



Lucas da Silva Carvalho

**Definição de trajetória robótica usando visão computacional: um modelo
baseado por identificação de marcadores de sinalização com aplicação em
Teoria dos Grafos**

CAMPINAS
2025

Lucas da Silva Carvalho

Definição de trajetória robótica usando visão computacional: um modelo baseado por identificação de marcadores de sinalização com aplicação em Teoria dos Grafos

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do diploma do Curso Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia Campus Campinas.

Orientador: Prof. Dr. Glauber da Rocha Balthazar.

CAMPINAS
2025

Ficha Catalográfica

Instituto Federal de São Paulo – Campus Campinas

Biblioteca – Pedro Augusto Pinheiro Fantinatti

Rosangela Gomes – CRB 8/8461

Carvalho, Lucas da Silva.

Definição de trajetória robótica usando visão computacional: um modelo baseado por identificação de marcadores de sinalização com aplicação em teoria dos grafos.

/ Lucas da Silva Carvalho. – Campinas, SP: [s.n.], 2025.

45 f. : il.

Orientador: Glauber da Rocha Balthazar.

Trabalho de Conclusão de Curso (graduação) – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Campus Campinas. Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, 2025.

1. Visão computacional. 2. Yolo v8. 3. Robótica. 4. Algoritmo de Dijkstra. 5. Desenvolvimento de jogos. I. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Campus Campinas. Curso de Tecnologia em Análise e Desenvolvimento de Sistemas. II. Título.

ATA N.º 5/2025 - TADS-CMP/DAE-CMP/DRG/CMP/IFSP

Ata de Defesa de Trabalho de Conclusão de Curso - Graduação

Na presente data, realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intitulado **Definição de trajetória robótica usando visão computacional: um modelo baseado por identificação de marcadores de sinalização com aplicação em Teoria dos Grafos**, apresentado(a) pelo(a) estudante Lucas da Silva Carvalho do Curso **SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS** (Campus Campinas). Os trabalhos foram iniciados às 19:00 pelo(a) Professor(a) presidente da banca examinadora, constituída pelos seguintes membros:

Membros	Instituição	Presença (Sim/Não)
Glauber da Rocha Balthazar	IFSP	Sim
Jose Américo dos Santos Mendonça	IFSP	Sim
Marcos Brandao Rios	IFSP	Sim

Observações:

A banca examinadora, tendo terminado a apresentação do conteúdo da monografia, passou à arguição do(a) candidato(a). Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final sobre o trabalho apresentado pelo(a) estudante, tendo sido atribuído o seguinte resultado:

Aprovado(a) Reprovado(a)

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, para constar, eu lavrei a presente ata que assino em nome dos demais membros da banca examinadora.

Campus Campinas, 20 de maio de 2025

Documento assinado eletronicamente por:

- **Glauber da Rocha Balthazar, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 20/05/2025 13:36:09.
- **Jose Americo dos Santos Mendonça, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 20/05/2025 15:44:52.
- **Marcos Brandao Rios, PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO**, em 22/05/2025 17:21:28.

Este documento foi emitido pelo SUAP em 20/05/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse <https://suap.ifsp.edu.br/autenticar-documento/> e forneça os dados abaixo:

Código Verificador: 953385

Código de Autenticação: af39a9b08b



*Dedico este trabalho aos meus familiares,
colegas de classe e professores do Instituto
que me auxiliaram nessa longa jornada.*

AGRADECIMENTOS

Agradeço à minha família, que sempre me apoiou para que eu chegasse até aqui.

Agradeço ao meu orientador que me auxiliou muito a resolver as dificuldades encontradas ao longo do projeto.

Agradeço aos meus colegas que sempre me proporcionaram ideias e soluções.

Agradeço a todos os professores e servidores do IFSP Campus Campinas, que contribuíram direta e indiretamente para a conclusão deste trabalho.

"O olho que observa influencia o que vê".

Frank Herbert

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema de navegação robótica autônoma baseado em visão computacional, utilizando marcadores de sinalização como referência para tomada de decisão. A pesquisa propõe uma solução para o desafio da navegabilidade robótica autônoma através da combinação de técnicas de *deep learning* para reconhecimento visual e algoritmos de Teoria dos Grafos para otimização de trajetórias. O sistema implementa um modelo de reconhecimento visual utilizando YOLO v8, treinado com um *dataset* personalizado de placas de sinalização desenvolvidas para este projeto. O sistema utiliza uma câmera, processando em tempo real imagens para detectar marcadores como "Reto", "Direita", "Esquerda" e "Pare". A tomada de decisão sobre a trajetória é realizada através do algoritmo de *Dijkstra*, que determina o caminho mais eficiente entre os pontos com base em um grafo dinâmico. A metodologia incluiu o desenvolvimento de marcadores visuais padronizados, criação de um *dataset* personalizado, treinamento no modelo YOLO v8, implementação do sistema em Python e testes em quatro cenários distintos.

Palavras-chave: Visão Computacional, YOLO v8, Robótica, Algoritmo de Dijkstra.

ABSTRACT

This work presents the development of an autonomous robotic navigation system based on computer vision, using signaling markers as references for decision-making. The research proposes a solution to the challenge of autonomous robotic navigability through the combination of deep learning techniques for visual recognition and Graph Theory algorithms for trajectory optimization. The system implements a visual recognition model using YOLO v8, trained with a custom dataset of signage plates developed specifically for this project. The system uses a camera to process images in real time, detecting markers such as "Forward", "Right", "Left", and "Stop". Decision-making regarding the trajectory is performed using Dijkstra's algorithm, which determines the most efficient path between points based on a dynamic graph. The methodology included the development of standardized visual markers, creation of a custom dataset, training of the YOLO v8 model, implementation of the system in Python, and testing in four distinct scenarios.

Keywords: Computer Vision, YOLO v8, Robotic, Dijkstra's Algorithm.

LISTA DE FIGURAS

Figura 1 – Gráfico Python.....	18
Figura 2 – Modelo visual (Direita).....	21
Figura 3 – Dataset customizado extraído do site roboflow.....	22
Figura 4 – Gráfico Python.....	23
Figura 5 – Asus Tinker Board.....	23
Figura 6 – Exemplo de detecção em tempo real.....	24
Figura 7 – Funcionamento do sistema com aplicação do algoritmo de Dijkstra.....	25
Figura 8 – Distribuição de instâncias por classe e visualização de características físicas dos objetos no conjunto de dados.....	28
Figura 9 – Curva Recall-Confidence do modelo YOLOv8 para as quatro classes.....	29
Figura 10 – Curva F1-Confidence do modelo YOLOv8 para detecção de sinais de trânsito...	30
Figura 11 – Detecção em cenário iluminado.....	35
Figura 12 – Detecção em cenário escuro.....	36
Figura 13 – Taxa de acerto x Confiança em caminhos curtos médios e longos.....	37
Figura 14 – Taxa de acerto x Distância em distâncias de 100cm e 200cm.....	38
Figura 15 – Taxa de confiança x Distância em distâncias de 100cm e 200cm.....	38
Figura 16 – Taxa de acerto x Confiança em caminhos curtos médios e longos.....	39
Figura 17 – Taxa de confiança x Cenário.....	40

SUMÁRIO

1 INTRODUÇÃO.....	11
1.1 Motivação.....	11
1.2 Justificativa/hipótese.....	12
1.3 Objetivo Geral.....	13
1.3.1 <i>Objetivos específicos</i>	13
2 FUNDAMENTAÇÃO TEÓRICA.....	14
2.1 Robótica Móvel.....	14
2.2 Machine learning e Deep Learning.....	15
2.3 Visão Computacional.....	15
2.4 Yolo V8.....	16
2.5 Python.....	17
2.6 Teoria dos grafos.....	18
3 METODOLOGIA.....	20
3.1 Modelagem das placas de sinalização (marcadores de sinalização).....	20
3.2 Obtenção do dataset para treinamento.....	20
3.3 Integração e Implementação.....	21
3.4 Teste de acurácia e reconhecimento.....	23
3.5 Aplicação prática.....	23
3.6 Aplicação na modelagem matemática de Grafos.....	24
4 TREINAMENTO.....	25
4.1 Análise do treinamento.....	25
5 ESTRUTURA DO ALGORITMO.....	29
5.1 Movimentação com GPIO.....	29
5.2 Detecção utilizando Yolo.....	30
5.3 Lógica de navegação com base no algoritmo de Dijkstra.....	31
6 RESULTADOS.....	33
6.1 Cenários.....	33
6.2 Análise estatística.....	34
6.3 Análise inferencial.....	38
7 CONCLUSÃO E TRABALHOS FUTUROS.....	39
REFERÊNCIAS.....	40

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Os robôs desempenham um papel essencial na indústria moderna, especialmente na Indústria 4.0. Desde 2004, o número de robôs industriais multifuncionais na Europa quase dobrou (Klenert et al., 2022). Com a informatização e métodos de produção autônomos, esses robôs tornaram o ambiente de trabalho mais seguro, flexível e colaborativo permitindo sua integração em espaços de trabalho com humanos de forma produtiva e econômica (Bahrin et al., 2016).

O tópico da robótica móvel desencadeia uma possibilidade de estudos como autonomia, navegação, otimização, aprendizagem e cooperação, sendo todos estes estudos desafios para os desenvolvedores (Jahn et al., 2020). Dentre estas possibilidades, a visão computacional permite que robôs e humanos compartilhem informações visuais ajudando robôs a entenderem e interagirem com o mundo de forma muito próxima como os humanos fazem. A área avançou rapidamente entre 2010 e 2020 impulsionada pelo *deep learning* e pela popularização de sensores de profundidade e cor (Red, Green e Blue [RGB-D]) como o Microsoft Kinect, que possibilitou novas aplicações como reconstrução 3D em tempo real (Santos e Khoshelham, 2015). Isso otimizou a utilização de técnicas como detecção de objetos, classificação de imagens e segmentação em nível de pixel para localização e identificação de objetos em cenas. No contexto da interação humano-robô a visão computacional permite detectar e analisar humanos em cenas visuais por meio de métodos como detecção facial, estimativa de pose e rastreamento de movimento (Robinson et al., 2023).

Segundo Wang e Liao (2024) a detecção de objetos é uma tarefa fundamental em visão computacional pois apoia uma variedade de tarefas subsequentes como segmentação de instâncias, rastreamento de múltiplos objetos, análise e reconhecimento de comportamento e reconhecimento facial. Devido a sua relevância a detecção de objetos tem sido tema de pesquisa popular nos últimos anos (Zou et al., 2023; Cazzato et al., 2020; Gupta et al., 2020). Além disso, com a crescente utilização de Internet of Things (IoT), dispositivos móveis e drones a capacidade de realizar detecção de objetos em tempo real tornou-se essencial para várias aplicações do mundo real incluindo direção autônoma, robôs industriais, autenticação de identidade, saúde inteligente e vigilância visual (Suo et al., 2023; Kaur et al., 2023; Xiao et al., 2023). Além disso, o uso da detecção de objetos por meio da

visão computacional é uma ferramenta muito útil para a naveabilidade dos robôs, sendo imprescindível para determinar sua localização dentro de um ambiente (Morar et al., 2020; Tang et al., 2020).

Neste contexto surge a possibilidade da aplicação de um sistema de navegação robótico utilizando a tecnologia de visão computacional que seja orientado por marcadores (placas indicadoras de sinalização como: “siga em frente”, “vire à direita”, “vire à esquerda” etc.). Diversos trabalhos (Luo et al., 2024; Jayasinghe et al., 2022; Marques et al., 2022) citam a utilização de marcadores no trânsito como norteadoras para orientação de modelos robóticos autônomos. Neste trabalho pretende-se explorar como essa tecnologia pode ser utilizada na naveabilidade autônoma utilizando modelos matemáticos de Teoria dos Grafos associados com identificação visual de marcadores de sinalização como motor de tomada de decisão.

1.2 JUSTIFICATIVA/HIPÓTESE

O uso de robôs na Indústria 4.0 não é mais novidade, e a necessidade de automação para tarefas repetitivas e complexas sempre esteve entre as prioridades da indústria moderna (Soori et al., 2024; William et al., 2024). No entanto, com a evolução das tecnologias de visão computacional e sua integração crescente com a robótica, surge uma nova abordagem para a execução de tarefas que antes exigiam sensores tradicionais, delimitadores físicos ou supervisão humana constante (Che et al., 2024; Dudek e Jenkin 2024; Robinson et al., 2023).

A popularização da visão computacional tem expandido significativamente as capacidades dos robôs industriais, permitindo que eles “enxerguem” e interpretem o ambiente em tempo real. Esta tecnologia tornou-se essencial para facilitar a navegação autônoma, reconhecimento de padrões e precisão em ambientes dinâmicos e com maior grau de complexidade (Dudek e Jenkin 2024). Quando integrada com robôs, a visão computacional oferece a capacidade de identificar e seguir trajetórias, detectar obstáculos e reagir a mudanças no ambiente, possibilitando um deslocamento eficiente e seguro (Robinson et al., 2023). Diversas pesquisas recentes (Morar et al., 2020; Tang et al., 2020) se orientam por meio da visão computacional para a identificação de objetos e obstáculos, definindo assim trajetórias e determinando uma naveabilidade para robôs móveis.

Diante disso, este trabalho propõe a hipótese de que um robô possa se deslocar em uma trajetória previamente definida utilizando-se exclusivamente de marcadores de sinalização como definido em trabalhos correlatos (Luo et al., 2024; Jayasinghe et al., 2022;

Marques et al., 2022). Assim, o robô poderá identificar marcadores visuais para ajustar seu movimento e alcançar o destino esperado, tornando o método de navegação visual uma alternativa viável e acessível para o problema da navegação robótica. A hipótese H0 afirma que o uso de visão computacional para identificação de marcadores, combinados com um modelo matemático de navegação são suficientes para determinar uma trajetória de deslocamento para um robô; a hipótese H1, alternativa, afirma que o uso destes marcadores não são suficientes para determinar trajetórias de deslocamento para um robô. Para execução da pesquisa, pretende-se utilizar um elemento robótico móvel previamente existente que possui recursos de processamento computacional e sistema de câmeras sendo dispensado o uso de ambiente virtual simulado; este robô permitirá a inserção do programa de navegação por marcadores utilizando a linguagem de programação Python e sua execução será realizada em ambiente controlado *indoor* gerando assim dados experimentais para uso em treinamento e teste da identificação das placas marcadores.

1.3 OBJETIVO GERAL

Treinar um robô para navegação autônoma com o uso de marcadores de sinalização obtidos pelo mapeamento de um conjunto de placas de marcação (placas de sinalização).

1.3.1 OBJETIVOS ESPECÍFICOS

- Construir um *dataset* de imagens de placas de sinalização para definir os marcadores de tomada de decisão;
- Treinar um modelo algorítmico de visão computacional para identificar os marcadores de sinalização;
- Implementar e testar o modelo em uma estrutura robótica já existente;
- Implementar o modelo matemático de Teoria dos Grafos para determinação da tomada de decisão de movimentação baseada nos marcadores identificados.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ROBÓTICA MÓVEL

O campo de pesquisa da Robótica vem se tornado cada vez mais popular nas últimas décadas. Jahn (2020) apresenta algumas capacidades que os robôs apresentam: navegação, autonomia, otimização/aprendizagem, cooperação multi-robô, segurança, Interação humano-robô, segurança de dados, confiabilidade e eficiência energética; além disso também apresenta diversas implementações possíveis na robótica: capacidade de tomada de decisão em tempo real, aprendizado de máquina para identificação de obstáculos com o uso de visão computacional, computação em nuvem para persistência de informações e base de dados para autonomia, monitoramento do ambiente, modularidade por meio de design de componentes independentes e desenvolvimento baseado em modelos utilizáveis em ambientes virtuais de treinamento. Para os desenvolvedores, tornar possível todas as capacidades e implementações possíveis para os robôs é um desafio, e o futuro pode ampliar a lista, com novas tendências (Jahn et al., 2020).

Robôs móveis podem se mover de forma autônoma em diferentes ambientes, ou seja, sem a assistência de operadores humanos porque a autonomia de um robô se dá quando ele é capaz de decidir suas próprias ações para realizar tarefas, usando um sistema de percepção (Jahn et al., 2020; Rubio et al., 2019). Além disso, precisa de uma unidade de cognição ou sistema de controle para coordenar todos os seus subsistemas. Os conceitos básicos consistem em quatro campos: locomoção (lida com mecanismos eletrônicos de movimentação, cinemática, dinâmica e teoria de controle); percepção (abrange tecnologias como visão computacional e sensores); cognição (analisa os dados dos sensores e toma decisões do robô); e navegação (envolve algoritmos de planejamento, teoria da informação e inteligência artificial) (Rubio et al., 2019).

Navegação e localização são grandes desafios da robótica, essenciais para que robôs autônomos executem tarefas com eficiência em aplicações do mundo real. A implementação de visão computacional (Computational Vision - CV) oferece uma perspectiva gráfica para sistemas robóticos, permitindo o rastreamento e a detecção de objetos (Dudek e Jenkin 2024). Diversos trabalhos (Che et al., 2024; Guerrero-Osuna et al., 2023; Robinson et al., 2023) utilizam CV para complementar aplicações de aprendizado de máquina como uma solução tecnológica para ensinar navegação ao robô.

2.2 MACHINE LEARNING E DEEP LEARNING

O aprendizado de máquina (Machine Learning - ML) se concentra em automatizar a construção de modelos analíticos sendo um campo técnico em rápido crescimento (situado na interseção da ciência da computação e estatística) fundamental para a inteligência artificial e ciência de dados (Baduge et al., 2022). O avanço recente em ML é impulsionado pelo desenvolvimento de novos algoritmos de aprendizado e pela disponibilidade crescente de dados online e computação de baixo custo. Estes métodos de aprendizado de máquina estão sendo adotados em várias áreas, como saúde, manufatura, educação, modelagem financeira, policiamento e marketing, promovendo decisões mais baseadas em evidências (Jordan, Mitchell 2015).

Sistemas inteligentes que oferecem capacidades de inteligência artificial dependem de machine learning o qual se refere a habilidade de sistemas em aprender a partir de dados de treinamentos específicos para automatizar a construção de modelos analíticos e resolver tarefas associadas (Baduge et al., 2022). Janiesch et al., 2021 afirma que o deep learning é um conceito de machine learning que se baseia no uso de redes neurais artificiais e cita que em muitas aplicações, modelos de aprendizado profundo superam modelos de aprendizado de máquina tradicionais e abordagens convencionais de análise de dados.

Voulodimos, 2018 complementa afirmando que o crescimento do deep learning nos últimos anos deve-se em grande parte aos avanços que permitiu no campo da visão computacional. Além disso, métodos de deep learning vem superando velhas técnicas de machine learning, sendo a visão computacional uma das áreas mais proeminentes (Janiesch et al., 2021). A ambição de desenvolver um sistema para simular o cérebro humano motivou o desenvolvimento de redes neurais, consequentemente levando ao desenvolvimento de sistemas de visão computacional (Voulodimos, 2018).

2.3 VISÃO COMPUTACIONAL

Peters (2017) define a visão computacional como sendo um campo da inteligência artificial focado em capacitar computadores a entender e interpretar informações visuais de imagens ou vídeos por meio do desenvolvimento de algoritmos e técnicas para extrair padrões e conhecimentos a partir de dados visuais, assim, imitando a percepção visual humana. A visão computacional tem amplas aplicações, como saúde, veículos autônomos, robótica, segurança, realidade aumentada, análise de conteúdo, e automação industrial. Ela continua avançando junto com as técnicas de *deep learning* e a disponibilidade de *datasets*

de grande escala, levando a um progresso significativo no reconhecimento visual (Pandey, 2023).

A detecção de objetos é o processo de identificar instâncias de objetos de uma determinada classe (como humanos, aviões ou pássaros) em imagens e vídeos digitais (Peters, 2017). Uma abordagem comum envolve a criação de várias janelas de candidatos (regiões de uma imagem que são selecionadas como potenciais áreas onde objetos de interesse podem estar localizados) que são então classificadas usando recursos de aprendizagem de máquina. Pandey (2023) afirma que parte da maioria dos trabalhos sobre detecção de objetos usa variações de Convolutional Neural Networks (CNNs) incluindo técnicas como def-pooling, CNNs em cascata supervisionadas de forma fraca e CNNs sensíveis a subcategorias. Ainda que menos comuns, algumas abordagens utilizam modelos profundos alternativos, como redes de crença profunda (Deep Belief Networks - DBN) para detecção em imagens de sensoriamento remoto e reconhecimento de objetos 3D, autoencoders empilhados para detecção de órgãos em imagens médicas e detecção de objetos salientes em vídeos (Voulodimos, 2018).

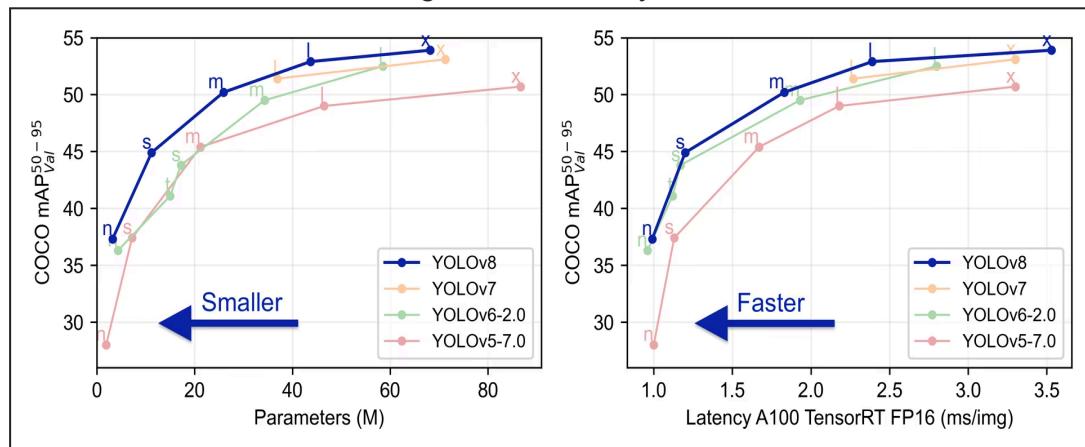
2.4 YOLO V8

Entre os muitos algoritmos de detecção de objetos em tempo real o YOLO (You Only Look Once) desenvolvido nos anos recentes, se destaca. Introduzido por Joseph Redmon em 2015, o YOLO trouxe avanços significativos ao realizar a detecção de objetos em uma única etapa, diferenciando-se dos métodos de duas etapas, como o R-CNN, que usa propostas de objetos e classificação separadamente. Além da detecção de objetos, o YOLO também é utilizado em outros campos de visão computacional, como segmentação de instâncias, estimativa de pose e detecção 3D, consolidando-se como uma abordagem central para sistemas em tempo real. YOLOv8 é uma versão reformulada do YOLOv5 que traz otimizações significativas de código e mudanças arquitetônicas, como a adaptação do ELAN (Efficient Layer Aggregation Network) do YOLOv7 e conexões residuais adicionais. Embora não seja uma nova versão do YOLO propriamente dita, YOLOv8 funciona como uma plataforma de integração tecnológica que conecta APIs para várias tarefas de pós-processamento, incluindo segmentação de instâncias, estimativa de pose e rastreamento de múltiplos objetos. Suas otimizações de código melhoraram o desempenho de treinamento em cerca de 30%, e a API permite fácil conexão com tarefas adicionais, atraindo profissionais e equipes de pesquisa e desenvolvimento (WANG, Chien-Yao, 2024).

A Figura 1 apresenta uma comparação das versões da família YOLO (You Only Look Once) em termos de precisão média ponderada (mAP) e latência média, avaliando o desempenho de cada versão no conjunto de validação COCO. No primeiro gráfico (à esquerda), o eixo X indica a quantidade de parâmetros do modelo em milhões (M), refletindo o tamanho e a complexidade de cada versão, enquanto o eixo Y mostra a métrica de precisão média. Observa-se que o YOLOv8 apresenta uma precisão superior.

No segundo gráfico (à direita), o eixo X representa a latência média de inferência em milissegundos por imagem (ms/img) em um hardware A100 com TensorRT em precisão FP16, enquanto o eixo Y mantém a métrica de precisão média. Observa-se que o YOLOv8 consegue alcançar alta precisão com menor latência, o que indica maior eficiência para aplicações em tempo real. Em resumo, o YOLOv8 mostra-se superior em precisão e velocidade em relação às versões anteriores, sendo mais adequado para aplicações que exigem alta precisão e baixa latência.

Figura 1 – Gráfico Python



Fonte: Ultralytics (2024)

2.5 PYTHON

De acordo com Krishna et al., (2023) a linguagem Python é amplamente utilizada, conhecida por sua sintaxe simples e código comprehensível, tornando-a popular entre desenvolvedores, engenheiros e amadores. Suas características de código aberto e a vasta gama de bibliotecas disponíveis facilitam a programação eficiente. Guido van Rossum, o criador do Python (Krishna et al., 2023), desenvolveu a linguagem na década de 1990. Desde então, várias versões foram lançadas, com as versões 3.6 e 3.9 sendo as mais utilizadas após o fim do suporte à versão 2.7. O Python é um projeto de código aberto mantido pela Python Software Foundation. Sendo uma linguagem de alto nível, os programas em Python são fáceis de entender e executam em uma máquina virtual,

permitindo a criação de códigos portáteis e executáveis em várias plataformas. Além disso, Python é dinamicamente tipada e utiliza um interpretador para processar tipos em tempo de execução, abstraindo otimizações subjacentes e integrando bibliotecas em Fortran e C para cálculos intensivos.

Esta linguagem foi escolhida para este projeto pois as bibliotecas e frameworks disponíveis facilitam a implementação e integração da visão computacional em um sistema. Além disso, Python é amplamente utilizado nas áreas de Ciência de Dados, computação científica, análise de dados e Aprendizado de Máquina. Seu sucesso deve-se em parte a bibliotecas focadas em ciência de dados e machine learning, como NumPy, Pandas e TensorFlow, bibliotecas de alto desempenho desenvolvidas em linguagens estáticas como C++ e Fortran devido à lentidão do interpretador Python (Castro, O, 2023).

2.6 TEORIA DOS GRAFOS

Segundo Majeed (2020) a teoria dos grafos teve início com o problema das pontes de Königsberg em 1735, abordado por Euler, que criou o conceito de grafo e de grafos Eulerianos. Em 1840, A. F. Möbius introduziu os conceitos de grafo completo e bipartido. Posteriormente, Gustav Kirchhoff usou grafos em 1845 para resolver problemas em circuitos elétricos com uma estrutura de árvore sem ciclos. Em 1852, Thomas Gutherie desenvolveu o problema das quatro cores, crucial para aplicações em computação gráfica. Em 1856, Thomas Kirkman e Hamilton conceberam o grafo Hamiltoniano, que teve grande impacto nos estudos de caminhos e rotas. O termo "grafo" foi introduzido em 1878 por James Sylvester, e novas contribuições surgiram ao longo dos anos, incluindo a teoria de colorabilidade de Ramsey em 1941. Com o tempo, a teoria dos grafos passou a ser aplicada em áreas modernas, como análise de redes sociais, big data, processamento de linguagem natural (NLP) e reconhecimento de padrões.

A teoria dos grafos, hoje se aplica amplamente em todas as disciplinas científicas e em diversas tecnologias modernas de informação e computação (Sporns, O, 2018). Além disso, possui aplicações amplas e únicas em ciência da computação e redes sociais, abrangendo criptografia, codificação, análise de redes neurais, planejamento de conservação e segmentação de imagens. Na ciência da computação a teoria é usada para *clustering* de documentos web, processamento de sinal e descoberta de conhecimento em redes sociais (Majeed et al., 2020).

O renomado cientista da computação holandês E. W. Dijkstra propôs, em 1958, o algoritmo de caminho mais curto que leva seu nome — o algoritmo de Dijkstra — uma

solução que teve grande impacto. O algoritmo de Dijkstra realiza uma varredura exhaustiva: para encontrar o caminho mais curto em um grafo, ele registra todas as rotas possíveis de forma expansiva (em camadas), garantindo que nenhuma rota se repita. Em seguida, calcula o comprimento de cada uma e compara os resultados para identificar o menor caminho (Heming et al., 2022).

3 METODOLOGIA

3.1 MODELAGEM DAS PLACAS DE SINALIZAÇÃO (MARCADORES DE SINALIZAÇÃO)

Nesta etapa foram modeladas as placas de sinalização a serem utilizadas para o machine learning e, posteriormente, utilizadas no sistema de navegação do robô. Essa modelagem envolve a definição dos tipos de placas que serão incluídas no escopo do projeto, tais como: parada obrigatória, virar à esquerda, virar à direita e seguir reto. Estas placas serão impressas no tamanho A4 padrão, todas na cor preta e com fundo branco. Foram realizados diversos testes experimentais para identificar a melhor forma de modelo que se adapta ao YoloV8.

Figura 2 – Modelo visual (Direita)



Fonte: Elaborado pelo autor (2025)

3.2 OBTENÇÃO DO DATASET PARA TREINAMENTO

Para obter o banco de dados que será utilizado no treinamento do modelo de reconhecimento e identificação dos marcadores de sinalização, foi necessário a criação de um *dataset* próprio e personalizado com imagens das placas previamente modeladas. Para isso foi utilizado a câmera do robô que gravou as imagens durante o seu deslocamento em um ambiente controlado com a presença destes marcadores de sinalização. Com as imagens prontas, foi utilizado o site “roboflow” para realizar as marcações, foram marcados 185 frames para o dataset. Após a obtenção do *dataset*, as imagens estavam prontas para serem

pré-processadas para melhorar o desempenho do modelo e posterior utilização pelo YOLO v8.

Figura 3 – Dataset customizado extraído do site roboflow



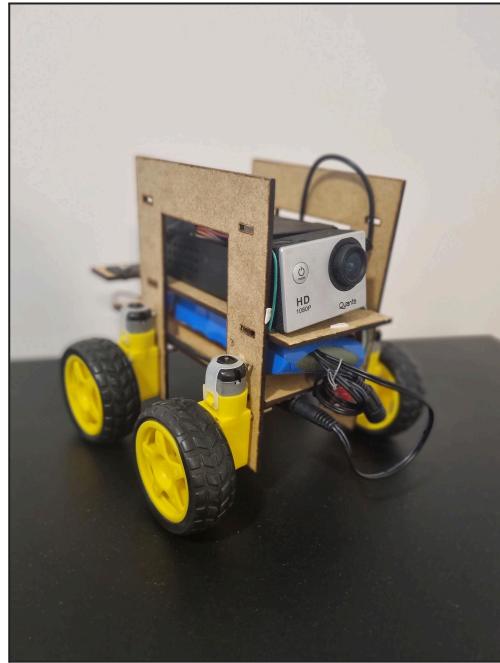
Fonte: Elaborado pelo autor (2025)

3.3 INTEGRAÇÃO E IMPLEMENTAÇÃO

A etapa de integração envolve unir os componentes do sistema robótico com o modelo de detecção visual treinado e também todo o hardware utilizado para executar o sistema de detecção dos marcadores de sinalização incluindo:

- Integração do Modelo YOLOv8 treinado e integrado ao sistema de visão computacional, que processa o vídeo capturado em tempo real pela câmera;
- Uso de Python para implementação da inteligência artificial do robô; e
- Implementação dos sistemas integrados ao robô.

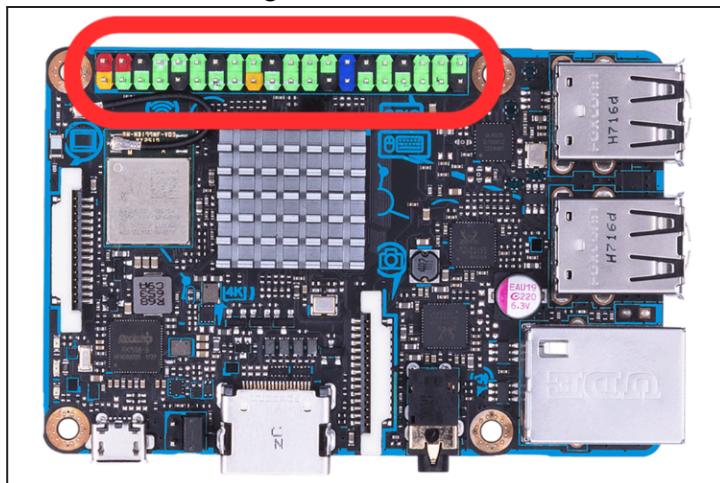
Figura 4 – Gráfico Python



Fonte: Elaborado pelo autor (2025)

A estrutura robótica (Figura 4) foi fornecida pelo professor orientador do projeto, provinda de um trabalho anterior de sua autoria. A estrutura serviu como base para a implementação do sistema. A estrutura utiliza o computador de placa única Asus TinkerBoard que possui 28 portas GPIO onde foram conectados os motores e permitindo por algoritmo escrito em Python o controle desses motores.

Figura 5 – Asus Tinker Board



Fonte: tinker-board.asus.com (editada)

A figura 5 apresenta o computador Asus Tinker Board, se trata de um SBC (Single Board Computer - Computador de Placa Única) destacando em vermelho as 28 portas GPIOs que o computador apresenta. As portas GPIO são portas de entrada e saída de propósito geral, elas permitiram a inserção de motores ao modelo e o controle de ligar e desligar essas portas com a biblioteca ASUS.GPIO.

As configurações do Asus Tinker Board utilizado são:

- Processador Quad-Core RK3288
- Placa gráfica integrada - ARM® Mali™-T764 GPU*1
- Memória RAM de 2GB
- 28 GPIOs configuráveis
- Sistema Operacional Tinker OS

3.4 TESTE DE ACURÁCIA E RECONHECIMENTO

A avaliação da eficácia do sistema foi conduzida por testes em ambientes simulados visando medir taxa de detecção, falsos positivos e tempo de resposta. Isso pôde ser feito pelo software YOLOv8 junto ao vídeo teste gravado pelo robô, o software forneceu os dados necessários para a avaliação (Figura 6). Neste contexto foram utilizados os métodos de avaliação: precisão, acurácia, taxa de acerto e confiabilidade do modelo.

Figura 6 – Exemplo de detecção em tempo real



Fonte: Elaborado pelo autor (2025)

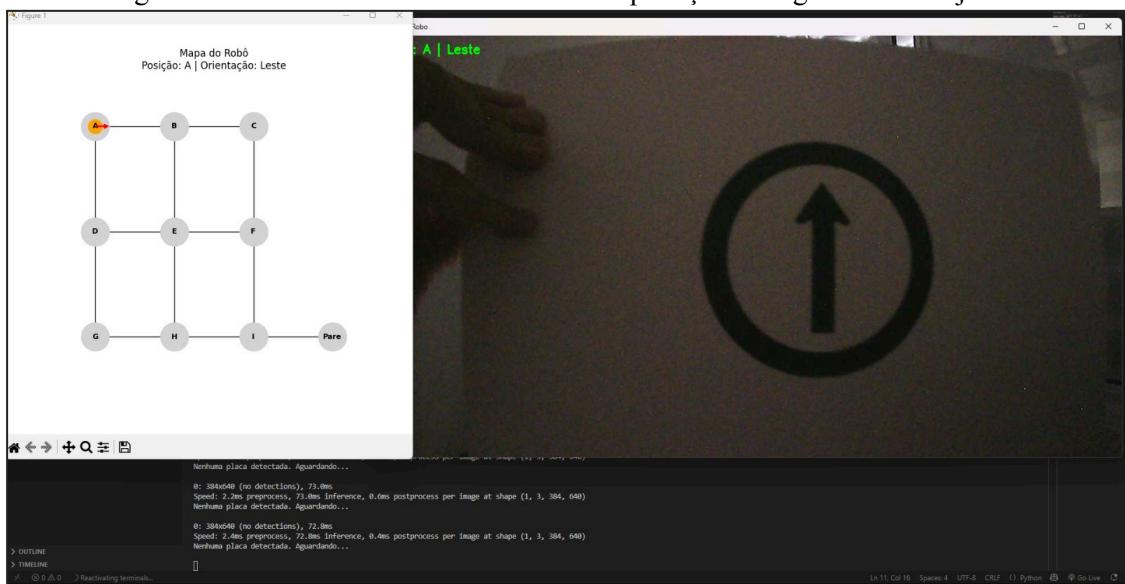
3.5 APLICAÇÃO PRÁTICA

Após finalizado os processos do sistema, a arquitetura foi reajustada, considerando os resultados e observações obtidas pelos testes, para criar um modelo final. Esse modelo foi utilizado para realizar testes empíricos no qual a acurácia do modelo foi avaliado em prática com a detecção dos marcadores; em seguida, com base em uma acurácia em torno de 70 a 80% conforme previsto por diversos autores (Moru et al., 2022; Yuan et al., 2021; Wang et al., 2017).

3.6 APLICAÇÃO NA MODELAGEM MATEMÁTICA DE GRAFOS

A teoria dos grafos foi utilizada para representar as rotas possíveis com base nas placas de sinalização identificadas pelo robô. Cada vértice representa um caminho possível, as arestas do grafo, são os caminhos e possuem distâncias (pesos) diferentes. No início do programa, o robô recebe uma indicação do melhor caminho possível, dependendo do seu ponto de início e destino. Ele aceita rotas alternativas, nesse caso, o algoritmo roda novamente buscando um novo caminho otimizado. Para isso, foi aplicado o algoritmo de Dijkstra (Figura 7).

Figura 7 – Funcionamento do sistema com aplicação do algoritmo de Dijkstra



Fonte: Elaborado pelo autor (2025)

Em um sistema voltado a testes foi desenvolvido um mapa gráfico para facilitar a visualização da implementação do algoritmo de Dijkstra. Com essa implementação o sistema se tornou capaz de se localizar em um mapa onde cada ponto (vértice) possui um peso (distância) diferente, assim, o robô consegue se localizar e prever o caminho mais curto para o seu destino.

4 TREINAMENTO

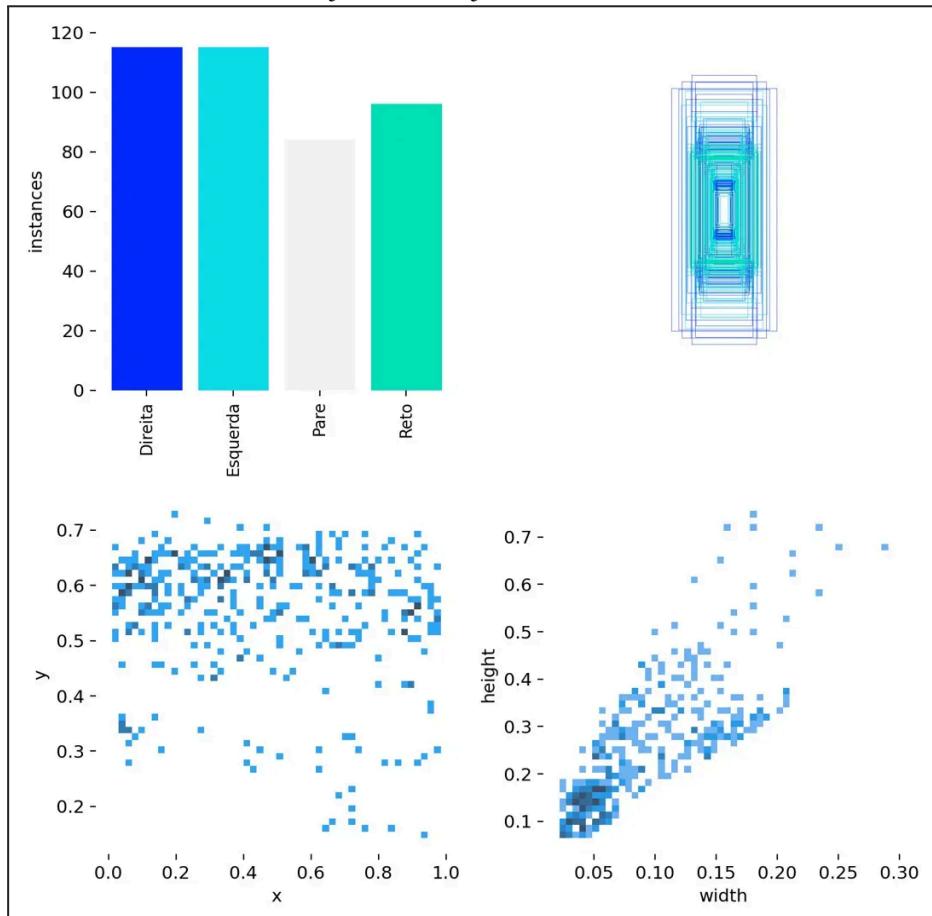
O treinamento utilizando o modelo YOLOv8 permitiu a obtenção de dados relevantes para a análise do desempenho, comportamento e confiabilidade da rede. A partir desses dados, foi possível realizar uma avaliação criteriosa para verificar se o modelo apresentou resultados satisfatórios para ser implementado no sistema.

4.1 ANÁLISE DO TREINAMENTO

O treinamento foi realizado com YOLOv8 utilizando o modelo “yolov8n” com 50 “epochs” e tamanho de imagem de 640px, o mesmo utilizado na construção do dataset, foram feitos 3 testes diferentes, onde foram alterados os marcadores visuais com base nos resultados obtidos. No último treinamento realizado, os dados demonstraram um bom desempenho geral, com uma precisão média (P) de 90,9%, revocação (R) de 84,7%, e uma mAP@0.5 de 94,6%. Já a métrica mais rigorosa, mAP@0.5:0.95, atingiu 80,9%, o que indica que o modelo teve bom desempenho em diferentes níveis de sobreposição (IoU). As classes "Direita" e "Reto" obtiveram os melhores resultados, com mAP@0.5 acima de 99% e 91%, respectivamente.

A Figura 8 apresenta informações sobre a composição do dataset utilizado para treinar o modelo YOLO v8. Na parte superior, observa-se a distribuição de instâncias entre as quatro classes de sinais de trânsito: "Direita" (~115 instâncias), "Esquerda" (~115 instâncias), "Pare" (~85 instâncias) e "Reto" (~95 instâncias).

Figura 8 – Distribuição de instâncias por classe e visualização de características físicas dos objetos no conjunto de dados



Fonte: Elaborado pelo autor (2025)

A visualização também inclui gráficos de dispersão mostrando a relação entre as coordenadas (x , y) dos objetos nas imagens e suas dimensões (largura e altura). Nota-se que a maioria dos sinais de trânsito tende a se concentrar em certas regiões da imagem, particularmente na parte central-superior (y entre 0.5 e 0.6). Quanto às dimensões, observa-se uma tendência de agrupamento em tamanhos específicos, com a maioria dos objetos apresentando proporções relativamente pequenas da imagem total. A concentração de objetos com tamanhos similares sugere um conjunto de dados com certa homogeneidade, o que pode facilitar o aprendizado do modelo, mas também pode limitar sua capacidade de generalização para cenários mais diversos.

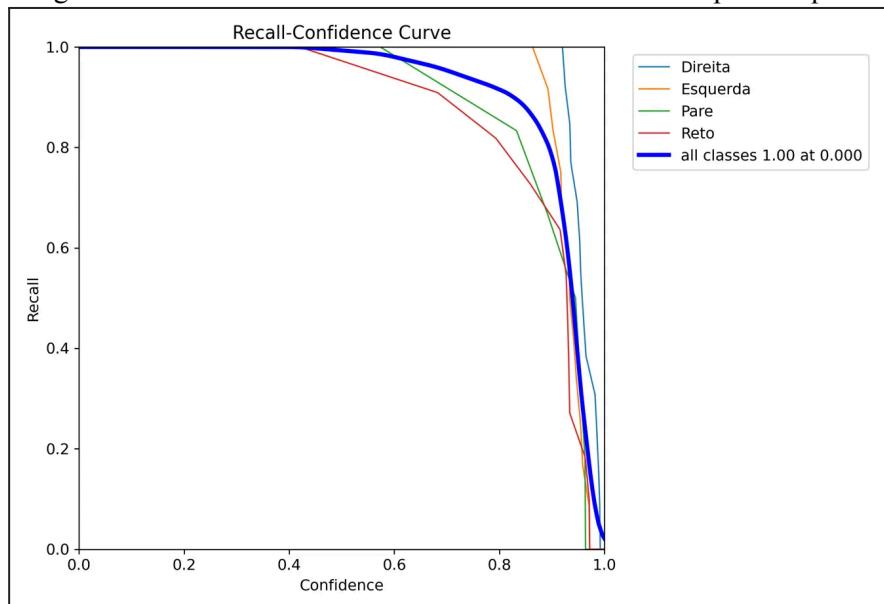
As Figuras 9 e 10 apresentam duas métricas complementares fundamentais para a avaliação do desempenho do modelo de detecção:

A **Curva Recall-Confidence** (Figura 9) demonstra como a capacidade do modelo de detectar todos os objetos relevantes (recall) varia em função do limiar de confiança aplicado. Observa-se que o modelo mantém recall perfeito (1.0) até um limiar de confiança de aproximadamente 0.4, após o qual começa a decair gradualmente. Isto indica que, com

um limiar de confiança baixo a moderado, o modelo é capaz de detectar praticamente todas as instâncias dos sinais de trânsito presentes nas imagens.

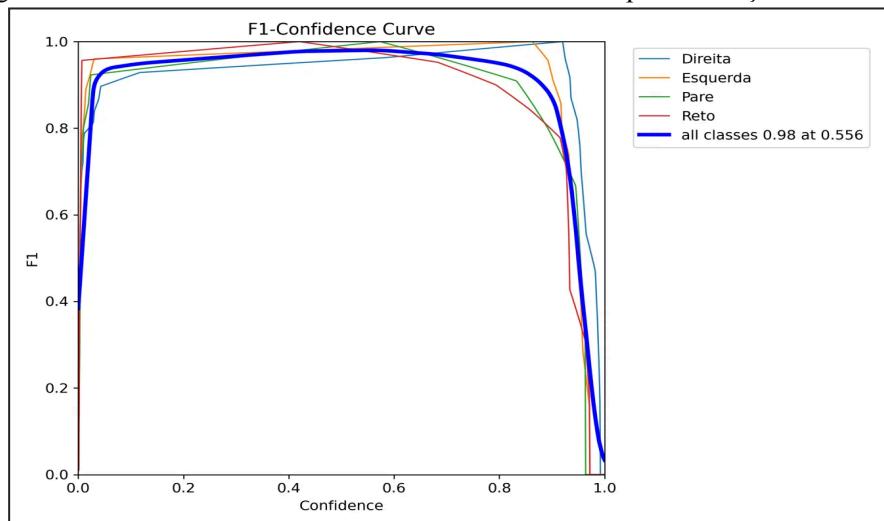
A **Curva F1-Confidence** (Figura 10) apresenta a relação entre o valor F1 (média harmônica entre precisão e recall) e o limiar de confiança. Esta métrica é particularmente útil para identificar o limiar ótimo de operação do modelo. Os resultados mostram que o modelo atinge seu pico de desempenho ($F1 = 0.98$) com um limiar de confiança de aproximadamente 0.556.

Figura 9 – Curva Recall-Confidence do modelo YOLOv8 para as quatro classes



Fonte: Elaborado pelo autor (2025)

Figura 10 – Curva F1-Confidence do modelo YOLOv8 para detecção de sinais de trânsito



Fonte: Elaborado pelo autor (2025)

4.1.1 Conclusões do treinamento

Os resultados obtidos na avaliação do modelo YOLOv8 para detecção de sinais de trânsito demonstram desempenho bom, com métricas muito próximas do ideal. O modelo alcançou um F1-score máximo de 0.98 e manteve alto recall até limiares de confiança relativamente elevados.

É importante reconhecer algumas limitações deste estudo. O desempenho alto pode indicar similaridade excessiva entre os ambientes de treinamento e teste. Além disso, o número limitado de classes e a possível homogeneidade das condições de captura das imagens podem não representar adequadamente os desafios de um cenário real de aplicação.

5 ESTRUTURA DO ALGORITMO

Para realizar o algoritmo que irá controlar o robô, é necessário analisar suas funcionalidades principais que se integram para formar um comportamento autônomo:

- Movimentação do robô com controle via GPIO
- Detecção de placas em tempo real utilizando Yolo
- Lógica de navegação e tomada de decisão com base no algoritmo de Dijkstra

Para implementar essas funcionalidades foi utilizado python e bibliotecas como:

- | | | |
|--------------------|---|-----------------------|
| ● ASUS.GPIO | - | Controle dos motores |
| ● CV2 | - | Captura de vídeo |
| ● time/sleep | - | Controle de tempo |
| ● ultralytics/YOLO | - | Detecção visual |
| ● heapq | - | Algoritmo de Dijkstra |

5.1 MOVIMENTAÇÃO COM GPIO

A movimentação física do robô foi realizada através do controle dos pinos da placa controladora do Asus Tinker Board usando a biblioteca ASUS.GPIO, que envia sinais elétricos para os motores. Esse controle é implementado na função “mover_robo”, e configurado no início do código.

```
in1, in2, enA = 35, 36, 33
in3, in4, enB = 37, 38, 32

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup([enA, in1, in2, in3, in4, enB], GPIO.OUT)
GPIO.output([in1, in2, in3, in4], GPIO.LOW)
GPIO.output([enA, enB], GPIO.HIGH)

p_enA = GPIO.PWM(enA, 1000)
p_enB = GPIO.PWM(enB, 1000)
VEL = 25
p_enA.start(VEL)
p_enB.start(VEL)

def mover_robo(direcao):
    comandos = {
        "Pare": (GPIO.LOW, GPIO.LOW, GPIO.LOW, GPIO.LOW),
        "Reto": (GPIO.LOW, GPIO.HIGH, GPIO.HIGH, GPIO.LOW),
        "Esquerda": (GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.LOW),
        "Direita": (GPIO.LOW, GPIO.HIGH, GPIO.LOW, GPIO.LOW)
    }
    if direcao in comandos:
        GPIO.output([in1, in2, in3, in4], comandos[direcao])
        printf("Movendo: {direcao}")
```

O trecho inicial configura os pinos GPIO usados para controlar os motores. Os pinos enA e enB controlam a velocidade via PWM (modulação por largura de pulso), e os pares de pinos in1/in2 e in3/in4 definem o sentido de rotação dos motores.

A função “mover_robo” centraliza os comandos de movimentação com base na direção desejada. Cada direção corresponde a uma combinação de sinais lógicos (GPIO.HIGH ou GPIO.LOW) que são enviados aos motores. O comando GPIO.output(...) aplica os sinais elétricos diretamente aos pinos configurados, fazendo com que os motores girem e o robô se move conforme esperado.

A velocidade foi fixada com o valor VEL = 25, suficiente para movimentar o robô de forma previsível e compatível com os tempos usados na lógica de decisão.

5.2 DETECÇÃO UTILIZANDO YOLO

A capacidade de "enxergar" o ambiente e interpretar sinais visuais foi adaptada utilizando uma câmera para capturar imagens em tempo real e o modelo de detecção YoloV8 para reconhecer os marcadores visuais.

```
# Inicialização da câmera
camera = cv2.VideoCapture(0)
camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 640)

# Carregamento do modelo
modelo = YOLO('best.pt')

# Dentro do loop principal
ret, frame = camera.read()
if not ret:
    print("Erro na câmera")
    break

resultados = modelo(frame)
direcao_detectada = None
confianca = 0.0

for resultado in resultados[0].boxes:
    conf, classe = resultado.conf[0], int(resultado.cls[0])
    if conf > CONFIDENCE_THRESHOLD and classe < len(CLASSES):
        direcao_detectada = CLASSES[classe]
        confianca = float(conf)
        break
```

O objeto “camera” é criado com cv2.VideoCapture(0) e configurado para capturar imagens com resolução 640x640. O modelo YOLOv8 é carregado a partir do arquivo best.pt. A cada iteração do loop, um novo frame é capturado pela câmera. Esse frame é processado pelo modelo YOLO, que retorna uma lista de detecções (caixas, classes e confianças). O sistema ignora detecções com confiança abaixo de

CONFIDENCE_THRESHOLD = 0.7. A classe detectada é convertida em texto com base na lista CLASSES, o que permite interpretar que tipo de movimento o robô deve executar em seguida.

5.3 LÓGICA DE NAVEGAÇÃO COM BASE NO ALGORITMO DE DIJKSTRA

Para que o robô tome decisões coerentes e siga o caminho mais eficiente até seu destino, foi implementado o algoritmo de Dijkstra, que calcula o caminho de menor custo entre dois pontos em um grafo. Esse grafo representa o mapa do ambiente por onde o robô pode se movimentar.

```
# Estrutura de pesos entre os nós abreviada (representando o mapa)
PESOS = {
    ('A', 'B'): 3, ('B', 'C'): 4, ..., ('S', 'T'): 3
}
# Adiciona pesos bidirecionais
for (a, b) in list(PESOS):
    PESOS[(b, a)] = PESOS[(a, b)]

# Geração das conexões válidas para cada ponto do mapa
letras = 'ABCDEFGHIJKLMNPQRST'
DIRECOES = {}
for idx, nome in enumerate(letras):
    row, col = divmod(idx, 5)
    vizinhos = {}
    if row > 0: vizinhos['Norte'] = letras[idx - 5]
    if row < 3: vizinhos['Sul'] = letras[idx + 5]
    if col > 0: vizinhos['Oeste'] = letras[idx - 1]
    if col < 4: vizinhos['Leste'] = letras[idx + 1]
    DIRECOES[nome] = vizinhos

# Algoritmo de Dijkstra para encontrar o menor caminho
def dijkstra(origem, destino):
    grafo = {}
    for atual, viz in DIRECOES.items():
        grafo[atual] = {
            v: PESOS.get((atual, v), 3) for d, v in viz.items()
        }

    fila = [(0, origem)]
    caminhos = {origem: (None, 0)}
    visitados = set()

    while fila:
        custo, atual = heapq.heappop(fila)
        if atual in visitados: continue
        visitados.add(atual)
        if atual == destino: break
        for viz, peso in grafo[atual].items():
            novo = custo + peso
            if viz not in caminhos or novo < caminhos[viz][1]:
                caminhos[viz] = (atual, novo)
                heapq.heappush(fila, (novo, viz))

    caminho = []
    atual = destino
    while atual != origem:
        caminho.append(atual)
        atual = caminhos[atual][0]
    caminho.append(origem)
    caminho.reverse()
```

```
while atual:  
    caminho.insert(0, atual)  
    anterior = caminhos.get(atual)  
    if anterior is None: break  
    atual = anterior[0]  
return caminho
```

O ambiente em que o robô se movimenta é representado por um conjunto de nós (letras A até T) conectados por arestas com pesos, que correspondem ao tempo estimado de deslocamento entre dois pontos. Para cada ponto (nó), é registrada a direção possível (Norte, Sul, Leste, Oeste) e o próximo ponto acessível. Isso permite ao robô interpretar qual caminho seguir com base em sua orientação atual.

A função “dijkstra(origem, destino)” calcula o caminho com menor custo acumulado entre os dois pontos usando o algoritmo clássico de Dijkstra com fila de prioridade (heapq), após cada movimentação do robô, a função é chamada novamente, recalculando o caminho ideal a partir da nova posição até o destino. Isso permite replanejar rotas caso ocorram desvios ou obstáculos.

6 RESULTADOS

O sistema desenvolvido neste trabalho soluciona o desafio do deslocamento robótico autônomo através do uso de machine learning com visão computacional e algoritmos.

6.1 CENÁRIOS

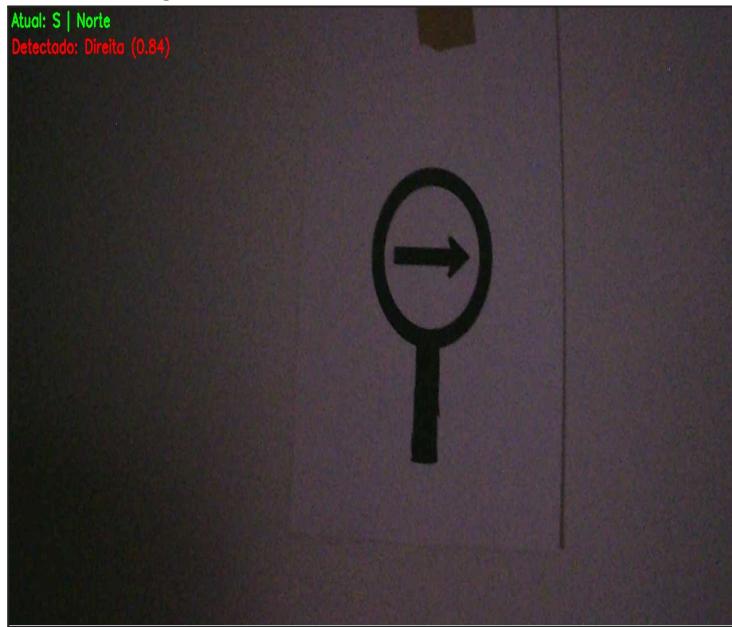
Para testar, em um cenário real, o funcionamento do sistema desenvolvido, foram criados 4 cenários de teste realizando 15 repetições de execução para cada cenário (60 repetições no total) a fim de realizar uma análise estatística e testar a confiabilidade do modelo. O cenário A e B testa a capacidade de detecção do sistema em um ambiente iluminado, acima dos 200 lux, em distâncias de 100 e 200 metros. Os cenários C e D serviram para testar o sistema em um ambiente escuro, entre 20 a 50 lux, em distâncias de 100 e 200 metros, também foi considerado caminhos (movimentos necessários para que o robô alcance o destino) curtos (6 movimentos), médios (8 movimentos) e longos (13 movimentos), observa-se que o cenário D foi necessário alterar o Grau de confiança aceito, apesar de isso influenciar diretamente os testes, foi uma medida necessária para que o sistema funcionasse corretamente diante as condições desse cenário.

Figura 11 – Detecção em cenário iluminado



Fonte: Elaborado pelo autor (2025)

Figura 12 – Detecção em cenário escuro



Fonte: Elaborado pelo autor (2025)

Tabela 1 - Cenários

Cenário	Distância da placa	Condições Visuais	Grau de confiança
A	100cm	Cenário Iluminado	0.7
B	200cm	Cenário Iluminado	0.7
C	100cm	Cenário Escuro	0.7
D	200cm	Cenário Escuro	0.6

Fonte: Elaborado pelo autor (2025)

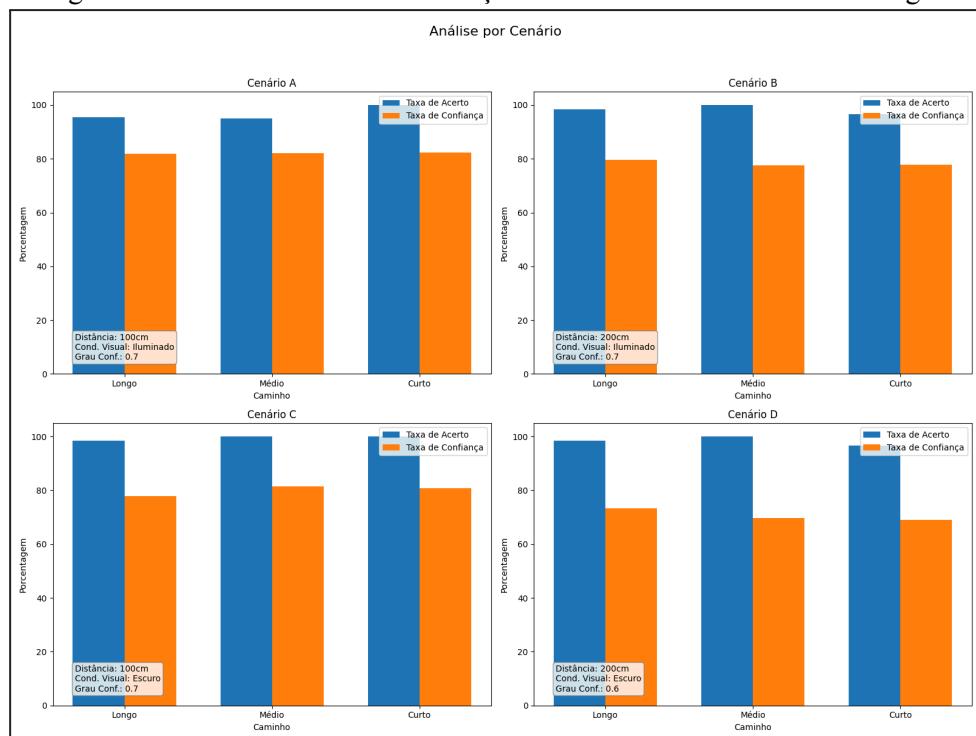
Foi Testado o comportamento e confiabilidade do modelo quanto à distância a qual consegue detectar os modelos visuais (placas), à confiança no momento da detecção, à taxa de acerto ao percorrer o caminho (se houveram erros ou não), à caminhos curtos e longos, e à influência de condições visuais não ideais. O grau de confiança aceito determina o mínimo de confiança que o modelo precisa ter para aceitar a detecção, isso influencia diretamente a taxa de confiabilidade, por isso, se manteve um padrão, entretanto foi necessário reduzi-la em um caso pois o modelo não funcionaria de forma eficiente sem a alteração.

6.2 ANÁLISE ESTATÍSTICA

Nesta seção, foram aplicadas análises estatísticas descritivas e comparativas para avaliar o desempenho do sistema de navegação robótica nos diferentes cenários testados. A

avaliação estatística dos resultados foi fundamental para validar o desempenho do sistema proposto. Os dados coletados incluem a taxa de acerto no reconhecimento dos marcadores e a confiança média atribuída pelo modelo em cada detecção.

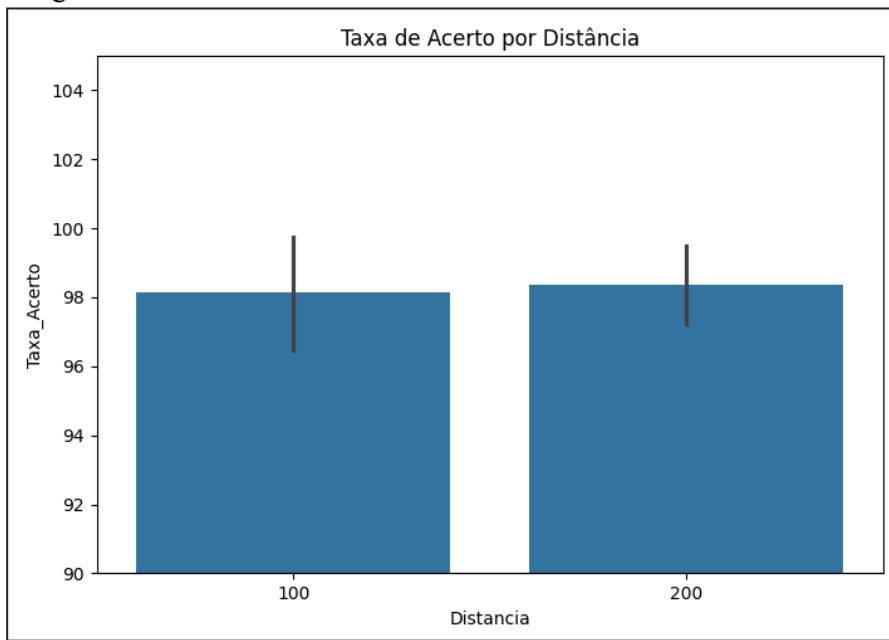
Figura 13 – Taxa de acerto x Confiança em caminhos curtos médios e longos



Fonte: Elaborado pelo autor (2025)

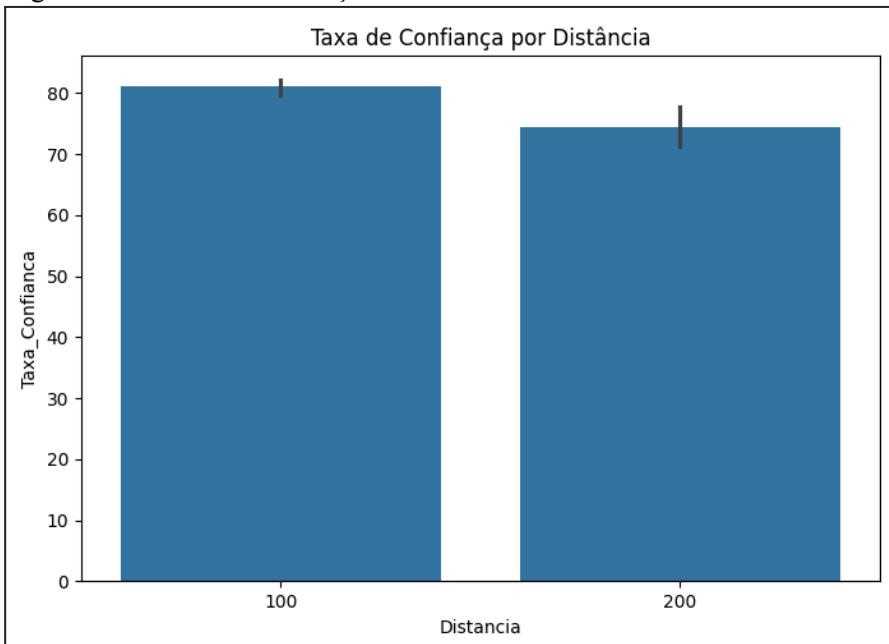
A figura 13 apresenta a média das taxas de acerto e taxas de confiança entre todos os cenários (A, B, C e D) em caminhos longos (13 movimentos), médios (8 movimentos) e curtos (6 movimentos). Pode-se observar que o tamanho do caminho teve praticamente nenhuma influência nos testes, a taxa de acerto também obteve não obteve quase nenhuma variância em todos os cenários, já a taxa de confiança teve uma variância maior quando comparada aos cenários e menor quando comparada ao tamanho dos caminhos.

Figura 14 – Taxa de acerto x Distância em distâncias de 100cm e 200cm



Fonte: Elaborado pelo autor (2025)

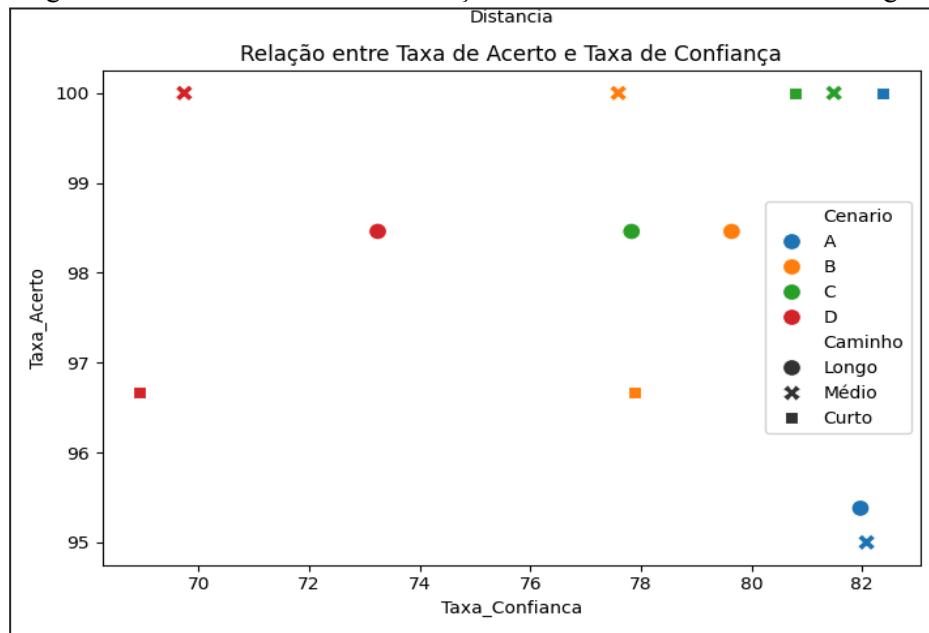
Figura 15 – Taxa de confiança x Distância em distâncias de 100cm e 200cm



Fonte: Elaborado pelo autor (2025)

As figuras 14 e 15 apresentam um cenário de testes onde é comparado a influência de distâncias de 100cm e 200cm nas taxas de acerto e confiança. Pode-se observar que as distâncias praticamente não possuem nenhum tipo de variância e relacao a taxa de acerto, ou seja, nas duas distâncias os testes estão apresentando poucos erros, já observando a taxa de confiança, pode se observar nitidamente a queda de valores em uma distância mais distante (200cm).

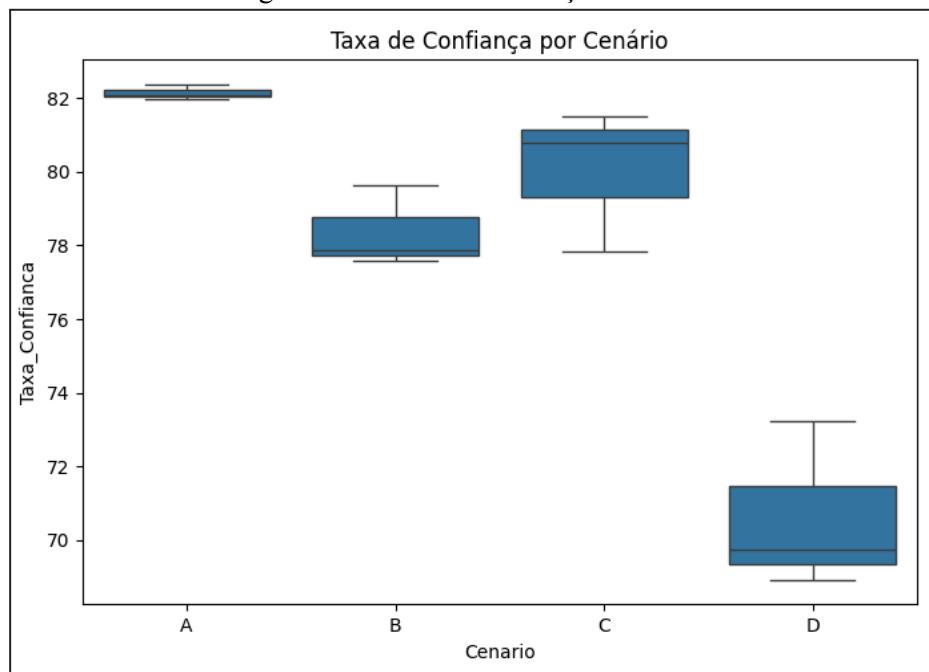
Figura 16 – Taxa de acerto x Confiança em caminhos curtos médios e longos



Fonte: Elaborado pelo autor (2025)

O gráfico da Figura 16 mostra o desempenho geral dos cenários diante da taxa de acerto e taxa de confiança em caminhos curtos (6 movimentos), médios (8 movimentos) e longos (13 movimentos). O resultado de maior valor nesse teste foi o cenário A utilizando um caminho curto, o que foi representado pelo quadrado azul.

Figura 17 – Taxa de confiança x Cenário



Fonte: Elaborado pelo autor (2025)

A figura 17 apresenta a relação entre confiança e cenário. Pode-se observar, em todos os cenários (A, B, C e D), o nível médio da taxa de confiança encontrada nos testes e, também, a variância que a taxa de confiança apresenta em seu respectivo cenário. Nesse

caso, a variância em nenhum caso é muito relevante e o cenário que melhor se sobressai é o cenário a com uma taxa de confiança de aproximadamente 82%.

6.3 ANÁLISE INFERENCIAL

A ANOVA (Análise de Variância) foi escolhida como método estatístico nesta análise por ser adequada quando se deseja comparar mais de duas médias para verificar se existem diferenças estatisticamente significativas entre elas. O objetivo era identificar se variáveis como cenário, caminho, distância e condição visual influenciam a Taxa de Acerto e a Taxa de Confiança. Como cada variável possui mais de dois níveis e o interesse é avaliar o efeito de cada uma sobre uma variável dependente contínua, a ANOVA é o teste mais apropriado.

Resultados obtidos com a análise inferencial:

- **Taxa de Acerto:** Nenhuma das variáveis analisadas apresentou relação significativa com a taxa de acerto. Os p-valores obtidos foram:
 - Cenário: 0,4237
 - Caminho: 0,7636
 - Distância: 0,8399
 - Condição Visual: 0,2333

Conclusões: indica que o desempenho do sistema (em termos de acerto) manteve-se estável, independentemente das variações nas condições dos testes.

- **Taxa de Confiança:** Duas das quatro variáveis foram analisadas, apresentaram relação significativa com a taxa de confiança. Os p-valores obtidos foram:
 - Cenário: **0,0001** (significativo)
 - Distância: **0,0076** (significativo)
 - Condição Visual: 0,0685 (não significativo)
 - Caminho: 0,9828 (não significativo)

Com esses resultados, pode-se concluir que o cenário e a distância influenciam significativamente a taxa de confiança do modelo, ou seja, certos contextos e distâncias alteram a percepção de confiabilidade do sistema, mesmo que a taxa de acerto não tenha mudado. Isso é importante, pois mostra que a usabilidade e a confiança subjetiva do usuário podem variar conforme as condições externas.

7 CONCLUSÃO E TRABALHOS FUTUROS

Este projeto desenvolveu um sistema de navegação robótica autônoma baseado na identificação de placas de sinalização utilizando visão computacional e Teoria dos Grafos. Através do uso do modelo YOLO v8, aliado a um conjunto personalizado de marcadores visuais e também ao algoritmo de Dijkstra responsável pela definição da trajetória. Foi possível implementar um sistema capaz de reconhecer sinais de forma eficaz e tomar decisões de movimentação com base em um grafo gerado dinamicamente. A análise estatística indicou robustez na execução do trajeto e uma boa estabilidade nos resultados, validando a hipótese de que a navegação visual baseada em marcadores é uma alternativa viável para robôs móveis.

A modelagem matemática por grafos foi essencial para que o robô reconhecesse sua posição e distância entre os pontos, assim, encontrando rotas otimizadas entre os pontos de interesse.

Por se tratar de um trabalho que utiliza algoritmos de machine learning a possibilidade de melhorias e expansão são inúmeras. A melhoria da estrutura robótica poderia melhorar a performance do sistema assim permitindo melhorias em diversos aspectos. O algoritmo de Dijkstra e o mapeamento poderia possibilitar o deslocamento utilizando-se de caminhos reais, seria preciso melhorar a orientação do robô.

REFERÊNCIAS

- BADUGE, S; THILAKARATHNA, S; PERERA, J; ARASHPOUR, M; SHARAFI, P; TEODOSI, B; SHRINGI, A; MENDIS, P. Artificial intelligence and smart vision for building and construction 4.0: machine and deep learning methods and applications. **Automation In Construction**, [S. l.], v. 141, p. 104440, set. 2022. Elsevier BV. DOI: <https://doi.org/10.1016/j.autcon.2022.104440>. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0926580522003132>. Acesso em: 17 nov. 2024.
- BAHRIN, M., OTHMAN, M., AZLI, N., & TALIB, M. (2016). INDUSTRY 4.0: A REVIEW ON INDUSTRIAL AUTOMATION AND ROBOTIC. **Jurnal Teknologi (Sciences & Engineering)**, [S. l.], v. 78, n. 6-13, 2016. DOI: <https://doi.org/10.11113/JT.V78.9285>. Disponível em: <https://journals.utm.my/jurnalteknologi/article/view/9285>. Acesso em: 11 fev 2025.
- CASTRO, O; BRUNEAU, P; SOTTET, J; & TORREGROSSA, D. (2023). Landscape of High-Performance Python to Develop Data Science and Machine Learning Applications. **ACM Computing Surveys**, 56, 1 - 30. DOI: <https://doi.org/10.1145/3617588>. Disponível em: <https://dl.acm.org/doi/10.1145/3617588>. Acesso em: 1 jun 2025.
- CAZZATO, Dario; CIMARELLI, Claudio; SANCHEZ-LOPEZ, Jose Luis; VOOS, Holger; LEO, Marco. A Survey of Computer Vision Methods for 2D Object Detection from Unmanned Aerial Vehicles. **Journal Of Imaging**, [S.L.], v. 6, n. 8, p. 78, 4 ago. 2020. MDPI AG. DOI: <https://doi.org/10.3390/jimaging6080078>. Disponível em: <https://www.mdpi.com/2313-433X/6/8/78>. Acesso em: 9 maio 2025.
- CHE, Chang; ZHENG, Haotian; HUANG, Zengyi; JIANG, Wei; LIU, Bo. Intelligent Robotic Control System Based on Computer Vision Technology. Cornell University, abr. 2024. ArXiv. DOI: <https://doi.org/10.48550/arXiv.2404.01116>. Disponível em: <https://arxiv.org/abs/2404.01116>. Acesso em: 23 maio 2025.
- DUDEK, Gregory; JENKIN, Michael. **Computational Principles of Mobile Robotics**. 3. ed. Cambridge: Cambridge University Press, 2024.
- GUPTA, Ashish Kumar; SEAL, Ayan; PRASAD, Mukesh; KHANNA, Pritee. Salient Object Detection Techniques in Computer Vision—A Survey. **Entropy**, [S.L.], v. 22, n. 10, p. 1174, 19 out. 2020. MDPI AG. DOI: <http://dx.doi.org/10.3390/e22101174>. Disponível em: <https://www.mdpi.com/1099-4300/22/10/1174>. Acesso em: 13 dez 2024.
- Guerrero-Osuna, H., Nava-Pintor, J., Olvera-Olvera, C., Ibarra-Pérez, T., Carrasco-Navarro, R., & Luque-Vega, L. (2023). Educational Mechatronics Training System Based on Computer Vision for Mobile Robots. **Sustainability**. DOI: <https://doi.org/10.3390/su15021386>. Disponível em: <https://www.mdpi.com/2071-1050/15/2/1386>. Acesso em: 14 dez 2024.
- Heming Bing, Lu Lai. Improvement and Application of Dijkstra Algorithms. **Academic Journal of Computing & Information Science (2022)**, Vol. 5, Issue 5: 97-102. DOI: <https://doi.org/10.25236/AJCIS.2022.050513>. Disponível em: <https://francis-press.com/papers/6807>. Acesso em: 20 fev 2025.

- Jahn, U., Hess, D., Stampf, M., Sutorma, A., Röhrig, C., Schulz, P., & Wolff, C. (2020). A Taxonomy for Mobile Robots: Types, Applications, Capabilities, Implementations, Requirements, and Challenges. **Robotics**, 9, 109. DOI: <https://doi.org/10.3390/robotics9040109>. Disponível em: <https://www.mdpi.com/2218-6581/9/4/109>. Acesso em: 13 dez 2024.
- Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. **Electronic Markets**, 31, 685 - 695. DOI: <https://doi.org/10.1007/s12525-021-00475-2>. Disponível em: <https://link.springer.com/article/10.1007/s12525-021-00475-2>. Acesso em: 9 maio 2025.
- JAYASINGHE, Oshada; HEMACHANDRA, Sahan; ANHETTIGAMA, Damith; KARIYAWASAM, Shenali; WICKREMASINGHE, Tharindu; EKANAYAKE, Chalani; RODRIGO, Ranga; JAYASEKARA, Peshala. Towards Real-time Traffic Sign and Traffic Light Detection on Embedded Systems. **2022 IEEE Intelligent Vehicles Symposium (IV)**, [S.L.], p. 723-728, 5 jun. 2022. IEEE. DOI: <http://dx.doi.org/10.1109/iv51971.2022.9827355>. Disponível em: <https://ieeexplore.ieee.org/document/9827355>. Acesso em: 9 maio 2025.
- Jordan, M., & Mitchell, T. (2015). Machine learning: Trends, perspectives, and prospects. **Science**, 349, 255 - 260. DOI: <https://doi.org/10.1126/science.aaa8415>. Disponível em: <https://www.science.org/doi/10.1126/science.aaa8415>. Acesso em: 6 jun 2025.
- KAUR, Ishmeet; JADHAV, Adwaita Janardhan. Survey on Computer Vision Techniques for Internet-of-Things Devices. **2023 IEEE International Conference On Industry 4.0, Artificial Intelligence, And Communications Technology (Iaict)**, [S.L.], p. 244-250, 13 jul. 2023. IEEE. DOI: <http://dx.doi.org/10.1109/iaict59002.2023.10205899>. Disponível em: <https://ieeexplore.ieee.org/document/10205899>. Acesso em: 6 jun 2025.
- KLENERT, David; FERNÁNDEZ-MACÍAS, Enrique; ANTÓN, José-Ignacio. Do robots really destroy jobs? Evidence from Europe. **Economic And Industrial Democracy**, [S.L.], v. 44, n. 1, p. 280-316, 26 jan. 2022. SAGE Publications. DOI: <http://dx.doi.org/10.1177/0143831x211068891>. Disponível em: <https://journals.sagepub.com/doi/10.1177/0143831X211068891>. Acesso em: 6 jun 2025.
- Krishna Kumar Mohbey, Malika Acharya, **Basics of Python Programming: A Quick Guide for Beginners**, Bentham Science Publishers (2023). Disponível em: <https://doi.org/10.2174/97898151796371230101>. https://www.eurekaselect.com/ebook_volume/3629. Acesso em: 6 jun 2025.
- Luo, Y., Ci, Y., Jiang, S. et al. A novel lightweight real-time traffic sign detection method based on an embedded device and YOLOv8. **J Real-Time Image Proc** 21, 24 (2024). DOI: <https://doi.org/10.1007/s11554-023-01403-7>. Disponível em: <https://link.springer.com/article/10.1007/s11554-023-01403-7>. Acesso em: 25 out 2024.
- Majeed, A., & Rauf, I. (2020). Graph Theory: A Comprehensive Survey about Graph Theory Applications in Computer Science and Social Networks. **Inventions**. DOI: <https://doi.org/10.3390/inventions5010010>. Disponível em: <https://www.mdpi.com/2411-5134/5/1/10>. Acesso em: 26 out. 2024

MARQUES, Rafael; RIBEIRO, Tiago; LOPES, Gil; RIBEIRO, A.. YOLOv3: traffic signs & lights detection and recognition for autonomous driving. Proceedings Of The 14Th International Conference On Agents And Artificial Intelligence, [S.L.], p. 818-826, 2022.

SCITEPRESS - Science and Technology Publications. DOI:

<http://dx.doi.org/10.5220/0010914100003116>. Disponível em:

<https://www.scitepress.org/Link.aspx?doi=10.5220/0010914100003116>. Acesso em: 26 out. 2024.

MICROSOFT. Python in Visual Studio Code: August 2024 Release. [S.I.]. DevBlogs, 2024. Disponível em:

https://devblogs.microsoft.com/python/python-in-visual-studio-code-august-2024-release/?wt.mc_id=developermscom. Acesso em: 11 dez. 2024.

MORAR, Anca; MOLDOVEANU, Alin; MOCANU, Irina; MOLDOVEANU, Florica; RADOI, Ion Emilian; ASAVEI, Victor; GRADINARU, Alexandru; BUTEAN, Alex. A Comprehensive Survey of Indoor Localization Methods Based on Computer Vision.

Sensors, [S.L.], v. 20, n. 9, p. 2641, 6 may 2020. MDPI AG. DOI:

<http://dx.doi.org/10.3390/s20092641>. Disponível em:

<https://www.mdpi.com/1424-8220/20/9/2641>. Acesso em: 9 dez. 2024.

MORU, D. K.. The effects of camera focus and sensor exposure on accuracy using computer vision. Nigerian Journal Of Technology, [S.L.], v. 41, n. 3, p. 585-590, 2 nov. 2022. **African Journals Online (AJOL)**. DOI: <http://dx.doi.org/10.4314/njt.v41i3.19>. Disponível em: <https://www.ajol.info/index.php/njt/article/view/235328>. Acesso em: 30 jun. 2024.

Pandey, M. (2023). Computer Vision. **International Journal for Research in Applied Science and Engineering Technology**. DOI: <https://doi.org/10.22214/ijraset.2023.54701>. Disponível em: <https://www.ijraset.com/best-journal/computer-vision>. Acesso em: 6 jun 2025.

PETERS, James F. **Foundations of Computer Vision**. Winnipeg: Springer, 2017. E-book. Disponível em: <https://link.springer.com/book/10.1007/978-3-319-52483-2>. Acesso em 10 nov 2024.

Robinson, Nicole; TIDD, Brendan; CAMPBELL, Dylan; KULIĆ, Dana; CORKE, Peter. Robotic Vision for Human-Robot Interaction and Collaboration: A Survey and Systematic Review. **ACM Transactions on Human-Robot Interaction**, 2023. DOI: <https://doi.org/10.1145/3570731>. Disponível em: <https://dl.acm.org/doi/10.1145/3570731>. Acesso em: 6 jun 2025.

Rubio, F., Valero, F., & Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. **International Journal of Advanced Robotic Systems**, 16. DOI: <https://doi.org/10.1177/1729881419839596>. Disponível em: <https://journals.sagepub.com/doi/10.1177/1729881419839596>. Acesso em: 4 jun 2025.

SANTOS, Daniel Rodrigues dos; KHOSHELHAM, Kourosh. MAPEAMENTO 3D DE AMBIENTES INTERNOS USANDO DADOS RGB-D. **Boletim de Ciências Geodésicas**, [S.L.], v. 21, n. 3, p. 442-464, set. 2015. FapUNIFESP (SciELO). DOI: <http://dx.doi.org/10.1590/s1982-21702015000300025>. Disponível em: <https://www.scielo.br/j/bcg/a/rQ8JrZg9TxRWnm7XndsWYLN/?lang=pt>. Acesso em: 4 jun 2025.

- SOORI, Mohsen; DASTRES, Roza; AREZOO, Behrooz; JOUGH, Fooad Karimi Ghaleh. Intelligent robotic systems in Industry 4.0: a review. **Journal Of Advanced Manufacturing Science And Technology**, [S.L.], v. 4, n. 3, p. 2024007-2024007, 2024. Huatuo Culture Media Co., Limited. DOI: <http://dx.doi.org/10.51393/j.jamst.2024007>. Disponível em: <http://www.jamstjournal.com/en/article/doi/10.51393/j.jamst.2024007>. Acesso em: 4 jun 2025.
- SPORN, O. (2018). Graph theory methods: applications in brain networks. **Dialogues in Clinical Neuroscience**, 20, 111 - 121. DOI: <https://doi.org/10.31887/DCNS.2018.20.2/osporns>. Disponível em: <https://www.tandfonline.com/doi/full/10.31887/DCNS.2018.20.2/osporns>. Acesso em: 5 jun 2025.
- SUO, Jinli; ZHANG, Weihang; GONG, Jin; YUAN, Xin; BRADY, David J.; DAI, Qionghai. Computational Imaging and Artificial Intelligence: the next revolution of mobile vision. **Proceedings Of The Ieee**, [S.L.], v. 111, n. 12, p. 1607-1639, dez. 2023. Institute of Electrical and Electronics Engineers (IEEE). DOI: <http://dx.doi.org/10.1109/jproc.2023.3338272>. Disponível em: <https://ieeexplore.ieee.org/document/10355958>. Acesso em: 6 jun 2025.
- TANG, Yunchao; CHEN, Mingyou; WANG, Chenglin; LUO, Lufeng; LI, Jinhui; LIAN, Guoping; ZOU, Xiangjun. Recognition and Localization Methods for Vision-Based Fruit Picking Robots: a review. **Frontiers In Plant Science**, London, Uk, v. 11, n. 25, p. 110-126, 19 may 2020. Frontiers Media SA. DOI: <http://dx.doi.org/10.3389/fpls.2020.00510>. Disponível em: <https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2020.00510/full>. Acesso em: 6 jun 2025.
- ULTRALYTICS. YOLOv8 Overview. 2024. Disponível em: <https://docs.ultralytics.com/models/yolov8/#overview>. Acesso em: 22 out 2024.
- Voulodimos, A., Doulamis, N., Protopapadakis, E., Deep Learning for Computer Vision: A Brief Review, **Computational Intelligence and Neuroscience**, 2018, 7068349, 13 pages, 2018. DOI: <https://doi.org/10.1155/2018/7068349>. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1155/2018/7068349>. Acesso em: 12 set 2024.
- WANG, Chien-Yao; LIAO, Hong-Yuan Mark. YOLOv1 to YOLOv10: The fastest and most accurate real-time object detection systems. **Taiwan: Academia Sinica; National Taipei University of Technology; National Chung Hsing University**, 2024. DOI: <https://arxiv.org/abs/2408.09332>. Disponível em: <https://arxiv.org/abs/2408.09332>. 22 out 2024.
- WANG, Zhenzhou. Increasing the accuracy for computer vision systems by removing the noise. 2017 3Rd **International Conference On Control, Automation And Robotics (Iccar)**, [S.L.], p. 371-374, abr. 2017. IEEE. DOI: <http://dx.doi.org/10.1109/iccar.2017.7942720>. Disponível em: <https://ieeexplore.ieee.org/document/7942720>. Acesso em: 11 dez. 2024.
- WILLIAM, P.; CHOUBEY, Siddhartha; CHOUDHURY, Dipti Sahoo. Mobile robot path planning using an optimization algorithm. **Procedia Computer Science**, [S.L.], v. 165, p. 555-564, 2019. Elsevier BV. DOI: <http://dx.doi.org/10.1016/j.procs.2020.01.069>. Disponível

em: <https://www.sciencedirect.com/science/article/pii/S1877050920300776?via%3Dihub>. Acesso em: 6 jun 2025.

XIAO, Jiaping; ZHANG, Rangya; ZHANG, Yuhang; FEROSKHAN, Mir. Vision-based Learning for Drones: a survey. **Arquivx: Computer Science > Robotics**, Ssi, v. 1, n. 1, p. 1-1, dez. 2023. ArXiv. DOI: <http://dx.doi.org/10.48550/ARXIV.2312.05019>. Disponível em: <https://arxiv.org/abs/2312.05019>. Acesso em: 5 jun 2025.

YUAN, Lu; CHEN, Dongdong; CHEN, Yi-Ling; CODELLA, Noel; DAI, Xiyang; GAO, Jianfeng; HU, Houdong; HUANG, Xuedong; LI, Boxin; LI, Chunyuan. Florence: a new foundation model for computer vision. **Arquivx: Computer Science > Computer Vision And Pattern Recognition**, Si, v. 1, n. 1, p. 1-1, nov. 2021. ArXiv. DOI: <http://dx.doi.org/10.48550/ARXIV.2111.11432>. Disponível em: <https://arxiv.org/abs/2111.11432>. Acesso em: 4 jun 2025.

ZOU, Zhengxia; CHEN, Keyan; SHI, Zhenwei; GUO, Yuhong; YE, Jieping. Object Detection in 20 Years: a survey. **Proceedings Of The Ieee**, [S.L.], v. 111, n. 3, p. 257-276, mar. 2023. Institute of Electrical and Electronics Engineers (IEEE). DOI: <http://dx.doi.org/10.1109/jproc.2023.3238524>. Disponível em: <https://ieeexplore.ieee.org/document/10028728>. Acesso em: 4 jun 2025.