. Assim como no bloco de rede de políticas existe uma camada *flatten* seguida por blocos totalmente conectados, onde o número dentro das caixas indica a quantidade de neurônios presente em cada camada.

O algoritmo 1 mostra de forma geral o processo de treinamento.

3.4 Configuração de Treinamento

Para o treinamento da rede o autor optou pela não utilização do módulo de estratégia disponível nos códigos da equipe Carrossel Caipira, ao invés disso foram utilizadas coordenadas objetivo geradas de forma aleatória juntamente com os Campos Potenciais Harmônicos para traçar um trajeto para o agente. Caso o robô atinja o objetivo global com a pose desejada o mesmo receberá uma recompensa grande, caso ele saia da trajetória ou colida receberá uma recompensa negativa, nos outros casos a recompensa depende de um fator de desconto aplicado a diferença de distância até o objetivo entre dois momentos t, além de uma penalidade a cada momento. A equação 3.1 exibe a relação entre recompensas

$$r_t = \begin{cases} R_{crash}, & \text{caso colida ou saia do trajeto} \\ R_{reach}, & \text{caso atinja o objetivo} \\ \gamma_p((d_{t-1} - d_t) \triangle t - C), & \text{em outros casos} \end{cases} \tag{3.1}$$

Algorithm 1 Treinamento AsDDPG

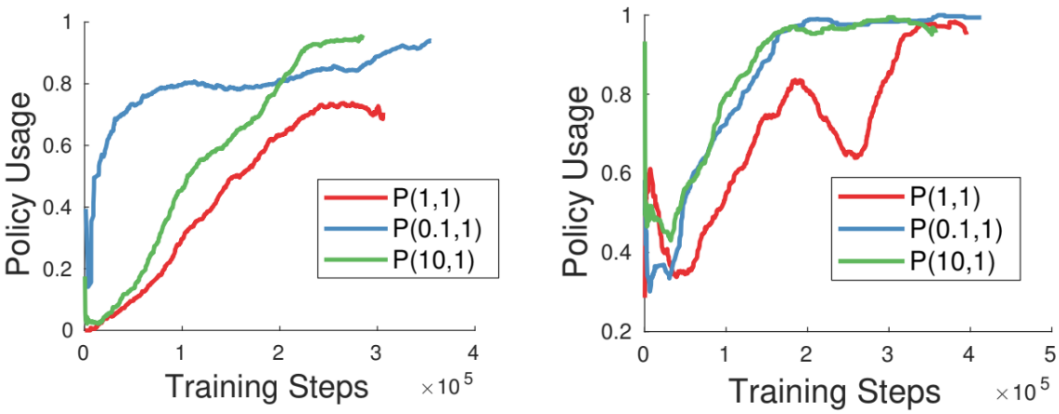
Inicializa $A(x, a|\theta^A)$, $Q(x, \sigma|\theta^Q)$ e $\pi(x|\theta^\pi)$
Inicializa as redes θ^A, θ^Q e θ^π
Inicializa buffer de replay R e ϵ
for passo = 1; passo $<$ T **do**
    Reseta o ambiente
    Obtem a observação inicial
    for passo = 1, T **do**
        Infere $Q_{policy}, Q_P = Q(x_t, \sigma|\theta^Q)$
        $\sigma_t = argmax([Q_{policy}, Q_P])$
        if $\sigma_t == 1$ **then**
            $a_t = \pi(x_t|\theta^\pi + \epsilon)$
        else
            a_t proveniente do controlador externo
        end if
        Executa a_t e recebe r_t, x_{t+1}
        Armazena a transição em R
        Otimiza a rede crítica DQN
        Atualiza a política de gradientes
        Atualiza a rede alvo
    end for
end for

No ambiente de treinamento utilizou-se dois sinais de entrada, um enviado pelo próprio simulador que diz as poses dos robôs observado pelo mesmo, e o outro é uma imagem do ambiente simulado, a qual é enviada para a rede neural convolucional buscando extrair características relevantes.

3.5 Medidas de Avaliação

Para avaliar a convergência da rede são utilizadas duas principais medidas, o primeiro diz respeito à recompensa média obtida pelo agente no decorrer do processo de treinamento e o segundo a variação entre o uso da rede de política aprendida em detrimento do controlador básico. Baseando-se nos resultados prévios obtidos por Xie et al. (2018) os parâmetros escolhidos para o controlador foi P(1, 1), conforme indicado na figura 10 entre outros valores elencados P(0.1,1) é um controlador lento e faz com que o agente demore a chegar no destino, fazendo com que seja facilmente seja substituído pela política e P(10,1) faz com que o agente se locomova muito rapidamente, geralmente passando do ponto destino e tendo que voltar para atingir o mesmo. Xie et al. (2018) utiliza dois cenários de treino, um vazio chamado *empty world* e outro chamado *simple world* que conta com 4 obstáculos retangulares posicionados ao redor da origem do ambiente, de onde o agente inicia sua movimentação, criando assim dificuldade para a movimentação do mesmo.

Figura 10 – Uso da política da rede no decorrer do treinamento



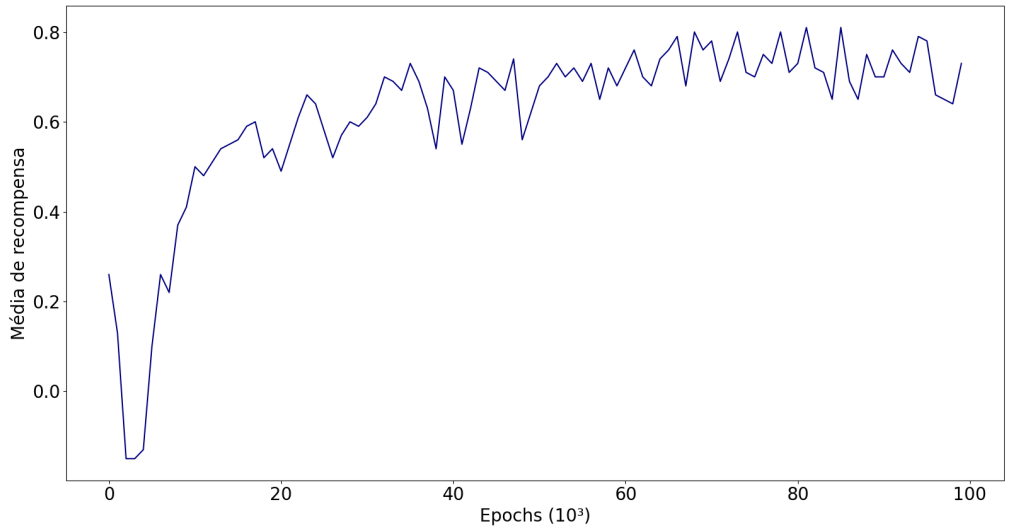
(a) Empty world (b) Simple world

Fonte: Xie et al. (2018)

4 Resultados

A figura 11 mostra a recompensa média obtida pelo agente no decorrer do treinamento. Pode-se observar que a convergência ocorre de forma rápida e estável para padrões de aprendizagem por reforço onde em média são necessários entre 1 e 10 milhões de passos de treinamento para que as redes possam convergir, no presente trabalho 100 mil passos foram suficientes para convergência. A recompensa média obtida pelo agente foi de 0,7214 dentro de um intervalo de [-2,2]. Este comportamento pode ser atribuído ao modelo de recompensa, que a possibilita que o agente receba recompensas durantes vários passos de iteração, reduzidas quando não são estados terminais e acrescidas quando o objetivo é alcançado ou o agente fracassa em sua navegação.

Figura 11 – Curva de aprendizado do modelo



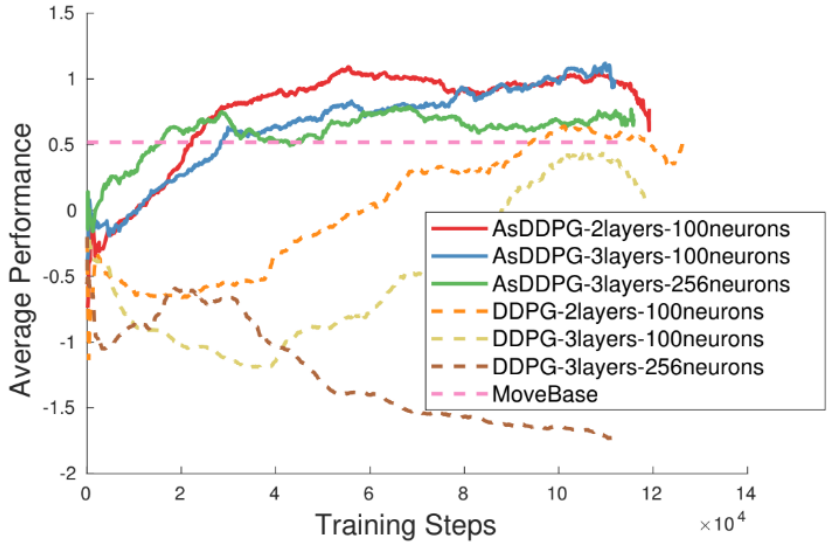
Fonte: Autor

Outro ponto interessante de ser observado é que o agente recebe pouca quantidade de recompensa negativa em seus momentos iniciais de treinamento, ao contrário da implementação sem aprendizado assistido exposta por Xie et al. (2018), isso acontece pois, com a assistência do controlador PID o agente não necessita realizar tantas ações aleatórias no começo do processo de aprendizagem, uma vez que este controlador passa a ser empregado no lugar da rede de política.

Não foram realizados testes da mesma configuração de rede DDPG sem a assistência de um controlador PID, porém, tomando como base os resultados obtidos por Xie et al. (2018)

é esperado que os resultados da implementação do presente trabalho mantenha o mesmo comportamento do autor original, como exibido na figura 12.

Figura 12 – Curvas de aprendizado do modelo na implementação original



Fonte: Xie et al. (2018)

A figura 13 por sua vez mostra o uso da política do agente no decorrer do processo de treinamento. Nota-se que, conforme esperado, nos momentos iniciais o controlador externo é usado mais vezes, porém, ao decorrer do treinamento a política passa a ser mais utilizada pois, segundo a rede DQN esta passa a ter melhor performance quando comparada ao comando do controlador PID. A política de controle foi utilizado em 58,77% das vezes, o que indica que após o treinamento, os comandos da rede de política são mais relevantes aos agentes em maior parte das vezes.

A tabela 1 traz a comparação de algumas medidas entre a política aprendida pela rede neural artificial após finalizado o treinamento e o controlador PID. Estas medidas foram obtidas através de testes padronizados e espelhados nos dois modelos, também foi mantida a mesma distância Manhattan do agente ao objetivo em todos os casos de teste. No total foram realizados 100 cenários de teste. O número de recompensa negativa diz respeito à quantas vezes o agente foi penalizado com o maior valor de recompensa negativa ao sair do trajeto traçado pelo campo potencial ou colidiu com algum obstáculo.

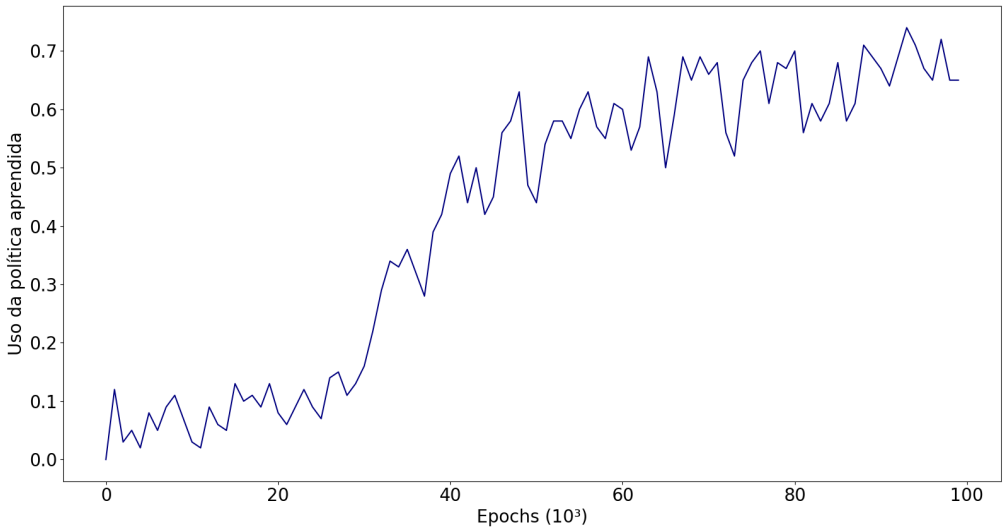
Tabela 1 – Medidas de comparação entre comandos

Sinal de Controle	Tempo médio (frames)	Recompensa negativa
Rede de política	347	17
Controlador PID	368	29

Fonte: Elaborado pelo Autor

Qualitativamente percebe-se uma melhora na locomoção do agente ao utilizar os

Figura 13 – Uso da política aprendida



Fonte: Elaborado pelo Autor

comandos da rede de política, uma vez que não foram notados comportamentos estranhos no agente após a conclusão do treinamento da rede, o que indica que o mesmo não ficou preso em um ótimo local. Em tarefas de movimentação que geralmente são problemáticas como alcançar objetivos próximos aos limites dos campos o agente obteve maior eficiência utilizando a rede de política do que o controlador externo, colidindo com as laterais do campo em menos ocasiões e alcançando seu objetivo.

Foram realizadas tentativas de criar uma interface que unisse o modelo ao restante do código da equipe de Futebol de Robôs, porém, empecilhos surgiram e impossibilitaram a mesma. A primeira e mais óbvia é que o código do time é escrito em linguagem C++, desta forma a interface de comunicação não foi eficiente a ponto de possibilitar a aplicação durante uma partida. Outro ponto limitante foi a capacidade computacional disponível, simular 6 agentes em uma partida dinâmica utilizando tal rede foi muito custoso ao *hardware* disponível. A tentativa de implementação em ambiente real também foi falha uma vez que os robôs reais estiveram disponíveis tardiamente, além de ter uma limitação grande em relação à bateria disponível, durando pouco mais de 4 horas e impossibilitando o treinamento com estes.

5 Conclusão

A robótica é uma área de pesquisa interessante e em franco crescimento que tende a cada vez mais estar presente no cotidiano das pessoas em produtos inovadores, seja em linhas de produção fabris aumentando a produtividade das empresas e diminuindo a escassez de recursos ou em atividades domésticas onde atua como facilitador para diversas tarefas. Como anteriormente exposto, tal área vem se beneficiando de técnicas de inteligência artificial, principalmente em sua vertente de aprendizagem por reforço em todos os níveis hierárquicos de planejamento e controle de agentes. Neste capítulo são trazidos os principais resultados obtidos e suas contribuições para o campo da robótica e também possíveis trabalhos futuros.

5.1 Resultados Obtidos

A proposta inicial de implementação de um controlador para robôs de acionamento diferencial utilizando técnicas de aprendizado de máquina foi cumprida ao implementar um modelo capaz de controlar agentes, permitindo que estes se locomovam de forma a atingir suas poses objetivo dentro do ambiente simulado. Foi alcançado um modelo adaptado ao ambiente de futebol de robôs e suas especificidades, atuando no simulador oficial da categoria e escolhendo ações e comandos dentro do escopo do ambiente.

Uma das vantagens da utilização de técnicas de inteligência artificial no controle de agentes robóticos é a possibilidade de adaptação à diferentes cenários, sem que seja necessário um grande esforço para realizar alterações para que o modelo de controle funcione no mesmo, uma vez que se espera que a própria rede neural artificial realizará o processo de aprendizagem sozinha. Neste sentido, quando se compara as técnicas de controle utilizadas anteriormente pela equipe baseada em resultados empíricos e várias horas de teste com diversas variáveis, buscando uma que melhor se adeque ao sistema, a rede de políticas oferece uma alternativa muito menos trabalhosa.

A arquitetura de rede neural artificial utilizada, baseada no trabalho de Xie et al. (2018) apresentou resultados que podem ser interessantes para a equipe de futebol de robôs Carrossel Caipira e a modalidade *IEEE Very Small Size Soccer* em termos gerais, ao unir um controlador externo e acelerar o processo de treinamento das redes.

Outro ponto interessante foi o modelo de recompensa utilizado, este juntamente com o uso de um controlador externo possibilitou uma convergência rápida do modelo, uma vez que em toda iteração do processo de aprendizagem o agente recebia uma recompensa, mesmo que pequena. Esta recompensa contínua possibilita que o agente saiba qual o efeito da ação selecionada para o sistema naquele momento, o problema ao utilizar apenas recompensas

esparsas é que, em muitos cenários o agente passa por diversos passos de treinamento sem saber se suas ações foram positivas ou negativas e assim podem acabar presos em ótimos locais ao receber uma recompensa positiva ocasional.

Dentre os objetivos iniciais do trabalho não foi possível a transferência do conhecimento para o ambiente real. A isso se atribui um atraso no desenvolvimento dos agentes, que só estiveram disponíveis no começo de outubro de 2022, vale ressaltar que o desenvolvimento e confecção dos robôs não estava dentro do escopo do presente trabalho. Com os robôs disponíveis, além do tempo escasso para implementação foram encontradas dificuldades em configurar um ambiente de comunicação à rádio com os robôs com bom desempenho, uma vez que durante o processo de treinamento em ambiente simulado o mesmo ocorreu de forma síncrona, o que não é possível no ambiente real e a interação entre rede neural artifical e agentes é assíncrona.

5.2 Trabalhos Futuros

Os resultados aqui obtidos abrem possibilidades para pesquisas na área de controle de robôs autônomos e inteligência artifical aplicada a robótica de forma geral. Algumas das possibilidades e necessidades elencadas durante o desenvolvimento do projeto são:

- Unificação do módulo de controle desenvolvido ao restante do código da equipe, unindo assim os módulos de estatégia, controle, visão e comunicação.
- Transferência de aprendizado e aplicação do atual modelo em ambiente real, configurando um ambiente tolerante à falhas e que possa ser utilizado por mais tempo de treinamento.
- Realizar tarefas de treinamento com objetivos específicos que se adaptem melhor ao comportamento esperado de cada um dos agentes, criando então modelos de controle mais ajustados às tarefas dos goleiros, volantes ou atacantes.
- Explorar outros modelos, arquiteturas e abordagens de controle, entre os outros trabalhos revisados os que mais chamam atenção são as *Intention-Net* propostas por Gao et al. (2017) e a arquitetura de treinamento e rede de Pokle et al. (2019).

Referências

ARULKUMARAN, K.; DEISENROTH, M. P.; BRUNDAGE, M.; BHARATH, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, Institute of Electrical and Electronics Engineers (IEEE), v. 34, n. 6, p. 26–38, nov 2017. Disponível em: <https://doi.org/10.1109\%2Fmsp.2017.2743240>.

CONNOLLY, C.; GRUPEN, R. The application of harmonic functions to robotics. *Journal of Robotic Systems*, v. 10, p. 931 – 946, 10 1993.

FIRASIM. 2022. <https://github.com/robocin/FIRASim>. Acessado por último em 23 de novembro de 2022.

GAO, W.; HSU, D.; LEE, W. S.; SHEN, S.; SUBRAMANIAN, K. Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation. *CoRR*, abs/1710.05627, 2017. Disponível em: <http://arxiv.org/abs/1710.05627>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

IRODOVA, M.; SLOAN, R. Reinforcement learning and function approximation. In: . [S.l.: s.n.], 2005. p. 455–460.

KHATIB, O. *Commande dynamique dans l'espace opérationnel des robots manipulateurs en présence d'obstacles*. Tese (Doutorado) — École nationale supérieure de l'aéronautique et de l'espace, 1980.

LILLICRAP, T. P.; HUNT, J. J.; PRITZEL, A.; HEESS, N.; EREZ, T.; TASSA, Y.; SILVER, D.; WIERSTRA, D. *Continuous control with deep reinforcement learning*. arXiv, 2015. Disponível em: <https://arxiv.org/abs/1509.02971>.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; GRAVES, A.; ANTONOGLOU, I.; WIERSTRA, D.; RIEDMILLER, M. A. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. Disponível em: <http://arxiv.org/abs/1312.5602>.

OPEN Dynamics Engine. 2022. <https://www.ode.org/>. Acessado por último em 05 de dezembro de 2022.

POKLE, A.; MARTÍN-MARTÍN, R.; GOEBEL, P.; CHOW, V.; EWALD, H. M.; YANG, J.; WANG, Z.; SADEGHIAN, A.; SADIGH, D.; SAVARESE, S.; VÁZQUEZ, M. Deep local trajectory replanning and control for robot navigation. *CoRR*, abs/1905.05279, 2019. Disponível em: <http://arxiv.org/abs/1905.05279>.

ROSSETTI, R. C. B.; CORREIA, J. V. M.; CASTRO, P. H. A. de; MONTES, G. R.; LEE, A. K. M.; PEGORARO, R.; BATISTA1, M. *Descrição do Time Carrossel Caipira de Futebol de Robôs da Unesp de Bauru*. 2022.

SHABBIR, J.; ANWER, T. A survey of deep learning techniques for mobile robot applications. *CoRR*, abs/1803.07608, 2018. Disponível em: <http://arxiv.org/abs/1803.07608>.

SILVER, D.; LEVER, G.; HEESS, N.; DEGRIS, T.; WIERSTRA, D.; RIEDMILLER, M. Deterministic policy gradient algorithms. *31st International Conference on Machine Learning, ICML 2014*, v. 1, 06 2014.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. Disponível em: <http://incompleteideas.net/book/the-book-2nd.html>.

TAI, L.; LIU, M. Deep-learning in mobile robotics - from perception to control systems: A survey on why and why not. *CoRR*, abs/1612.07139, 2016. Disponível em: <http://arxiv.org/abs/1612.07139>.

VIEIRA, F. C. *Controle Dinâmico de Robôs Móveis com Acionamento Diferencial*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Natal - RN, 2005.

XIAO, X.; LIU, B.; WARNELL, G.; STONE, P. Motion control for mobile robot navigation using machine learning: a survey. *CoRR*, abs/2011.13112, 2020. Disponível em: <https://arxiv.org/abs/2011.13112>.

XIE, L.; WANG, S.; ROSA, S.; MARKHAM, A.; TRIGONI, N. Learning with training wheels: Speeding up training with a simple controller for deep reinforcement learning. *CoRR*, abs/1812.05027, 2018. Disponível em: <http://arxiv.org/abs/1812.05027>.