Renato Sousa Bezerra

Detecção de Falhas de Movimentação para Robôs Omnidirecionais Baseada em Dados

Universidade Federal de Pernambuco
graduacao@cin.ufpe.br
www.cin.ufpe.br/~secgrad

Recife
2021

Renato Sousa Bezerra

Detecção de Falhas de Movimentação para Robôs Omnidirecionais Baseada em Dados

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Área de Concentração: *Inteligência Computational*
Orientador: *Edna Natividade da Silva Barros*

Recife
2021

AGRADECIMENTOS

Primeiramente gostaria de agradecer a minha família, principalmente minha mãe e meus irmãos, que me deram o melhor suporte possível nessa longa jornada. Sem vocês seria impossível chegar até aqui.

Agradeço a minha orientadora, Professora Edna Barros, por todo apoio durante a realização desse trabalho. Agradeço também aos professores e companheiros do RobôCIn por todos esses anos de competições e vitórias.

Por fim, não poderia esquecer dos meus amigos que tanto me ajudaram nesse caminho: Lucas Cavalcanti, Roberto Fernandes, Cristiano Oliveira, Walber Macedo, Juliana Damurie, Victor Sabino, Heitor Rapela, Felipe Martins, Matheus Gonçalves, Raphael Brito, Amaro Assunção, entre outros. Obrigado por compartilhar momentos tão importantes comigo.

“Nobody ever figures out what life is all about, and it doesn’t matter. Explore the world. Nearly everything is really interesting if you go into it deeply enough."

–Richard P. Feynman

ABSTRACT

The industry has increasingly used autonomous mobile robotics for its ability to iterate with dynamic environments and real-time decision-making. This class of robots is characteristic for movement based on vision, sensors, and artificial intelligence, and with the advancement of these areas, we can see its application in new contexts such as factories, warehouses, and hospitals. The robot world cup Robotics Competition (RoboCup) was created to foment the study of robotics, presenting multiple challenges in a competition format between researchers. One of the main and complex categories of autonomous robot soccer is called Small Size League (SSL), which is characterized by teams of 6 players competing in a match. The movement of the robots in the field depends on a complex mechanical system with multiple components, where the malfunction of any step can cause unexpected behavior and may even generate irreversible problems, such as the uselessness of the motors. The robots must be monitored to avoid such problems, and possible failures are identified as soon as possible. This paper presents a comparative study of techniques for identifying omnidirectional robot motion faults in real-time. Starting from an initial approach where the paths of robots with and without failures are visually compared, following to an analytical study, which consists in the observation of expected and observed velocities. Finally, from a data-driven approach, statistical models were analyzed from historical behaviors to detect the presence or not of faults, and it was possible to build a classifier based on K nearest neighbors with an accuracy of 98.59

Keywords: Robotics. Fault Detection. Machine Learning

RESUMO

A robótica móvel autônoma tem sido cada vez mais utilizada pela industria por sua capacidade de iteração com ambientes dinâmicos e tomada de decisão em tempo real. Essa classe de robôs tem como principal característica a movimentação baseada em visão, sensores e inteligência artificial, e com o avanço dessas áreas, podemos ver a sua aplicação em novos contextos como fábricas, armazéns e hospitais. Para fomentar o estudo da robótica, a copa do mundo de robôs (RoboCup) foi criada, apresentando múltiplos desafios no formato de competição entre pesquisadores. Uma das principais e complexas categorias de futebol de robôs autônomos é chamada Small Size League (SSL), que se caracteriza por times de 6 jogadores disputando uma partida. A movimentação dos robôs em campo depende de um complexo sistema mecânico com múltiplos componentes, em que o mau funcionamento de qualquer etapa pode ocasionar em um comportamento não esperado, podendo até gerar problemas irreversíveis, como inutilidade dos motores. Para evitar tais problemas, é preciso que os robôs sejam monitorados, e possíveis falhas sejam identificas o mais rápido possível. Esse trabalho apresenta um estudo comparativo de técnicas para identificação de falhas de movimentação de robôs omnidirecionais em tempo real. Partindo de uma abordagem inicial onde as trajetórias de robôs com e sem falhas são comparadas visualmente, seguindo para um estudo analítico, que consiste na observação das velocidades esperadas e observadas. Por fim, a partir de uma abordagem baseada em dados, foram analisados modelos estatísticos a partir de comportamentos históricos para identificar ou não a presença de falhas, foi e possível construir um classificador baseando em K vizinhos mais próximos com uma acurácia de 98.59%.

Palavras-chave: Robótica. Detecção de Falhas. Aprendizagem de Máquina.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

cm	Centímetro
kNN	K-Nearest Neighbors
m	Metro
MLP	Multilayer Perceptron
mm	Milímetro
RMAs	Robôs Móveis Autônomos
RoboCup	Robotics Competition
SSL	Small Size League
SVM	Support Vector Machine
UFPE	Universidade Federal de Pernambuco
VGAs	Veículos Guiados Automatizados

LISTA DE SÍMBOLOS

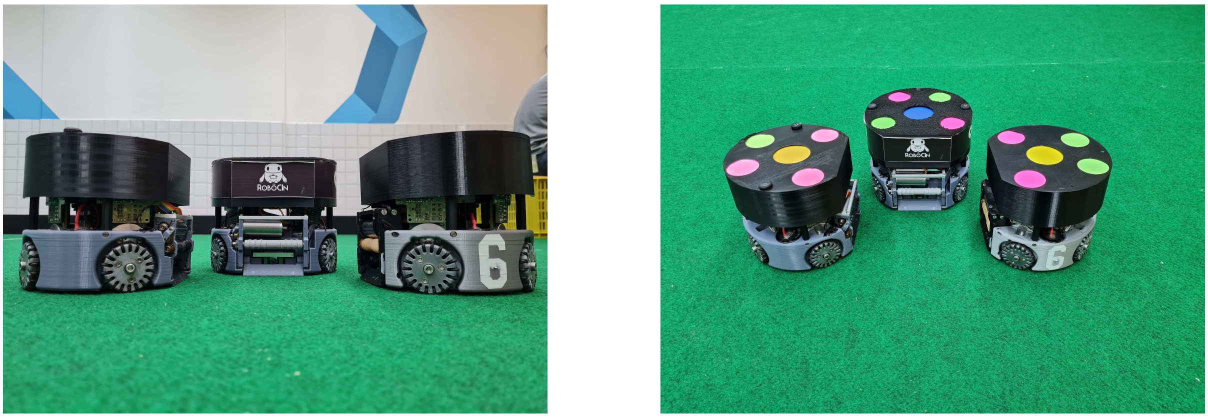
φ	Função de Ativação

SUMÁRIO

1
INTRODUÇÃO

Diferente de Veículos Guiados Automatizados (VGAs), os Robôs Móveis Autônomos (RMAs) possuem uma movimentação baseada em visão, sensores, inteligência artificial, entre outros. Dessa maneira, robôs autônomos conseguem se adaptar facilmente à ambientes dinâmicos como fábricas, armazéns e hospitais [Fragapane *et al.*, 2021]. Essas características tornam possível o uso da robótica em competições como futebol de robôs.

A Robotics Competition (RoboCup), maior competição mundial de robôs, possui uma categoria chamada Small Size League (SSL), que se caracteriza por times de 6 robôs, totalmente autônomos, de diâmetro até 180mm e altura até 150mm, que se movimentam em um campo de (9 x 6) m. A Figura 1 apresenta os robôs do time de robótica do Cento de Informática usados nessa categoria em diferentes ângulos.



(a) Visão frontal e lateral

(b) Visão Superior

Figura 1: Robôs utilizados pelo RobôCIn na categoria SSL.

A competição possui um sistema de visão global, que pode ser composto por uma ou mais câmeras, localizadas a aproximadamente 4 metros de altura, sobre o campo. As imagens são processadas por um *software open-source* chamado *SSL-Vision* [Zickler *et al.*, 2009] responsável por detectar e localizar os robôs e bola dentro do campo. Após a detecção, as posições de todos objetos são enviados para o software de estratégia de cada equipe. Cada time possui um programa, usualmente chamado de *coach* que recebe as informações processadas pelo *SSL-Vision*, planeja a melhor estratégia para vencer a partida e envia as velocidades desejadas para cada robô em campo. Todo esse fluxo é ilustrado na Figura 2.

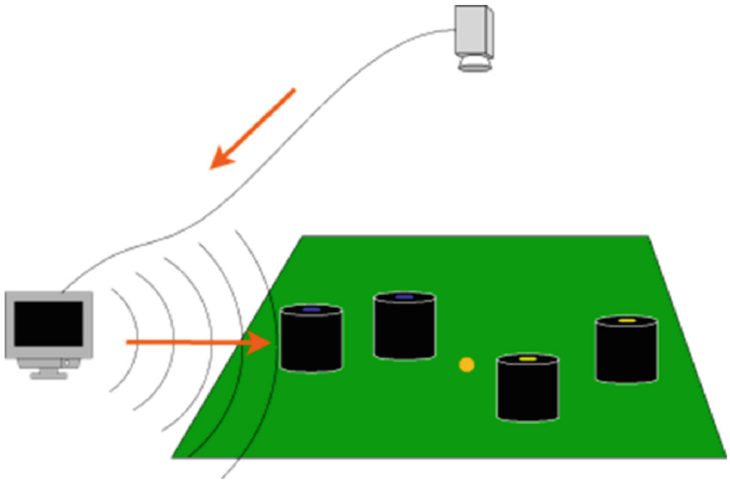
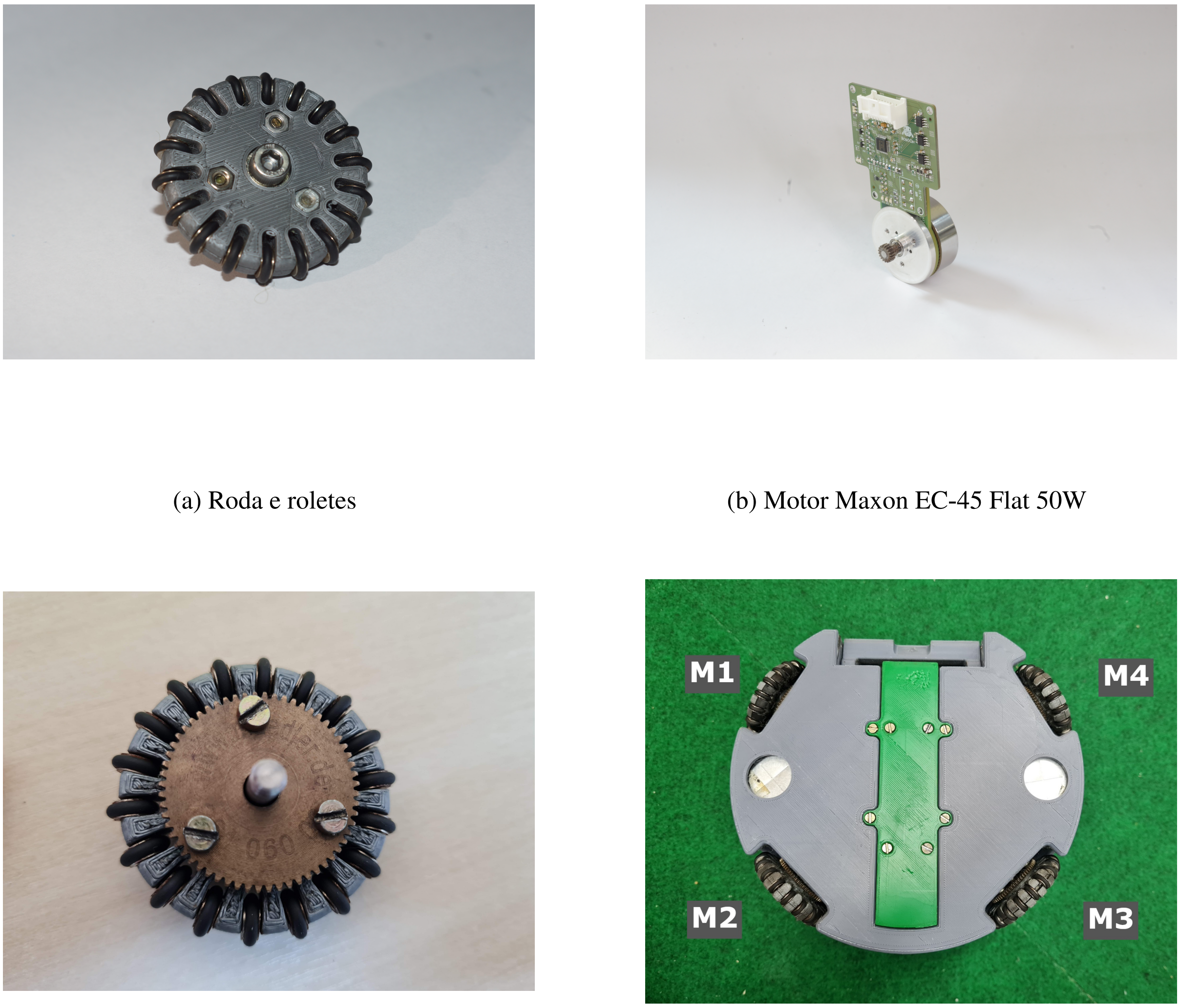


Figura 2: Estrutura geral da Small Size League [Weitzenfeld *et al.*, 2014].

As regras e objetivos do jogo são semelhantes aos do futebol comum, e para alcançar o sucesso em uma partida é primordial que os robôs tenham uma movimentação precisa, otimizada e rápida. Para isso são utilizados robôs omnidirecionais com quatro rodas e motores, que conseguem se movimentar em todas as direções do campo e também rotacionar em torno do seu centro de gravidade [Rojas & Förster, 2006]. As rodas possuem roletes, Figura 3a, e estão dispostas de maneira que o robô é capaz de se movimentar em qualquer direção do plano xy, incluindo movimentos laterais. O movimento de cada roda é gerado por um motor *brushless Maxon EC-45 Flat 50W*, Figura 3b, e o movimento é transferido por uma engrenagem presa à roda, Figura 3c. A Figura 3d apresenta uma visão inferior do robô, assim podemos observar como as rodas são posicionadas, e também a que motor está conectada. A roda frontal direita está conectada ao motor *M1*, a traseira direita ao motor *M2* e assim por diante.



(a) Roda e roletes

(b) Motor Maxon EC-45 Flat 50W

(c) Engrenagem responsável pela transferência do movimento

(d) Posição e referência de cada roda

Figura 3: Componentes diretamente responsáveis pela movimentação de um robô da categoria SSL.

Problemas de montagem, avaria ou até acumulo de sujeiras presentes no campo podem causar danos a movimentação de um robô. Quando isso acontece é exigido uma maior potência dos motores para que assim seja possível executar o caminho planejado, e por isso, muitas vezes tais problemas são impossíveis de se identificar visualmente. Por outro lado, quando os componentes mecânicos do robô são expostos a essa sobrecarga, danos irreversíveis e maiores podem acontecer. Por isso, tais falhas devem ser identificadas o mais rápido possível.

É essencial que o comportamento do robô seja monitorado afim de identificar falhas antes que problemas irreversíveis ocorram. Esse monitoramento precisa ser eficiente e identificar possíveis falhas em tempo real [Verma *et al.*, 2004].

Na literatura, felizmente existem diversas abordagens para detecção de falhas que podem ser aplicadas a sistemas robóticos. Uma das principais técnicas de detecção de falhas é a baseada em dados, que torna possível identificar a presença de falhas em tempo real a partir de modelos estatísticos e dados históricos de comportamentos [Khalastchi & Kalech, 2018]

O principal objetivo deste trabalho consistiu em um estudo sobre o uso de técnicas de inteligência computacional para detecção de falhas na movimentação de robôs omnidirecionais. De forma secundária, foi possível analisar outras abordagens como a comparação visual e analítica para o mesmo fim. Esse estudo será utilizado pelo time de robótica do Centro de Informática, o *RobôCIn*, na construção de um modelo de detecção de falhas *on line* que será utilizado em competições, dado que assim será possível realizar substituição e manutenção no decorrer de partidas, e também evitar que robôs com pequenas falhas, difíceis de identificar apenas pela observação, continuem sobrecarregando os componentes mecânicos e eletrônicos, comprometendo suas atividades.

O presente trabalho está organizado da seguinte maneira: O capítulo 2 contém uma introdução a detecção de falhas, seguindo de conceitos de aprendizagem de máquina como algoritmos, métodos de validação e avaliação. No capítulo 3 é apresentado o processo de definição e rotulação das falhas, assim como todos os passos necessários para a construção e comparação de modelos de detecção de falhas. No capítulo 4 os resultados são apresentados e discutidos. Finalmente, no capítulo 5, a conclusão desse trabalho é apresentada.

2

FUNDAMENTAÇÃO TEÓRICA

O presente capítulo tem como objetivo apresentar conceitos importantes para o entendimento do trabalho. Inicialmente será apresentada a área de estudo de detecção de falhas, e em seguida a área de aprendizagem de máquina, assim como as particularidades dos algoritmos propostos para a resolução do problema. Por fim, métodos de validação e avaliação de classificadores serão descritos.

2.1 DETECÇÃO DE FALHAS

A detecção de falha consiste em um processo de decisão binária que tem como objetivo identificar a presença ou não de falha, e esse processo pode variar de acordo com que tipo de falha se deseja identificar e a abordagem escolhida. As falhas podem ser classificadas como problemas de *hardware*, envolvendo componentes físicos como controladores, sensores, etc; problemas de *software*, envolvendo problemas de algoritmos, como má implementação; e problemas interação, que podem ocorrer por conta de fatores externos como outros robôs ou seres humanos [Khalastchi & Kalech, 2018].

As abordagens para a detecção de falhas podem ser divididas em 3 grupos. As técnicas baseadas em modelo, que consiste na construção de um modelo analítico que possibilita a comparação de informações esperadas com informações observadas. A abordagem baseada em dados, que utiliza dados históricos de comportamentos falhos e livres de falhas para construção de modelos estatísticos de classificação. E por fim, a abordagem baseada em conhecimento é caracterizada por uma combinação da duas abordagens anteriores , quando é desejável também o diagnóstico. Nesse caso uma técnica baseada em dados pode realizar a detecção, e a técnica baseada em modelo provém o diagnóstico [Khalastchi & Kalech, 2018].

Com o passar dos anos, os sistemas robóticos passaram a produzir cada vez mais dados para o monitoramento do seu comportamento. Com um volume tão grande de evidências de comportamento, a abordagem de detecção baseada em dados está se tornando cada vez mais popular [Dai & Gao, 2013]. Essa abordagem utiliza técnicas de aprendizagem de máquina para criar modelos de classificação a partir dos dados gerados pelos robôs, como informações de velocidade, aceleração e posicionamento. Os atributos mais importantes são usados para identificar padrões que caracterizam um comportamento normal ou defeituoso. Quando os modelos são treinados, esses padrões são identificados, e assim conseguem classificar novas entradas (conjuntos de dados ainda não visto pelo modelo).

2.2 APRENDIZAGEM DE MÁQUINA

Aprendizagem de máquina é uma área de pesquisa da ciência da computação que foca no estudo e desenvolvimento de técnicas de reconhecimento de padrões. Os padrões são identificados por modelos matemáticos a partir de amostras de dados que podem ser previamente rotuladas ou não, conhecidas como dados de treinamento, e devem ser capazes de predições sem que tenham sido explicitamente programados para isso [Zhang, 2020].

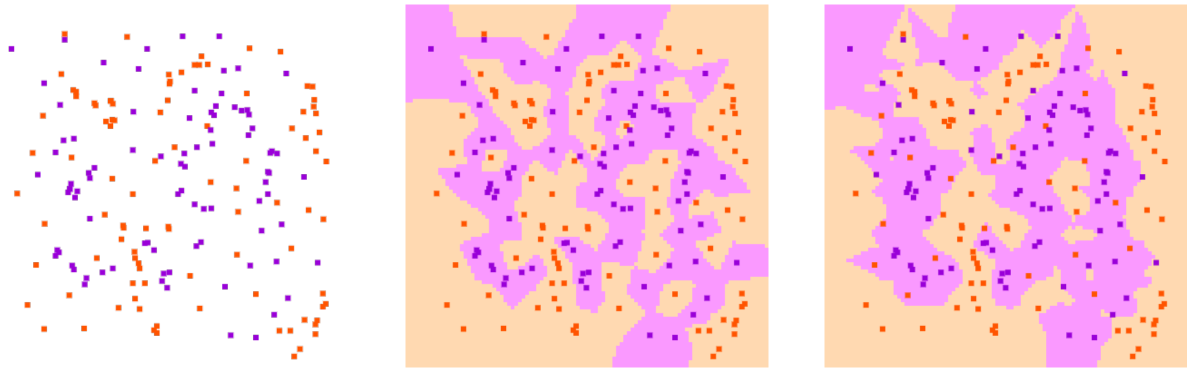
Os problemas de classificação em aprendizagem de máquina podem ser divididos em supervisionados e não supervisionados. Algoritmos não-supervisionados tem como objetivo identificar padrões em conjuntos de dados que não possuem classificação prévia, em outras palavras, essa classe de algoritmos realiza uma tarefa de segmentação apenas utilizando as características das entidades e semelhanças entre elas. Uma aplicação para esse tipo de modelo é a segmentação de clientes com base no comportamento de compra para criação de estratégias de *marketing* assertivas para cara grupo. Por outro lado, algoritmos de aprendizagem supervisionada precisam de exemplos rotulados na fase de treinamento. Um exemplo para essa classe de algoritmos é o problema de previsão do tempo, onde é possível classificar se vai chover ou não a partir de exemplos rotulados e informações histórias de temperatura, umidade, precipitação, etc.

Com base no problema proposto, onde a base de dados disponível será construída a partir falhas simuladas e assim, dados previamente classificados, algumas técnicas de aprendizagem supervisionada serão exemplificadas nas sub sessões seguintes.

2.2.1 K-Vizinhos Mais Próximos

Em inglês, K-Nearest Neighbors (kNN), é uma abordagem de classificação que utiliza a distância (euclidiana, Manhattan, ou outra) entre as entidades de teste (que vão ser classificadas) e as K entidades mais próximas, do conjunto de treinamento, onde se tem conhecimento da classe. O parâmetro K define a quantidade de vizinhos mais próximos que devem ser considerados para a classificação da nova entidade e deve ser definido antes do processo de treinamento [Pinto & Cerquitelli, 2019].

A Figura 4 ilustra o resultado processo de treinamento do algoritmo e como o parâmetro K pode alterar as fronteiras das classes. Na Figura 4a podemos observar um conjunto de dados aleatórios em duas dimensões, com duas classes indicadas por duas cores. Quando definido o K igual a 1, onde apenas o vizinho mais próximo será observado para a classificação de um novo ponto, temos as fronteiras das classes como mostra Figura 4b. Quando o K é definido como 3, as fronteiras das classes se posicionam como mostra Figura 4c. Dessa maneira podemos observar como diferentes configurações podem construir modelos diferentes, e escolher o melhor valor para o parâmetro faz parte do processo de treinamento e análise do modelo.



(a) Distribuição dos dados (b) Fronteiras das classes com k = 1 (c) Fronteiras das classes com k = 3

Figura 4: Ilustração do treinamento de kNN com diferentes valores para k.

2.2.2 Árvore de Decisão

A árvore de decisão é uma abordagem de modelagem preditiva baseada em regras, que é facilmente interpretada se observada como uma técnica hierárquica de divisão de domínio. O processo de treinamento consiste na divisão recursiva do conjunto de dados (nó pai) em dois subconjuntos (nós filhos), onde cada um possui um ganho de informação maior que o nó pai. Em um problema de classificação, o ganho de informação pode ser interpretado como a facilidade de agrupar entidades da mesma classe em um conjunto de dados. Então, na etapa de divisão, é definido qual atributo será utilizado de forma que maximize o ganho de informação. Assim, os nós filhos sempre serão mais fácies de classificar que os seus pais [Suthaharan, 2016].

A Figura 5 ilustra um exemplo de árvore de decisão para diagnóstico de um paciente. Cada elipse representa um teste em um atributo do conjunto de dados do paciente. Os quadrados representam o diagnóstico Monard & Baranauskas [2003].

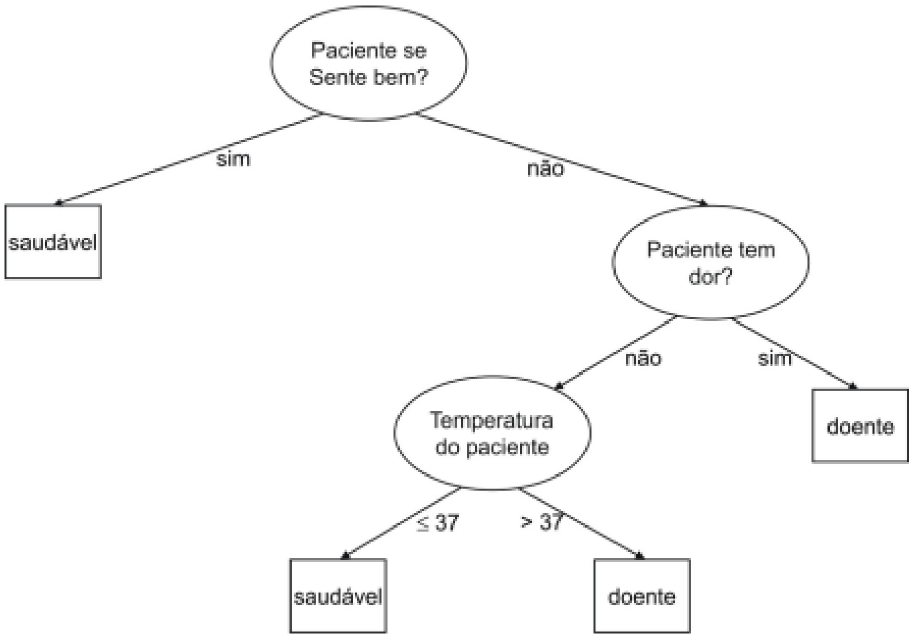
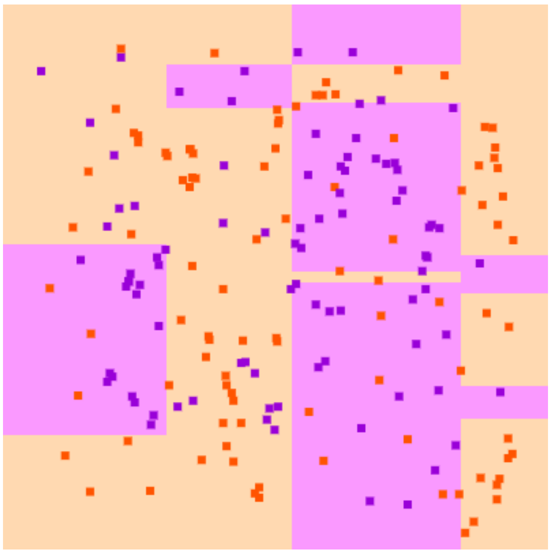
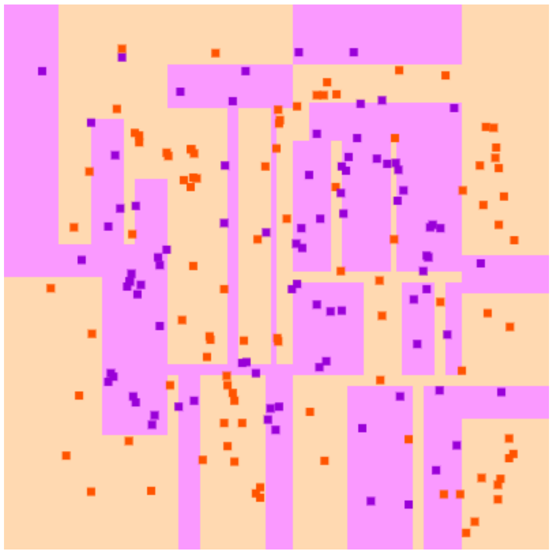


Figura 5: Exemplo de árvore de decisão para diagnóstico de pacientes Monard & Baranauskas [2003].

A profundidade da árvore está diretamente ligada a quantidade de divisões que foram feitas, e o quão específico está cada subconjunto final (ou folha). A Figura 6 mostra exemplos de fronteiras de decisão para o mesmo conjunto de dados, mas como profundidades máximas diferentes. Podemos observar que o modelo com profundidade maior, ilustrado na Figura 6b, consegue criar mais grupos, e ainda mais específicos do que o modelo com profundidade menor, ilustrado na Figura 6a. Note que as áreas dos nós são definidos por retas paralelas aos eixos, isso acontece porque os subconjuntos são definidos a partir de segmentações nos atributos.



(a) Fronteira de decisão para árvore de profundidade máxima igual 5



(b) Fronteira de decisão para árvore de profundidade máxima igual 10

Figura 6: Fronteira das classes para árvore de decisão com diferentes profundidades máximas.

2.2.3 Perceptron Multicamadas

Perceptron Multicamadas, também conhecido como Multilayer Perceptron (MLP), consiste em um sistema interconectado de neurônios (ou nós), ilustrado na Figura 7, onde modelo o representa um mapeamento não linear entre um vetor de entrada e um vetor de saída [Gardner & Dorling, 1998]. A primeira camada recebe os atributos das entidades e é chamada camada de entrada. A camada de saída indica o resultado do classificador. As camadas intermediárias são chamadas de camadas ocultas.

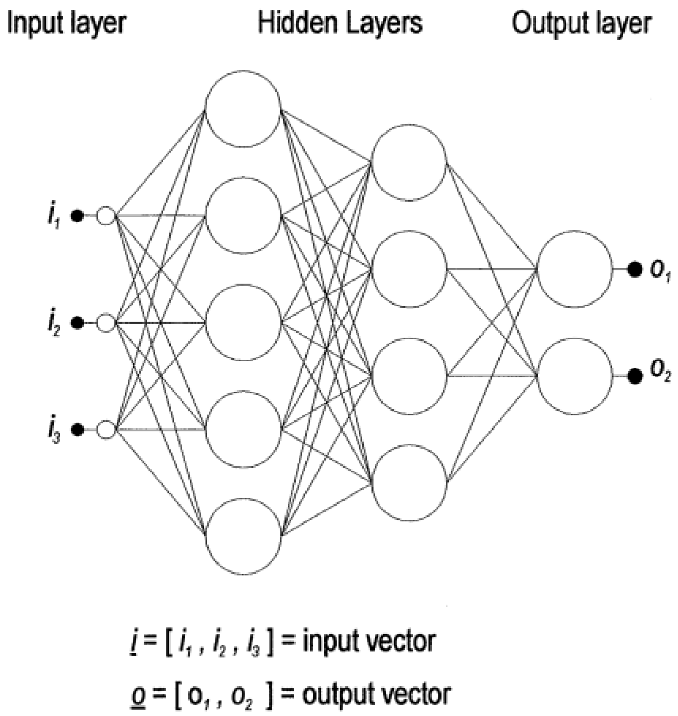


Figura 7: Um perceptron multicamadas com duas camadas escondidas [Gardner & Dorling, 1998].

Os nós, também chamados de neurônios, recebem como entrada o produto entre os sinais de entrada x_i e os pesos w_{ki}, e seu comportamento é ilustrado na Figura 8. A saída de cada neurônio é composta pela saída da função soma (equação 2.1) aplicada a uma função de ativação, que pode ser logística (equação 2.3) , degrau (equação 2.2), linear ou outra.

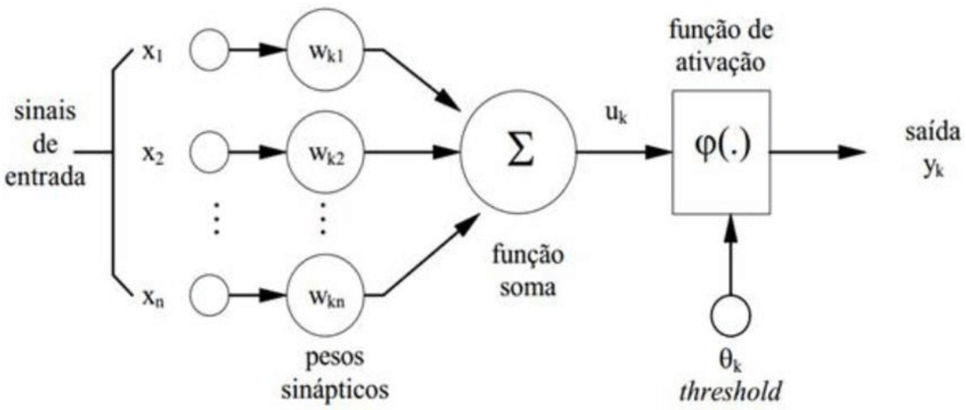


Figura 8: Modelo de um neurônio artificial [Assis *et al.*, 2016].

$$u_k = \sum_{i=1}^{n} W_{ki} * X_i \tag{2.1}$$

$$\varphi(u) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases} \tag{2.2}$$

$$\varphi(u) = \frac{1}{1+e^{-u}} \tag{2.3}$$

A etapa de aprendizagem em redes neurais é feita com o auxilio do algoritmo de retro propagação do erro (*backpropagation*), que recebe esse nome pelo fato do gradiente da função erro caminhar da camada de saída até a camada de entrada ajustando os pesos da rede. Os cálculos parciais do gradiente de uma camada são reutilizados no cálculo do gradiente para a camada anterior.

2.3 MÉTODOS DE VALIDAÇÃO

2.3.1 Validação Cruzada

A validação cruzada é umas das técnicas mais usadas na re amostragem de dados para estimar o verdadeiro erro das predições e ajuste de parâmetros de modelos de aprendizagem de máquina [Berrar, 2019]. O método também é conhecido por *k-fold*, que consiste em dividir o conjunto total de dados em k subconjuntos mutuamente exclusivos do mesmo tamanho, onde um dos sub conjuntos é usado para teste, e os outros *k-1* são usados para treinamento, como mostra a Figura 9. Depois que todos os subconjuntos forem usados como teste, após k iterações, as métricas resultantes são calculadas a partir da média dos valores em cada iteração.

Figura 9: Exemplo de validação cruzada com 5 *folds* onde o conjunto de dados é dividido em 5 subconjuntos e em cada iteração um é utilizado como teste.

2.4 MÉTRICAS DE AVALIAÇÃO DOS MODELOS

Escolher a melhor métrica para a avaliação de modelos de aprendizagem de máquina, é de suma importância, mas muitas vezes torna-se uma tarefa difícil. Para classificadores, diversas métricas podem ser utilizadas e nessa sessão serão apresentadas algumas das mais importantes.

Dependendo da aplicação, analisar apenas uma métrica pode não prover a melhor avaliação da solução, e por isso é necessário o uso de métricas auxiliares. Por exemplo, problemas envolvendo diagnóstico médicos não devem observar apenas a taxa de acerto, mas também a taxa de falsos negativos, para evitar que pacientes com diagnóstico positivo sejam classificados como saudáveis.

As seguintes métricas assumem valores entre 0 e 1 e quanto maior, melhor é o resultado.

2.4.1 Acurácia

Corresponde a quantidade de acertos do modelo, dividido pelo total de registros. A equação 2.4 apresenta como a acurácia pode ser calculada. O fator *Verdadeiros Positivos* (ou *True Positive*) corresponde às predições corretas da classe 1, e o fator *Verdadeiros Negativos* (ou *True Negative*), as predições corretas da classe 0. Caso exista mais classes, o numerador também deve conter fatores que indicam a quantidade predições corretas em cada.

$$Acurácia = \frac{Verdadeiros\ Positivos + Verdadeiros\ Negativos}{Total} \tag{2.4}$$

Quando um modelo apresenta 0.7 de acurácia, podemos dizer que ele acerta 7 entre 10 predições.

2.4.2 Precisão

A precisão indica o percentual de acerto quando o modelo classifica como positivo, ou seja, de todos os dados classificados como positivos, quantos são realmente positivos. Quando a precisão é igual a 1 indica que não houve nenhum falso positivo. A equação 2.5 indica como calcular a métrica.

$$Precisão = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Positivos} \tag{2.5}$$

2.4.3 Revocação

A revocação, também conhecido como *recall*, corresponde a porcentagem de dados classificados como positivos comparado com a quantidade real de positivos existentes. Quando o *recall* é igual a 1 significa que não houve nenhum falso negativo. A equação 2.6 mostra como calcular essa métrica.

$$Revocação = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos} \tag{2.6}$$

3

METODOLOGIA

Nessa capítulo será apresentando o processo para a construção e avaliação do detector de falha. Inicialmente é preciso definir quais problemas serão simulados e posteriormente, identificados como falhas. Essas definições estão descritas na seção 3.1. Na seção 3.2 é definido os movimentos que devem ser executados nos testes para futuras comparações e análises. A construção da base de dados é definida na seção 3.3, e o treinamento dos modelos, na seção 3.4.

3.1 ROTULAÇÃO DE TIPOS DE FALHAS

As falhas de movimentação podem decorrer de diversos problemas envolvendo os componentes mecânicos. Esses problemas podem acontecer devido ao acúmulo de sujeira presente no campo, falta de manutenção, ou montagem errada. Nessa seção serão definidos os problemas mais comuns, os quais devem ser detectados.

3.1.1 Roda travada

Esse problema é o mais crítico, pela possibilidade de danificar o motor, e pode acontecer devido a algum problema de montagem ou acúmulo de sujeira. Quando ocorre, o motor aumenta a potência aplicada para conseguir se mover da maneira planejada, podendo assim queimar componentes eletrônicos do controlador do motor.

3.1.2 Eixo Solto

Podemos dizer que o eixo está solto quando ele perde a fixação com a base do robô, e começa a rotacionar com roda. Isso prejudica a transferência de movimento do motor. Nesse caso, a falha não apresenta nenhuma resistência ao movimento desejado.

3.1.3 Sem Engrenagem

Caracterizamos esse problema quando a engrenagem, anteriormente fixada à roda, se solta. O motor continua o seu trabalho livremente, mas o movimento não é transferido para a roda, o que prejudica o movimento planejado.

As falhas descritas em 3.1.2 e 3.1.3 apresentam dificuldade maior para identificação visual. Como não apresentam resistência ao movimento, os motores perfeitos se esforçam mais para corrigir a ausência do motor falho.

3.2 MOVIMENTAÇÕES AVALIADAS

Para atender diversos propósitos, os comportamentos foram separados em dois grupos, que serão descritos nas próximas subseções.

Todos comportamentos foram executados pelo mesmo robô, com velocidade máxima de 1.5 metro por segundo, apenas variando as rodas quando necessário simular alguma falha.

3.2.1 Movimentação Predeterminada

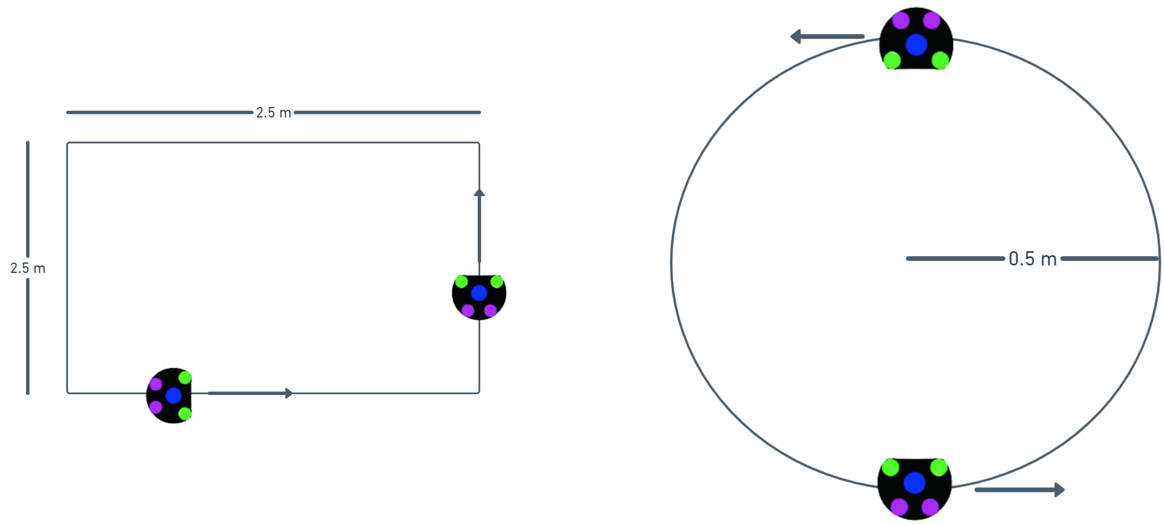
Para possibilitar a comparação e análise visual das trajetórias de robôs com e sem falhas, é necessária a definição de comportamentos predeterminados, ilustrados na Figura 10.

3.2.1.1 Trajetória Quadrada

Foram definidos quatro pontos formando um quadrado de lado 2.5 m, onde o robô se move com a sua frente direcionada para o próximo ponto. Quando o robô alcança o ponto, realiza uma rotação de 90 graus e se desloca para o próximo ponto. Esse comportamento é ilustrado na Figura 10a.

3.2.1.2 Trajetória Circular

O robô realiza um movimento seguindo uma circunferência de raio 0.5 m, no sentido anti-horário e com sua frente sempre direcionada para o centro, como mostra a Figura 10b. Podemos dizer que esse é movimento mais complexo, pois o robô está se deslocando de lado, corrigindo seu ângulo a todo instante, e ainda ajustando o raio da circunferência do seu trajeto se movendo para frente ou para trás.



(a) Trajetória Quadrada (b) Trajetória Circular

Figura 10: Movimentos predeterminados executados para testes.

3.2.2 Movimentação Livre

O comportamento livre visa simular situações reais de uma partida. Limitando o espaço, foi usada a metade do campo (de dimensões 4.5 x 6m). Um robô e uma bola são iniciados em posições aleatórias, e o robô deve se mover até a bola e chutar em gol. Esse movimento foi repetido pelo menos 3 vezes para cada falha em cada uma das roda, ou seja, o agente executou o movimento pelo menos 12 vezes para cada falha mapeada (4 rodas x 3 jogadas).

3.3 BASE DE DADOS

A seguinte seção tem como objetivo descrever o processo de construção e processamento da base de dados que será utilizada para identificação de padrões.

3.3.1 Construção

A construção da base de dados é centralizada no *coach*, *software* localizado no computador responsável por receber e processar dados da visão e do robô, para definir a melhor estratégia para a partida. A Figura 11 descreve a taxa de atualização de cada módulo do sistema. O processo de escrita é realizado 60 vezes por segundo, que corresponde à taxa de atualização do módulo mais lento. Os dados são escritos em tempo real em um arquivo no formato *csv*.

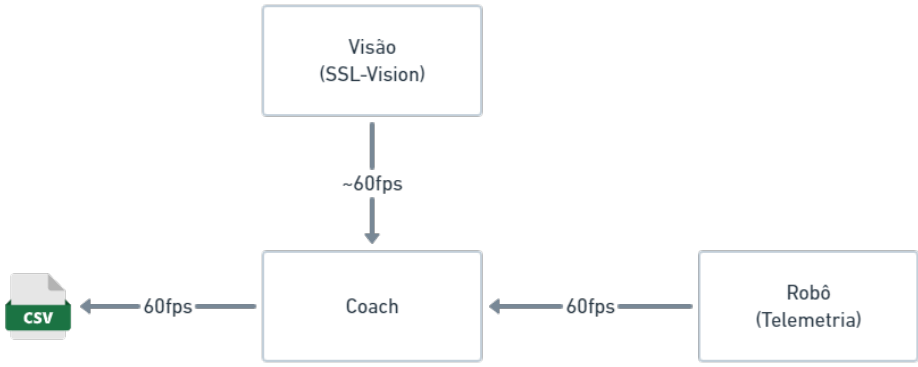


Figura 11: Fluxo de dados para escrita da base de dados.

Foram selecionadas todas as informações presentes no *coach* e relevantes para a movimentação dos robôs. A Tabela 1 apresenta as informações disponíveis e em que módulo a informação é coletada. As informações dos robôs são geradas por sensores embarcados e enviados para o *coach* via rede sem fio. As informações provindas da visão são enviadas pelo *SSL-Vision* [Zickler *et al.*, 2009], via rede cabeada, que processa as imagens das câmeras e identifica a posição e ângulo de cada robô. Em posse das informações citadas anteriormente, o *coach* decide a melhor estratégia e envia a velocidade que cada robô deve executar no momento.

Tabela 1: Variáveis utilizadas para construção da base de dados e suas fontes.

Nº	**Variável**	**Fonte do Dado**
1	Velocidade da roda M1	Robô
2	Velocidade da roda M2	Robô
3	Velocidade da roda M3	Robô
4	Velocidade da roda M4	Robô
5	Posição do robô no eixo x	Visão
6	Posição do robô no eixo y	Visão
7	Ângulo do robô	Visão
8	Velocidade desejável no eixo x	Coach
9	Velocidade desejável no eixo y	Coach
10	Velocidade angular desejável	Coach

Após definir os dados que serão exportados e como, é possível construir a base de dados. Nesse processo foi executado todos os comportamentos descritos na seção 3.2. A Figura 12 ilustra como foi construída a base de dados.

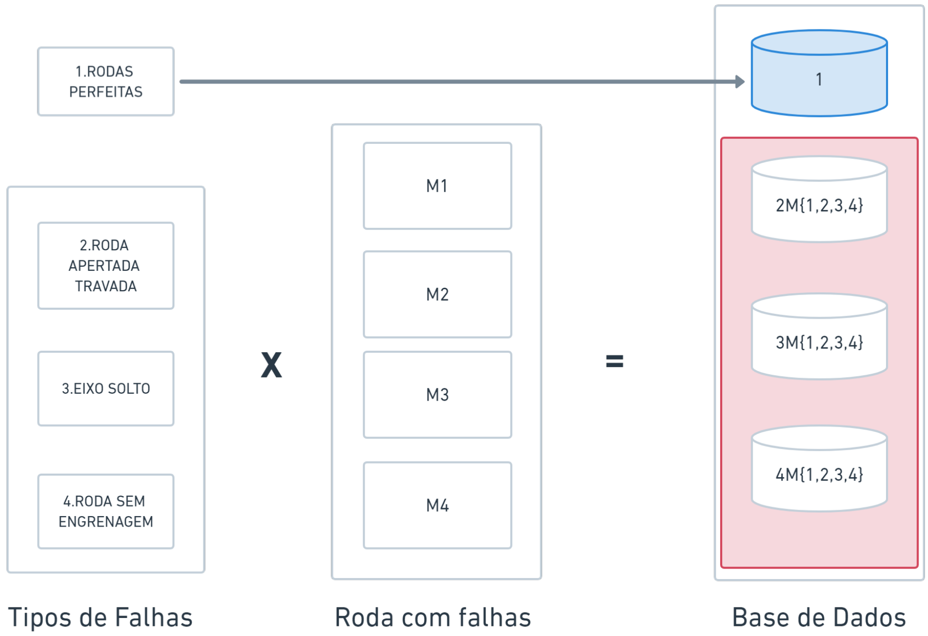


Figura 12: Processo de construção da base de dados usada para detecção de falhas.

3.3.2 Pré-Processamento e Rotulação

Antes da rotulação é necessária uma etapa de limpeza dos dados, para que possíveis erros do processo de construção ou inconsistências dos dados não prejudiquem a etapa de classificação. Foi possível observar instâncias com velocidades nulas para os motores, em todas os comportamentos. Esses casos acontecem quando é iniciado a etapa de exportação antes que velocidades desejáveis sejam enviadas para o robô. O *coach* já calculou as velocidades, mas ainda não as enviou. Como essa não é uma situação comum, essas entradas foram retiradas.

Foram executados comportamentos onde o robô apresenta ou não falhas, e as entradas precisam ser rotuladas para que seja possível utilizar algoritmos de aprendizagem supervisionada. Uma nova coluna foi criada, nomeada de *target*, indicando ou não a presença de falha. A nova coluna recebe o valor **0** quando ausente de falha, e **1** caso contrário. A Tabela 2 ilustra a distribuição de registros por classe.

Tabela 2: Quantidade de registros em cada classe após o processo de rotulação em valor absoluto e percentual.

Classe 0	**Classe 1**
9542 (47.06%)	10735 (52.94%)

3.4 TREINAMENTO DE MODELOS

O conjunto de dados construído na seção anterior foi utilizado para treinamento e avaliação dos classificadores descritos na seção 2.2. Como os classificadores propostos possuem parâmetros que podem contribuir para a aderência do modelo ao problema, diferentes parametrizações foram testadas. A Tabela 3 ilustra os parâmetros testados para cada algoritmo.

Tabela 3: Parâmetros dos classificadores analisados para a detecção de falhas.

Modelos			**Parâmentros**		
Árvore de Decisão	Razão de ganho	Mínimo objetos por folha = 2			
KNN	Distância euclidiana	K = 1, 3, 5			
MLP	Camadas escondidas = 2, 8, 10	Taxa de aprendizagem = 0.3	Tamanho do lote = 100	Épocas = 500	Função de ativação = Sigmóide

Para todos modelos, com diferentes parâmetros, foi aplicada a validação cruzada, com 10 *folders*. E para garantir que a divisão em subconjuntos no processo de validação não enviese o resultado, todo o processo é repetido 10 vezes e as métricas definidas na seção 2.4 serão geradas a partir da média das todas iterações.

4

RESULTADOS

Essa capítulo apresenta os resultados comparativos das abordagens propostas. Como descrito no capítulo 3, a metodologia propõe comparar diversas abordagens para detecção de falhas de movimentação. O capítulo irá apresentar inicialmente uma abordagem visual, que opõe as trajetórias de comportamentos com a presença ou não de falhas. Posteriormente, uma abordagem analítica é apresentada, onde foi experimentada a comparação entre as velocidades instantâneas e desejadas de cada robô. Por fim, os resultados de modelos de classificação são apresentados e comparados.

4.1 COMPARAÇÃO VISUAL

A Figura 13 apresenta a trajetória de um robô perfeito, já a Figura 14 apresenta as trajetórias com diferentes tipos de falhas e sem falhas. Aqui podemos constatar o grau de dificuldade para indicar qual caminho pertence a um comportamento perfeito, e qual não. Tamanha dificuldade se deve ao fato da alta taxa de atualização da visão e do *coach*, tornando possível a atuação para correção de trajetórias e ângulos incomuns, exigindo mais potência dos outros motores.

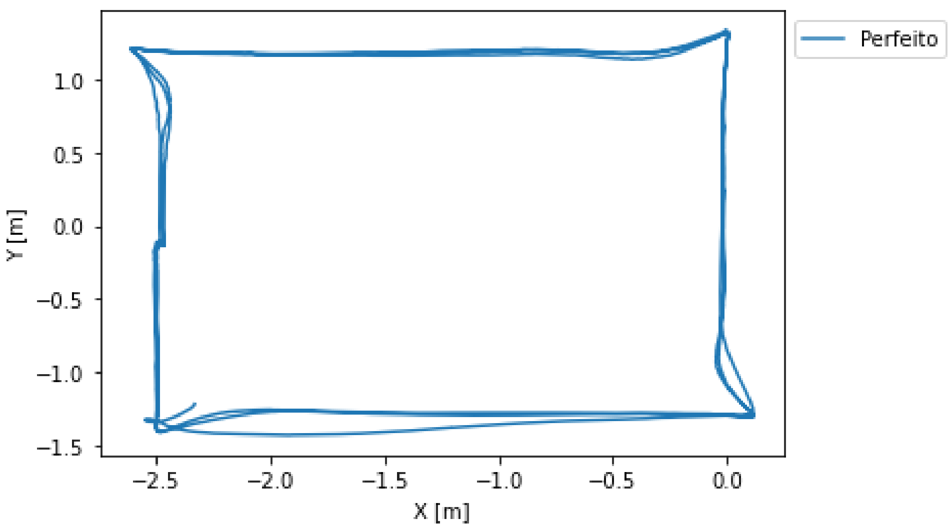


Figura 13: Trajetória quadrada descrita na seção 3.2.1.1 executada por robôs perfeito.

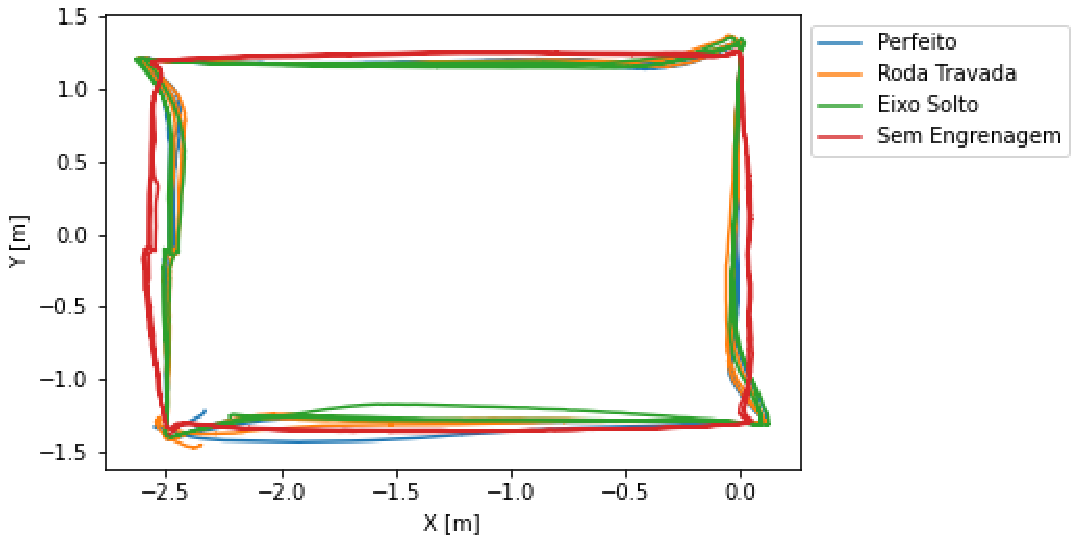


Figura 14: Trajetória quadrada executada por robôs perfeitos e com diferentes falhas.

Por se tratar de um movimento complexo e exigir mais das quatro rodas, a análise das trajetórias circulares tornou possível identificar ao menos uma das falhas. A Figura 15 ilustra o caminho executado por um robô sem falhas. Já a Figura 16 apresenta as trajetórias de robôs em diversas configurações, e foi possível identificar que o movimento do robô sem engrenagem se difere dos demais. A trajetória com raio menor do que o esperado, ocorre pela falta de movimento de um dos motores, o que dificulta a rotação do robô, fazendo com que o *coach* tente corrigir a angulação continuamente.

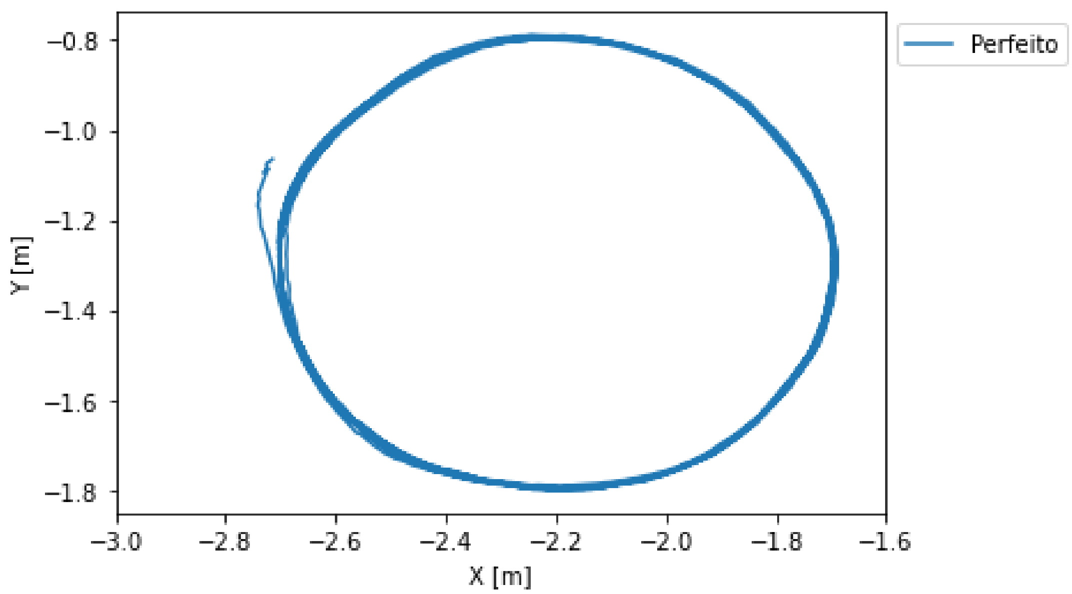


Figura 15: Trajetória circular descrita na seção 3.2.1.2 executada por um robôs perfeito.

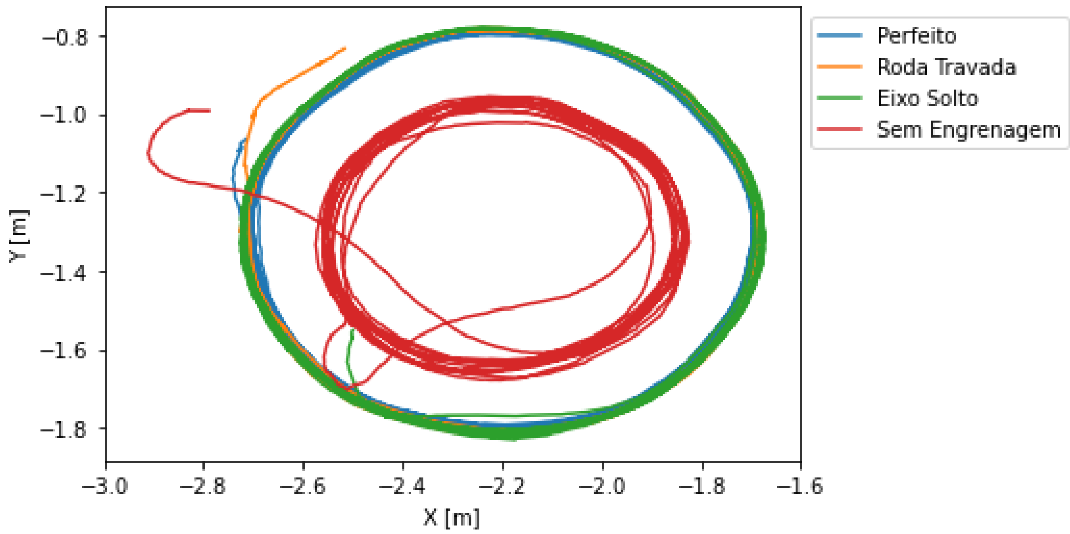


Figura 16: Trajetória circular executada por robôs perfeitos e com diferentes falhas.

4.2 ABORDAGEM ANALÍTICA

A partir das velocidades instantâneas em cada roda, é possível calcular as velocidades do robô em cada eixo e também a velocidade angular. Se compararmos essas velocidades com as velocidades enviadas pelo *coach*, podemos calcular o erro e assim tentar identificar falhas.

Com a ajuda dos diagramas de caixa, *box plots*, podemos comparar visualmente as distribuições dos erros. As caixas representam os intervalos entre o 1º e 3º quartis (responsáveis por 50% dos dados), ou seja, os dados mais frequentes. A barra dentro da caixa representa a mediana. E os pontos representam valores atípicos em comparação as distribuições, *outliers*.

A Figura 17 ilustra a distribuição dos erros das velocidades no eixo x com o robô em diferentes configurações de falhas. Podemos observar que os limites das caixas são semelhantes, assim como suas medianas. A falha de Eixo Solto, possui um intervalo maior composto por *outliers*, mas que não podem ser utilizados como regra de classificação porque são eventos raros.

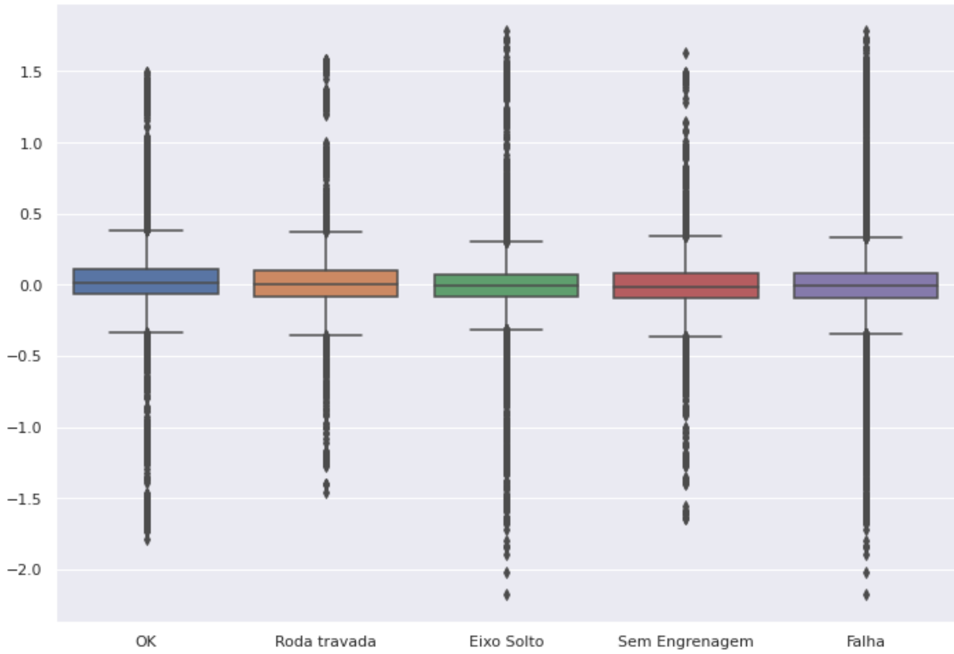


Figura 17: Erro da velocidade no eixo X em metros por segundo das diferentes configurações do robô.

Tabela 4: Diferença entre velocidade instantânea do robô e a velocidade desejada no eixo X em metros por segundo.

	Sem Falhas	**Roda Travada**	**Eixo Solto**	**Sem Engrenagem**	**Todas Falhas**
Média	-0.007778	0.054686	-0.003250	0.003859	0.010793
Min	-1.787410	-1.454010	-2.173850	-1.640390	-2.173850
25%	-0.068291	-0.083688	-0.081283	-0.097920	-0.090753
50%	0.012100	0.002730	-0.003833	-0.012170	-0.005247
75%	0.112740	0.098159	0.072770	0.078000	0.079409
Max	1.497000	1.581901	1.783602	1.627000	1.783602

Com a Tabela 4 é possível comparar numericamente e concluir que apenas com os erros não é possível detectar erros. O mesmo comportamento pode ser visto para o eixo Y e velocidade angular, Figura 18 (Tabela 5) e Figura 19 (Tabela 6), respectivamente.

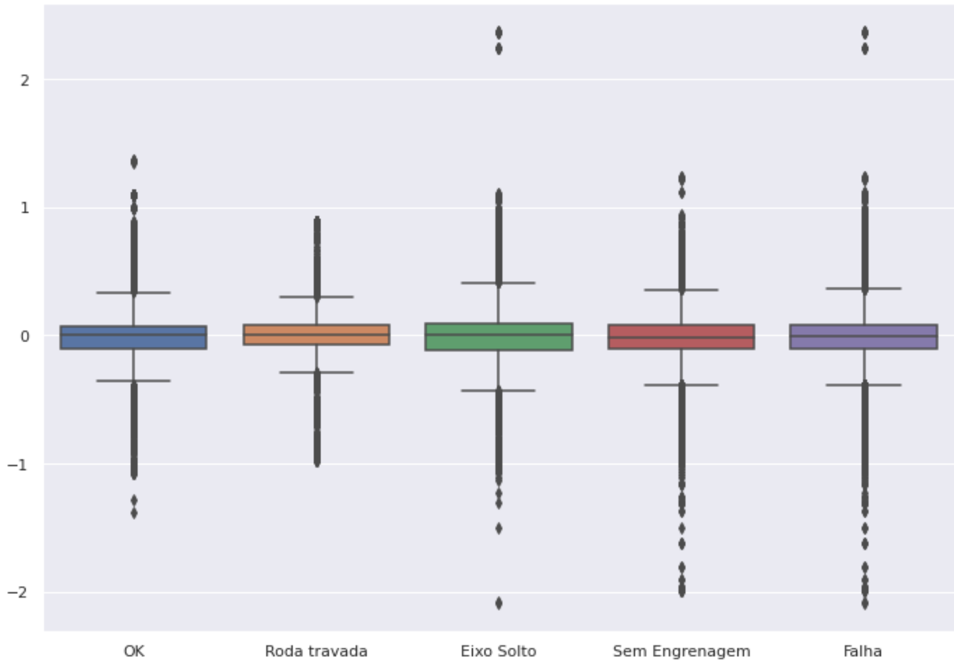


Figura 18: Erro da velocidade no eixo Y em metros por segundo das diferentes configurações do robô.

Tabela 5: Diferença entre velocidade instantânea do robô e a velocidade desejada no eixo Y em metros por segundo.

	Sem Falhas	**Roda Travada**	**Eixo Solto**	**Sem Engrenagem**	**Todas Falhas**
Média	-0.013264	-0.015071	-0.024106	-0.036227	-0.027731
Min	-1.377000	-0.974010	-2.088970	-1.988443	-2.088970
25%	-0.068291	-0.083688	-0.081283	-0.097920	-0.090753
50%	0.000017	0.008253	0.001826	-0.016320	-0.004146
75%	0.072707	0.078404	0.095952	0.080057	0.084544
Max	1.367502	0.888144	2.370640	1.240811	2.370640

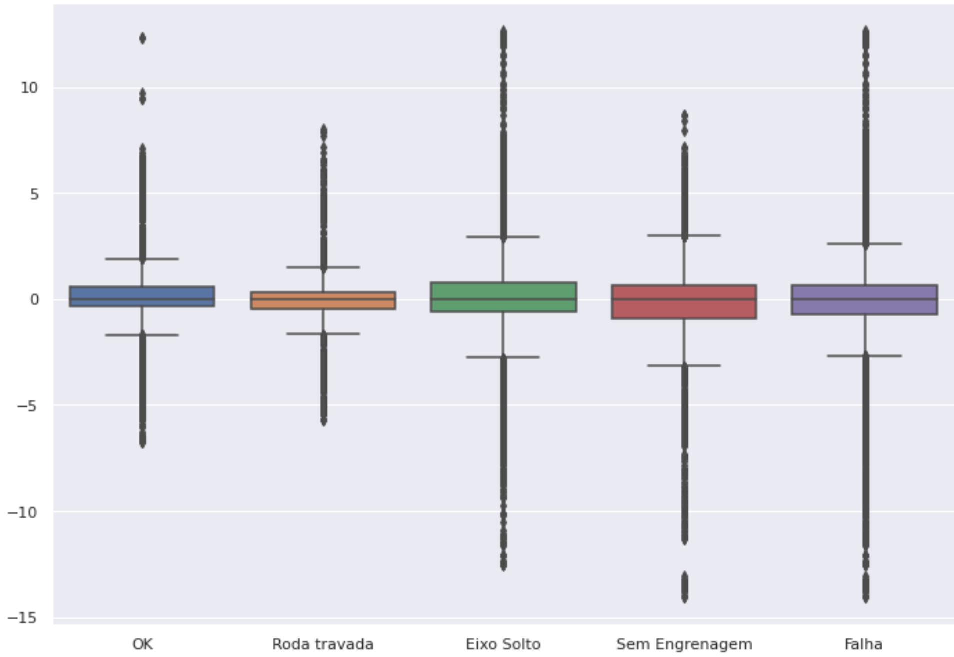


Figura 19: Erro da velocidade angular em radiano por segundo das diferentes configurações do robô.

Tabela 6: Diferença entre velocidade angular instantânea do robô e a velocidade desejada em radianos por segundo.

	Sem Falhas	**Roda Travada**	**Eixo Solto**	**Sem Engrenagem**	**Todas Falhas**
Média	0.229352	-0.074064	0.205560	-0.299028	-0.068947
Min	-6.723290	-5.717020	-12.516090	-14.035200	-14.035200
25%	-0.348105	-0.476094	-0.615830	-0.887136	-0.707265
50%	0.015023	-0.019646	-0.005738	-0.008930	-0.010660
75%	0.560423	0.319755	0.810310	0.655453	0.627652
Max	12.323600	8.001220	12.650340	8.654650	12.650340

A maior dificuldade na tentativa de detectar falhas por essa abordagem pode ocorrer pela falta de um componente temporal, visto que existe um *delay* até que o robô consiga atingir a velocidade desejada pelo *coach*.

4.3 UTILIZANDO TÉCNICAS DE APRENDIZAGEM DE MÁQUINA

Visto que as duas abordagens anteriores não são suficientes para a detecção de falhas de movimentação, é preciso partir para uma abordagem mais complexa, onde é possível identificar padrões mais difíceis de observar.

A partir dos dados obtidos na seção 3.3 foi possível treinar e avaliar os modelos de K-vizinhos mais próximos, árvores de decisão e perceptrons multicamada de acordo com acurácia, precisão e revocação, citadas na seção 2.4. As métricas podem ser comparadas com a ajuda da Tabela 7.

Tabela 7: Métricas dos modelos de detecção de falhas.

	KNN (k = 1)	**KNN (k = 3)**	**KNN (k = 5)**	**Árvore de Decisão**	**MLP (h = 2)**	**MLP (h = 8)**	**MLP (h = 10)**
Acurácia	0.9859	0.9736	0.9637	0.9783	0.7484	0.8441	0.8628
Precisão	0.9852	0.9720	0.9639	0.9783	0.8052	0.8829	0.9010
Revocação	0.9849	0.9720	0.9587	0.9756	0.6261	0.7729	0.9776

Ao analisar os resultados, é possível observar que os classificadores de K-vizinhos mais próximos conseguem se adequar bem ao problema. O modelo com K=1, apresenta acurácia de 0.9859, precisão de 0.9852 e revocação 0.9849. O classificador resultante da árvore de decisão também apresenta um resultado promissor. E por fim, os classificadores perceptron multicamadas, não atingiram resultados satisfatórios com os parâmetros testados.

5 CONCLUSÃO

Robôs móveis autônomos tem sido utilizados cada vez mais na indústria pela sua capacidade de iteração com o ambiente e tomada de decisão em tempo real. Sua principal característica é a movimentação baseada em visão, sensores e inteligência artificial, proporcionando assim, a aplicação em novos contextos como fábricas, armazéns e hospitais.

A copa do mundo de robôs, RoboCup, foi criada para fomentar o estudo de robótica de acordo com diversos desafios em forma de competição entre pesquisadores. Uma das principais categorias de futebol de robôs autônomos é chamada de Small Size League (SSL), que se caracteriza por dois times de 6 jogadores de diâmetro máximo 180mm disputando uma partida em um campo de *(9 x 6) m*.

A movimentação dos robôs em campo, depende de um sistema mecânico complexo e com múltiplos componentes. Quando um ou mais desses componentes apresenta uma falha, pode comprometer todo o sistema, e caso não seja identificado em tempo hábil, pode culminar em problemas irreversíveis como inutilidade de motores. Por isso é essencial que o comportamento do sistema seja monitorada afim de identificar falhas o mais rápido possível.

Esse trabalho apresentou um estudo comparativo de abordagens para identificação falhas de movimentação de robôs omnidirecionais em tempo real. Partindo de uma abordagem inicial onde as trajetórias de robôs perfeitos e com falhas são comparadas. Posteriormente, uma abordagem analítica que compara a velocidade instantânea do robô com a velocidade desejável para completar a trajetória previamente definida. E por fim, uma abordagem baseada em dados que constrói modelos estatísticos a partir de comportamentos históricos para indicar a presença ou não de falhas.

Com os resultados apresentados, foi possível concluir que a detecção de falhas a partir de uma abordagem visual não é eficiente. Assim como a abordagem analítica, onde a comparação entre as velocidades desejadas e observadas não se mostra suficiente para a conclusão da ocorrência de falhas. Possivelmente pela falta de uma componente temporal, visto que é necessário um intervalo para que a velocidade desejada seja atingida. Por outro lado, a abordagem baseada em dados, se mostrou promissora com altos valores para acurácia.

O modelo da família *K vizinhos mais próximos*, com $K = 1$, classificou corretamente a presença ou não de falhas em 98.59% dos casos. O classificador apresentou uma precisão de

0.9852, mostrando que consegue avaliar muito bem quando a amostra em questão tem falha. O alto índice de precisão garante uma baixa taxa de falso positivos, que em situação de jogo diminui as chances de retirada de robôs no decorrer de partidas para manutenção sem que exista alguma falha. O modelo apresentou também o maior resultado de revocação, 0.9849, que mostra a capacidade identificar corretamente 98.49% amostras com falha. Esse valor indica uma baixa taxa de falso negativo, aonde o erro está presente e o modelo classifica como comportamento perfeito.

Como base nesse estudo, o time de robótica do Centro de Informática, *RobôCIn*, irá utilizar o modelo resultante como mais adequado para detecção de falhas em tempo real, em situações de treino e partidas.

Como trabalhos futuros, indico o estudo de séries temporais e aprendizagem não supervisionada. Séries temporais pode se mostrar uma boa abordagem para a comparação analítica. Já modelos não supervisionados adicionam a vantagem de detecção de falhas de difíceis reprodução, e até identificação de falhas não esperadas.

REFERÊNCIAS

Assis, R., Cossío, J. B., Cunha, A., Pitombo, C., & Fernandes Jr, J. (2016). Uso de redes neurais artificiais para o desenvolvimento de modelos de previsão da condição de pavimentos de aeroportos.

Berrar, D. (2019). Cross-validation. In Ranganathan, S., Gribskov, M., Nakai, K., & Schönbach, C., editors, *Encyclopedia of Bioinformatics and Computational Biology*, 542–545. Academic Press, Oxford.

Dai, X. & Gao, Z. (2013). From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis. *IEEE Transactions on Industrial Informatics*, 9(4):2226–2238.

Fragapane, G., de Koster, R., Sgarbossa, F., & Strandhagen, J. O. (2021). Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European Journal of Operational Research*.

Gardner, M. W. & Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636.

Khalastchi, E. & Kalech, M. (2018). On fault detection and diagnosis in robotic systems. *ACM Computing Surveys (CSUR)*, 51(1):1–24.

Monard, M. C. & Baranauskas, J. A. (2003). Indução de regras e árvores de decisão. *Sistemas Inteligentes-Fundamentos e Aplicações*, 1:115–139.

Pinto, R. & Cerquitelli, T. (2019). Robot fault detection and remaining life estimation for predictive maintenance. *Procedia Computer Science*, 151:709–716.

Rojas, R. & Förster, A. G. (2006). Holonomic control of a robot with an omnidirectional drive. *KI-Künstliche Intelligenz*, 20(2):12–17.

Suthaharan, S. (2016). Decision tree learning. In *Machine Learning Models and Algorithms for Big Data Classification*, 237–269. Springer.

Verma, V., Gordon, G., Simmons, R., & Thrun, S. (2004). Real-time fault diagnosis [robot fault diagnosis]. *IEEE Robotics & Automation Magazine*, 11(2):56–66.

Weitzenfeld, A., Biswas, J., Akar, M., & Sukvichai, K. (2014). Robocup small-size league: Past, present and future. In *Robot Soccer World Cup*, 611–623.

Zhang, X.-D. (2020). Machine learning. In *A Matrix Algebra Approach to Artificial Intelligence*, 223–440. Springer.

Zickler, S., Laue, T., Birbach, O., Wongphati, M., & Veloso, M. (2009). Ssl-vision: The shared vision system for the robocup small size league. In *Robot Soccer World Cup*, 425–436.