



UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Dissertação de Mestrado

**Modelo para Planejamento de Caminho Livre de Colisão
para Manejo de Guindaste**

Rafaela Iovanovich Machado

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande - FURG, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Computação

Orientadora: Prof^a. Dr^a. Silvia Silva da Costa Botelho
Co-orientador: Dr. Matheus Machado dos Santos

Rio Grande, 2023

Ficha Catalográfica

M149m Machado, Rafaela Iovanovich.

Modelo para planejamento de caminho livre de colisão para
manejo de guindaste / Rafaela Iovanovich Machado. – 2023.
51 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande –
FURG, Programa de Pós-Graduação em Computação, Rio
Grande/RS, 2023.

Orientadora: Dra. Silvia Silva da Costa Botelho.

Coorientador: Dr. Matheus Machado dos Santos.

1. Aprendizado por reforço 2. Guindaste 3. Planejamento de
caminho I. Botelho, Silvia Silva da Costa II. Santos, Matheus
Machado dos III. Título.

CDU 004

Catálogo na Fonte: Bibliotecário José Paulo dos Santos CRB 10/2344

DISSERTAÇÃO DE MESTRADO

Modelo para Planejamento de Caminho Livre de Colisão para Manejo de Guindaste

Rafaela Iovanovichi Machado

Banca examinadora:

Documento assinado digitalmente
 **CARLOS RODRIGUES ROCHA**
Data: 04/09/2023 11:20:44-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Carlos Rodrigues Rocha (IFRS)

Firmado digitalmente por
com.apple.kerberos.kdc
DN: CN=com.apple.kerberos.kdc, O=System Identity
Razón: Estoy aprobando este documento con mi firma legalmente vinculante
Ubicación:
Fecha: 2023.09.01 14:07:10-0300
Exit: PDF Reader Version: 1.2.0.1

com.apple.kerberos.kdc

Prof. Dr. Vinicius Menezes de Oliveira (FURG)

Documento assinado digitalmente
 **SILVIA SILVA DA COSTA BOTELHO**
Data: 01/09/2023 10:45:12-0300
Verifique em <https://validar.iti.gov.br>

Profa. Dra. Silvia Silva da Costa Botelho (FURG)
Orientadora

*Education is our passport to the future, for tomorrow belongs to the people who
prepare for it today. — MALCOLM X*

RESUMO

MACHADO, Rafaela Iovanovichi. **Modelo para Planejamento de Caminho Livre de Colisão para Manejo de Guindaste**. 2023. 51 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

O Brasil possui um número elevado de plataformas para extração de petróleo e gás natural, um ramo que envolve um alto grau de complexidade em diversos aspectos. Estes empreendimentos movimentam um grande volume de carga com um peso elevado, o que torna inviável utilizar apenas recursos humanos para alocação dos carregamentos. Por isso, as unidades marítimas de produção de petróleo utilizam os guindastes para carga e descarga de componentes, produtos químicos e alimentos.

Visando auxiliar as operações de guindastes, este estudo tem como objetivo desenvolver um modelo de planejamento de caminho livre de colisão para a manipulação destes equipamentos. Por este ser um problema complexo e de difícil aquisição de dados, torna-se inviável a utilização de metodologias empregadas em aprendizado supervisionado e não supervisionado, pois estas necessitam de dados anteriores para o treinamento e testes com o modelo.

Por isso, para o desenvolvimento do trabalho optou-se pela utilização de técnicas de aprendizado por reforço (*reinforcement learning* RL), pois nos algoritmos de RL o agente aprende a tomar ações para maximizar uma recompensa acumulada ao longo do tempo, tendo como objetivo aprender a melhor estratégia de ação em uma determinada situação interagindo com o ambiente ao qual está inserido.

Neste contexto, foi desenvolvido um ambiente de simulação para viabilizar o treinamento e testes com algoritmos de RL. Foram testados os algoritmos *Deep Deterministic Policy Gradient* e *Soft Actor-Critic*. Seus desempenhos são apresentados e comparados, bem como a sugestão de melhorias e trabalhos futuros.

Palavras-chave: Aprendizado por reforço, guindaste, planejamento de caminho.

ABSTRACT

MACHADO, Rafaela Iovanovichi. **Model for Collision Free Path Planning for Crane Handling**. 2023. 51 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

Brazil has a large number of offshore platforms for oil and natural gas extraction, a sector that involves a high degree of complexity in several aspects. These undertakings move a large volume of cargo with a high weight, which makes it unfeasible to use only human resources to allocate shipments. For this reason, offshore oil production units use cranes to load and unload components, chemicals and food.

In order to assist crane operations, this study aims to develop a collision-free path planning model for crane manipulation. Given that, this is a complex problem with challenging data acquisition, so the use of methodologies employed in supervised and unsupervised learning becomes impractical, as they require prior data for model training and testing.

Hence, for this work's development, it was decided to use reinforcement learning (RL) techniques. In RL algorithms the agent learns to take actions to maximize an accumulated reward over time, aiming to learn the best strategy of action in a given situation interacting with the environment to which it is inserted.

In this context, a simulation environment was developed to enable training and testing with RL algorithms. The Deep Deterministic Policy Gradient and Soft Actor-Critic algorithms were tested, their performances are presented and compared, as well as suggestions for improvements and future work.

Keywords: Reinforcement learning, crane, path planning.

LISTA DE FIGURAS

1	Exemplo de Guindaste [1]	13
2	Classificação x Regressão (adaptado de [2])	17
3	Agrupamento de dados (Adaptado de [3])	18
4	Interação agente-ambiente no aprendizado por reforço (adaptado de [4])	19
5	Ator-Crítico	24
6	Visão geral do sistema	28
7	Modelo 3D do guindaste	30
8	Ferramenta para ajustes de PID	31
9	Ambiente simulado	31
10	Visão geral do guindaste	32
11	Arquitetura do sistema (adaptado de [5])	33
12	Máxima recompensa	35
13	Exemplo de caminho de guindaste navegando em direção ao alvo, ilustrado na Fig. quatro obstáculos (objetos laranja), alvo (objeto ver- melho) e o caminho linha (azul).	36
14	Posições dos obstáculos no cenário	37
15	Gráfico de recompensa DDPG sem novas regras de recompensa . . .	39
16	Gráfico de recompensa DDPG com novas regras de recompensa . . .	39
17	Gráfico de recompensa SAC sem novas regras de recompensa	40
18	Gráfico de recompensa SAC com novas regras de recompensa	40

LISTA DE TABELAS

1	Especificações de <i>hardware</i> e <i>software</i>	27
2	Posição dos obstáculos	37
3	Estrutura DDPG	38
4	Estrutura SAC	38
5	Comparação de Resultados: DDPG x SAC	41

LISTA DE ABREVIATURAS E SIGLAS

CAD	Computer Aided Design
DPG	Deterministic Policy Gradient
DP	Dynamic programming
DDPG	Deep Deterministic Policy Gradient
SAC	Soft Actor Critic
PPO	Proximal Policy Optimization
DQN	Deep Q-Network
D3QN	Dueling Double Deep Q-Network
IHM	Interface Humano-Máquina
LiDAR	Light Detection and Ranging
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Process
PID	Proporcional, integral e derivativo
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
ROS	Robot Operating System
SAC	Soft Actor Critic
SL	Supervised Learning

SUMÁRIO

1	Introdução	12
1.1	Motivação	15
1.2	Objetivos	16
2	Revisão Teórica	17
2.1	Aprendizado Supervisionado	17
2.2	Aprendizado Não Supervisionado	18
2.3	Aprendizado por Reforço	18
2.3.1	Processo de Decisão de Markov	20
2.3.2	Política	21
2.3.3	Recompensa	21
2.3.4	Função de Valor	21
2.3.5	Modelo de Ambiente	22
2.3.6	Q-Learning	22
2.3.7	Aprendizado por Reforço Baseado em Modelo e Não Baseado em Modelo	23
2.3.8	Deep Deterministic Policy Gradient	23
2.3.9	Soft Actor-Critic	24
2.4	Trabalhos Relacionados	25
3	Metodologia	27
3.1	Configuração de Equipamento Utilizado	27
3.2	Visão Geral da Proposta	27
3.3	Modelagem do Ambiente	28
3.3.1	Robot Operating System, Unified Robotic Description Format e Gazebo .	29
3.3.2	Modelo do Guindaste	29
3.3.3	Modelo do Ambiente	31
3.4	Um Modelo para Planejamento de Caminho Baseado em Algoritmos de Aprendizado por Reforço	32
3.4.1	Estrutura dos Algoritmos: DDPG e SAC	33
3.4.2	Base Estrutural de Comunicação Entre os <i>Scripts</i>	34
3.5	Política de Recompensa	34
4	Testes e Resultados	37
4.1	Ambiente de Testes e Parametrização dos Modelos	37
4.2	Resultados DDPG	38
4.3	Resultados SAC	40
4.4	Comparando Resultados	41

4.5	Dificuldades Encontradas	42
5	Conclusão	44
	Referências	46

1 INTRODUÇÃO

No Brasil, existem 246 blocos de petróleo e gás natural em fase de exploração, sendo destes, 138 blocos marítimos e 108 blocos terrestres. Após a conclusão dessa fase e a eventual declaração de comercialidade, iniciam-se as etapas de desenvolvimento e produção [6].

A fase de exploração tem por objetivo encontrar e avaliar jazidas de petróleo e ou gás natural. Já a fase de produção, é aquela em que as acumulações que tiveram sua viabilidade comercial comprovada dão origem a um campo produtor, sendo desenvolvidas e postas em produção para abastecer o mercado. Essa fase começa com a perfuração de poços produtores e a instalação de plataformas de produção no mar ou em terra. Por fim, a fase de desenvolvimento é a etapa em que toda a infraestrutura necessária para a produção do campo é implantada, incluindo a construção de estradas, instalações de armazenamento e tratamento de petróleo, estações de compressão de gás, entre outras [6].

As operações *offshore* envolvidas na exploração e produção requerem manipulação de uma quantidade significativa de carga, frequentemente com pesos substanciais, levando a necessidade de uso de guindastes para efetuar o carregamento e descarregamento de componentes, produtos químicos e alimentos [7] [8].

Pode-se definir os guindastes como “dispositivos de elevação e rotação montados em pedestais [...] para transferência de materiais ou pessoas para embarcações ou estruturas marítimas”, conforme ilustrado na Figura 1, e “são normalmente montados em uma estrutura de plataforma fixa (suportada no fundo) ou flutuante usada em operações de perfuração e produção”[9].

A operação de guindastes *offshore* requer a aplicação de padrões rigorosos, tanto para garantir a segurança quanto a sua eficiência, uma vez que operam em condições adversas em alto mar, onde a falta de controle sobre a carga transportada pode acarretar na perda da carga útil movimentada [10]. Além disso, esses guindastes *offshore* de alto desempenho possuem uma operação complexa, que depende significativamente das habilidades do operador para coordenar o seu movimento [11]. Tais operações, frequentemente realizadas com precisão junta a junta, requerem uma combinação de atitudes, reflexos e experiência do operador.



Figura 1: Exemplo de Guindaste [1]

No que tange a saúde e segurança do operador, é importante destacar que a atividade executada em plataformas de extração de petróleo é uma das mais perigosas do setor marítimo [12], sendo responsável por um impacto significativo na saúde dos funcionários embarcados. Pode-se destacar o efeito social, haja visto os longos períodos isolados em alto mar e separados de suas famílias [13]. Também observa-se a interferências na saúde física. Por exemplo, o alto nível de ruído acústico gera consequências negativas na saúde [13], bem como percebe-se maior propensão ao consumo de álcool e a mudança nos padrões alimentares. Consequentemente a saúde mental também pode ser comprometida, pela presença de sintomas de ansiedade e depressão [14]. Dessa forma, o setor procura constantemente por inovações que conduzam a melhores padrões de desempenho e segurança [15].

Um dos principais riscos na operação de guindastes é a limitada visibilidade do operador, e isso se torna um grande motivo de preocupação [16] [17]. Câmeras vem sendo acopladas ao equipamento, expandindo o campo de visão disponível para o guindasteiro. No entanto, se torna uma tarefa difícil acompanhar a movimentação do ambiente através de um único monitor sem nenhuma informação, como por exemplo, a posição do trabalhador em relação ao guindaste ou à carga. A baixa visibilidade ou ponto cego faz com que o operador tenha dificuldade em identificar quaisquer pessoas ou objetos na zona de trabalho [18].

Os contínuos avanços tecnológicos estão possibilitando cada vez mais o desenvolvimento de simuladores e técnicas para auxiliar e capacitar o guindasteiro. Essas inovações também desempenham um papel crucial na aprimorada percepção do operador em relação ao ambiente de carga e descarga, assim como na clara visualização do caminho que o

guindaste percorre durante suas operações de movimentação [19].

Esta dissertação está relacionada às questões relativas a geração de caminho livre de obstáculos no contexto de guindastes autônomos. O planejamento de caminho é um desafio computacional de grande importância no campo da robótica. O principal objetivo do planejamento de caminho é encontrar a melhor rota entre a posição atual do robô e o alvo. Para a maioria dos robôs, o planejamento de caminho ideal é o caminho mais curto entre dois locais. No entanto, existem desafios que necessitam planejar um caminho que atenda a alguns requisitos para lidar com condições incertas, como obstáculos fixos e móveis no ambiente atual.

A geração de caminho é obtida informando a posição final desejada das juntas atuadas e a partir dos conceitos de controle é realizado o cálculo do caminho desejado. Dada a complexidade dos cenários de carga e descarga envolvendo guindastes *offshore*, o planejamento de caminho livre de obstáculos é uma tarefa não trivial.

Diferentes fabricantes de guindastes implicam em diferentes modelos dinâmicos, cuja inércia e características dos atuadores tornam-se singulares para cada equipamento. As condições meteorológicas (neblina, vento) também modificam totalmente tanto a percepção dos operadores, como as inércias de suas juntas, componentes e cargas. Outro fator relevante é a diversidade da natureza das cargas (centro de gravidade).

Estes elementos exigem operadores experientes para a movimentação destes equipamentos. No caso da busca por algoritmos autônomos para planejamento de caminho, técnicas tradicionais devem ser capazes de modelar fisicamente o cabedal de fenômenos envolvidos.

Para realizar a carga e descarga, os guindastes apanham o carregamento do local, como de um navio por exemplo, e iniciam uma trajetória de locomoção, que inclui o içamento, deslocamento horizontal e descida no ponto de descarga [15]. Ao se buscar dotar o equipamento de planejamento de trajetória autônomo, pode-se vincular duas formas para tal: (1) fornecer o caminho desejado no espaço operacional; ou (2) fornecer o caminho desejado no espaço das juntas. No primeiro caso, é necessário calcular o algoritmo da cinemática inversa para transformar a trajetória fornecida da ferramenta nas trajetórias das juntas.

Neste trabalho propõe-se o uso do aprendizado por reforço (*Reinforcement Learning* - RL), tendo como base os conceitos utilizados em carros autônomos para gerar caminhos livre de colisões [20]. O RL é um tipo de aprendizagem de máquina focada em obter uma recompensa a partir de uma determinada ação. A técnica utiliza sistemas de realimentação para estimular o comportamento desejado e recompensar as ações corretas ou punir as ações erradas [4]. Os algoritmos de aprendizado por reforço empregados no trabalho são *Deep Deterministic Policy Gradient* (DDPG) [21] e *Soft Actor Critic* (SAC) [22], e serão apresentados no decorrer do trabalho.

Algumas vantagens sobre outros métodos é que o aprendizado por reforço não neces-

sita de dados anteriores para realizar o treinamento. O agente aprende com o ambiente ao qual está inserido, além de visar alcançar o comportamento ideal de um modelo dentro de um contexto específico, para maximizar seu desempenho. Essa característica é útil quando a única forma de coletar informações sobre o ambiente é interagindo com ele [23].

Os algoritmos de RL podem ser utilizados para gerar trajetórias comportamentais assumindo um módulo de percepção separado para extrair as coordenadas do objeto e os limites de velocidade [24]. Após a obtenção desses dados, é viável aplicar uma política previamente treinada por meio de RL para planejar a trajetória desejada. Isso significa que o algoritmo aprende a tomar decisões sobre como mover o objeto ao longo do tempo baseando-se nas informações obtidas pelo módulo de percepção, com o objetivo de atingir uma trajetória planejada de maneira eficaz e otimizada.

1.1 Motivação

O ambiente em alto mar apresenta uma série de fatores que impactam negativamente na saúde física e mental dos operadores de guindastes. Por isto, traçar um caminho livre de colisão poderá melhorar o processo de movimentação de carga e descarga auxiliando o guindasteiro na tomada de decisão, podendo assim, mitigar colisões e acidentes.

Outro tópico relevante, que diz respeito tanto aos operadores do guindaste quanto aos outros funcionários atuantes na plataforma, é a segurança. Sendo assim, fornecer uma ferramenta visando contribuir para a redução de risco de acidentes pode gerar um ambiente menos perigoso.

Estudar um tema bastante pesquisado e aplicá-lo em uma área específica é de suma importância, haja visto que as conclusões deste trabalho poderão contribuir para futuros estudos, além do fato de identificar novos problemas e objetivos para novas pesquisas.

E por fim, pesquisar sobre o aprendizado por reforço, uma das áreas do *machine learning*, é outro aspecto motivador, pois a maioria dos algoritmos de planejamento de caminhos tradicionais tem baixa adaptabilidade e são adequados apenas para situações com modelos físicos bem conhecidos e comportados.

Existem algoritmos como o A* que é um algoritmo de busca informada que depende fortemente do cálculo do custo de movimento de uma determinada posição até o destino [25]. Além de difícil estimativa no cenário *offshore*, alguns percursos que levam ao destino não são previamente conhecidos, por isso, o uso do algoritmo A* não encontrará o caminho mais curto, mesmo sendo um caminho viável [26].

Com o aprendizado por reforço (RL), os agentes podem otimizar estratégias por meio da interação com ambientes desconhecidos por métodos de tentativa e erro e acumular recompensas (o valor das recompensas pode ser positivo ou negativo) para aprender o padrão de comportamento ideal. Além disso, RL tem boa generalidade e é adequado para ambientes desconhecidos [27].

1.2 Objetivos

Com o exposto até o presente momento, pode-se concluir que o caminho a ser realizado pelos guindastes de alta performance localizados em plataformas de extração de petróleo apresentam um alto grau de complexidade e diversos fatores podem impactar negativamente na operação das cargas e na saúde humana dos funcionários embarcados. Visando minimizar estes problemas, este trabalho apresenta como objetivo geral: *desenvolver um modelo baseado em aprendizado por reforço para planejamento de caminho livre de colisão para operação de guindastes*. O modelo será avaliado em um ambiente simulado.

De forma mais precisa, baseando-se em métodos de aprendizado por reforço, propõe-se os seguintes objetivos específicos:

- Aprofundar os conhecimentos em aprendizado por reforço e planejamento de caminho para guindastes;
- Propor um algoritmo de aprendizado por reforço que se adéque às necessidades do trabalho;
- Criar modelo em ambiente simulado e avaliar o modelo criado.

A estrutura do trabalho será dividida da seguinte forma: neste primeiro capítulo foi apresentada uma introdução sobre a problemática proposta, os objetivos gerais e específicos do trabalho. No segundo capítulo serão discutidos alguns fundamentos relacionados ao entendimento da temática. No terceiro capítulo será apresentada a metodologia da pesquisa. No quarto capítulo serão descritos os testes e os resultados. E no quinto capítulo será apresentada a conclusão do trabalho.

2 REVISÃO TEÓRICA

Os algoritmos de aprendizado de máquina (Machine Learning) são classificados em três categorias: o aprendizado supervisionado (Supervised Learning), aprendizado não supervisionado (Unsupervised Learning) e o aprendizado por reforço (Reinforcement Learning) [20].

2.1 Aprendizado Supervisionado

A aprendizagem supervisionada se baseia em inferência indutiva onde o modelo é normalmente treinado usando dados rotulados para resolver problemas de classificação ou regressão. Algoritmos de aprendizagem supervisionada induzem modelos a partir desses dados de treinamento e estes podem ser usados para classificar outros dados não rotulados [28].

Esse tipo de aprendizado é análogo ao aprendizado humano, onde experiências anteriores são utilizadas para adquirir novos conhecimentos a fim de melhorar nossa capacidade de realizar tarefas do mundo real [29].

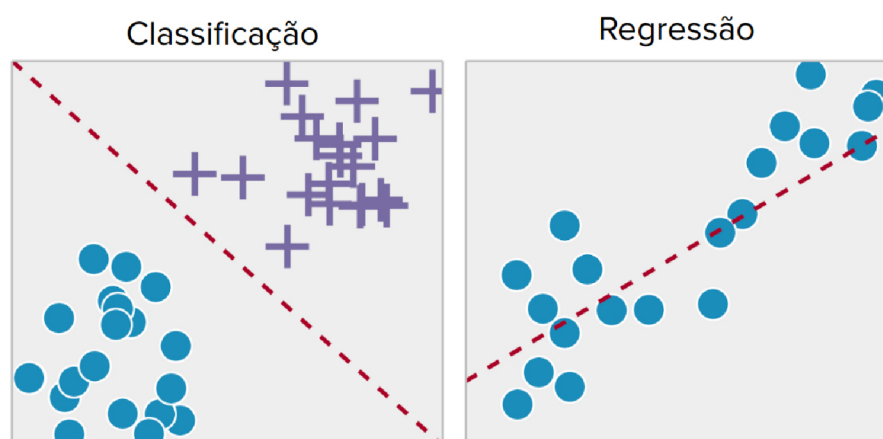


Figura 2: Classificação x Regressão (adaptado de [2])

Como exemplificado na Figura 2, quando se trata de problemas de classificação, pretende-se prever a categoria de um certo dado, como por exemplo, classificar um objeto como sendo da categoria cruz ou da categoria círculo. De forma similar, os problemas de

regressão utilizam dados de entrada para prever uma resposta, e a sua principal diferença é que um estima um rótulo para os dados e o outro um valor numérico para os dados [28].

2.2 Aprendizado Não Supervisionado

Aprendizagem não supervisionada geralmente pode ser vista como um método de modelagem da densidade de probabilidade das entradas. Os algoritmos podem, portanto, classificar, rotular e/ou agrupar os pontos de dados contidos nos conjuntos sem ter qualquer orientação externa na execução dessa tarefa [30]. Em outras palavras, o aprendizado não supervisionado permite que o sistema identifique padrões dentro de conjuntos de dados por conta própria.

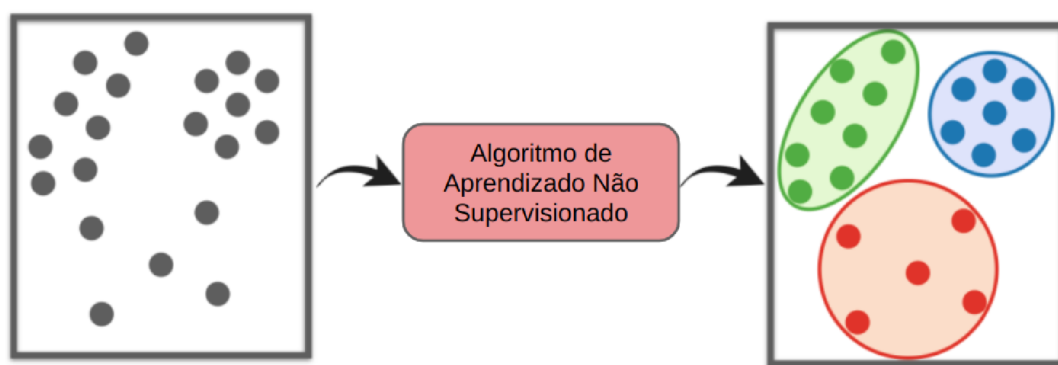


Figura 3: Agrupamento de dados (Adaptado de [3])

Como mostra a Figura 3, no aprendizado não supervisionado, um sistema de inteligência agrupa as informações não classificadas de acordo com semelhanças e diferenças, embora não haja categorias fornecidas [31].

2.3 Aprendizado por Reforço

Aprendizado por reforço (*Reinforcement learning* - RL) é o estudo de um mapeamento de situações para ações de modo a maximizar uma recompensa escalar ou sinal de reforço. O algoritmo não precisa ser informado diretamente sobre quais ações tomar, como na maioria das formas de aprendizado de máquina, mas, em vez disso, ele deve descobrir quais ações geram mais recompensas experimentando-as. Nos casos mais interessantes e desafiadores, uma ação pode afetar não apenas a recompensa imediata, mas também a próxima situação e, conseqüentemente, todas as recompensas subsequentes. A busca por tentativa e erro e a recompensa retardada são as duas características distintivas mais importantes da aprendizagem por reforço [4].

No RL, um agente autônomo aprende a melhorar seu desempenho em uma tarefa atribuída interagindo com seu ambiente. Agentes de RL não são informados explicitamente como agir por um especialista. Em vez disso, o desempenho de um agente é

avaliado por uma função de recompensa. Para cada estado experimentado, o agente escolhe uma ação e recebe uma recompensa ocasional de seu ambiente com base na utilidade de sua decisão. O objetivo do agente é maximizar as recompensas recebidas ao longo de sua vida [20].

O RL considera o problema de como um agente deve atuar em seu ambiente ao longo do tempo de forma a maximizar algum sinal de recompensa escalar. Pode-se formalizar a interação de um agente com seu ambiente com o Processo de Decisão de Markov (MDP) [32].

Um dos principais desafios do aprendizado por reforço é gerenciar o *trade-off* entre *exploration*, que implica em seguir um novo caminho, e *exploitation*, que segue uma trajetória já conhecida. Para maximizar as recompensas que recebe, um agente deve explorar seu conhecimento selecionando ações que resultam em altas recompensas. Por outro lado, para descobrir tais ações benéficas, é preciso correr o risco de tentar novas ações que possam levar a recompensas maiores do que as ações atuais mais valorizadas para cada estado do sistema. Em outras palavras, o agente de aprendizagem tem que explorar o que já sabe para obter recompensas, mas também tem que explorar o desconhecido para fazer melhores seleções de ação no futuro [20].

O agente é a entidade que treina-se para tomar decisões corretas, o ambiente é o entorno com o qual o agente interage, o estado (S) define a situação atual do agente, a ação (A) é a escolha que o agente faz no intervalo de tempo atual (t), e a política é o processo de pensamento por trás da escolha de uma ação [33].

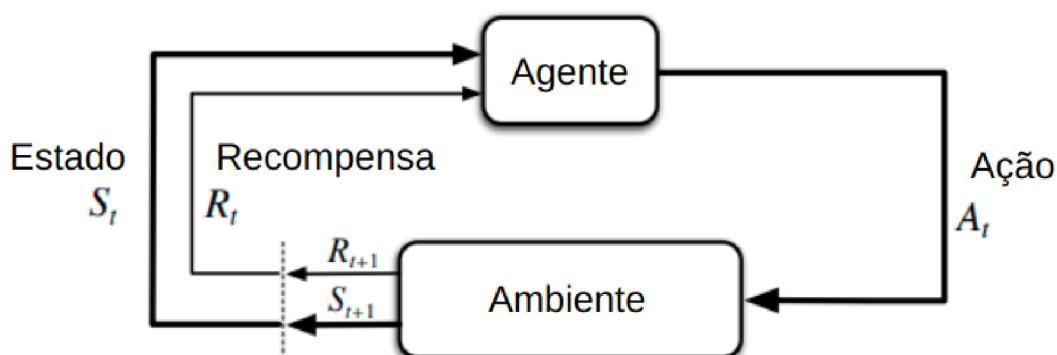


Figura 4: Interação agente-ambiente no aprendizado por reforço (adaptado de[4])

Como apresentado na Figura 4, depois de realizar ação (A_t) o agente recebe uma recompensa (R_{t+1}) imediata que pode ser positiva ou negativa. Cabe ao agente e, portanto, ao algoritmo de aprendizado, usar e interpretar essa recompensa. A aprendizagem por reforço pode ser vista como uma abordagem de tentativa e erro. Como o agente não é informado explicitamente sobre qual ação tomar, ele só pode avaliar as ações com as recompensas que recebeu. Para que essa avaliação seja eficiente, o agente deve interagir continuamente com o meio ambiente e adaptar sua estratégia quanto às recompensas que

obtem [34].

O sinal de reforço no aprendizado por reforço origina-se da recompensa imediata do *feedback* ambiental. Esse valor de recompensa indica a qualidade da ação executada em vez de comunicar à máquina sobre qual é a ação correta. O processo da máquina interagindo com o ambiente pode ser considerado como um Processo de decisão de Markov (MDP) [35].

2.3.1 Processo de Decisão de Markov

Uma cadeia de Markov é um modelo matemático que descreve a sequência de possíveis eventos em um problema e modela os estados que descrevem as possíveis situações com as quais um agente pode se deparar [36]. O MPD supõe que a probabilidade de transição para um novo estado, dado o estado e ações atuais, é independente de todos os estados e ações anteriores [37], como descrito na Equação 1.

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, S_2, \dots, S_t) \quad (1)$$

A probabilidade de passar para o estado S_{t+1} dado que se está em S_t é igual a probabilidade de passar para o estado S_{t+1} dados todos os estados em que se esteve em seu histórico [38].

Um MDP é uma tupla (S, A, T, R) onde:

- S é um conjunto de estados em que o processo pode estar;
- A é um conjunto de ações que podem ser executadas em diferentes épocas de decisão;
- $T : S \times A \times S \rightarrow [0 ; 1]$ é uma função que dá a probabilidade de o sistema passar para um estado $s' \in S$, dado que o processo estava em um estado $s \in S$ e o agente decidiu executar uma ação $a \in A$ (denotada $T(s'|s, a)$);
- $R : S \times A \rightarrow R$ é uma função que dá o custo (ou recompensa) por tomar uma decisão $a \in A$ quando o processo está em um estado $s \in S$.

No aprendizado por reforço, em um tempo t um agente no estado S_t executa uma ação de acordo com a política π . O agente faz a transição para o estado S_{t+1} e recebe uma recompensa R_t . A sequência de estados, ações e recompensas é dada como uma trajetória γ ao longo do ambiente T [37], como demonstra a Equação 2.

$$\gamma = (S_1, A_1, R_1), \dots, (S_T, A_T, R_T) \quad (2)$$

2.3.2 Política

Uma política define a maneira de como o agente de aprendizagem se comporta em um determinado momento. De forma simplificada, uma política é um mapeamento de estados percebidos do meio ambiente para ações a serem tomadas quando estiver nesses estados. Seria equivalente a um conjunto de regras ou associações de estímulo-resposta, tomando como base os princípios da psicologia.

Em alguns casos, a política pode ser uma função simples ou tabela de pesquisa, enquanto em outros pode envolver cálculos extensos, como um processo de pesquisa. A política é o núcleo de um agente de aprendizagem por reforço, no sentido de que ela sozinha é suficiente para determinar o comportamento [4].

Em algoritmos de aprendizado por reforço existem as políticas de comportamento e políticas alvo. Nas políticas de comportamento o agente determina a sua ação (comportamento) em um dado estado. Na política alvo, o agente aprende com as recompensas recebidas por suas ações para atualizar o valor Q (que corresponde ao par estado-ação). Sendo assim, se a política alvo é diferente da política de comportamento, o agente é chamado de *off-policy learner*. Já se a política alvo é a mesma de comportamento, o agente é chamado de *on-policy learner* [20]. A política alvo é tipicamente a política gulosa determinística em relação à estimativa atual da função de valor de ação. Esta política se torna uma política determinística ótima enquanto a política de comportamento permanece estocástica e mais exploratória [39].

2.3.3 Recompensa

Um sinal de recompensa define o objetivo de um problema de aprendizado por reforço. Em cada etapa de tempo o ambiente envia ao agente de aprendizagem por reforço um único valor, denominado recompensa. O único objetivo do agente é maximizar a recompensa total que recebe a longo prazo.

A recompensa, portanto, define quais são os eventos bons e ruins para o agente. Em um sistema biológico, podemos pensar nas recompensas como análogas às experiências de prazer ou dor. Eles são as características imediatas e definidoras do problema enfrentado pelo agente.

O sinal de recompensa é a base primária para alterar a política de comportamento; se uma ação selecionada pela política for seguida por uma recompensa baixa, a política pode ser alterada para selecionar alguma outra ação para aquela situação no futuro. Em geral, os sinais de recompensa podem ser funções estocásticas do estado do ambiente e das ações realizadas [4].

2.3.4 Função de Valor

Uma função de valor atribui um valor numérico a cada estado ou estado-ação em um ambiente, refletindo o quão bom é para um agente estar nesse estado ou tomar essa

ação. Ou seja, o valor de um estado é a quantidade total de recompensa que um agente pode esperar acumular no futuro a partir desse estado. Enquanto as recompensas determinam a desejabilidade intrínseca imediata dos estados ambientais, os valores indicam a desejabilidade a longo prazo dos estados depois de levar em consideração os estados que provavelmente se seguirão e as recompensas disponíveis nesses estados.

Por exemplo, um estado pode sempre render uma recompensa imediata baixa, mas ainda ter um valor alto porque é regularmente seguido por outros estados que geram recompensas altas, assim como o inverso também pode ser verdade. Para fazer uma analogia humana, as recompensas se assemelham ao prazer (se alto) e a dor (se baixo), enquanto os valores correspondem a um julgamento mais refinado e perspicaz de quão satisfeitos ou descontentes estamos pelo estado do ambiente resultante das ações tomadas [4].

2.3.5 Modelo de Ambiente

O modelo de ambiente ou mundo externo é algo que imita o comportamento do ambiente em algum sentido. Por exemplo, dada uma situação e uma ação, o modelo pode prever o próximo estado resultante e a próxima recompensa. Se houver $|S|$ estados e $|A|$ ações, então um modelo completo ocupará espaço proporcional ao tamanho $|S| \times |S| \times |A|$, porque mapeia a distribuição de probabilidade de cada par de estado-ação sobre os estados, fornecendo a probabilidade de cada possível estado como resultado para cada ação tomada nestes estados [4].

Diferentemente da aprendizagem supervisionada, na aprendizagem por reforço não há apresentação de pares de entrada/saída. Em vez disso, depois de escolher uma ação, o agente é informado sobre a recompensa imediata e o estado subsequente, mas não é informado qual ação teria sido de seu interesse a longo prazo. É necessário que o agente reúna experiência útil sobre os possíveis estados do sistema, ações, transições e recompensas ativamente para agir de forma otimizada. Além disso, outra diferença do RL em relação a aprendizagem supervisionada é que o desempenho em tempo real é importante e a avaliação do sistema costuma ser simultânea à aprendizagem [23].

2.3.6 Q-Learning

Q-learning é uma forma de aprendizagem por reforço livre de modelos ou políticas. Ele é descrito desta forma pois utiliza apenas a função de valor para selecionar as ações do agente, e pode ser visto como um método de programação dinâmica (DP) assíncrona. Ele fornece aos agentes a capacidade de aprender a agir de forma otimizada nos domínios markovianos experimentando as consequências das ações, sem exigir que eles construam mapas dos domínios [40].

O algoritmo Q-learning estima a função de valor de ação de estado, conforme a Equação 3:

$$Q_{t+1}(s, a) = (1 - \alpha_t(s, a))Q_t(s, a) + \alpha_t(s, a)(R_M(s, a) + \gamma \max_{b \in U(s')} Q_t(s', b)) \quad (3)$$

Onde s' é o estado alcançado a partir do estado s ao realizar a ação a no tempo t . Isto é, um processo assíncrono no sentido de que atualiza uma única entrada a cada etapa [41].

Inicialmente o Q-Learning não possui informações suficientes sobre os estados e as recompensas. Porém, ao frequentar o mesmo estado diversas vezes, pode experimentar ações diferentes e aprender com elas. Após um número suficiente de interações, que pode variar drasticamente conforme o problema, a função Q passa a convergir à política ótima [36].

2.3.7 Aprendizado por Reforço Baseado em Modelo e Não Baseado em Modelo

O aprendizado não baseado em modelo aprende a controlar sem aprender o modelo, enquanto no aprendizado baseado em modelo aprende o modelo e o utiliza para derivar o controlador. O agente entende o seu ambiente e, interagindo com ele, cria o modelo com base nessas interações, e, em seguida, escolhe a política ideal com base no modelo aprendido [23].

A principal vantagem de ter um modelo é que ele permite ao agente planejar pensando no futuro e estimando o que aconteceria para uma variedade de escolhas possíveis, decidindo explicitamente entre suas opções. Os agentes podem então obter os resultados do planejamento antecipado em uma política aprendida. A principal desvantagem é que um modelo real do ambiente geralmente não está disponível para o agente. Para isso, ele terá que aprender o modelo puramente por experiência, o que criará vários desafios. O maior desafio é que o viés no modelo pode ser explorado pelo agente, resultando em um agente com bom desempenho em relação ao modelo aprendido, mas com comportamento abaixo do ideal no ambiente real [42].

Algoritmos sem modelo buscam aprender as consequências de suas ações por meio de experiências adquiridas através de algoritmos como *Policy Gradient*, *Q-Learning*, etc. Em outras palavras, o algoritmo executará uma ação várias vezes e ajustará a política, a estratégia por trás de suas ações, para recompensas ideais, com base nos resultados [43].

Se o agente prever a recompensa por alguma ação antes de realizá-la planejando assim o que deve ser feito, podemos afirmar que o algoritmo é baseado em modelo. Enquanto, no aprendizado livre de modelo, o agente realmente executa uma ação para a partir do resultado aprender com ela.

2.3.8 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) é uma técnica de aprendizado por reforço *off-policy learner* usado para lidar com espaços de ação contínua com base no

algoritmo ator-crítico. O DDPG combina as abordagens de *Deterministic Policy Gradient* (DPG) e *Deep Q-Network* (DQN), onde o ator é uma rede de políticas que toma o estado como entrada e produz a ação contínua em vez de uma distribuição de probabilidade sobre as ações. O crítico é uma rede de valor Q que recebe estado e ação como entrada e gera o valor Q [21].

Como apresentado na Figura 5, o algoritmo faz uso de uma rede neural para a rede ator e outra para a rede crítica.

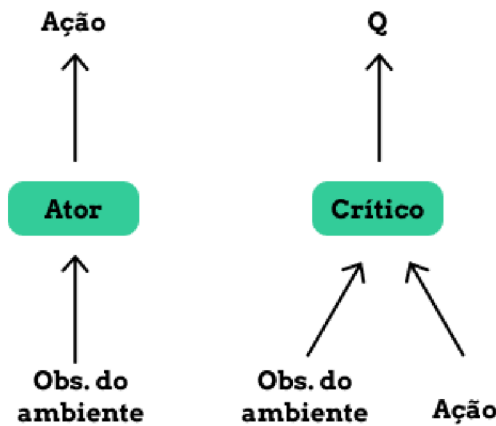


Figura 5: Ator-Crítico

Na rede ator sua entrada é o estado atual (observação do ambiente) que gera como saída uma ação para o espaço de ação contínuo. Na rede crítico, tem como entrada a observação do ambiente e a ação correspondente que gera como saída o valor Q estimado deste estado atual como a avaliação da ação, e ajusta constantemente sua função de valor e a ação gerada pelo ator [44]. A rede atual do Ator melhora continuamente a estratégia de ação de acordo com o valor Q. A rede alvo ator e crítico é usada principalmente para o processo de atualização subsequente [45]. O agente recebe diretamente dados de sensores para gerar uma política ótima que planeja seu caminho seguro até o alvo [46].

2.3.9 Soft Actor-Critic

Soft Actor-Critic (SAC) é um algoritmo de aprendizado por reforço, *off-policy learner*, utilizado para espaços de ação contínuo com base no algoritmo ator-crítico, *Q-learning* e aprendizado por reforço de entropia máxima [22] e [47].

Este algoritmo visa aprender simultaneamente uma política e duas redes de valor Q (crítico). O aprendizado por reforço de entropia máxima otimiza as políticas para maximizar o retorno esperado e a entropia esperada da política, ao maximizar a entropia, o SAC incentiva a política a explorar uma ampla gama de ações, levando a uma exploração mais eficiente. Isso tem uma conexão estreita com o *trade-off* entre *exploration* e *exploitation*: o aumento da entropia resulta em mais *exploration*, o que pode acelerar o aprendizado. Com isso, o algoritmo supera o problema da fragilidade, incentivando a rede de políticas

a explorar e não atribuir uma probabilidade muito alta a qualquer parte do intervalo de ações [22] e [47].

Em resumo, o SAC combina os benefícios da maximização da entropia para exploração, estimativa de valor Q mínimo para mitigar problemas de superestimação e o uso das redes críticas para uma melhor estimativa de valor.

2.4 Trabalhos Relacionados

Há situações do mundo real em que obter dados de mapeamento completos são inatingíveis. Um exemplo é a condução autônoma: é quase impossível mapear todas as vias e, mesmo com um mapeamento, qualquer alteração inesperada podem fazer com que o veículo se desloque para o local errado. Por meio dos algoritmos de aprendizado por reforço se torna possível aprender quais ações são mais vantajosas para atingir objetivos em ambientes incertos, permitindo a geração de múltiplos caminhos [23].

O planejamento de caminho é a tecnologia chave para robôs móveis autônomos. Tendo em vista a escassez de caminhos encontrados pelo algoritmo tradicional *best first search* (BFS) e *rapidly-exploring random trees* (RRT) que não são curtos e suaves o suficiente para a navegação do robô. O *Q-Learning*, um algoritmo clássico de aprendizado por reforço, é usado para encontrar um caminho global para robôs. Caminhos mais curtos e suaves podem ser obtidos por este algoritmo em comparação com o algoritmo BFS e RRT [48].

É uma tendência que os robôs substituam os humanos nos campos industriais com o aumento do custo da mão de obra e a preservação da saúde e bem estar humano, e por isso eles são amplamente utilizados para a execução de tarefas em ambientes industriais hostis. O DQN é um método eficiente de aprendizado por reforço, e tem sido usado para o planejamento de caminhos de robôs móveis em ambientes desconhecidos, auxiliando em um importante problema dos robôs móveis, que é planejar o seu caminho em um ambiente desconhecido. Porém, é importante salientar, que o DQN geralmente tem baixa velocidade de convergência [49].

Entre os algoritmos comuns para o planejamento de caminhos locais são: o método de campo potencial artificial, método de histograma vetorial, algoritmo genético, método de lógica difusa, método de aprendizado por reforço. Conforme definido na Seção 2.3, o algoritmo de aprendizado por reforço é um método de aprendizado que não requer nenhum conhecimento a priori do ambiente ou do próprio robô, no qual ele age enquanto percebe o estado atual do ambiente. O robô decide a próxima ação com base nas informações de recompensa. Atualmente, o algoritmo *Q-learning* de aprendizado por reforço é amplamente utilizado no planejamento de caminhos de robôs móveis. Não requer nenhum conhecimento empírico prévio ou compreensão do ambiente. Ele só precisa ajustar a ação por meio de tentativa e erro no ambiente desconhecido e aprender de forma independente

para atingir a meta de planejamento de caminhos. Embora a vantagem do *Q-learning* seja que o planejamento do caminho não requer conhecimento prévio, ele ainda tem as desvantagens de convergência lenta e baixa capacidade de generalização. Por isso, o *Q-learning* começou a se desenvolver em combinação com outros métodos [50].

Em comparação com os métodos tradicionais de planejamento de caminho, os baseados em aprendizado de reforço profundo não precisam construir todo o modelo de ambiente e podem realizar autoaprendizagem para mapeamento de ação, que possui alta flexibilidade. Através da interação contínua entre o robô móvel e o ambiente, o aprendizado por reforço profundo usa a estratégia de ação correspondente para determinar a próxima ação do robô móvel de acordo com o estado do robô e combina a função de recompensa para otimizar continuamente a estratégia de ação. Como um dos algoritmos típicos de aprendizado por reforço profundo aplicado ao planejamento de caminhos, o DDPG pode treinar o modelo em ambiente de simulação autoconstruído e ser aplicado diretamente ao ambiente real, com forte capacidade de generalização [45].

No trabalho de Xiaogang et al. [51], apresenta-se uma arquitetura *Dueling Double DQN* - (D3QN) para navegação de robôs, utilizando uma câmera para percepção do ambiente considerando apenas as imagens como entrada para navegar por um ambiente. Os resultados do experimento mostram que o robô móvel pode atingir os alvos desejados sem colidir com nenhum obstáculo. Porém, algoritmos como o DQN utilizam ações discretas, o que pode ser um problema em um ambiente com grande número de obstáculos. Em [52], propõe-se a utilização do *Soft Actor Critic* (SAC) para treinar a rede, buscando políticas de navegação que permeiem executar tarefas em ambientes não estruturados capazes de evitar mínimos locais.

A proposta apresentada por Guo et al. [53], utiliza o aprendizado por reforço profundo para propor um modelo de planejamento de caminho autônomo para navios não tripulados baseado em DRL, a essência do modelo é que o agente encontra independentemente o caminho mais eficiente através do método enumerado, que pode estar mais próximo da manipulação humana. Ao mesmo tempo, o movimento do navio é transformado em um problema de controle de movimento contínuo, de modo que esteja em conformidade com as características reais de movimento do navio. Em Wang et al. [54] e Zhao et al. [27] empregam-se diferentes algoritmos de RL para planejamento de caminho de robôs móveis.

Cho e Han [55] apresentam um estudo demonstrando diferentes estratégias de recompensa utilizando os algoritmos SAC e *Proximal Policy Optimization* (PPO) para tarefas de içamento com guindastes, neste estudo, a abordagem proposta e os resultados demonstram que os modelos neurais baseados em RL podem aprender como controlar um guindastes do tipo torre durante as tarefas de içamento, observando as informações que podem ser reconhecidas pelos operadores humanos e obtendo as recompensas de acordo com as estratégias de treinamento.

3 METODOLOGIA

Neste capítulo será descrita a metodologia utilizada para desenvolver o projeto. Para isso, serão apresentadas as especificações de hardware utilizadas, a visão geral da proposta, as ferramentas e *software* escolhidos para realização da pesquisa, como foi elaborada a modelagem do guindaste, e a configuração dos ganhos do controlador.

3.1 Configuração de Equipamento Utilizado

Para o desenvolvimento do trabalho foi utilizado um computador *desktop* cujas especificações de *hardware* e *software* podem ser observadas na Tabela 1:

Tabela 1: Especificações de *hardware* e *software*

Componente	Especificação
Processador	Intel CORE I3 9ª geração
Memória RAM	12GB
Placa de vídeo	Intel® UHD Graphics 630 (CFL GT2)
Sistema operacional	Ubuntu 18.04.6 LTS
ROS	ROS1 Melodic
Gazebo	versão 9.0.0

3.2 Visão Geral da Proposta

A Figura 6 apresenta uma visão panorâmica dos elementos que constituem o sistema. A seção de simulação aborda a representação visual e física dos elementos do cenário, bem como as entradas do módulo RL. Por sua vez, o módulo de controle desempenha o papel de determinar o novo valor de recompensa e, quando necessário, reiniciar a simulação. O algoritmo de aprendizado por reforço (RL) assume a responsabilidade de conduzir o processo de aprendizagem e refinamento das estratégias.

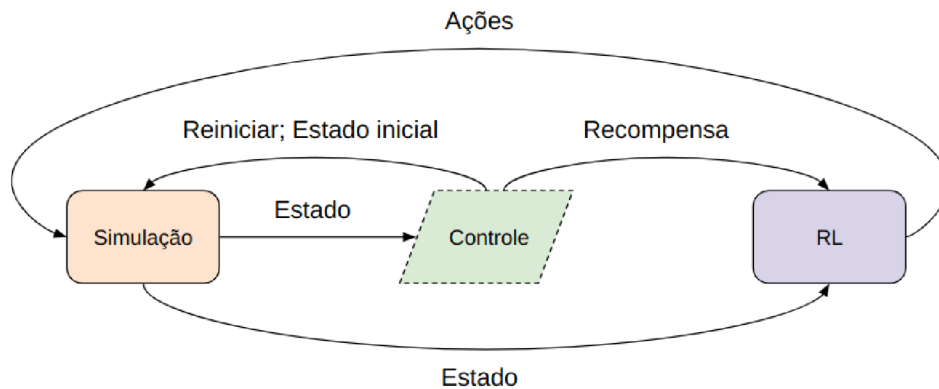


Figura 6: Visão geral do sistema

A simulação corresponde à parte visual e física do guindaste, onde estão presentes as juntas e seus controladores, bem como, o sensor *laser* e os objetos do ambiente. Na simulação obtém-se as entradas do módulo RL, sendo elas, as leituras do *laser*, a posição atual das juntas do guindaste e a distância da ponta da corda do guindaste até o alvo desejado.

O módulo de controle desempenha diversas funções cruciais no sistema, isso inclui, reiniciar a simulação para permitir que o agente explore diferentes caminhos e situações, acrescenta obstáculos ao ambiente para aumentar a complexidade e desafio da tarefa, e calcula a recompensa com base nos parâmetros de entrada do modelo. No entanto, é importante ressaltar que o módulo de controle é utilizado exclusivamente durante a fase de treinamento. Quando o sistema é implantado em um ambiente de produção, o conhecimento adquirido durante o treinamento é aproveitado para guiar as decisões do agente. Durante a produção, o agente usará as estratégias aprendidas para interagir com o ambiente e tomar ações que maximizem suas recompensas, sem a necessidade contínua de intervenção do módulo de controle.

O algoritmo RL é responsável por receber a recompensa e calcular a próxima ação (nova posição das juntas) para enviar a simulação, instigando o agente a receber a maior recompensa. A saída do sistema é uma nova posição do guindaste.

3.3 Modelagem do Ambiente

A proposta desta dissertação é usar o aprendizado por reforço para aprender as corretas ações a serem tomadas pelo guindaste. Neste contexto é importante disponibilizar um ambiente onde possam ser representados os diferentes estados possíveis do guindaste no ambiente de forma a oportunizar diferentes políticas de recompensa. Assim, opta-se por desenvolver um cenário de simulação composto por um guindaste e cargas simuladas fisicamente, onde o algoritmo de aprendizado por reforço será aplicado e convergirá para caminhos coerentes com a movimentação da carga. Propõe-se o uso do *framework* ROS

e do simulador Gazebo, bem como descreve-se um modelo de guindaste e de ambiente que serão utilizados em simulação para o aprendizado com os algoritmos DDPG e SAC. A seguir estes elementos são descritos.

3.3.1 Robot Operating System, Unified Robotic Description Format e Gazebo

O ROS é um *software* de código aberto que permite a comunicação entre processos de *software* e *hardware*, tornando mais fácil a construção de sistemas robóticos complexos. Além disso, o ROS conta com uma comunidade ativa de desenvolvedores que oferecem suporte por meio de plataformas como o ROS *Answers* e o ROS *Wiki*, bem como uma infraestrutura de comunicação robusta. Isso torna o ROS uma escolha popular para desenvolvedores de robótica em todo o mundo [56].

O *Robot Operating System* possui um conjunto de bibliotecas e ferramentas *open source* para desenvolvimento de robôs, e uma infraestrutura de comunicação e troca de mensagens que podem ser facilmente enviadas para uma variedade de ferramentas de visualização e teleoperação, como o envio de mensagens através de nós (ROS *Topic*), gravação e reprodução de mensagens (ROS *bag*), chamada remota de processos (ROS *Service*), entre outros. O ROS também suporta interfaces de *hardware* para muitos componentes de robôs, como câmeras, LIDARs e controladores de motor [57].

O *Unified Robotic Description Format* (URDF) é um formato de arquivo XML usado no ROS para descrever todos os elementos de um robô tornando possível a integração com *softwares* de simulação como o Gazebo [58].

O Gazebo é um simulador que permite testar algoritmos, projetar robôs, executar testes de regressão e treinar o sistema de IA usando cenários realistas. Ele também possui ferramentas para criar simulações de robôs com precisão e eficiência em complexos ambientes internos e externos, levando em considerações as propriedades físicas e dinâmicas dos objetos simulados [59].

3.3.2 Modelo do Guindaste

O uso de RL para o fornecimento de caminho para o guindaste requer um ambiente perceptivo que permita ao sistema aprender as melhores ações a serem tomadas. Para tal, propõe-se um ambiente de simulação que reproduz o guindaste e seu cenário de operação. Para isso, utilizou-se um software de *Computer Aided Design* (CAD) e através desta ferramenta, desenvolveu-se a modelagem 3D do guindaste.

O processo de modelagem digital do objeto foi desenvolvido pela pesquisadora do trabalho a partir de um modelo finalizado, disponível em <https://grabcad.com/library/crane-170>. Avaliando a estrutura já elaborada, foram realizados alguns ajustes visando adequar o guindaste, pois no modelo base utilizado existiam muitos detalhes de estrutura que causavam erros na exportação no formato necessário.

Modificando a base da estrutura, simplificando algumas das partes do guindaste e

inserindo elementos geométricos para complementar o modelo final, o resultado destas modificações é apresentado na Figura 7.

Foi escolhido este tipo de *software* para desenvolvimento da modelagem do guindaste, pois através de uma pesquisa identificou-se que ele atenderia as necessidades básicas do projeto e conseguiria gerar o arquivo de descrição de robôs, *Unified Robot Description Format* (URDF) necessário para a integração com o ROS1 e o simulador Gazebo.

Com o uso do URDF é possível incorporar nos *scripts* do modelo os controladores das juntas e o *plugin Hokuyo* que simula o laser responsável por captar as informações de distâncias utilizadas como entrada do sistema, bem como a inclusão do *bumper* utilizado para auxiliar na identificação de colisões com os obstáculos.

Esses elementos são de suma importância para a implementação dos algoritmos de RL no guindaste, haja visto que os controladores são os responsáveis pela interação do *software* com o modelo simulado e o laser identifica os objetos em cena (estado atual do ambiente).

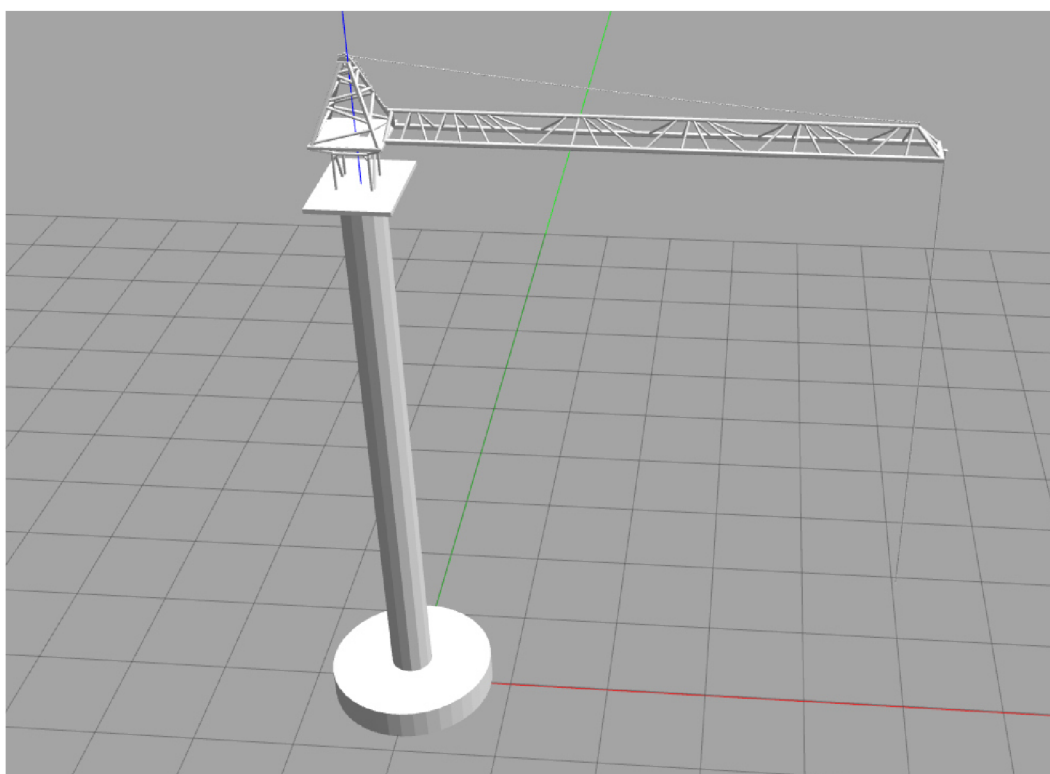


Figura 7: Modelo 3D do guindaste

Os ganhos do controlador foram definidos em um *framework* ROS chamado *rqt_gui*, onde é possível adicionar valores aos ganhos do controlador *PID* em cada junta controlada do modelo simulado e observar como será o novo comportamento das juntas atribuindo valores por meio do *message publisher* e acompanhando através de um gráfico em uma interface humano-máquina (IHM) como mostra a Figura 8. Após a definição dos ganhos do controlador, estes foram adicionados às definições de controle do guindaste para melhorar

o controle das juntas.

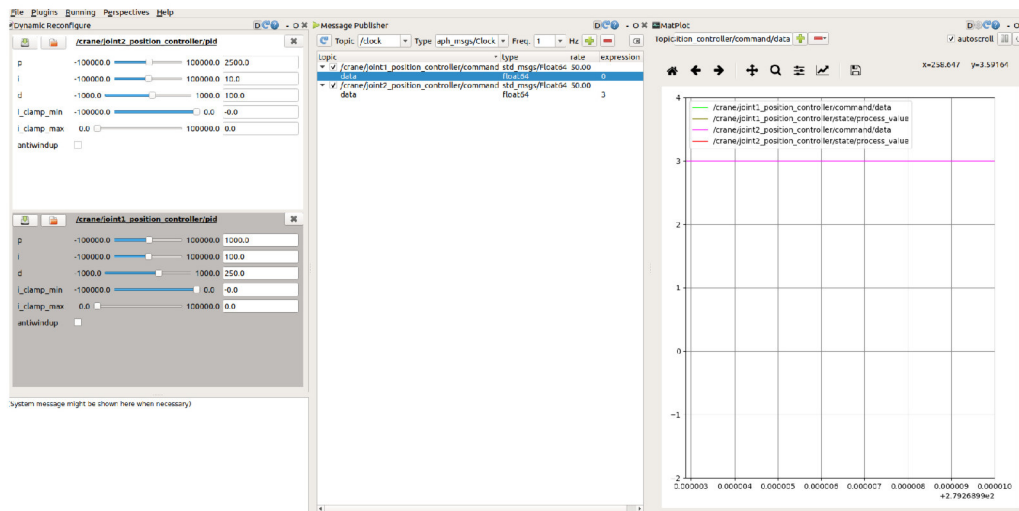


Figura 8: Ferramenta para ajustes de PID

3.3.3 Modelo do Ambiente

O ambiente simulado é apresentado na Figura 9. Nele, estão presentes o guindaste (objeto na cor cinza), quatro obstáculos (objetos na cor laranja), um alvo dinâmico e de posição aleatória a cada episódio de treinamento (objeto na cor vermelha) e o laser posicionado na ponta da haste do guindaste, com seu feixe (objeto na cor azul) apontado para baixo, identificando os elementos no cenário.

Define-se um episódio de treinamento como uma série de etapas em que o agente pode realizar ações. Cada etapa corresponde a uma movimentação executada pelo agente no cenário. Ao chegar no número correspondente de etapas o episódio é finalizado independentemente se o agente atingiu seu objetivo.

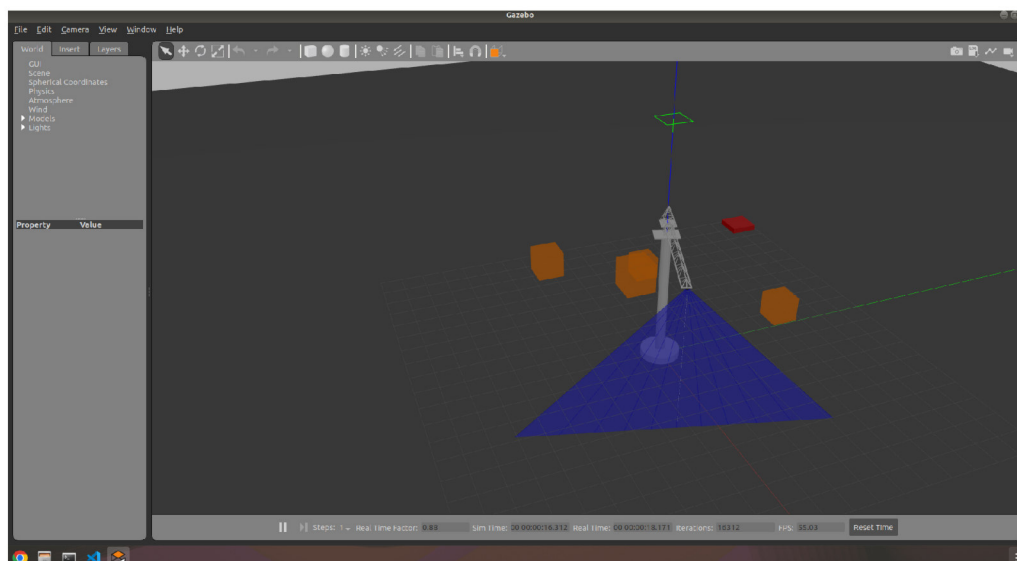


Figura 9: Ambiente simulado

Na Figura 10, pode-se observar de forma mais clara os componentes do guindaste, sendo a junta 1 definida como uma junta contínua e a junta 2 como junta prismática. Além disso, para facilitar e acelerar a identificação de colisões foi adicionado a junta 2 um *plugin* ROS chamado *bumper*, este *plugin* é definido como um sensor que retorna contatos de colisão. A intenção de incorporá-lo ao sistema foi de identificar colisões de maneira mais rápida e eficiente, sendo este utilizado em conjunto com os dados recebidos a partir do sensor laser.

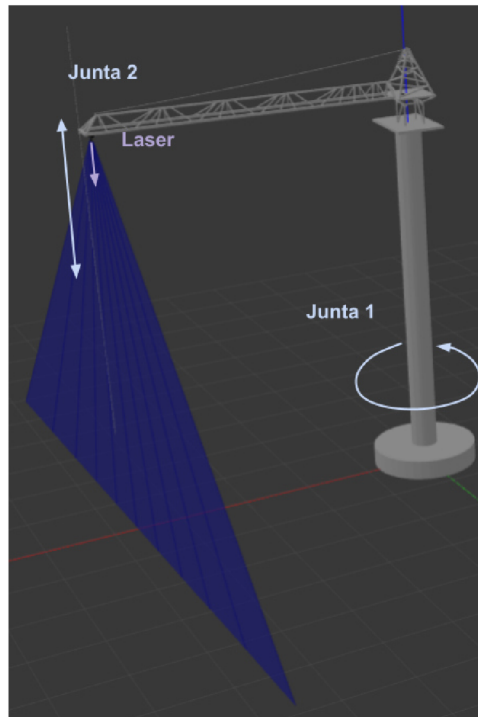


Figura 10: Visão geral do guindaste

O sensor *laser* desempenha um papel essencial na percepção do ambiente, contribuindo para as informações que constituem a entrada do módulo RL. Com base nesses valores, juntamente com a posição atual do agente, a distância atual da ponta da lança do guindaste ao alvo, bem como o valor de recompensa, o algoritmo de RL calcula a próxima ação a ser executada, transmite para as articulações do guindaste um novo comando para alterar suas posições objetivando a aproximação ao alvo.

3.4 Um Modelo para Planejamento de Caminho Baseado em Algoritmos de Aprendizado por Reforço

Nesta pesquisa serão averiguados dois diferentes algoritmos de aprendizado por reforço para o planejamento de caminho de operação de guindaste, ambos estados da arte, sendo eles o DDPG e SAC.

O DDPG é uma versão aprimorada do algoritmo ator-crítico, que usa as redes neurais

profundas para estimar a função de política ideal em vez de escolher a ação com base em uma distribuição específica. Este algoritmo pode ser facilmente aplicado a problemas complexos e estruturas de rede maiores com simplicidade e convergência [60]. Ele possui dois modelos, o ator e o crítico. A rede ator toma o estado atual como entrada e produz uma ação contínua, ao invés de uma distribuição de probabilidade sobre as ações. Já a rede crítica recebe o estado e a ação atual como entrada e gera um valor Q atualizado.

Já o algoritmo SAC engloba entropia da política como regularização, incentivando a exploração estocástica, levando-o a uma política mais suave e mais exploratória, ajudando na exploração mesmo nas fases finais do treinamento, onde outros algoritmos de RL geralmente convergem para uma solução abaixo do ideal devido à falta de exploração [22].

3.4.1 Estrutura dos Algoritmos: DDPG e SAC

Nesta subseção iremos abordar as estruturas utilizadas nos algoritmos de aprendizado por reforço implementados no trabalho.

O *script* do algoritmo DDPG é composto pela rede ator e pela rede crítico contendo camadas de função ReLU, que é uma função de ativação não linear, enquanto que o *buffer* de repetição é utilizado para atualizar os parâmetros da rede neural.

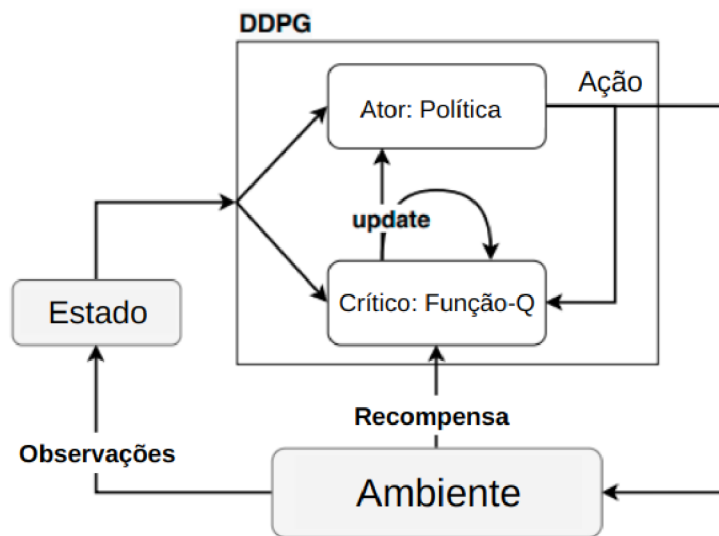


Figura 11: Arquitetura do sistema (adaptado de [5])

O *buffer* de repetição é um vetor circular de tamanho fixo responsável por armazenar as tuplas de experiências vivenciadas pelo sistema. Cada tupla é composta por informações como recompensa, estado atual, ação e próximo estado. Em seguida, utilizamos as experiências contidas no *buffer* para atualizarmos as redes. Além disso, adiciona-se ruído às ações para incentivar a exploração determinística. A abordagem típica para adicionar ruído é usar processos de ruído gaussiano, como o ruído *ornstein-uhlenbeck*,

que fornece uma exploração suave ao longo do tempo.

A Figura 11 mostra, de forma resumida, a arquitetura do sistema de aprendizado baseado em DDPG. A rede ator decide a ação a ser tomada, enquanto a rede crítico retorna ao ator o quão boa foi a ação tomada e como deverá se ajustar para tomar a próxima ação.

Assim como o algoritmo DDPG, o SAC é composto por uma rede ator, pela rede crítico e o uso de *buffer* para armazenamento de experiências, porém o que o difere é que na implementação do agente, realizamos a introdução de dois críticos a rede.

Também no SAC, a implementação de ruído é diferente, isso ocorre porque o SAC lida com políticas estocásticas, e portanto pode usar diretamente a entropia da distribuição de ações como uma medida de exploração. O ruído é uma parte intrínseca da política estocástica, e a entropia da política é incorporada na função de recompensa para incentivar a exploração. O agente busca maximizar a soma das recompensas esperadas e a entropia da política, que por sua vez ajuda a manter uma distribuição diversificada de ações.

3.4.2 Base Estrutural de Comunicação Entre os *Scripts*

Posterior a criação das redes, se fez necessária a identificação das juntas, do sensor laser e dos controladores do modelo em uma linguagem que tornasse esta comunicação possível. Para isso, utilizou-se a linguagem *Python*, pois esta possui bibliotecas disponíveis para a comunicação com o ROS, além de existirem diversas bibliotecas para implementação de algoritmos de aprendizado por reforço.

Criou-se um *script* para receber os dados dos *publishers* e *subscribers* dos controladores das juntas do guindaste, bem como o tópico de leitura do laser e *bumper*. Além disto, criou-se uma função para calcular a distância atual das juntas ao alvo, a detecção de colisão com os objetos de cena, as recompensas por chegar ao alvo e punições por colisões.

Além do *script* mencionado anteriormente, fez-se necessário o desenvolvimento de um código responsável pela inclusão dinâmica do alvo no cenário.

Estes elementos são utilizados a cada episódio de treinamento definindo as movimentações das juntas do guindaste para encontrar o alvo, baseando-se nas recompensas recebidas.

3.5 Política de Recompensa

O agente interpreta o *feedback* do ambiente por meio de um sistema de recompensas e punições que se manifestam como pontuações positivas (recompensas) ou negativas (punições). Nos primeiros testes com os algoritmos o sistema de recompensas é configurado para identificar colisões por meio do sensor *laser* e *bumper*, caso ocorra, o sistema é penalizado com uma pontuação negativa equivalente a menos dez (-10). Além disso, a outra condição avaliada pelo sistema de recompensas é se o agente não consegue atingir

seu objetivo dentro de 100 etapas em um único episódio de treinamento. Se essa situação ocorrer, o episódio é finalizado e o agente não recebe nenhuma recompensa.

Nos instantes em que o laser não identifica nenhum objeto, o agente seguirá mapeando o ambiente e não receberá nenhuma pontuação. Porém, quando o sensor *laser* identifica um objeto a uma distância maior que dois metros de distância, ele indica para o agente tomar uma ação a fim de evitar colisão. Esta ação pode ser por exemplo, girar a haste do guindaste ou elevar a lança.

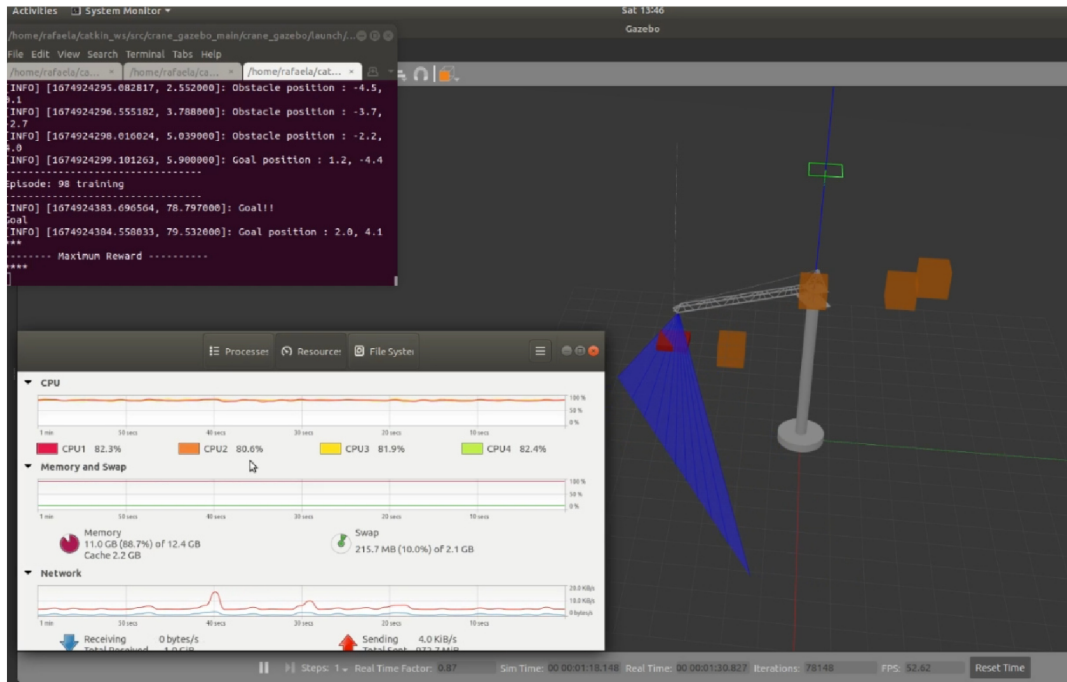


Figura 12: Máxima recompensa

Para receber uma recompensa positiva igual a cem pontos (+100), o agente precisa encontrar o alvo e isso ocorre quando a distância da ponta da haste do guindaste em relação a posição do alvo for menor do que 1 metro. Como mostra a Figura 12, ao atingir o alvo o sistema recebe a recompensa máxima, o alvo muda sua posição e o agente tenta encontrá-lo novamente neste mesmo episódio do treinamento. Caso ele não consiga encontrar o alvo novamente no mesmo episódio e ocorra uma colisão, ele receberá uma punição e perderá dez (-10) pontos de sua recompensa acumulada.

Além de trazer a comparação entre os dois algoritmos de aprendizado por reforço para identificar um caminho livre de colisões para o guindaste, o presente trabalho traz como contribuição a modificação do sistema de recompensa para tentar aprimorar o alcance do objetivo por meio do agente.

Senso assim, o sistema de recompensa proposto é definido da seguinte maneira:

- Recompensa positiva: +2 quando a distância atual ao alvo for menor que a anterior;
- Recompensa negativa: -1 quando a distância atual ao alvo for maior que a anterior;

- Recompensa positiva: +100 quando o agente encontra o alvo sem bater em nenhum obstáculo;
- Recompensa negativa: -10 quando o agente bate em algum obstáculo;

Atribui-se recompensas positivas quando o agente se aproxima do alvo e recompensas negativas quando se afasta para tentar induzir o agente a continuar avançando quando sua aproximação ao alvo for diminuindo.

A seguir são apresentadas exemplificações de um caminho realizado pelo agente até encontrar o alvo.

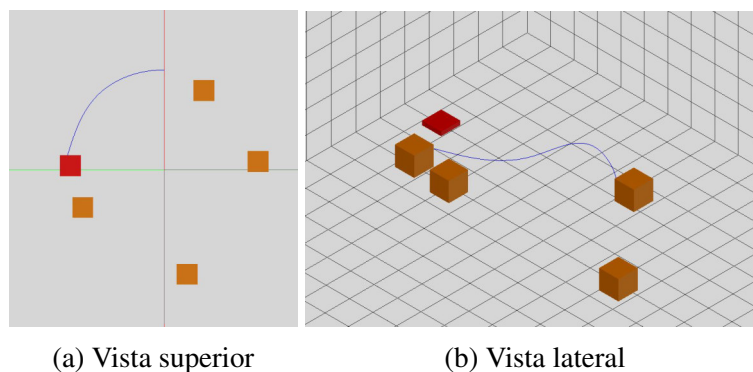


Figura 13: Exemplo de caminho de guindaste navegando em direção ao alvo, ilustrado na Fig. quatro obstáculos (objetos laranja), alvo (objeto vermelho) e o caminho linha (azul).

Na Figura 13, pode-se observar a vista superior e lateral do caminho que o agente percorreu para alcançar o alvo, é relevante destacar que se considera que o agente completou seu objetivo quando a distância da ponta da lança for menor que um metro.

4 TESTES E RESULTADOS

Nesta seção iremos apresentar os testes e resultados obtidos a partir das implementações dos algoritmos de aprendizado por reforço DDPG e SAC, com base nas metodologias delineadas nas seções anteriores.

4.1 Ambiente de Testes e Parametrização dos Modelos

Os testes realizados com os algoritmos de RL foram executados no ambiente descrito na Seção 3. Devido a algumas dificuldades encontradas, expostas na Seção 4.5, se fez necessário fixar as posições dos obstáculos para todos os episódios de treinamento. Na Tabela 2 estão as posições dos obstáculos no ambiente.

Tabela 2: Posição dos obstáculos

x	y	z
-2.5	-3.8	2.5
0.7	-4.5	2.4
3.1	3.4	2.5
-4.5	1.3	2.4

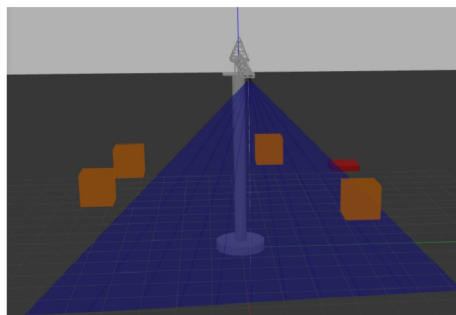


Figura 14: Posições dos obstáculos no cenário

Apesar da fixação dos obstáculos no ambiente, o alvo mantém sua posição dinâmica no cenário. Toda vez que o agente atinge o alvo ou a cada novo episódio de treinamento,

sua posição é alterada. A Figura 14 apresenta uma ilustração do cenário com os obstáculos nas posições indicadas na Tabela 2.

Nas Tabelas 3 e 4 são apresentadas as parametrizações de rede dos algoritmos DDPG e SAC:

Tabela 3: Estrutura DDPG

Parâmetro	Valor
Otimizador	Adam
Taxa de aprendizado ator	1×10^{-4}
Taxa de aprendizado crítico	1×10^{-3}
Buffer	10^4
Nº de camadas ocultas	2
Nº de unidades ocultas por camada	250
Nº de amostras por minilote	256
Ativação	ReLU

Tabela 4: Estrutura SAC

Parâmetro	Valor
Otimizador	Adam
Taxa de aprendizado	3×10^{-4}
Buffer	10^4
Nº de camadas ocultas	2
Nº de unidades ocultas por camada	256
Nº de amostras por minilote	256
Ativação	ReLU

A estrutura do algoritmo SAC segue a utilizada na implementação fornecida em[22].

4.2 Resultados DDPG

A seguir serão apresentados os resultados obtidos utilizando o algoritmo DDPG. A configuração da rede foi apresentada na Seção 3.4 e o sistema de recompensa apresentado anteriormente na Seção 4.1.

Ao gerar o gráfico dos valores de recompensa do treinamento utilizando somente as regras de compensação descritas anteriormente sem as novas contribuições para o sistema de recompensa, é possível observar que o agente consegue alcançar o alvo em diversos episódios do treinamento.

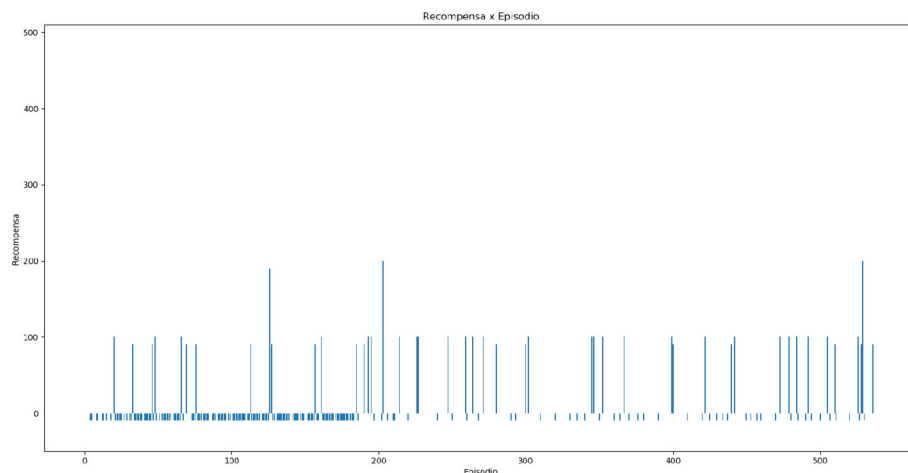


Figura 15: Gráfico de recompensa DDPG sem novas regras de recompensa

Os picos no gráfico representam os momentos que o agente atingiu seu objetivo e encontrou o alvo pelo menos uma vez durante o episódio de treinamento recebendo a recompensa máxima de (+100) pontos.

Posteriormente, foram adicionadas as novas contribuições para o sistema de recompensas. Nas situações em que a distância atual do guindaste ao alvo foi menor do que a distância anterior ao alvo, a variável de recompensa será igual a dois pontos positivos (+2). Contudo, nas situações em que a distância do guindaste ao alvo for maior do que a distância alcançada anteriormente, a variável de recompensa será igual a menos um ponto negativo (-1).

Com o incremento desta nova recompensa, como pode ser observado na Figura 16, ocorreu um incremento significativo de vezes que o agente encontrou o alvo, inclusive houve um acréscimo no número de vezes que ele encontrou o alvo mais de uma vez no mesmo episódio de treinamento.

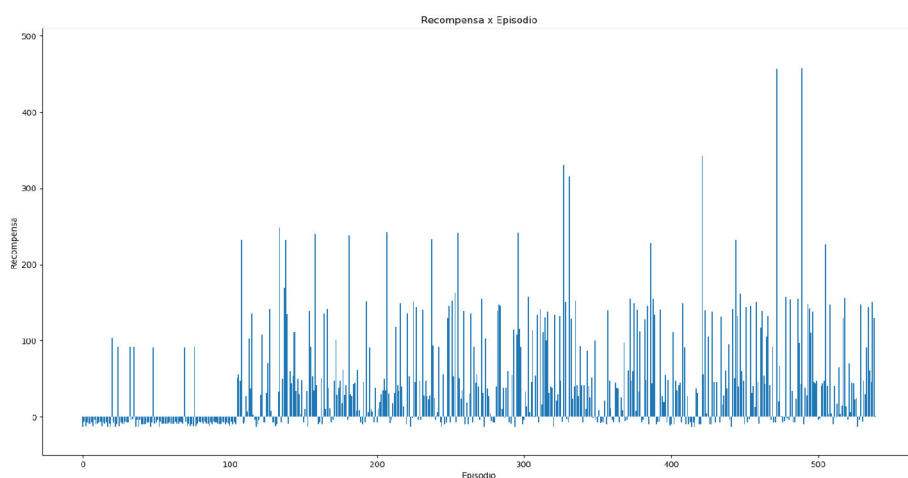


Figura 16: Gráfico de recompensa DDPG com novas regras de recompensa

É importante salientar que na elaboração dos gráficos os dois sistemas foram testados

pelo mesmo período de tempo e com a mesma quantidade de obstáculos presentes no cenário.

4.3 Resultados SAC

A seguir serão apresentados os resultados obtidos utilizando o algoritmo SAC. A configuração da rede foi apresentada na Seção 3.4. Ao gerar o gráfico dos valores de recompensa do treinamento sem o acréscimo por se aproximar do alvo ou decréscimo por se afastar do alvo, é possível observar que o agente consegue alcançar o objetivo em diversos episódios do treinamento. Também é possível observar que o número de vezes em que o agente encontra o alvo em um mesmo episódio é superior quando comparamos os resultados obtidos com o DDPG.

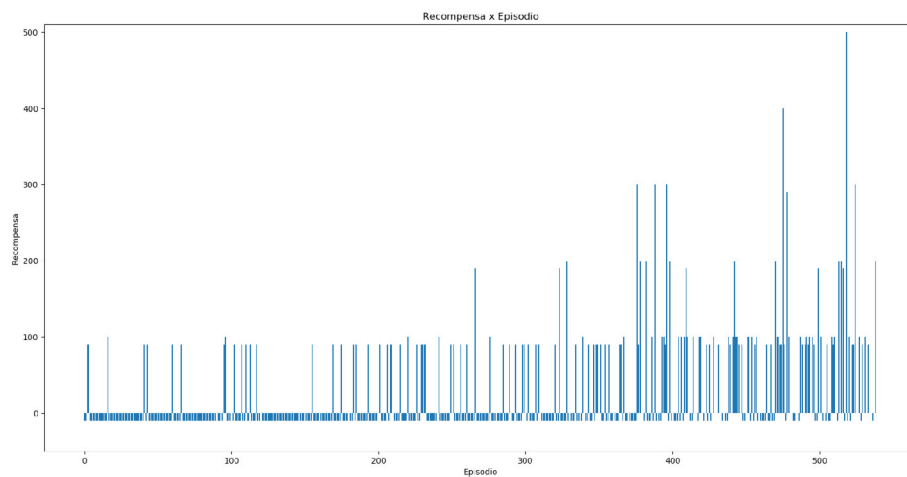


Figura 17: Gráfico de recompensa SAC sem novas regras de recompensa

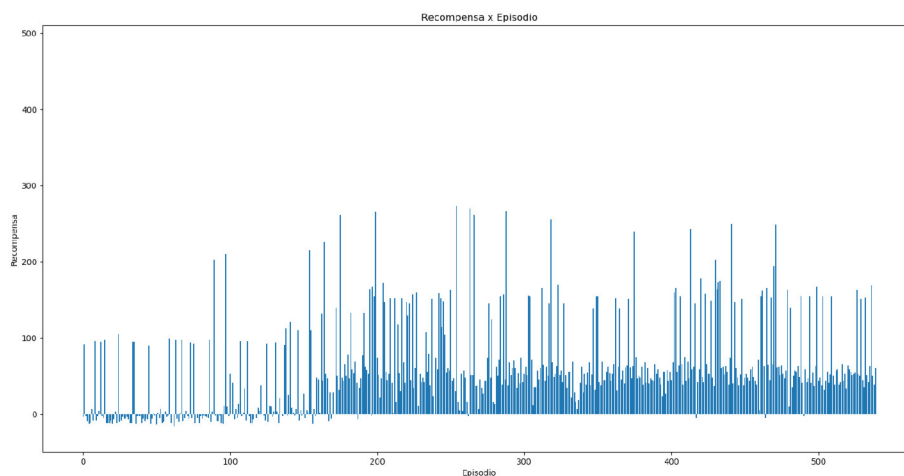


Figura 18: Gráfico de recompensa SAC com novas regras de recompensa

Na Figura 18 observamos os resultados obtidos após a inserção das novas regras de recompensa.

Observando os gráficos apresentados nas Figuras 17 e 18, nota-se que a introdução de novas recompensas não afetou positivamente o desempenho do algoritmo, pois sem esta adição ele apresentou uma melhor constância no aprendizado conseguindo alcançar o alvo mais vezes durante o mesmo episódio de treinamento.

4.4 Comparando Resultados

Nesta subseção serão apresentados os comparativos entre os dois algoritmos.

Como pode ser observado na Tabela 5, realizamos a comparação entre os resultados do algoritmo DDPG com e sem as novas recompensas propostas e o desempenho do SAC com e sem as recompensas propostas em um total de 539 episódios.

Tabela 5: Comparação de Resultados: DDPG x SAC

	Vezes que alcançou o alvo	Vezes que alcançou o alvo (%)
DDPG com novas recompensas	123	22.8
DDPG sem novas recompensas	46	8.53
SAC com novas recompensas	109	20.22
SAC sem novas recompensas	135	25.04

Nota-se uma distinção notável nos efeitos das recompensas propostas quando aplicadas aos algoritmos DDPG e SAC. No algoritmo DDPG, a introdução das novas recompensas propostas resultam em melhorias positivas. No entanto, no algoritmo SAC este incremento não repercute positivamente no seu desempenho.

Conforme discutido por Haarnoja et al. [22], o SAC demonstra sensibilidade ao dimensionamento de sinal de recompensa, onde maiores magnitudes de recompensa correspondem a entradas mais baixas. Para pequenas magnitudes de recompensa, a política torna-se quase uniforme e, conseqüentemente, falha em explorar o sinal de recompensa, resultando em degradação substancial do desempenho. Para grandes magnitudes de recompensa, o modelo aprende rapidamente no início, mas a política se torna quase determinística levando a mínimos locais ruins devido à falta de exploração adequada. Com o dimensionamento de recompensa correto o modelo equilibra *exploration* e *exploitation*, levando a um aprendizado mais rápido e a um melhor desempenho.

No caso do algoritmo DDPG, ele emprega metas suavizadas para atualizar suas redes de valor, contribuindo para mitigar instabilidades no treinamento. Contudo, essa abordagem pode conferir uma certa sensibilidade às mudanças abruptas nas recompensas recebidas.

Em resumo, os algoritmos DDPG e SAC reagem diferentemente à manipulação das

recompensas e essas diferenças resultam de suas respectivas características e sensibilidades a diversos fatores, como exploração, aprendizado e ajustes dos parâmetros internos.

4.5 Dificuldades Encontradas

Inicialmente, foi considerada a utilização do *Open AI Gym* [61] para o desenvolvimento do trabalho. Contudo, ao tentar utilizar as bibliotecas disponíveis não foi obtido sucesso por conta das suas incompatibilidades com o ROS1, sendo necessário a utilização do ROS2.

Entretanto, os esforços de estudo e experimentação voltados para o ROS2 não resultaram no sucesso esperado, particularmente no que refere-se ao desenvolvimento do ambiente e sua integração com o *Open AI Gym*. A fim de abordar essas dificuldades, uma alternativa viável foi desenvolver um ambiente personalizado para a execução dos algoritmos de aprendizado por reforço utilizando o ROS1.

Posterior a estas tentativas de desenvolvimento do *Open AI gym*, as principais adversidades encontradas foram no desenvolvimento do ambiente gráfico de testes criado dentro do Gazebo. Uma vez que o guindaste foi modelado em 3D no software de CAD e o modelo resultante foi exportado para o formato URDF (*Unified Robot Description Format*), deparou-se com diversas incompatibilidades ao definir as articulações do guindaste. Essas incompatibilidades comprometiam a possibilidade de incorporar os controladores necessários para o funcionamento adequado do sistema.

Além das dificuldades encontradas, percebeu-se uma perda de desempenho do computador devido a utilização de toda memória *RAM* disponível no equipamento ao executar repetidamente vários episódios de treinamento de maneira consecutiva. Para abordar essa situação, foram implementadas algumas tentativas para mitigar esse problema. Entre essas medidas, destaca-se a incorporação de um recurso que salva o modelo a cada 100 episódios realizados no treinamento e, posteriormente, encerra o processo. Em seguida, um *script* carrega o modelo salvo e reinicia o treinamento a partir do último ponto armazenado. Este artifício fez com que a memória do computador não fosse totalmente comprometida com o treinamento da rede, porém após testes mais robustos foram encontradas inconsistências na sequência de aprendizado, descartando a utilização desta solução. Subsequentemente, foi aplicado o uso do *garbage collector* como sugerido por um membro da banca de qualificação do trabalho, porém também sem sucesso.

Após muitas pesquisas, foi encontrado um fórum a respeito de vazamento de memória causado por um erro na instalação de uma das bibliotecas utilizadas nos algoritmos de aprendizado. Seguindo as indicações do fórum, foi realizada a formatação de disco e uma nova instalação de todo o *setup*, incluindo o sistema operacional, ROS, Gazebo, Python e demais bibliotecas. Após a nova instalação do sistema, notou-se que o problema de memória havia sido resolvido.

Ao resolver a questão relacionada à memória, o ambiente permitiu a condução de treinamentos ininterruptos. No entanto, durante esse processo contínuo, foi identificado um erro intrínseco a um dos serviços próprios do Gazebo, conhecido como *gazebo_msgs.srv Delete Model*. Quando esse método era chamado de maneira repetida, apresentava falhas e não conseguia remover adequadamente os objetos do cenário. Diante dessa dificuldade, que poderia comprometer os testes e a obtenção de resultados, foi tomada a decisão de manter os obstáculos em posições fixas. Nesse cenário, apenas o alvo era modificado de maneira dinâmica a cada episódio de treinamento, permitindo que os testes prosseguissem e resultados fossem gerados de forma confiável.

5 CONCLUSÃO

O número de acidentes causados pela pouca visibilidade dos operadores de guindaste são alarmantes, pois estes equipamentos são fortemente dependentes de atitudes, reflexos, experiência do guindasteiro e situações climáticas.

Visando mitigar o número de acidentes e facilitar a operação do equipamento, este trabalho teve como objetivo desenvolver um modelo para planejamento de caminho livre de colisão para manipulação de guindastes através do desenvolvimento de um sistema utilizando aprendizado por reforço.

Este é um problema complexo que não possui uma base de dados robusta com livre acesso. Consequentemente, não sendo possível utilizar as metodologias empregadas em aprendizado supervisionado e não supervisionado, pois estas necessitam de dados anteriores. Desta forma, optou-se pela utilização do aprendizado por reforço, pois este método aplica políticas e estratégias de recompensa e penalidade, fazendo com que o sistema aprenda com o ambiente ao qual está inserido.

A aprendizagem por reforço é uma estrutura promissora para lidar com as tarefas de tomada de decisão. No entanto, encontrar caminhos livres de colisão e com eficiência em torno de outros agentes do ambiente é um grande desafio, haja visto que exige a previsão de movimentação e a antecipação de padrões de interação em tempo real.

É sabido que a maior parte dos algoritmos de planejamento de caminhos tradicionais tem baixa adaptabilidade e são adequados apenas para situações com modelos ambientais. Porém, os agentes do RL podem otimizar as estratégias através da interação com ambientes desconhecidos por métodos de tentativa e erro e acumulando recompensas para aprender o padrão de comportamento ideal. Também pode-se destacar que o RL tem boa generalidade e é adequado para ambientes desconhecidos [27].

Com o que foi apresentado na Seção 4, pode-se concluir que o uso de aprendizado por reforço é um campo promissor para o planejamento de caminho livre de colisões para guindastes. No contexto apresentado no trabalho, as técnicas utilizadas conseguiram mostrar certa evolução ao desempenhar um caminho que busca encontrar o alvo nos episódios de treinamento nos cenários propostos.

Acredita-se que ainda existam muitos testes a serem realizados, principalmente au-

mentando o número de etapas máximas dentro de um mesmo episódio de treinamento. Entende-se que a limitação máxima de 100 etapas por episódio pode ser demasiadamente baixa levando em consideração a complexidade do ambiente.

Além disso, se faz necessária a realização de testes com outras políticas de recompensa, pois apesar da introdução de incentivos nas situações em que o agente se aproxima do alvo, isso pode levá-lo a sempre tentar ir pelo caminho mais longo a fim de receber mais recompensas, fazendo com que a prioridade, além de não colidir com obstáculos, seja percorrer um caminho maior para obter mais recompensas, o que pode se tornar perigoso em um ambiente real devido as circunstâncias de manipulação dos guindastes.

Indica-se o aprimoramento das políticas de recompensa para maximizar a evolução do agente sem incentiva-lo a buscar somente a maior recompensa; realizar treinamentos com um número maior de etapas por episódio, pois a complexidade do ambiente faz com que o agente necessite um tempo maior para alcançar o objetivo; a realização de um número mais robusto de treinamentos com um *buffer* maior, assim possibilitando que o agente armazene uma quantidade mais robusta de experiências para aprimorar seu aprendizado.

Também sugere-se averiguar o uso de *Hindsight Experience Replay* para melhorar a eficiência de aprendizado de um agente ao lidar com recompensas escassas. Em vez de depender apenas dos objetivos e recompensas originais definidos para a tarefa, o HER aproveita o resultado de um episódio, mesmo que não tenha atingido o objetivo desejado, para criar novas experiências com metas alternativas [62].

Como resultado desta dissertação, elenca-se a publicação dos seguintes trabalhos científicos:

- Machado, R. I., Dos Santos, M. M., Botelho, S. S., ***Enhancing Crane Handling Safety: A Deep Deterministic Policy Gradient Approach to Collision-Free Path Planning***. *IEEE International Conference on Industrial Informatics, INDIN 2023* Lemgo, Alemanha;
- Machado, R. I., Dos Santos, M. M., Botelho, S. S., ***Safe Crane Handling with Collision-Free Path Planning using DDPG***. *IEEE Latin American Robotics Symposium - LARS 2023* Salvador, Bahia
- Machado, R. I., Dos Santos, M. M., Botelho, S. S., ***Modelo para Planejamento de Caminho Livre de Colisão para Manejo de Guindaste utilizando DDPG***. *Simpósio Brasileiro de Automação Inteligente - SBAI 2023* Manaus, Amazonas

Com isso, conclui-se que o trabalho realizado atendeu a proposta de desenvolver os conhecimentos da aluna acerca do aprendizado por reforço, sendo capaz de avaliar que o desempenho dos algoritmos DDPG e SAC são alternativas possíveis para criar um modelo de planejamento de caminho livre de colisões para o manejo de guindastes.

REFERÊNCIAS

- [1] J. Lawson, “Compact board offshore crane.” <http://surl.li/goapg>, 2018. Online, acesso: janeiro 2023.
- [2] L. Edell, “Is machine learning the new epm black?.” <http://surl.li/gnznl>, 2015. Online, acesso: novembro 2021.
- [3] A. Jeffares, “Supervised vs unsupervised learning in 3 minutes.” <http://surl.li/gnzng>, 2018. Online, acesso: novembro 2021.
- [4] A. G. Sutton, Richard S. and Barto, *REINFORCEMENT LEARNING: AN INTRODUCTION*. MIT press, 2018.
- [5] M. Handaoui, J.-E. Dartois, J. Boukhobza, O. Barais, and L. d’Orazio, “Releaser: A reinforcement learning strategy for optimizing utilization of ephemeral cloud resources,” in *2020 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 65–73, IEEE, 2020.
- [6] G. N. e. B. Agência Nacional do Petróleo, “Anuário estatístico brasileiro do petróleo, gás natural e biocombustível: 2022.” <http://surl.li/gnzmt>, 2022. Online, acesso: fevereiro 2023.
- [7] M. P. Abraçado, F. J. d. C. M. Duarte, and P. D. Béguin, “Microprojetos em usos imprevisíveis: O caso da movimentação de cargas em plataformas offshore,” *Human Factors in Design*, vol. 10, no. 19, 2021.
- [8] R. S. d. Miranda, “Análise dinâmica de um guindaste offshore sobre pedestal,” 2019.
- [9] A. P. Institute, “Specification for offshore pedestal mounted cranes.” <http://surl.li/gnznc>, 2004. Online, acesso: fevereiro 2023.
- [10] J. Neupert, T. Mahl, B. Haessig, O. Sawodny, and K. Schneider, “A heave compensation approach for offshore cranes,” in *2008 American Control Conference*, pp. 538–543, IEEE, 2008.

- [11] H.-C. Park, S. Chakir, Y.-B. Kim, and D.-H. Lee, "A robust payload control system design for offshore cranes: Experimental study," *Electronics*, vol. 10, no. 4, 2021.
- [12] A. Sneddon, K. Mearns, and R. Flin, "Stress, fatigue, situation awareness and safety in offshore drilling crews," *Safety science*, vol. 56, pp. 80–88, 2013.
- [13] K. M. Sutherland and R. H. Flin, "Stress at sea: a review of working conditions in the offshore oil and fishing industries," *Work & Stress*, vol. 3, no. 3, pp. 269–285, 1989.
- [14] S. Pavičić Žeželj, O. Cvijanović Peloza, F. Mika, S. Stamenković, S. Mahmutović Vranić, and S. Šabanagić Hajrić, "Anxiety and depression symptoms among gas and oil industry workers," *Occupational Medicine*, vol. 69, no. 1, pp. 22–27, 2019.
- [15] B. Henrique Miranda Galhardo, "Guindaste offshore: Modelagem e simulação baseadas em helicoides," 2018. Dissertação (Mestrado) - Instituto Militar de Engenharia.
- [16] M. F. Milazzo, G. Ancione, V. Spasojevic Brkic, *et al.*, "Safety in crane operations: An overview on crane-related accidents," in *Proceedings of the 6th International Symposium on Industrial Engineering, SIE*, pp. 36–39, 2015.
- [17] T. Cheng and J. Teizer, "Modeling tower crane operator visibility to minimize the risk of limited situational awareness," *Journal of Computing in Civil Engineering*, vol. 28, no. 3, p. 04014004, 2014.
- [18] T. Sutjaritvorakul, A. Vierling, and K. Berns, "Data-driven worker detection from load-view crane camera," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 37, pp. 864–871, IAARC Publications, 2020.
- [19] J. R. Juang, W.-H. Hung, and S. C. Kang, "Simcrane 3d+: A crane simulator with kinesthetic and stereoscopic vision," *Advanced Engineering Informatics*, vol. 27, no. 4, pp. 506–518, 2013.
- [20] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.
- [21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

- [22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.
- [23] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.
- [24] E. J. C. d. Souza, *Controle anti-oscilatório de tempo mínimo para guindaste usando a programação linear*. PhD thesis, Universidade de São Paulo, 2009.
- [25] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [26] Y. Hu, L. Yang, and Y. Lou, “Path planning with q-learning,” in *Journal of Physics: Conference Series*, vol. 1948, p. 012038, IOP Publishing, 2021.
- [27] Y. Zhao, X. Wang, R. Wang, Y. Yang, and F. Lv, “Path planning for mobile robots based on tpr-ddpg,” in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2021.
- [28] P. Cunningham, M. Cord, and S. J. Delany, *Machine Learning Techniques for Multimedia*. Springer, 2008.
- [29] B. Liu, *Web Data Mining*. Springer, 2011.
- [30] G. E. Hinton, T. J. Sejnowski, *et al.*, *Unsupervised learning: foundations of neural computation*. MIT press, 1999.
- [31] J.-O. Palacio-Niño and F. Berzal, “Evaluation metrics for unsupervised learning algorithms,” *arXiv preprint arXiv:1905.05667*, 2019.
- [32] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, “Curriculum learning for reinforcement learning domains: A framework and survey,” *Journal of Machine Learning Research*, pp. 1–50, 2020.
- [33] Q. Hu and W. Yue, *Markov decision processes with their applications*, vol. 14. Springer Science & Business Media, 2007.
- [34] P. P. Reddy, “Autonomous car: Deployment of reinforcement learning in various autonomous driving applications,” *EasyChair Preprint*, no. 1305, 2019.
- [35] J. Yu, Y. Su, and Y. Liao, “The path planning of mobile robot by neural networks and hierarchical reinforcement learning,” *Frontiers in Neurorobotics*, vol. 14, p. 63, 2020.

- [36] Í. d. C. Mota, “Aprendizagem por reforço utilizando q-learning e redes neurais artificiais em jogos eletrônicos,” 2018.
- [37] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, “Navigating occluded intersections with autonomous vehicles using deep reinforcement learning,” in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 2034–2039, IEEE, 2018.
- [38] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, “An introduction to deep reinforcement learning,” *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, p. 219–354, 2018.
- [39] H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton, “Toward off-policy learning control with function approximation,” in *ICML*, vol. 10, pp. 719–726, 2010.
- [40] C. J. C. H. Watkins, *Learning from Delayed Rewards*. PhD thesis, King’s College, Oxford, 1989.
- [41] E. Even-Dar, Y. Mansour, and P. Bartlett, “Learning rates for q-learning,” *Journal of machine learning Research*, vol. 5, no. 1, 2003.
- [42] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, *et al.*, “Model-based reinforcement learning for atari,” *arXiv preprint arXiv:1903.00374*, 2019.
- [43] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau, *et al.*, “An introduction to deep reinforcement learning,” *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018.
- [44] J. C. Jesus, J. A. Bottega, M. A. Cuadros, and D. F. Gamarra, “Deep deterministic policy gradient for navigation of mobile robots in simulated environments,” in *2019 19th International Conference on Advanced Robotics (ICAR)*, pp. 362–367, IEEE, 2019.
- [45] H. Gong, P. Wang, C. Ni, and N. Cheng, “Efficient path planning for mobile robot based on deep deterministic policy gradient,” *Sensors*, vol. 22, no. 9, p. 3579, 2022.
- [46] W. Othman and N. Shilov, “Deep reinforcement learning for path planning by cooperative robots: Existing approaches and challenges,” in *2021 28th Conference of Open Innovations Association (FRUCT)*, pp. 349–357, IEEE, 2021.
- [47] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, “Soft actor-critic algorithms and applications,” *arXiv preprint arXiv:1812.05905*, 2018.

- [48] P. Gao, Z. Liu, Z. Wu, and D. Wang, “A global path planning algorithm for robots using reinforcement learning,” in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1693–1698, IEEE, 2019.
- [49] S. Lin, A. Liu, J. Wang, and X. Kong, “A review of path-planning approaches for multiple mobile robots,” *Machines*, vol. 10, no. 9, p. 773, 2022.
- [50] Z. Liu, Q. Wang, and B. Yang, “Reinforcement learning-based path planning algorithm for mobile robots,” *Wireless Communications and Mobile Computing*, vol. 2022, 2022.
- [51] X. Ruan, D. Ren, X. Zhu, and J. Huang, “Mobile robot navigation based on deep reinforcement learning,” in *2019 Chinese control and decision conference (CCDC)*, pp. 6174–6178, IEEE, 2019.
- [52] V. R. Miranda, A. A. Neto, G. M. Freitas, and L. A. Mozelli, “On the generalization of deep reinforcement learning methods in the problem of local navigation,” *arXiv preprint arXiv:2209.14271*, 2022.
- [53] S. Guo, X. Zhang, Y. Zheng, and Y. Du, “An autonomous path planning model for unmanned ships based on deep reinforcement learning,” *Sensors*, vol. 20, no. 2, p. 426, 2020.
- [54] Y. Wang, Y. Fang, P. Lou, J. Yan, and N. Liu, “Deep reinforcement learning based path planning for mobile robot in unknown environment,” in *Journal of Physics: Conference Series*, vol. 1576, p. 012009, IOP Publishing, 2020.
- [55] S. Cho and S. Han, “Reinforcement learning-based simulation and automation for tower crane 3d lift planning,” *Automation in Construction*, vol. 144, p. 104620, 2022.
- [56] R. O. System, “Ros.” <https://www.ros.org/>, S/D. Online, acesso: novembro 2021.
- [57] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, “Ros: an open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.
- [58] Gazebo, “Why gazebo?” <https://classic.gazebosim.org/>, S/D. Online, acesso: setembro 2022.
- [59] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2149–2154, IEEE, 2004.

- [60] Y. Du, X. Zhang, Z. Cao, S. Wang, J. Liang, F. Zhang, and J. Tang, “An optimized path planning method for coastal ships based on improved ddpg and dp,” *Journal of Advanced Transportation*, vol. 2021, 2021.
- [61] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.
- [62] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, “Hindsight experience replay,” *arXiv preprint arXiv:1707.01495*, 2018.