

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Victor Hugo Girardi

Sistema de monitoramento de vagas em estacionamento utilizando câmera PTZ

Blumenau
2025

Victor Hugo Girardi

Sistema de monitoramento de vagas em estacionamento utilizando câmera PTZ

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.
Orientador: Prof. Dr. Ciro André Pitz

Blumenau
2025

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Girardi, Victor Hugo
Sistema de monitoramento de vagas em estacionamento utilizando câmera PTZ / Victor Hugo Girardi ; orientador, Ciro André Pitz, 2025.
64 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Blumenau, Graduação em Engenharia de Controle e Automação, Blumenau, 2025.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Câmera PTZ. 3. Detecção de Veículos. 4. Estacionamento Inteligente. 5. Visão Computacional. I. Pitz, Ciro André. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Victor Hugo Girardi

Sistema de monitoramento de vagas em estacionamento utilizando câmera PTZ

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Controle e Automação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 23 de outubro de 2025.

Banca Examinadora:



Documento assinado digitalmente
Ciro Andre Pitz
Data: 31/10/2025 12:51:11-0300
CPF: ***.728.089-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Ciro André Pitz, Dr.
Universidade Federal de Santa Catarina



Documento assinado digitalmente
Carlos Roberto Moratelli
Data: 31/10/2025 12:59:30-0300
CPF: ***.999.369-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Carlos Roberto Moratelli, Dr.
Universidade Federal de Santa Catarina



Documento assinado digitalmente
Marcos Vinicius Matsuo
Data: 31/10/2025 12:26:08-0300
CPF: ***.580.029-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Marcos Vinicius Matsuo, Dr.
Universidade Federal de Santa Catarina

Dedico este trabalho à minha família, em especial aos meus pais, Viviane e Juvanildo, que foram meus pilares em toda a jornada, e à minha namorada, Giovana, pelo apoio constante.

AGRADECIMENTOS

Agradeço aos meus pais, Viviane e Juvanildo, por serem a base da minha jornada, oferecendo apoio, força e todo o suporte emocional e financeiro, sempre com muito amor. Vocês foram fundamentais para a conclusão dessa etapa, sem vocês não conseguiria chegar a este momento tão especial.

Agradeço a todos os meus familiares, em especial aos meus avós — Vera, Jair, Cecília e Silvestre — que me apoiaram de diversas maneiras e foram fundamentais em todo o meu crescimento.

À minha companheira, Giovana, por me incentivar em todos os momentos com muito amor, sendo imprescindível e estando ao meu lado ao longo de toda a minha jornada. Você foi essencial não apenas para a conclusão deste trabalho, mas também para a minha formação pessoal.

Ao meu orientador, Ciro, agradeço por todo o apoio e auxílio no desenvolvimento deste trabalho. Seus ensinamentos e orientações foram fundamentais para a realização deste projeto.

“Inteligência é a capacidade de se adaptar à mudança.” (Stephen Hawking)

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema para monitoramento de vagas de estacionamento utilizando uma câmera PTZ e técnicas de visão computacional. O objetivo é oferecer uma alternativa de menor custo e maior escalabilidade em comparação a sistemas tradicionais baseados em sensores individuais por vaga. O sistema proposto consiste no uso de um modelo treinado para detecção de veículos, definição de regiões de interesse correspondentes às vagas e integração com a API Tuya para controle da câmera. Uma API REST é implementada para disponibilizar o estado das vagas em tempo real. A validação do sistema é realizada através de imagens estáticas, vídeos anotados e fluxo RTSP. Nas etapas com imagens e vídeos são utilizadas métricas como precisão, revocação e F1-score, enquanto no RTSP é realizada análise qualitativa do desempenho. Os resultados demonstram a viabilidade da solução proposta e evidenciam limitações relacionadas ao critério de decisão adotado e às restrições do equipamento utilizado.

Palavras-chave: Câmera PTZ; Detecção de Veículos; Estacionamento Inteligente; Visão Computacional; YOLO.

ABSTRACT

This work presents the development of a system for parking space monitoring using a PTZ camera and computer vision techniques. The objective is to provide a lower-cost and more scalable alternative compared to traditional systems based on individual sensors for each parking spot. The method consists of using a trained model for vehicle detection, defining regions of interest corresponding to the parking spaces, and integrating the Tuya API for camera control. A REST API is implemented to provide the occupancy status of the spaces in real time. Validation is carried out using static images, annotated videos, and RTSP streaming. In the image and video stages, metrics such as precision, recall, and F1-score are applied, while in the RTSP test the performance is analyzed qualitatively. The results confirm the feasibility of the proposed solution and highlight limitations related to the adopted decision criterion and the restrictions of the equipment used.

Keywords: Computer Vision; PTZ Camera; Smart Parking; Vehicle Detection; YOLO.

LISTA DE FIGURAS

Figura 1 – Detecção de ocupação de vagas de estacionamento utilizando sensores magnéticos. 16
Figura 2 – Detecção de ocupação de vagas de estacionamento utilizando sensor infravermelho. 17
Figura 3 – Detecção de vagas de estacionamento usando técnicas de visão computacional. 18
Figura 4 – Câmera IP alimentada por um *switch PoE*. 19
Figura 5 – Aquisição de Imagem. 20
Figura 6 – Câmera Dome. 22
Figura 7 – Câmera Bullet. 22
Figura 8 – Câmera PTZ. 23
Figura 9 – Eixos de movimentação *pan* e *tilt* de câmera PTZ. 25
Figura 10 – Influência da distância focal sobre o FoV. 26
Figura 11 – Exemplo de arquitetura de uma rede neural artificial totalmente conectada. 27
Figura 12 – Representação do conceito de campo receptivo em CNNs. 28
Figura 13 – Exemplo da operação de convolução em uma CNN. 28
Figura 14 – Exemplo da operação de *max pooling*. 29
Figura 15 – Exemplo de aprendizado hierárquico em redes convolucionais. 29
Figura 16 – Detecção de veículos e vagas com YOLO. 30
Figura 17 – Relatórios do treinamento na Ultralytics. 32
Figura 18 – Câmera interna Wi-Fi FHD PTZ WEG Home. 34
Figura 19 – Diagrama da arquitetura do projeto. 35
Figura 20 – Fluxograma do projeto. 37
Figura 21 – Exemplo de anotação nas vagas. 40
Figura 22 – Exemplo de imagem utilizada no treinamento. 41
Figura 23 – Detecção de vagas. 42
Figura 24 – Relatórios do treinamento na Ultralytics. 45
Figura 25 – Matriz de confusão normalizada no conjunto de validação. 46
Figura 26 – Exemplo de teste do modelo. 48
Figura 27 – Estacionamento da UFSC Blumenau. 48
Figura 28 – Posição da câmera. 49
Figura 29 – Exemplo de teste do modelo com vídeo. 50
Figura 30 – Sistema em funcionamento na posição 1. 51
Figura 31 – Sistema em funcionamento na posição 2. 51
Figura 32 – Falso negativo pelo ângulo da vaga. 52
Figura 33 – Falso positivo pela cor da superfície. 53

Figura 34 – Falso positivo pela cor da superfície. 53
Figura 35 – Falso negativo pela cor do carro. 54
Figura 36 – Falso negativo pela ROI e posição. 54
Figura 37 – Falso positivo pela borda da vaga com possível sujeira. 55
Figura 38 – Falha no fluxo RTSP. 55

LISTA DE TABELAS

Tabela 1 – Bibliotecas e componentes de *software* do protótipo. 34
Tabela 2 – Resumo do treinamento. 45
Tabela 3 – Desempenho do modelo ajustado ao domínio (30 imagens; 583 vagas). . 47
Tabela 4 – Desempenho do modelo genérico (COCO), classes: `car`, `truck`, `bus` e `motorbike`/`motorcycle`. 47
Tabela 5 – Desempenho em vídeo (N=90) com o modelo ajustado ao domínio (`best.pt`). 49
Tabela 6 – Desempenho em vídeo (N=90) com o modelo genérico `yolo11s.pt` (COCO); classes consideradas: `car`, `truck`, `bus`, `motorbike`/`motorcycle`. 49

SUMÁRIO

1 INTRODUÇÃO 14
1.1 OBJETIVOS DO TRABALHO 15
1.1.1 Objetivos Específicos 15
1.2 ORGANIZAÇÃO DO TRABALHO 15
2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 SISTEMAS DE MONITORAMENTO DE ESTACIONAMENTO . . . 16
2.1.1 Sensores para detecção de veículos 16
2.1.1.1 Sensores magnéticos 16
2.1.1.2 Sensores ultrassônicos 17
2.1.1.3 Sensores infravermelhos 17
2.1.1.4 Câmeras 18
2.1.1.5 Comparação entre diferentes abordagens 18
2.2 CÂMERAS DE SEGURANÇA 19
2.2.1 Aquisição de Imagem 20
2.2.2 Protocolo ONVIF 20
2.2.3 Plataforma Tuya 21
2.2.4 Modelos de Câmeras 21
2.2.5 Câmeras PTZ 22
2.2.5.1 Uso de Câmera PTZ para monitoramento de vagas de estacionamento 23
2.2.5.2 Movimentação e controle de câmeras PTZ 24
2.3 TÉCNICAS DE PROCESSAMENTO DE IMAGENS PARA DETECÇÃO DE VEÍCULOS 26
2.3.1 Redes Neurais Artificiais 26
2.3.1.1 Redes Neurais Convolucionais 27
2.3.2 You Only Look Once 30
2.3.3 Ultralytics 31
2.3.4 Dataset 31
2.3.5 Google Colab 32
2.3.6 OpenCV 32
3 SISTEMA PROPOSTO 33
3.1 *HARDWARE* UTILIZADO: CÂMERA PTZ WI-FI FHD 33
3.2 PROJETO DE SOFTWARE 34
3.2.1 Bibliotecas e componentes de software 34
3.2.2 Visão geral da arquitetura de Software 35
3.2.3 Integração Tuya e RTSP 36
3.2.4 Mapeamento de vagas por posição 39
3.2.5 Treinamento do modelo 40

3.2.6 Detecção e decisão de ocupação 42
3.2.7 API REST de consulta 42
3.2.8 Publicação do projeto de *software* 43
4 RESULTADOS E DISCUSSÕES 44
4.1 METODOLOGIA DE AVALIAÇÃO 44
4.2 TREINAMENTO DO MODELO 44
4.3 IMAGENS ESTÁTICAS 46
4.4 VÍDEO/RTSP 47
4.4.1 Cenário e posicionamento da câmera 47
4.4.2 Resultados em vídeo 49
4.4.3 Resultado com fluxo RTSP 50
4.5 ANÁLISE DOS ERROS 52
4.5.1 Erros em Imagens Estáticas 52
4.5.2 Erros em vídeos 53
4.5.3 Erros no fluxo RTSP 55
4.5.4 Limitações 56
5 CONCLUSÃO 57
REFERÊNCIAS 58

1 INTRODUÇÃO

A crescente demanda por eficiência na utilização de espaços urbanos tem feito crescer a busca por soluções tecnológicas inovadoras no gerenciamento de estacionamentos. Com o aumento da população e a urbanização das cidades, vagas de estacionamento em locais de grande circulação, como shoppings, centros comerciais e edifícios corporativos, têm-se tornado escassas. Segundo a Organização das Nações Unidas (ONU), até 2050 cerca de 68% da população mundial viverá em áreas urbanas, o que implica em um aumento significativo da demanda por infraestrutura urbana eficiente, como estacionamentos (UNITED NATIONS, 2018). A procura por vagas se torna uma tarefa frustrante e demorada, causando aumento nos congestionamentos e, consequentemente, um maior consumo de combustível. Em áreas urbanas movimentadas, a procura contribui significativamente para o aumento de tráfego, cerca de 34% (HAMPSHIRE; SHOUP, 2018). O cenário ambiental também é diretamente impactado, tendo em vista que os veículos geram gases poluentes durante a procura por vagas, principalmente no aumento das emissões de CO_2

A qualidade de vida dos motoristas é diretamente afetada, pois perdem tempo em longas buscas por lugares para estacionar. A falta de informações em tempo real sobre a disponibilidade de vagas causa ineficiência no uso dos estacionamentos. Além disso, estacionamentos com sensores em cada vaga têm um grande custo de implementação e requerem um projeto de engenharia muito maior. Diante desse cenário, torna-se necessário o investimento em soluções que otimizem a utilização de estacionamentos urbanos.

Estudos têm demonstrado a eficácia de sistemas inteligentes de estacionamento baseados em tecnologias como visão computacional e internet das coisas (*IoT - Internet of Things*). O uso de redes neurais combinado com algoritmos de detecção de objetos tem se mostrado uma solução eficiente para detecção em tempo real da ocupação de vagas; além disso, fornece informações precisas aos motoristas (NITHYA *et al.*, 2022).

Tecnologias de monitoramento, como câmeras PTZ (*Pan-Tilt-Zoom*) integradas a algoritmos de visão computacional, oferecem soluções inovadoras para o acompanhamento inteligente de vagas disponíveis. O protocolo ONVIF (*Open Network Video Interface Forum*) padroniza a comunicação entre aplicação e câmera, permitindo empregar equipamentos de diferentes fabricantes sem alterações de código específicas (ONVIF, 2025b). Complementarmente, plataformas de IoT em nuvem, como a Tuya, disponibilizam APIs e SDKs para controle remoto e orquestração de dispositivos, leitura de estados e integração com serviços de *backend*. Enquanto o ONVIF provê a interoperabilidade no nível do dispositivo, a Tuya simplifica o gerenciamento, a autenticação e a integração do parque de câmeras em aplicações web/serviços, facilitando a implementação do sistema (TUYA INC., 2023). Essas tecnologias trazem ganhos diretos na experiência do usuário ao diminuir o tempo de busca por vagas, além de oferecer aos administradores do estacionamento ferramentas para supervisão e análise operacional.

1.1 OBJETIVOS DO TRABALHO

O principal objetivo deste trabalho é desenvolver um sistema inteligente de mapeamento de vagas em um estacionamento utilizando uma câmera PTZ em conjunto com um algoritmo de visão computacional.

1.1.1 Objetivos Específicos

Os objetivos específicos deste trabalho são listados a seguir:

- Implementar um algoritmo de visão computacional capaz de detectar e identificar a ocupação de vagas em imagens capturadas por câmera PTZ.
- Configurar e integrar uma câmera PTZ com suporte ao protocolo da Tuya, permitindo o controle remoto e a captura das imagens de diferentes ângulos.
- Treinar um modelo para identificação de vaga ocupada ou vazia.
- Implementar uma saída do sistema com a disponibilização de uma API REST de consulta.
- Avaliar a eficácia do sistema proposto em diferentes condições de luminosidade e ângulo de captação, comparando os resultados obtidos com o estado real das vagas.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado da seguinte forma. No Capítulo 2, apresentam-se os fundamentos teóricos que dão suporte ao desenvolvimento do sistema proposto, abordando conceitos de visão computacional, redes neurais e o funcionamento de câmeras PTZ. No Capítulo 3, é descrito o desenvolvimento do sistema, incluindo o treinamento do modelo, a integração com a câmera PTZ e a disponibilização do status das vagas por meio de API. No Capítulo 4, são apresentados e discutidos os resultados obtidos, considerando imagens estáticas, vídeos e fluxo RTSP (*real-time streaming protocol*), bem como a análise de erros identificados. Por fim, no Capítulo 5, é apresentada a conclusão, destacando as contribuições do trabalho, suas limitações e propostas para estudos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos e tecnologias necessários para o desenvolvimento do sistema proposto. São abordados temas relacionados aos sistemas de monitoramento de estacionamento, funcionamento de câmeras PTZ, bem como às técnicas de processamento de imagem para detecção de veículos.

2.1 SISTEMAS DE MONITORAMENTO DE ESTACIONAMENTO

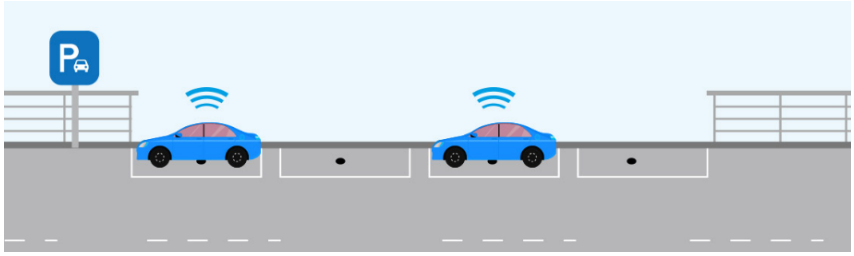
A dificuldade de encontrar vagas de estacionamento agrava o tráfego urbano e impacta negativamente a mobilidade, sendo responsável por aumentar cerca de 34% dos congestionamentos (HAMPSHIRE; SHOUP, 2018). Para reduzir esses problemas, sistemas de monitoramento de vagas surgem como uma solução eficiente ao disponibilizar informações em tempo real (JABBAR; TIEW; SHAH, 2024). Esses sistemas não apenas otimizam a utilização das vagas existentes, mas também reduzem o tempo de busca, o consumo de combustível e as emissões de poluentes. Outras funcionalidades, como reserva de vagas, tarifação dinâmica e gestão preditiva de demanda, são permitidas a partir dessas informações e alinham-se com os princípios da mobilidade urbana inteligente (ALKHEDER; RAJAB; ALZOUBI, 2016).

2.1.1 Sensores para detecção de veículos

2.1.1.1 Sensores magnéticos

Os sensores magnéticos detectam os veículos a partir de alterações no campo magnético terrestre, causadas pelas partes metálicas dos automóveis. Instalados no solo, como pode ser visto na Figura 1, esses sensores oferecem baixo consumo de energia, facilidade de instalação e boa eficácia (ZHANG *et al.*, 2022).

Figura 1 – Detecção de ocupação de vagas de estacionamento utilizando sensores magnéticos.



Fonte: Adaptado de (MOKOSMART, 2022).

A principal limitação dos sensores magnéticos é a sua suscetibilidade a interferências externas, especialmente as provenientes de campos magnéticos gerados por linhas

de alta tensão, sistemas ferroviários, além de veículos estacionados em vagas próximas. Apesar dessa limitação, sua aplicação é considerada adequada para o monitoramento de estacionamentos urbanos, com exemplos práticos em Shenzhen, na China (ZHANG *et al.*, 2022).

2.1.1.2 Sensores ultrassônicos

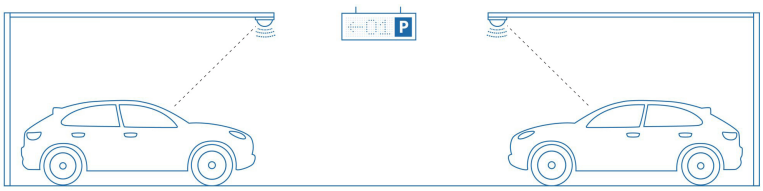
Os sensores ultrassônicos são amplamente utilizados na detecção de vagas de estacionamento, sendo uma solução de baixo custo. O princípio de funcionamento é baseado na emissão de ondas sonoras de alta frequência e na medição do tempo decorrido entre a transmissão do sinal ultrassônico e o recebimento do sinal refletido. A partir da medida de tempo, calcula-se a distância entre o sensor e o objeto responsável pela reflexão do sinal ultrassônico. Em seguida, a medida dessa distância é utilizada para verificar se há ou não algum automóvel estacionado.

Entre as principais vantagens dos sensores ultrassônicos estão a simplicidade de operação e a boa exatidão na medição de distâncias curtas, abaixo de 5 metros. Porém, devido à baixa resolução angular, esses sensores apresentam dificuldade na detecção das bordas de objetos, afetando a capacidade de delimitar exatamente o tamanho e o formato das vagas (SHAO; CHEN; CAO, 2018).

2.1.1.3 Sensores infravermelhos

Sensores infravermelhos também são utilizados em aplicações de detecção de vagas de estacionamento, como pode ser visto na Figura 2, principalmente em ambientes onde as condições de iluminação variam significativamente. O funcionamento é baseado na emissão de radiação infravermelha e na detecção da radiação refletida por objetos próximos, como veículos. Quando um automóvel entra no campo de visão do sensor, a radiação infravermelha é emitida e capturada por um receptor (KUMAR *et al.*, 2023). Esse sensor é utilizado principalmente em estacionamentos ao ar livre, onde ocorrem grandes mudanças de luz natural ao longo do dia. No entanto, sua eficácia pode ser afetada por obstáculos que bloqueiam o campo de visão, como outros veículos ou estruturas físicas.

Figura 2 – Detecção de ocupação de vagas de estacionamento utilizando sensor infravermelho.



Fonte: (PARKSOL, 2025).

2.1.1.4 Câmeras

O uso de câmeras para detecção de vagas de estacionamento tem se consolidado, sendo uma alternativa eficiente para estacionamentos externos. De modo geral, o uso de câmeras proporciona um custo menor em comparação com os sensores tradicionais, tendo em vista que, dependendo das especificações da câmera, é possível cobrir uma área grande de estacionamento. Com o posicionamento adequado, uma única câmera é capaz de substituir dezenas de sensores individuais, tornando o sistema mais econômico. Nesse contexto, câmeras PTZ podem auxiliar em estacionamentos maiores, ou até mesmo, para obter maior assertividade, capturando imagens de posições diferentes.

A detecção de vagas com câmeras é realizada por meio de técnicas de visão computacional, que permitem interpretar as imagens e identificar a ocupação, conforme exemplificado na Figura 3. Redes neurais são amplamente utilizadas nesse processo, pois são capazes de reconhecer padrões e diferenciar vagas livres de vagas ocupadas. Os maiores desafios das câmeras estão relacionados às condições climáticas adversas, sendo necessária uma iluminação e campo de visão adequados, além da escolha correta de equipamento, de acordo com a necessidade do local.

Figura 3 – Detecção de vagas de estacionamento usando técnicas de visão computacional.



Fonte: (ALMEIDA, P. R. de *et al.*, 2015).

2.1.1.5 Comparação entre diferentes abordagens

Cada abordagem de detecção de veículos tem suas vantagens e limitações, que devem ser consideradas dependendo das necessidades específicas do ambiente do estacionamento. Os sensores magnéticos são eficientes em termos de custo e exatidão, mas podem ser afetados por interferências externas, como campos magnéticos de linhas de alta tensão ou veículos estacionados próximos (ZHANG *et al.*, 2022).

Os sensores ultrassônicos oferecem baixo custo e boa exatidão em curtas distâncias, mas, devido à baixa resolução angular, apresentam dificuldades na detecção das bordas

dos veículos (SHAO; CHEN; CAO, 2018). Os sensores infravermelhos são indicados para situações de variação de iluminação, sendo eficazes ao ar livre. Porém, obstruções no campo de visão podem afetar sua eficácia (KUMAR *et al.*, 2023). Além disso, os sensores são instalados individualmente por vaga, gerando a necessidade de uma infraestrutura de comunicação.

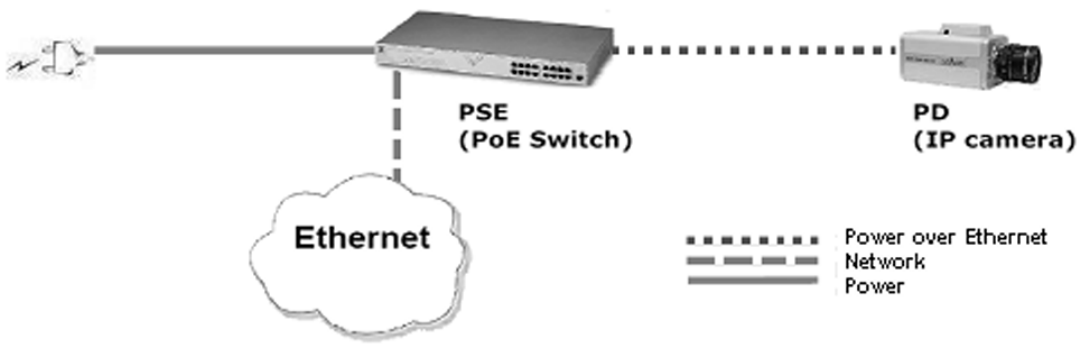
As câmeras, por outro lado, oferecem uma cobertura ampla e a capacidade de monitorar múltiplas vagas simultaneamente. Quando combinadas com técnicas de visão computacional e redes neurais, permitem uma detecção com grande eficácia, principalmente quando são do tipo PTZ. A maior dificuldade das câmeras são as questões de iluminação e de estarem posicionadas de maneira correta (KUMAR *et al.*, 2023).

2.2 CÂMERAS DE SEGURANÇA

As câmeras de segurança são componentes essenciais em sistemas de vigilância moderna, sendo amplamente adotadas em ambientes residenciais, comerciais e públicos devido à sua capacidade de dissuasão de crimes, registro de evidências e monitoramento em tempo real (VIGNE *et al.*, 2011). Elas capturam e transmitem imagens para sistemas de gravação, como NVRs *(Network Video Recorders)* ou servidores em nuvem, além de permitirem a visualização remota via dispositivos móveis.

Essas câmeras podem ser classificadas conforme a tecnologia de transmissão utilizada. As câmeras IP, por exemplo, operam em redes digitais Ethernet ou Wi-Fi e oferecem alta resolução (até 8MP/4K), compressão eficiente (e.g., H.265) e funcionalidades inteligentes, como análise de vídeo embarcada. Diferentemente das analógicas, que dependem de cabos coaxiais e *DVRs* para digitalização, as câmeras IP utilizam *switches* PoE *(Power over Ethernet)*, que consolidam energia e dados em um único cabo, conforme ilustrado na Figura 4. Já as câmeras analógicas HD (HD-TVI/AHD) atingem resoluções de até 1080p (WEG S.A., 2025), sendo uma opção de custo reduzido para migração de sistemas legados.

Figura 4 – Câmera IP alimentada por um *switch PoE*.

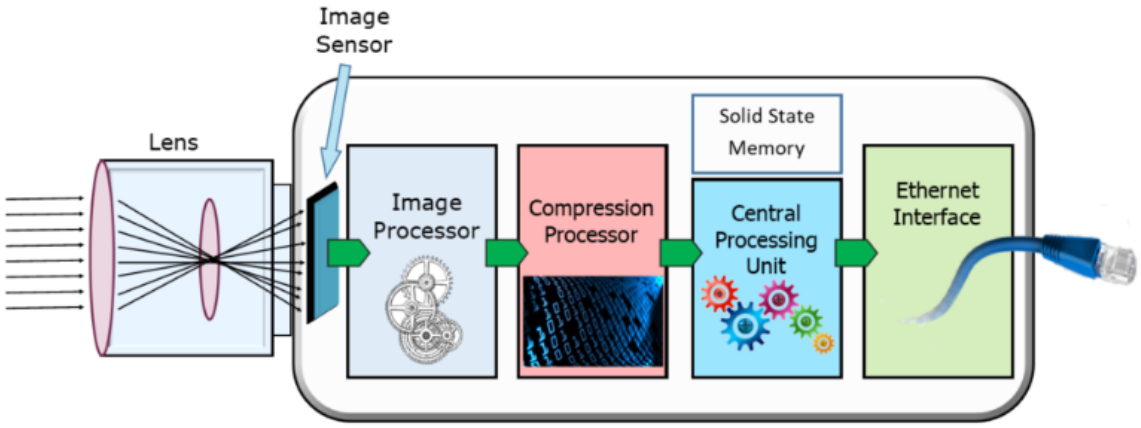


Fonte: (POPOVIC *et al.*, 2013).

2.2.1 Aquisição de Imagem

O processo de aquisição de imagens em câmeras IP envolve três etapas principais: captura, processamento interno e transmissão. Inicialmente, as lentes e o filtro óptico direcionam a luz para o sensor CCD ou CMOS, que converte a intensidade luminosa em sinais elétricos, gerando quadros (frames) (POPOVIC *et al.*, 2013). Em seguida, circuitos embarcados digitalizam e comprimem o vídeo (e.g., H.264/H.265), aplicando ajustes como correção de cor e redução de ruído para otimizar a largura de banda. Por fim, os dados são transmitidos por meio de protocolos de rede padronizados (e.g., ONVIF/Tuya) ou proprietários para servidores de armazenamento ou sistemas de análise (ONVIF, 2025b; TUYA INC., 2023). A Figura 5 ilustra o processo citado acima.

Figura 5 – Aquisição de Imagem.



Fonte: (KINTRONICS, 2022).

2.2.2 Protocolo ONVIF

Sistemas de videomonitoramento e dispositivos de segurança eletrônica dependem de protocolos de comunicação que padronizam a troca de dados e permitem o controle remoto de funcionalidades. Entre os mais utilizados, destaca-se o ONVIF, voltado à integração padronizada de câmeras IP, e que será o foco deste estudo, em virtude de sua ampla adoção e relevância na interoperabilidade entre dispositivos de diferentes fabricantes.

O ONVIF é um padrão global amplamente adotado na indústria de segurança eletrônica para assegurar a interoperabilidade entre câmeras IP, NVRs e *softwares* de gerenciamento de vídeo. Esse protocolo define um conjunto de padrões de comunicação baseados em *web services* (SOAP/XML), permitindo que dispositivos de diferentes fabricantes operem de forma integrada e oferecendo maior flexibilidade na implementação de soluções de monitoramento (ONVIF, 2025b).

Para facilitar a compatibilidade e a especificação dos dispositivos, o ONVIF é dividido em Perfis, que representam conjuntos de funcionalidades obrigatórias e opcionais implementadas por cada dispositivo (ONVIF, 2025a). Entre os principais perfis utilizados em aplicações de videomonitoramento, destacam-se:

- **Perfil S:** voltado para *streaming* de vídeo ao vivo, configuração de vídeo, controle PTZ, detecção de movimento e eventos. É o perfil mais comum em câmeras IP e NVRs.
- **Perfil T:** evolução do Perfil S, inclui suporte para compressão *H.265*, metadados de vídeo e analíticos embarcados, como detecção de objetos, rostos e leitura de placas veiculares OCR (*Optical Character Recognition*).
- **Perfil G:** voltado para gravação e reprodução de vídeo em dispositivos de borda, permitindo que clientes, como softwares VMS *(Video Management System)*, acessem gravações diretamente dos dispositivos.
- **Perfil M:** direcionado a metadados e analíticos de vídeo, incluindo dados de inteligência artificial e eventos analíticos estruturados, sendo comum em câmeras com recursos de detecção inteligente.

2.2.3 Plataforma Tuya

A Tuya é uma plataforma de integração de dispositivos conectados que disponibiliza recursos de controle remoto de câmeras e automação por meio de serviços em *cloud*, com autenticação por chaves do projeto e chamadas a *endpoints* regionais (TUYA INC., 2023). No contexto deste trabalho, a Tuya é utilizada para acionar *presets* de câmeras do tipo PTZ antes das inferências, permitindo reposicionar o enquadramento de forma reprodutível e independente do fabricante, enquanto o vídeo é acessado via fluxo local.

Além do acionamento de *presets*, a plataforma oferece comandos discretos de movimento (*up*, *down*, *left*, *right*) e de *zoom*, bem como consulta de estado do dispositivo. Assim, a Tuya complementa o uso de padrões abertos (e.g., ONVIF) ao simplificar a orquestração e o gerenciamento dos dispositivos em aplicações de *backend* e *web*.

2.2.4 Modelos de Câmeras

As câmeras de segurança podem ser diferenciadas pelo seu formato físico. Entre os modelos mais comuns utilizados em sistemas de vídeo monitoramento, destacam-se:

- **Câmeras Dome:** possuem formato semiesférico, como pode ser visto na Figura 6 e são instaladas, geralmente, no teto. Sua principal vantagem está na discrição e no amplo campo de visão, já que o invólucro transparente dificulta ao observador externo identificar a direção para a qual a lente está apontando. São indicadas para ambientes internos, como shoppings e escritórios, devido à estética e resistência ao vandalismo (ABESE, 2025).

- **Câmeras Bullet:** de formato cilíndrico alongado, representadas na Figura 7, são facilmente identificáveis e, por isso, muitas vezes usadas em áreas externas. Destacam-se pela capacidade de captação em longa distância (AVIGILON, 2025).
- **Câmeras PTZ:** permitem movimentação horizontal, vertical e ajuste de zoom remotamente, possibilitando cobertura dinâmica de grandes áreas sem necessidade de múltiplas unidades (AVIGILON, 2025). A Figura 8 apresenta um exemplo de câmera PTZ.

Além desses modelos, diversos outros formatos de câmeras estão disponíveis no mercado; entretanto, o foco deste estudo recai sobre as câmeras do tipo PTZ.

Figura 6 – Câmera Dome.



Fonte: (WEG, 2025).

Figura 7 – Câmera Bullet.



Fonte: (WEG, 2025).

2.2.5 Câmeras PTZ

As câmeras PTZ diferenciam-se das câmeras fixas pela capacidade de alterar o campo de visão dinamicamente, graças ao controle motorizado nos eixos horizontal *(pan)*,

Figura 8 – Câmera PTZ.



Fonte: (WEG, 2025).

vertical *(tilt)* e na aproximação por meio do zoom óptico. Essa flexibilidade permite que uma única câmera cubra uma área ampla e adapte sua cobertura de acordo com a necessidade, reduzindo a quantidade de dispositivos necessários (MICHELONI; RINNER; FORESTI, 2010)

Outra característica relevante é a integração dessas câmeras com protocolos de comunicação, como o ONVIF e Tuya, que permitem sua configuração e controle em *softwares* de gerenciamento de vídeo. Isso possibilita automatizar varreduras pré-programadas de áreas de interesse, além de realizar o acompanhamento em tempo real de eventos detectados por algoritmos de análise de vídeo (ONVIF, 2025a; TUYA INC., 2023).

O uso de câmeras PTZ também está associado a aplicações em ambientes amplos, como estacionamentos, estádios e vias urbanas, onde há necessidade de monitoramento dinâmico. Sua capacidade de aproximação por zoom óptico, sem perda significativa de qualidade, favorece a identificação de detalhes, como placas veiculares e a movimentação de pessoas (AVIGILON, 2025).

2.2.5.1 Uso de Câmera PTZ para monitoramento de vagas de estacionamento

As câmeras PTZ representam uma evolução significativa em sistemas de vigilância baseados em vídeo, especialmente no contexto de monitoramento de estacionamentos. Diferentemente das câmeras fixas, as PTZ permitem uma cobertura dinâmica de grandes áreas, podendo ser reposicionadas remotamente para focar em regiões de interesse específicas, como vagas individuais ou veículos em movimento (LEE *et al.*, 2005). Essa flexibilidade é particularmente útil em estacionamentos de grande porte, onde a instalação de múltiplas câmeras fixas pode ser economicamente inviável ou logisticamente complexa.

Em ambientes externos, os desafios incluem variações de iluminação, oclusões

parciais e a necessidade de capturar imagens suficientemente detalhadas para identificação veicular. A principal vantagem da câmera PTZ neste contexto é sua capacidade de adaptar o campo de visão e utilizar o zoom óptico para obter imagens de alta resolução de uma vaga ou veículo específico sob demanda, superando a limitação de resolução de uma câmera fixa que tenta cobrir uma área muito ampla (MARTÍN NIETO *et al.*, 2019). Para sistemas que exigem identificação detalhada (e.g., leitura de placas), o uso de PTZs é quase mandatório.

Além disso, quando integradas a sistemas de visão computacional, as câmeras PTZ podem executar movimentações de forma autônoma, ajustando o enquadramento em tempo real de acordo com a detecção de eventos relevantes. Por exemplo, algoritmos de análise de vídeo podem identificar a entrada de um veículo em determinada região e acionar automaticamente a câmera PTZ para acompanhar sua trajetória até a vaga escolhida, realizando o *zoom* quando necessário para obter maior nível de detalhe. Essa capacidade de resposta dinâmica transforma o monitoramento em um processo mais eficiente e reduz a dependência de intervenção humana (DAVANI; AL-HADRUSI; SARHAN, 2021).

2.2.5.2 Movimentação e controle de câmeras PTZ

A movimentação de câmeras PTZ envolve três graus de liberdade: *pan* (eixo horizontal), *tilt* (eixo vertical) e *zoom*, como pode ser visto na Figura 9. Em termos geométricos, *pan* e *tilt* são rotações da câmera em torno de eixos ortogonais do referencial do sensor, enquanto o *zoom* altera a distância focal efetiva e, portanto, o campo de visão FoV - *field of view*.

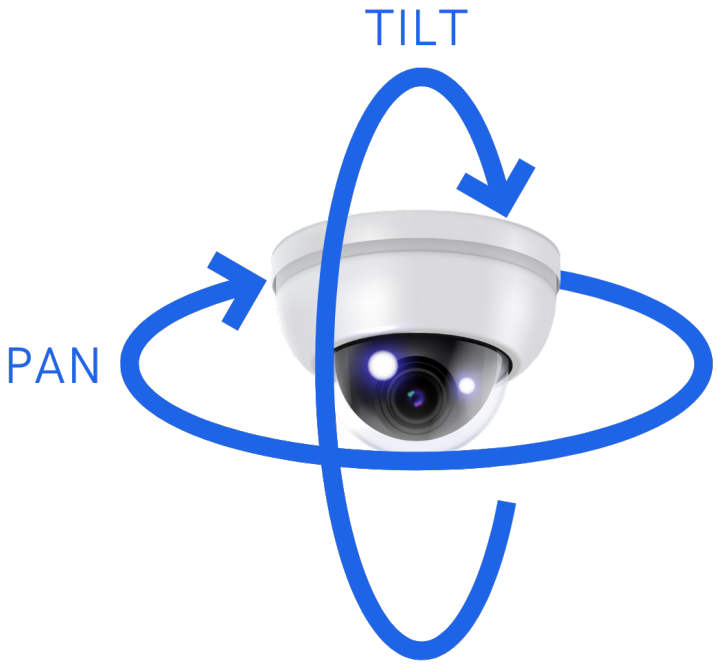
Do ponto de vista óptico, para um mesmo tamanho de sensor, a distância focal da lente determina diretamente o FoV. Distâncias focais curtas (e.g., 2,8 mm ou 3,6 mm) produzem um FoV, cobrindo uma área maior com menos detalhes por objeto. Por outro lado, distâncias focais longas (e.g., 12 mm) resultam em um FoV mais estreito, capturando uma área menor, porém com maior detalhamento (maior densidade de pixels por metro)(SAFWAN; MAHIM, 2025).

A Figura 10 mostra por que lentes varifocais, como as de câmeras PTZ (e.g., 2,8–12 mm), permitem ajustar o equilíbrio entre o FoV e o nível de detalhe de acordo com a necessidade do cenário, como é o caso de aplicações de visão computacional.

A movimentação das câmeras PTZ, utilizada para ajustar o campo de visão conforme a necessidade da aplicação, é realizada por meio de comandos enviados através de protocolos padronizados (e.g., ONVIF, Tuya) ou de interfaces proprietárias, como CGI ou SDK. Esses comandos podem ser executados tanto em uma rede local quanto via serviços em nuvem.

O ONVIF no perfil S define um serviço PTZ com operações normativas para movimentação absoluta, relativa e contínua, bem como leitura de estado e *presets*. Em particular, `AbsoluteMove` posiciona a câmera em uma coordenada-alvo de *pan*/*tilt*/*zoom*; `RelativeMove` aplica incrementos relativos à posição atual; `ContinuousMove` aceita um

Figura 9 – Eixos de movimentação *pan* e *tilt* de câmera PTZ.



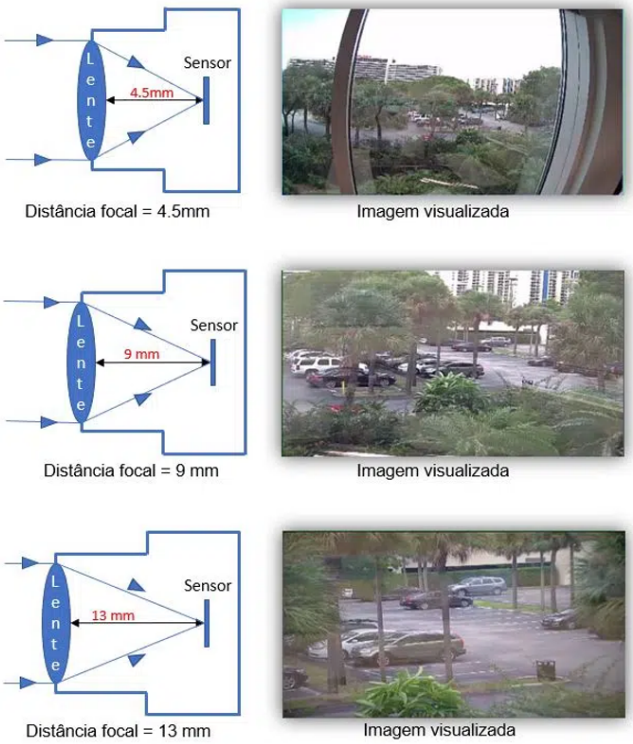
Fonte: (ISARSOFT GMBH, 2025).

vetor de velocidade (*pan*, *tilt*, *zoom*) aplicado por um tempo (`PTZTimeout`), devendo ser finalizado por `Stop`. O padrão também define `GetStatus`, `GetPresets`, `SetPreset`, `GotoPreset` e `SetHomePosition`/`GotoHomePosition`. Os sistemas de coordenadas são identificados por URIs (*spaces*); o *Generic Position Space* normaliza *pan*/*tilt* para $[-1, 1]$ e *zoom* para $[0, 1]$, enquanto os espaços de velocidade aceitam $[-1, 1]$. Para obter os limites e espaços suportados pela câmera, são consultados pelas operações `GetConfigurations` e `GetConfigurationOptions` (ONVIF, 2025a).

No ecossistema Tuya, o controle PTZ em nuvem ocorre pelo *endpoint REST* `POST /v1.0/cameras/{device_id}/configs/ptz`, com comandos discretos enumerados (e.g., `UP`, `DOWN`, `LEFT`, `RIGHT`, `STOP`). A plataforma também expõe SDKs para dispositivos móveis que encapsulam PTZ como *Data Points* (DPs), incluindo pan/tilt/stop e *zoom in/out/stop*, além de recursos como *presets* e *patrol/cruise* (TUYA INC., 2023).

Diferentemente do ONVIF (orientado à posição/velocidade normalizada e disponível em rede local), a plataforma da Tuya é orientada a comandos discretos e requer conectividade com a nuvem e autenticação do dispositivo no projeto. Já em protocolos proprietários, os fabricantes disponibilizam interfaces HTTP/CGI e SDKs com recursos não contemplados em protocolos padronizados, incluindo modos avançados de posicionamento e automações de rastreamento.

Figura 10 – Influência da distância focal sobre o FoV.



Fonte: (APRENDA CFTV, 2025).

2.3 TÉCNICAS DE PROCESSAMENTO DE IMAGENS PARA DETECÇÃO DE VEÍCULOS

A detecção automática de veículos e vagas de estacionamento é um problema clássico de visão computacional, que combina técnicas de processamento de imagens com modelos de aprendizado profundo. Nesse contexto, o processamento de imagens atua na preparação e manipulação de dados visuais — como filtragem, redimensionamento e transformações geométricas — de modo a fornecer informações adequadas para algoritmos de reconhecimento de padrões (SZELISKI, 2010; BRADSKI; KAEHLER, 2008).

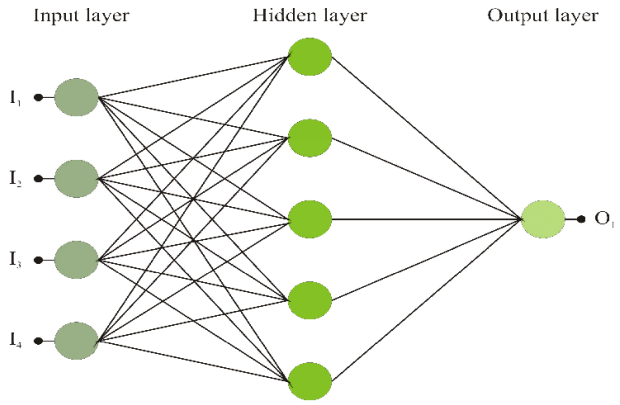
Nos últimos anos, o avanço das redes neurais convolucionais (CNNs) possibilitou o desenvolvimento de métodos de detecção em tempo real capazes de lidar com variações de iluminação, perspectiva e oclusões em ambientes complexos (LECUN; BENGIO; HINTON, 2015; HEIMBERGER *et al.*, 2021). Entre esses métodos, destacam-se arquiteturas da família YOLO (*You Only Look Once*), amplamente aplicadas em cenários de vigilância e monitoramento viário (REDMON *et al.*, 2015; JOCHER; QIU, 2024).

2.3.1 Redes Neurais Artificiais

As redes neurais artificiais (RNAs) são modelos computacionais inspirados na estrutura do cérebro humano, compostos por unidades chamadas neurônios artificiais. Cada neurônio recebe entradas, pondera-as por meio de pesos, aplica um termo de viés

e processa o resultado por uma função de ativação, que introduz não linearidade ao sistema. A organização em camadas: entrada, ocultas e saída, permite às RNAs aprender representações complexas de dados, ilustrada na Figura 11. Esse aprendizado ocorre por meio do ajuste iterativo dos pesos, em que algoritmos como o gradiente descendente e a retropropagação do erro atualizam os parâmetros a fim de minimizar a função de custo (GOODFELLOW; BENGIO; COURVILLE, 2016).

Figura 11 – Exemplo de arquitetura de uma rede neural artificial totalmente conectada.



Fonte: Adaptado de (SANTIN *et al.*, 2015).

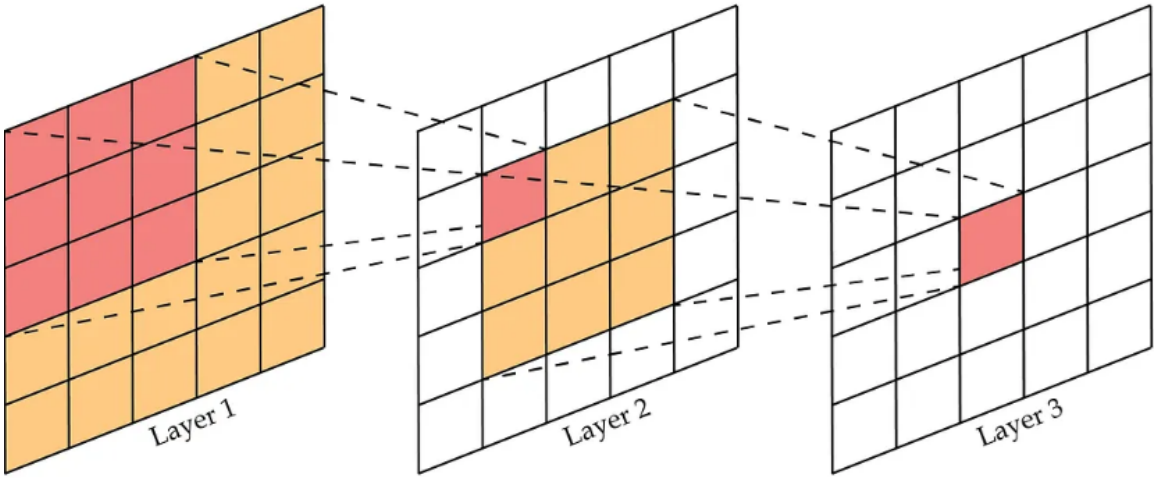
As RNAs têm ampla aplicação em reconhecimento de padrões, classificação e regressão, destacando-se em áreas como visão computacional, processamento de fala e previsão de séries temporais. Sua principal força reside na capacidade de generalizar a partir de exemplos, aprendendo representações de forma supervisionada ou não supervisionada. No contexto de visão computacional, redes neurais são utilizadas para identificar formas, objetos e padrões em imagens, estabelecendo as bases para arquiteturas mais avançadas, como as CNNs, que foram projetadas especificamente para lidar com dados visuais (LECUN; BENGIO; HINTON, 2015; SZELISKI, 2010).

2.3.1.1 Redes Neurais Convolucionais

As redes neurais convolucionais (CNNs) representam um dos maiores avanços da inteligência artificial aplicada à visão computacional. Diferentemente das redes neurais tradicionais, em que cada neurônio é conectado a todos os outros da camada anterior (como ilustrado na Figura 11), as CNNs utilizam o conceito de receptores locais, em que cada neurônio é conectado apenas a uma pequena região da entrada, formando o chamado campo receptivo. À medida que a rede se aprofunda, esses campos se expandem, permitindo capturar padrões mais complexos, como ilustrado na Figura 12 (LECUN; BENGIO; HINTON, 2015).

A principal característica das CNNs é a utilização de camadas convolucionais, responsáveis por aplicar filtros (*kernels*) sobre a imagem de entrada, como pode ser visto na Figura 13. Cada filtro percorre a imagem, operação conhecida como convolução, extraindo

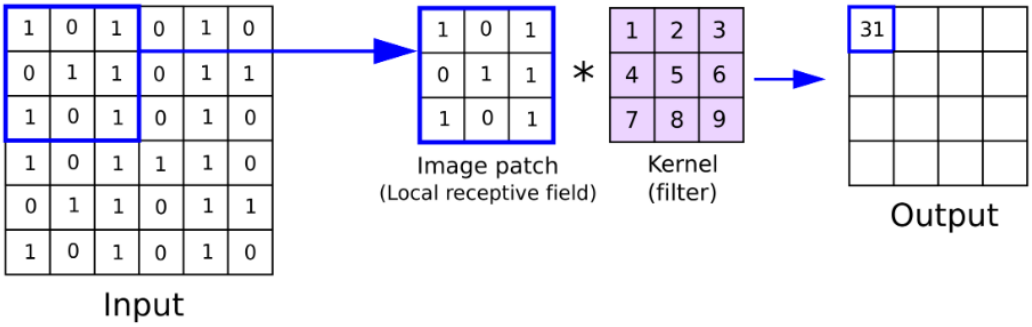
Figura 12 – Representação do conceito de campo receptivo em CNNs.



Fonte: Adaptado de (KALANTAR, 2023).

padrões específicos, como bordas, linhas, curvas ou texturas. Esses filtros são ajustados automaticamente durante o processo de treinamento, substituindo a etapa manual de extração de características que era comum em métodos clássicos de visão computacional (GOODFELLOW; BENGIO; COURVILLE, 2016).

Figura 13 – Exemplo da operação de convolução em uma CNN.

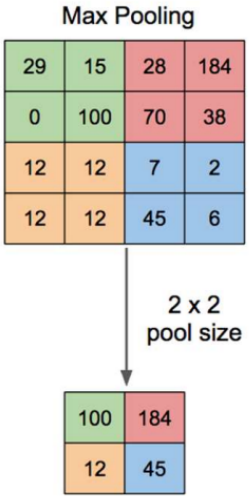


Fonte: (REYNOLDS, 2019).

Outro elemento fundamental das CNNs são as camadas de *pooling*, que reduzem a dimensionalidade dos mapas de características (*feature maps*). O *pooling* mais utilizado é o *max pooling*, que seleciona o valor máximo em uma região da imagem, ilustrado na Figura 14. Isso torna a rede mais robusta a pequenas variações de posição, iluminação ou escala de um objeto, o que é crucial em cenários de monitoramento de veículos, nos quais as condições do ambiente podem mudar significativamente.

Após diversas camadas convolucionais e de *pooling*, os dados são processados por camadas totalmente conectadas, responsáveis por integrar as informações extraídas e realizar a classificação ou regressão. No caso de detecção de veículos em estacionamentos,

Figura 14 – Exemplo da operação de *max pooling*.

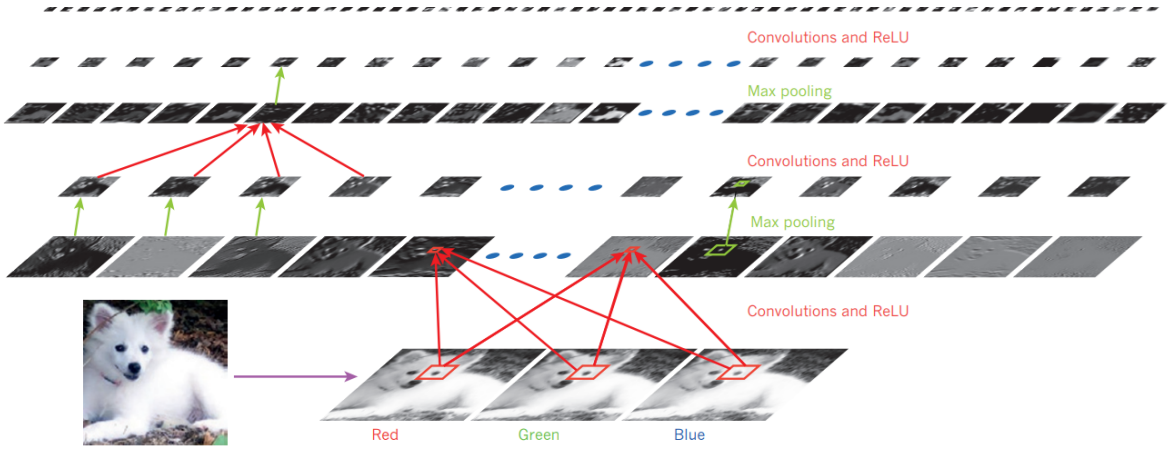


Fonte: Adaptado de (YANI; IRAWAN; SETININGSIH, 2019).

essas camadas finais são utilizadas para indicar a presença de um carro, sua posição e até mesmo a ocupação de uma vaga.

As CNNs apresentam uma propriedade chamada aprendizado hierárquico: nas primeiras camadas aprendem-se características simples, como bordas e linhas, enquanto nas camadas mais profundas são capturados padrões complexos, como rodas, faróis ou a silhueta de um automóvel. Essa capacidade torna as CNNs adequadas para lidar com os desafios da visão computacional moderna, como variações de perspectiva, iluminação e oclusões parciais (LECUN; BENGIO; HINTON, 2015), conforme ilustrado na Figura 15.

Figura 15 – Exemplo de aprendizado hierárquico em redes convolucionais.



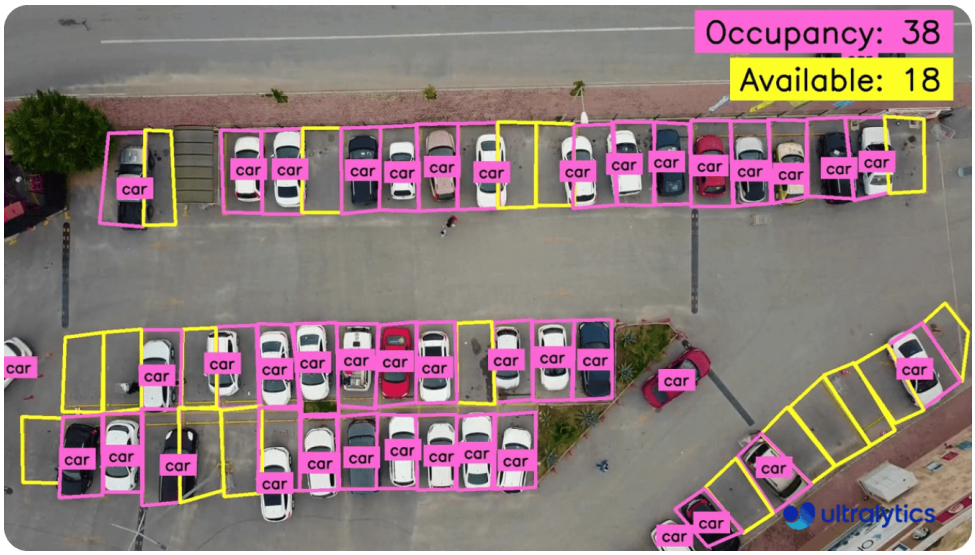
Fonte: (LECUN; BENGIO; HINTON, 2015).

2.3.2 You Only Look Once

A detecção de veículos em imagens e vídeos requer técnicas capazes de identificar objetos com rapidez e acurácia. Entre os algoritmos de maior destaque, encontra-se o YOLO (*You Only Look Once*), que se diferencia por tratar a tarefa de detecção de objetos como um problema de regressão direta. Em vez de utilizar múltiplas etapas, como proposto em abordagens tradicionais baseadas em *region proposals*, o YOLO realiza a previsão das caixas delimitadoras (*bounding boxes*) e das classes em uma única passada da rede neural convolucional (REDMON *et al.*, 2015). Essa característica torna o algoritmo extremamente rápido, possibilitando sua aplicação em cenários de monitoramento em tempo real, como o controle de veículos em estacionamentos, ilustrado na Figura 16.

O funcionamento do YOLO é baseado na divisão da imagem de entrada em uma grade (*grid*). Cada célula dessa grade é responsável por prever a presença de objetos, suas coordenadas e a probabilidade associada à classe. No caso da detecção de veículos, isso significa que o modelo é capaz de indicar simultaneamente a localização dos automóveis e distinguir se uma vaga está livre ou ocupada. Combinando camadas convolucionais profundas, o YOLO consegue extrair padrões hierárquicos de baixo e alto nível, identificando desde contornos simples até formas complexas, como a silhueta de um carro.

Figura 16 – Detecção de veículos e vagas com YOLO.



Fonte: (VINA, 2024).

Uma das principais vantagens do YOLO é sua capacidade de operar com baixa latência, mesmo em dispositivos com recursos computacionais limitados. Essa eficiência é crucial para aplicações de visão computacional em ambientes urbanos, nos quais o sistema precisa reagir rapidamente a mudanças na cena, como a entrada ou saída de veículos de um estacionamento. Além disso, sua arquitetura pode ser adaptada a diferentes tamanhos de modelos (e.g., versões *small*, *medium*, *large*), equilibrando velocidade e acurácia conforme as necessidades do projeto (JOCHER; QIU, 2024).

2.3.3 Ultralytics

A Ultralytics é a implementação de referência da família YOLO em Python, oferecendo uma interface unificada (CLI e API) para todo o ciclo de visão computacional: preparação de dados, treinamento, validação, avaliação, exportação e inferência em imagens e vídeo (JOCHER; QIU, 2024). Para detecção de veículos, disponibiliza modelos pré-treinados de diferentes portes (e.g., *small*, *medium*, *large*), possibilitando ajustar o equilíbrio entre acurácia e latência em cenários de monitoramento em tempo real.

No treinamento, a Ultralytics utiliza um arquivo `data.yaml` contendo caminhos de treino/validação e classes, além de um conjunto de hiperparâmetros que controlam otimização, tamanho da entrada e *data augmentation*. Esses elementos determinam a velocidade de convergência, a robustez a variações de cena e o compromisso entre acurácia e latência em detecção de veículos. Os principais elementos são listados a seguir:

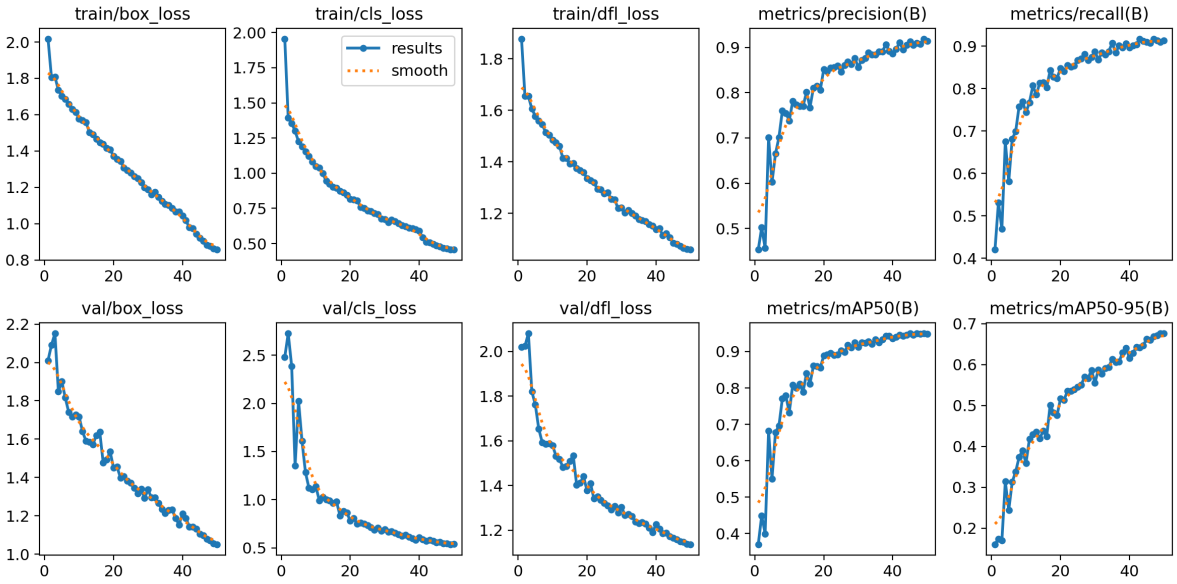
- ***Epochs* (`epochs`)**: número de passagens completas sobre o conjunto de treino; valores maiores tendem a aumentar a *mean Average Precision* (mAP) até certo ponto, mas elevam o risco de *overfitting*.
- **Tamanho do lote (`batch`)**: quantidade de imagens processadas em paralelo; *batches* maiores estabilizam o gradiente, porém exigem mais memória.
- **Resolução de entrada (`imgsz`)**: dimensão de entrada (e.g., `imgsz`=640); resoluções maiores ajudam na detecção de objetos pequenos, com maior custo de processamento.
- **Otimização e taxa de aprendizado** (`optimizer`, `lr0`, `lrf`, `warmup_epochs`): definem o regime de aprendizado, do estágio inicial ao estágio final, além do *warmup*; controlam a velocidade de convergência e a estabilidade do treino.
- **Regularização e perdas** (`weight_decay`; `box`, `cls`, `dfl`): `weight_decay` auxilia na generalização; `box`/`cls`/`dfl` ajustam a importância relativa das componentes da função de perda (caixa, classe e *Distribution Focal Loss*).

A avaliação ocorre a cada época no conjunto de validação, reportando métricas como mAP, curvas de perdas (caixa, classe e distribuição) e gráficos *precision–recall*, ilustrado na Figura 24. Em termos práticos, mais épocas tendem a melhorar a mAP até um platô; ganhos marginais e aumento do *gap* treino–validação indicam saturação ou *overfitting*. Para modelos *small* em vídeo em tempo real, valores típicos incluem `imgsz`=640, `batch` de dezenas de imagens e `epochs` na ordem de dezenas (e.g., 50).

2.3.4 Dataset

Um *dataset* (conjunto de dados) em visão computacional é uma coleção organizada de imagens acompanhadas de anotações que descrevem o que existe nelas. Para detecção de veículos, essas anotações costumam ser caixas delimitadoras (*bounding boxes*) e rótulos

Figura 17 – Relatórios do treinamento na Ultralytics.



Fonte: Elaborado pelo autor.

de classe (e.g., `car`), ou ainda polígonos/regiões de interesse (ROIs) das vagas quando o objetivo é indicar "livre/ocupada". Além das imagens e anotações, define-se uma divisão entre subconjuntos de treino, validação e teste, usada para treinar o modelo, ajustar hiperparâmetros e medir desempenho de forma justa (JOCHER; QIU, 2024).

2.3.5 Google Colab

O Google Colab é um ambiente de *notebooks* em nuvem que permite escrever e executar código Python no navegador, com acesso facilitado a GPUs/TPUs. Ele é voltado à ciência de dados e aprendizado de máquina, simplificando tarefas como instalar bibliotecas, carregar conjuntos de dados do Google Drive e acompanhar o treinamento de modelos em tempo real, sem precisar configurar uma máquina local com placa de vídeo. O Colab é amplamente utilizado em *deep learning*, pois disponibiliza uma placa de vídeo com bastante memória de vídeo, melhorando e reduzindo o tempo de treinamento de modelos (GOOGLE RESEARCH, 2025).

2.3.6 OpenCV

O OpenCV é uma biblioteca aberta para processamento de imagens e visão computacional. Ela oferece funções para ler fluxos de vídeo (e.g., RTSP), converter cores, redimensionar, filtrar ruído e desenhar elementos sobre os quadros (caixas, textos e polígonos). Em ambientes de estacionamento, o OpenCV é usado para pré-processar os *frames*, aplicar máscaras/ROIs das vagas e exibir, na própria imagem, as detecções e o estado de cada vaga. (BRADSKI; KAEHLER, 2008).

3 SISTEMA PROPOSTO

Neste capítulo apresenta-se o sistema proposto para monitoramento de vagas de estacionamento com uma única câmera PTZ, capaz de varrer múltiplas regiões de interesse por meio de posições pré-determinadas acionadas pela API da Tuya.

A solução compreende a aquisição de vídeo via RTSP e modelagem das vagas como regiões de interesse poligonais bidimensionais no plano da imagem definidas por uma lista ordenada de vértices x e y em *pixels* na resolução nativa. Além disso, o sistema proposto realiza a detecção automática de veículos por visão computacional e disponibiliza o estado de ocupação por meio de uma API REST.

Os conjuntos dessas regiões são organizados por posição pré-determinada da PTZ (uma coleção por posição) e armazenados em arquivos JSON com identificador único por vaga, assegurando rastreabilidade entre varreduras. A navegação entre pontos de observação utiliza posições configuradas no aplicativo Tuya e invocadas via nuvem; após cada reposicionamento, aplica-se um tempo de acomodação antes da captura para garantir estabilidade mecânica e fotométrica.

Nas seções seguintes detalham-se o *hardware* utilizado, a arquitetura de *software*, o procedimento de mapeamento de vagas, o modelo de detecção e os critérios de decisão, bem como integrações, desempenho e limitações do sistema, além do processo de treinamento do modelo.

3.1 *HARDWARE* UTILIZADO: CÂMERA PTZ WI-FI FHD

Para o desenvolvimento do projeto foi escolhida uma câmera interna Wi-Fi FHD PTZ da WEG Home, apresentada na Figura 18, com sensor CMOS de 1/2,7", resolução de 2 MP em 1920×1080, lente fixa de 4 mm sem controle de *zoom* e campo de visão aproximado de 95° na horizontal, 60° na vertical e 108° na diagonal. O conjunto óptico é auxiliado por IR (*Infrared*) de 850 nm com alcance efetivo de até 10 m, DNR 3D e WDR digital, o que melhora o desempenho do sistema em condições de baixa luminosidade. O corpo PTZ permite rotações de até 355° no eixo de *pan* e 110° no eixo de *tilt*, viabilizando cobrir múltiplas regiões de interesse com uma única unidade. O vídeo é codificado em H.264, com taxa de quadros adaptável de até 20 FPS, armazenável localmente e transmitido por rede Wi-Fi de 2,4 GHz.

Para integração utiliza-se a API da Tuya, que aciona posições pré-determinadas configuradas no aplicativo Tuya e, assim, reposiciona a PTZ entre diferentes regiões de observação. A aquisição do vídeo é feita via RTSP para processamento local. Como o acionamento de posição ocorre pela nuvem, adota-se um tempo de acomodação após cada comando para garantir estabilidade mecânica e fotométrica antes da captura.

Do ponto de vista do projeto, esses atributos do *hardware* se alinham a quatro frentes: cobertura espacial, desempenho fotométrico, orçamento de rede e latência. O

amplo curso de *pan* e *tilt* permite reapontar a mesma unidade para diferentes setores e reduzir o número de câmeras. Em baixa luminosidade, IR, DNR 3D e WDR digital estabilizam a fotometria do quadro, preservando a taxa de detecção. Por fim, o codificador H.264 com até 20 FPS mantém o fluxo estável com uso moderado de banda, viabilizando inferência quase em tempo real no *pipeline* proposto na seção 3.2.

Figura 18 – Câmera interna Wi-Fi FHD PTZ WEG Home.



Fonte: (WEG HOME, 2025).

3.2 PROJETO DE SOFTWARE

3.2.1 Bibliotecas e componentes de software

Para o desenvolvimento do protótipo, foi utilizada a linguagem Python em conjunto com um conjunto de bibliotecas especializadas que formam o *pipeline* de processamento do sistema. A Tabela 1 detalha os principais componentes de *software* empregados, suas versões e funções específicas no projeto.

Tabela 1 – Bibliotecas e componentes de *software* do protótipo.

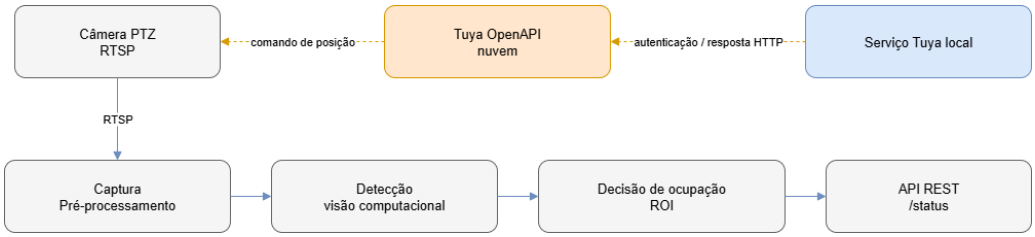
Componente	**Versão**	**Função**
Python	3.11.9	Linguagem/ambiente de execução.
Ultralytics (YOLO)	8.3.193 (`yolo11s.pt`)	Inferência de detecção de veículos.
OpenCV	4.12.0.88	Decodificação RTSP, operações de imagem e teste ponto–polígono.
NumPy	2.1.2	Estruturas numéricas e álgebra vetorial para o pipeline.
Flask	3.1.2	Exposição da API REST (`/status`).
tuya-connector	0.1.2	Integração com Tuya OpenAPI (acionar *presets* da PTZ).

3.2.2 Visão geral da arquitetura de Software

O diagrama da Figura 19 sintetiza a arquitetura lógica do sistema. A câmera PTZ fornece o fluxo de vídeo por RTSP. Um módulo de captura realiza o *grabbing* e o pré-processamento dos quadros. Em seguida, o módulo de detecção executa o modelo de visão computacional e o módulo decisor cruza as detecções com as regiões de interesse poligonais definidas por posição pré-determinada, inferindo o estado de ocupação por vaga. Por fim, um serviço HTTP expõe os resultados por meio de uma API REST.

A varredura do estacionamento é orquestrada por posições pré-determinadas acionadas pela API da Tuya via nuvem, que reposicionam a PTZ entre regiões de observação. Após cada comando de posição, aplica-se um tempo de acomodação para estabilização mecânica e fotométrica antes da captura. Dessa forma, o sistema opera com dois canais complementares: controle de posição pela Tuya OpenAPI e aquisição de vídeo local por RTSP.

Figura 19 – Diagrama da arquitetura do projeto.



Fonte: Elaborado pelo autor.

A Figura 20 descreve o fluxograma do sistema. A seguir, detalha-se cada componente mostrado no fluxograma:

1. **Início**: inicializa recursos do sistema e prepara o ambiente de execução.
2. **Carregar credenciais e configuração**: leitura de *Access ID*, *Access Key*, *endpoint* e *device id* da Tuya, além de parâmetros do *pipeline*. Inclui tempo de acomodação, ordem das posições pré-determinadas, limiares de detecção e mapeamentos dos arquivos JSON com as regiões de interesse.
3. **Autenticar na Tuya OpenAPI**: o cliente local estabelece sessão por HTTPS e obtém autorização para enviar comandos de posição.
4. **Selecionar posição pré-determinada**: define a próxima posição da PTZ na sequência de varredura e o conjunto de regiões de interesse correspondente.
5. **Enviar comando de posição Tuya OpenAPI para a câmera**: emissão do comando com o identificador da posição desejada. O dispositivo é acionado via nuvem, retornando um resultado em HTTP. Em seguida o sistema aguarda o tempo de acomodação antes da captura.

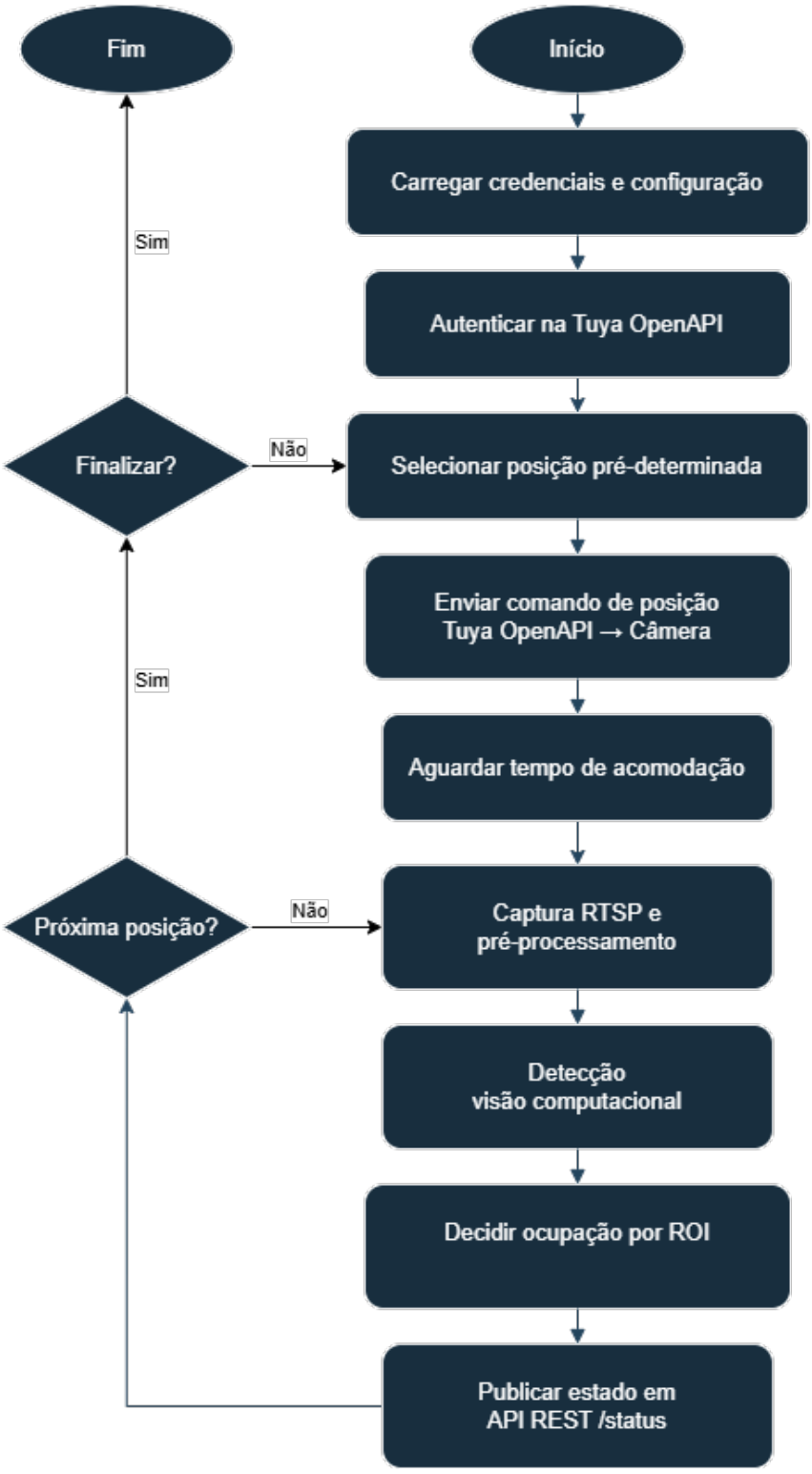
6. **Aguardar tempo de acomodação**: espera configurada para estabilização mecânica da PTZ e estabilização fotométrica do quadro.
7. **Captura RTSP e pré-processamento**: aquisição do quadro de vídeo, decodificação, conversão de espaço de cor e redimensionamento quando necessário.
8. **Detecção por visão computacional**: execução do modelo treinado, gerando caixas delimitadoras, escores e classes, com aplicação de um limiar de confiança.
9. **Decidir ocupação por ROI**: cruzamento entre detecções e regiões de interesse da posição ativa. As regiões são polígonos no plano da imagem definidos por lista ordenada de vértices x e y em *pixels* na mesma resolução do quadro avaliado. A regra de decisão usa o centro da caixa dentro do polígono, e o estado é atualizado a cada inferência respeitando o intervalo configurado.
10. **Publicar estado na API REST `/status`**: retorna JSON no formato `{"POSk": {"livres": [i..], "ocupadas": [j..]}}` para cada posição pré-determinada. Os índices referem-se à ordem das vagas no arquivo de regiões de interesse.
11. **Próxima posição**: se houver nova posição na sequência, segue; caso contrário, é obtido mais um quadro do vídeo.
12. **Finalizar**: se houver nova posição na sequência, retorna à seleção; caso contrário, encerra o ciclo.
13. **Fim**: encerramento controlado, com liberação de recursos de rede e fechamento do cliente da Tuya OpenAPI.

3.2.3 Integração Tuya e RTSP

Para habilitar o controle em nuvem, é criado um projeto no portal de desenvolvedor da Tuya em ambiente *cloud*, selecionando a categoria de câmeras e o centro de dados compatível com a região da conta do aplicativo móvel, pois essa escolha define o `API_ENDPOINT` utilizado no código. Após a criação do projeto, são registradas as credenciais `ACCESS_ID` e `ACCESS_KEY`. Em seguida, é vinculada a conta do aplicativo Tuya Smart ao projeto, de forma que os dispositivos da conta fiquem visíveis no projeto em nuvem; com isso, a câmera passa a constar na lista de dispositivos e é possível obter o seu `DEVICE_ID`. No próprio projeto, são autorizadas as APIs necessárias para dispositivos de câmera IP, incluindo as operações de *preset* expostas pela plataforma. Além disso, são publicadas as permissões para que as chamadas HTTP do cliente local sejam aceitas.

Com o projeto e as permissões ativas, são criadas no aplicativo Tuya as posições pré-determinadas usadas na varredura, nomeadas de modo consistente com o código (por exemplo, POS1 e POS2). Também é validado o acionamento manual dessas posições no aplicativo. A integração é então verificada ponta a ponta, executando uma chamada de `preset` a partir do cliente local, direcionada ao `DEVICE_ID` e ao identificador numérico da

Figura 20 – Fluxograma do projeto.



Fonte: Elaborado pelo autor.

posição configurada (1 para POS1 e 2 para POS2), observando a resposta HTTP e o movimento da câmera. No protótipo, `ACCESS_ID`, `ACCESS_KEY`, `API_ENDPOINT` e `DEVICE_ID`, bem como os identificadores das posições, são definidos no arquivo principal e repassados aos serviços na inicialização.

A configuração e as credenciais são definidas no arquivo principal e repassadas aos serviços na inicialização. O conjunto mínimo inclui `ACCESS_ID`, `ACCESS_KEY`, `API_ENDPOINT`, `DEVICE_ID` e `rtsp_url`. O trecho do código correspondente a essa etapa é apresentado no Código 1.

Código 1 – Trecho do main.py.

```
ACCESS_ID    = "..."   # client id do projeto Tuya
ACCESS_KEY   = "..."   # client secret do projeto Tuya
API_ENDPOINT = "https://openapi.tuyaus.com"
DEVICE_ID    = "..."   # id da camera no projeto

porta_rtsp = 5543
ip      = "192.168.0.xxx"
usuario= "admin"
senha   = "admin123"
rtsp_url = (
f"rtsp://{usuario}:{senha}@{ip}:{porta_rtsp}/live/channel0"
)
```

O módulo `tuya_service.py`, apresentado no Código 2, encapsula o diálogo com a Tuya OpenAPI. Na inicialização, cria-se um cliente `TuyaOpenAPI` com `endpoint`, `ACCESS_ID` e `ACCESS_KEY`, chamando `connect()` para estabelecer a sessão e armazenando o `DEVICE_ID`. O método `call_preset()` emite uma requisição `POST` com o comando `ipc_preset_action` e o identificador de posição convertido para *string*.

Código 2 – Trecho do tuya_service.py.

```
from tuya_connector import TuyaOpenAPI

class TuyaService:
    def __init__(
        self,
        access_id,
        access_key,
        endpoint,
        device_id
    ):
        self.openapi = TuyaOpenAPI(
            endpoint,
            access_id,
            access_key
```

```
        )
        self.openapi.connect()
        self.device_id = device_id
    def call_preset(self, preset_id: int):
        cmd = {
            "commands": [
                {
                    "code": "ipc_preset_action",
                    "value": str(preset_id)
                }
            ]
        }
        res = self.openapi.post(
            f"/v1.0/devices/{self.device_id}/commands",
            cmd
        )
        return res
```

3.2.4 Mapeamento de vagas por posição

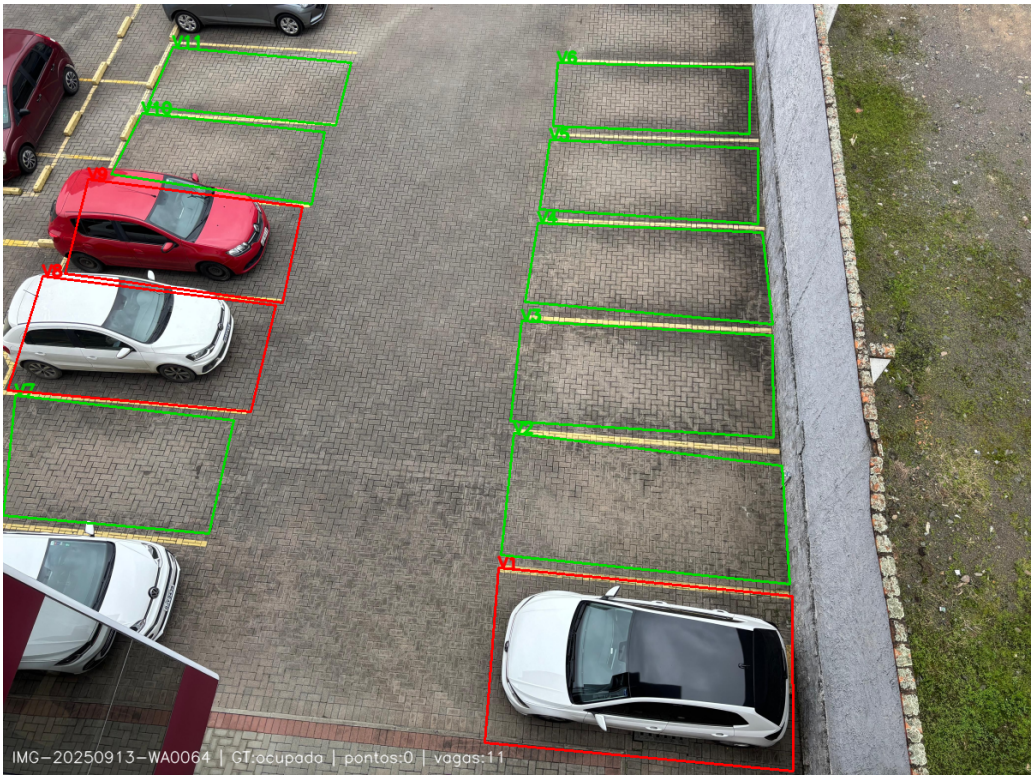
As vagas são descritas em arquivos JSON separados por posição da PTZ. Cada arquivo possui um objeto na raiz cuja chave identifica a coleção de vagas daquela posição, por exemplo `"1"`, mapeando para uma lista de objetos de vaga. Cada vaga contém o campo `points`, definido como uma lista ordenada de quatro vértices `[x,y]` em *pixels*, anotados na mesma resolução do quadro processado pelo RTSP ou pela imagem, como pode ser visto na Figura 21. A ordem das vagas na lista define os índices publicados no *endpoint* `/status` em `livres` e `ocupadas`. Um exemplo mínimo é mostrado no Código 3.

Código 3 – Arquivo JSON com as coordenadas das vagas.

```
{
  "1": [
    { "points": [[32,32],[11,150],[82,155],[94,32]] },
    { "points": [[110,35],[98,149],[152,153],[160,37]] }
  ]
}
```

O sistema mantém duas coleções em memória, uma por posição. O laço escolhe a coleção conforme o *preset* ativo, e a detecção só é executada se o tempo mínimo desde a última inferência desse *preset* (`infer_interval`) tiver sido atingido.

Figura 21 – Exemplo de anotação nas vagas.



Fonte: Elaborado pelo autor.

3.2.5 Treinamento do modelo

O treinamento utilizou um conjunto de imagens obtido no Roboflow (PROJECT, 2023), organizado em três partições: treino com 1618 imagens, validação com 215 imagens e teste com 84 imagens. O problema foi modelado com duas classes, `car` e `free`, conforme o arquivo `data.yaml`. As anotações seguem o formato aceito pela família YOLO, com caixas delimitadoras por instância de veículo. Foram incluídas amostras em diversos ângulos e condições de iluminação, como pode ser visto na Figura 22, contemplando sombras e variações de cena, com *auto-orient* aplicado no pré-processamento e redimensionamento para 640×640. O arquivo de configuração é apresentado no Código 4.

Código 4 – Arquivo de configuração do treinamento do modelo.

```
train: ../train/images
val:   ../valid/images
test:  ../test/images
nc: 2
names: ['car', 'free']
```

O treinamento ocorreu no Google Colab (GPU T4), partindo de `yolo11s.pt` com pesos pré-treinados e ajuste fino por 50 épocas, `batch` 40 e entrada 640. Os principais hiperparâmetros são descritos no Código 5.

Figura 22 – Exemplo de imagem utilizada no treinamento.



Fonte: (PROJECT, 2023).

Código 5 – Hiperparâmetros do treinamento.

```
model: yolo11s.pt
epochs: 50
batch: 40
imgsz: 640
pretrained: true
optimizer: auto
amp: true
seed: 0
warmup_epochs: 3
lr0: 0.01
lrf: 0.01
momentum: 0.937
weight_decay: 0.0005
iou: 0.7
max_det: 300
```

A validação é executada a cada época com `precision`, `recall` e `mAP`. Para operação no sistema, além das métricas globais, considera-se a taxa de falsos positivos/negativos por ROI (uma vez que a decisão final cruza o centro da `bbox` com o polígono da vaga). No protótipo, o limiar de confiança do serviço de detecção foi `conf_th = 0,30`, privilegiando a sensibilidade e delegando o controle de falsos positivos à etapa de decisão por ROI.

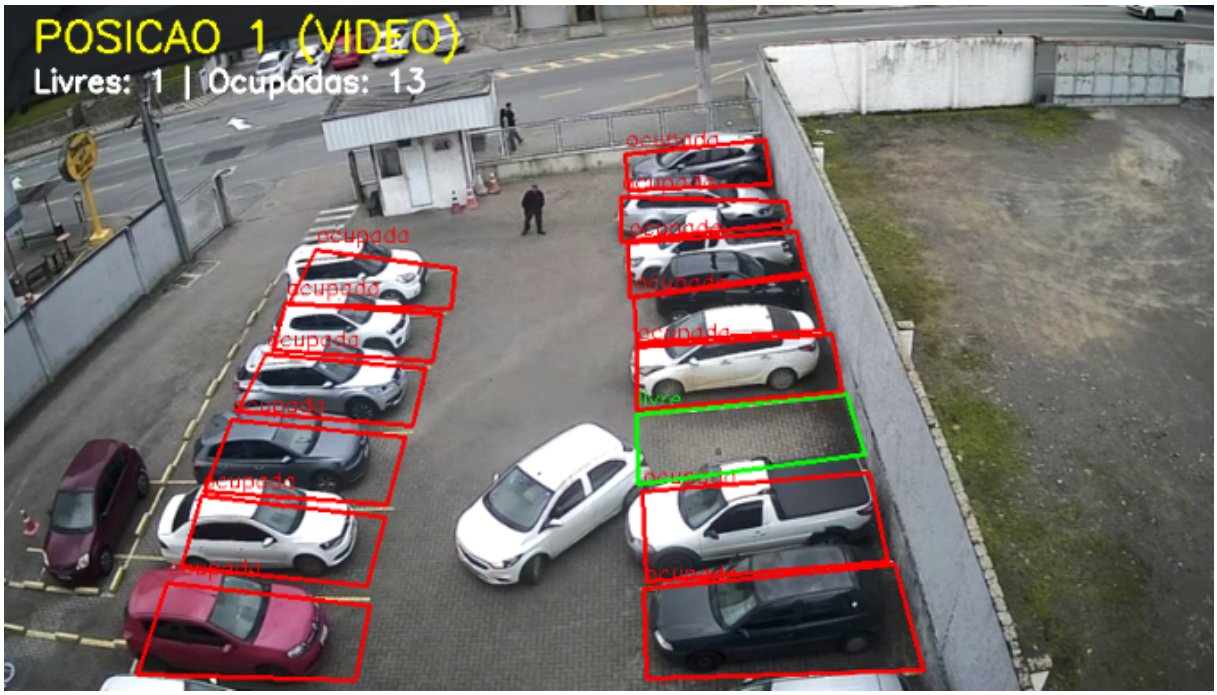
3.2.6 Detecção e decisão de ocupação

O módulo de detecção carrega o artefato `best.pt` (classes `car` e `free`) e filtra as detecções, mantendo apenas a classe `car`. O processamento usa cache por *preset* `last_status` e `last_time`) para reduzir custo computacional, executando a rede somente após o intervalo mínimo definido.

A decisão de ocupação é um cruzamento geométrico entre detecções e as ROIs da posição ativa. Cada vaga é um polígono convexo de quatro vértices `[x,y]` em *pixels*. Uma vaga é marcada como ocupada quando o centro da *bounding box* de uma detecção de carro está dentro (ou sobre a borda) do polígono da vaga. Vagas fora do formato esperado são ignoradas para a ocupação e tratadas como livres.

O resultado por posição é um dicionário com `livres` e `ocupadas`, cujos índices seguem a ordem das vagas no JSON. No laço principal, o resumo é publicado em `status_vagas["POS1"]` ou `status_vagas["POS2"]` conforme `preset_atual`, com proteção por `lock`. A sobreposição gráfica desenha os polígonos e rótulos `livre`/`ocupada`, apresentados na Figura 23, além dos totais por quadro, auxiliando a validação do *pipeline*.

Figura 23 – Detecção de vagas.



Fonte: Elaborado pelo autor.

3.2.7 API REST de consulta

O serviço HTTP é implementado em Flask (Python), executando localmente e respondendo em `application/json; charset=utf-8`. Para consumo externo, expõe um único *endpoint* que publica, em JSON, o estado de ocupação por *preset* da câmera PTZ.

Os índices das vagas seguem a ordem do arquivo de ROIs da posição correspondente. Dois exemplos de respostas são apresentados no Código 6.

Código 6 – Exemplo de resposta da API.

```
{
  "POS1": { "livres": [0,1,2,3], "ocupadas": [4,5,6,7,8,9,10,
  11,12,13,14,15] },
  "POS2": { "livres": [0,1,2,3,4,5,
  6,7,8,9], "ocupadas": [10,11,12,13] }
}
```

3.2.8 Publicação do projeto de *software*

O código-fonte do sistema é disponibilizado publicamente no repositório: https://github.com/vhugogirardi/TCC_MonitoramentoVagas. A organização do projeto segue uma estrutura modular, onde `main.py` coordena o fluxo principal, `vaga_service.py` implementa a detecção de veículos, `tuya_service.py` gerencia o controle da câmera PTZ via API Tuya e `api_service.py` expõe os dados via REST. O repositório inclui ainda diretórios para validação do modelo e resultados dos testes, com documentação de configuração e uso no `README.md`.

4 RESULTADOS E DISCUSSÕES

Este capítulo apresenta e discute os resultados do sistema proposto para monitoramento de vagas com câmera PTZ. Inicialmente, descreve-se a metodologia de avaliação. Em seguida, detalha-se o treinamento do modelo. Na sequência, apresentam-se os resultados em imagens estáticas e, depois, em vídeo/RTSP. Por fim, procede-se à análise de erros e à discussão comparativa e limitações, com ênfase nas implicações práticas.

4.1 METODOLOGIA DE AVALIAÇÃO

A avaliação foi conduzida em dois blocos: imagens estáticas com anotações por vaga e vídeos com quadros específicos anotados. No primeiro bloco, 30 imagens (583 vagas) foram avaliadas com entrada `imgsz`=640 e decisão por `centro-no-polígono`: uma vaga é prevista como ocupada se existir detecção `car` cujo centro da *bounding box* pertença ao polígono da vaga; caso contrário, é prevista como livre. Detecções fora das ROIs são ignoradas. Testaram-se três limiares de confiança: `conf` $\in \{0{,}30,\ 0{,}40,\ 0{,}50\}$.

As métricas reportadas por vaga foram: precisão, revocação, F1, além de contagens de verdadeiro positivo (TP), falso positivo (FP), falso negativo (FN) e verdadeiro negativo (TN). A precisão indica a proporção de previsões positivas que realmente são corretas, enquanto a revocação mede a capacidade do modelo de identificar todas as instâncias positivas. Já o F1 combina essas duas medidas por meio de sua média harmônica, oferecendo uma avaliação equilibrada entre a precisão e a revocação (ULTRALYTICS, 2025). Para o modelo ajustado (`best.pt`), considerou-se apenas a classe `car`; para o modelo genérico (`yolo11s.pt`) também foram aceitas `truck`, `bus` e `motorbike`/`motorcycle`.

No segundo bloco, quatro vídeos foram utilizados com quadros anotados por vaga (total $N = 90$). A inferência manteve `imgsz`=640, a mesma regra de decisão por polígono e os mesmos limiares de `conf`. Para cada vídeo/quadro avaliado, calcularam-se TP/FP/FN/TN por vaga e, em seguida, as métricas agregadas. Para transparência e inspeção qualitativa, geraram-se *overlays* por quadro com indicação textual de TP/FP/FN/TN, além do centro das detecções e do contorno das ROIs. Todas as avaliações utilizaram a mesma base de pesos e *pipeline* de pré-processamento, garantindo comparabilidade entre imagens estáticas e vídeo.

4.2 TREINAMENTO DO MODELO

O treinamento é realizado a partir do conjunto de dados público *Car Park Detection* disponível na plataforma Roboflow (PROJECT, 2023). O *dataset* contém 1917 imagens de estacionamentos anotadas com *bounding boxes*, divididas em três subconjuntos: *train* (1618 imagens, 84%), *validation* (215 imagens, 11%) e *test* (84 imagens, 4%). As anotações são fornecidas em duas classes: `car` e `free`.

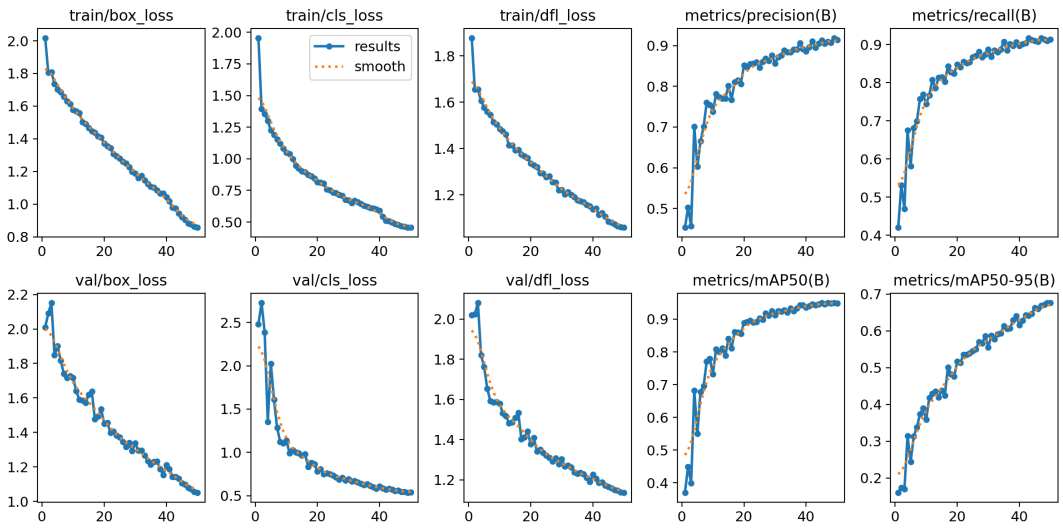
O modelo ajustado (`best.pt`) foi obtido por ajuste fino de `yolo11s.pt` sobre esse conjunto anotado. O treinamento utilizou GPU T4 no Google Colab com 50 épocas, `batch`=40 e `imgsz`=640, `warmup` de 3 épocas, `lr0`=0,01, `lrf`=0,01, `momentum`=0,937 e `weight_decay`=5e-4. O conjunto de validação foi avaliado a cada época, reportando mAP e curvas *precision–recall*. A Tabela 2 apresenta os resultados obtidos.

Tabela 2 – Resumo do treinamento.

Item	**Valor/Descrição**
Arquitetura/base	YOLOv11s com ajuste fino (`best.pt`)
Épocas / Batch / Img	50 / 40 / 640
Melhor época (val)	49 (mAP@50 máximo)
mAP@50 (val)	0,9498
mAP@50:95 (val)	0,676
Precisão (val)	0,9191
Revocação (val)	0,9095
Perdas (treino: início → fim)	`box_loss`: 2,01 → 1,06; `cls_loss`: 2,48 → 0,54; `dfl_loss`: 2,02 → 1,14

A Figura 24 apresenta os gráficos gerados pela Ultralytics durante o treinamento (perdas, mAP e curvas `precision-recall`). Observa-se tendência de queda das perdas de caixa/classe, estabilização do mAP nas últimas épocas e proximidade entre as curvas de treino e validação, sinalizando boa generalização ao domínio. O bom desempenho em validação se refletiu nos testes por vaga (Seção 3), sobretudo em revocação, quando comparado ao modelo genérico.

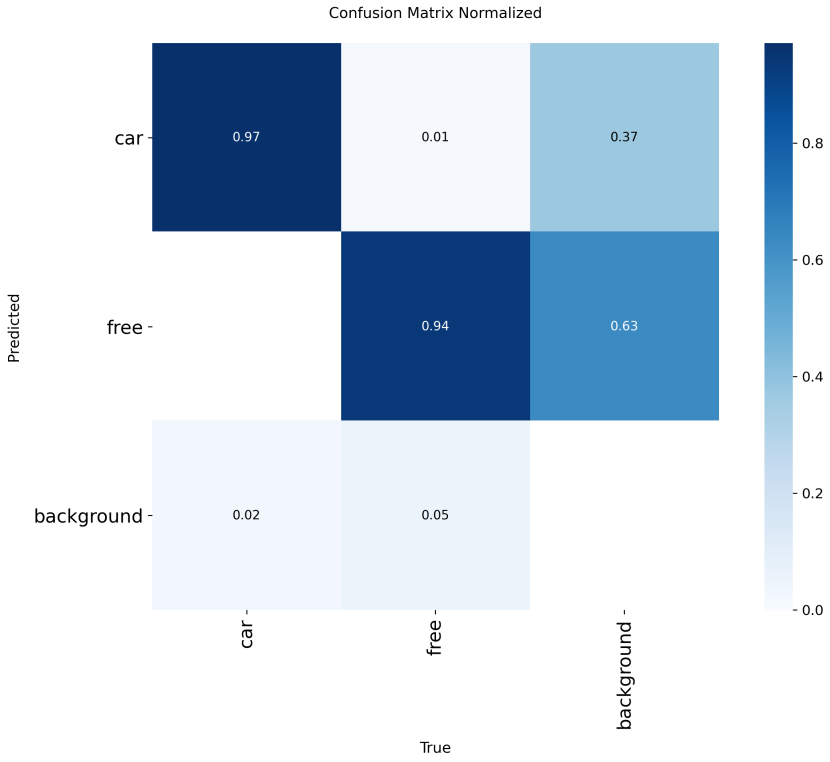
Figura 24 – Relatórios do treinamento na Ultralytics.



Fonte: Elaborado pelo autor.

A Figura 25 mostra a matriz de confusão normalizada por classe real no conjunto de validação. Observa-se alta acurácia nas classes de interesse: `car` com 0,97 e `free` com 0,94 na diagonal principal. Os principais erros concentram-se contra a classe `background` (e.g., 0,37 de `car` em `background` e 0,63 de `free` em `background`). No contexto do sistema, essas confusões têm impactos distintos: predições `background` fora das ROIs não afetam a decisão; já falsos positivos de `car` dentro de uma ROI podem marcar ocupação indevida. Esse efeito é mitigado pelo limiar de confiança escolhido na validação e pela regra "centroide da *bounding box* $\in$ ROI".

Figura 25 – Matriz de confusão normalizada no conjunto de validação.



Fonte: Elaborado pelo autor.

4.3 IMAGENS ESTÁTICAS

O conjunto de imagens estáticas foi composto por 30 imagens, totalizando 583 vagas anotadas com rótulos de referência ("livre"/"ocupada"). As inferências utilizaram entrada `imgsz`=640 e a regra de decisão `centro-no-polígono`: uma vaga é classificada como ocupada quando o centro da *bounding box* de algum veículo incide dentro do polígono da vaga; caso contrário, é classificada como livre. A Tabela 3 apresenta os resultados do modelo ajustado ao domínio (`best.pt`), e a Tabela 4, do modelo genérico (`yolo11s.pt`, COCO).

As detecções fora das ROIs são desconsideradas. Na decisão de ocupação, apenas a classe `car` é utilizada (isto é, a presença de `car` com centro da *bounding box* dentro do polígono implica vaga ocupada).

Tabela 3 – Desempenho do modelo ajustado ao domínio (30 imagens; 583 vagas).

Limiar (conf)	**Precisão**	**Revocação**	**F1**	**N**
0,30	0,981	0,880	0,928	583
0,40	0,988	0,853	0,916	583
0,50	0,988	0,816	0,894	583

Tabela 4 – Desempenho do modelo genérico (COCO), classes: `car`, `truck`, `bus` e `motorbike`/`motorcycle`.

Limiar (conf)	**Precisão**	**Revocação**	**F1**	**N**
0,30	1,000	0,291	0,451	583
0,40	1,000	0,274	0,430	583
0,50	1,000	0,258	0,410	583

Com `conf` = 0,30 (N=583), o modelo ajustado obteve Precisão = 0,981, Revocação = 0,880 e F1 = 0,928, com acurácia aproximada de 0,93 $(263 + 279)/583$. Foram contabilizados 263 verdadeiros positivos, 5 falsos positivos, 36 falsos negativos e 279 verdadeiros negativos; a Figura 26 apresenta um exemplo do teste, em que as cores indicam cada resposta do modelo: verde para verdadeiro positivo, cinza para verdadeiro negativo, laranja para falso positivo e vermelho para falso negativo. A incidência muito baixa de falsos positivos é coerente com a alta precisão; os falsos negativos concentram-se em vagas com veículos distantes, parcialmente ocluídos ou nas bordas do polígono (quando o centro da *bounding box* não ingressa na região).

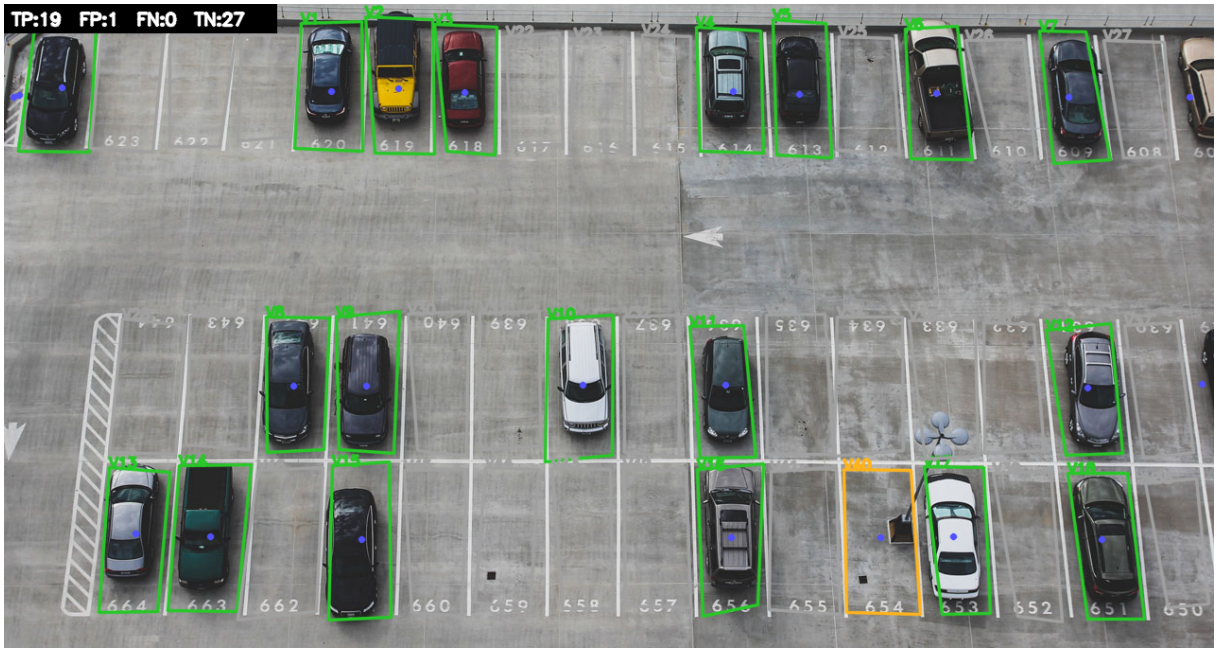
No conjunto de 30 imagens (583 vagas), o modelo ajustado apresentou maior assertividade global (F1) em todos os limiares testados, com destaque para `conf` = 0,30. A precisão permaneceu alta em ambos os modelos, sugerindo baixo risco de falsos positivos; a diferença decisiva esteve na revocação, sensivelmente superior no modelo ajustado em virtude da adequação ao domínio.

4.4 VÍDEO/RTSP

4.4.1 Cenário e posicionamento da câmera

Os ensaios foram realizados no estacionamento da Universidade Federal de Santa Catarina, campus Blumenau, como pode ser visto na Figura 27, com a câmera PTZ instalada na posição apresentada na Figura 28, mais especificamente no terceiro andar do prédio.

Figura 26 – Exemplo de teste do modelo.



Fonte: Adaptado de (TRANSPORTES, 2025).

Figura 27 – Estacionamento da UFSC Blumenau.



Fonte: (SANTA CATARINA, 2025).

Figura 28 – Posição da câmera.



Fonte: Elaborado pelo autor.

4.4.2 Resultados em vídeo

Os vídeos anotados (N=90 vagas) foram avaliados em três valores de `conf`. Observa-se o compromisso entre Precisão e Revocação: ao elevar `conf`, a Precisão tende a subir, porém a Revocação cai (mais falsos negativos). O melhor equilíbrio (maior F1) ocorreu em `conf` entre 0,30 e 0,40.

Tabela 5 – Desempenho em vídeo (N=90) com o modelo ajustado ao domínio (`best.pt`).

Limiar (`conf`)	**Precisão**	**Revocação**	**F1**	**TP**	**FP**	**FN**	**TN**	**N**
0,30	0,971	0,957	**0,964**	67	2	3	18	90
0,40	0,985	0,943	**0,964**	66	1	4	19	90
0,50	1,000	0,857	0,923	60	0	10	20	90

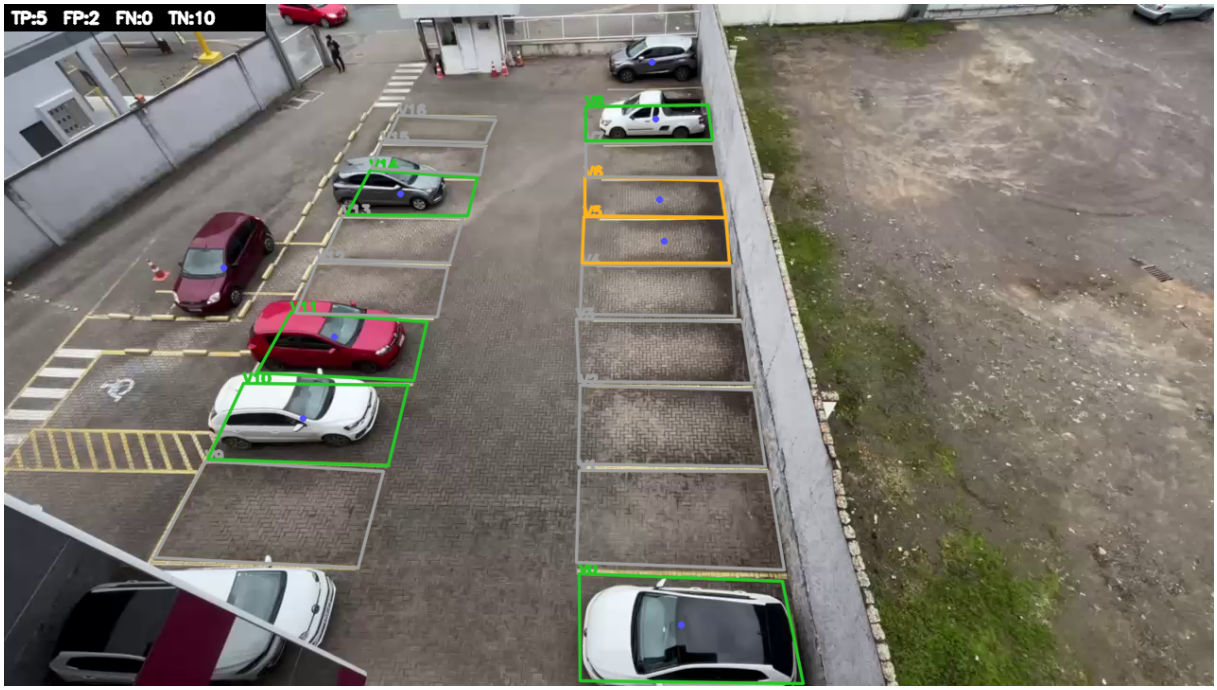
Tabela 6 – Desempenho em vídeo (N=90) com o modelo genérico `yolo11s.pt` (COCO); classes consideradas: `car`, `truck`, `bus`, `motorbike`/`motorcycle`.

Limiar (`conf`)	**Precisão**	**Revocação**	**F1**	**TP**	**FP**	**FN**	**TN**	**N**
0,30	1,000	0,914	0,955	64	0	6	20	90
0,40	1,000	0,900	0,947	63	0	7	20	90
0,50	1,000	0,857	0,923	60	0	10	20	90

Com `conf` em 0,30–0,40, o modelo ajustado manteve $F_1 \approx 0{,}964$ com alta

Precisão ($\geq$ 0,971) e Revocação ($\geq$ 0,943). O baseline `yolo11s.pt` também apresentou alta Precisão (1,000) nessa faixa, porém menor Revocação (0,914 $\rightarrow$ 0,900) e, portanto, F1 inferior (0,955 $\rightarrow$ 0,947)). Em `conf`=0,50, ambos atingem Precisão = 1,000, mas à custa de Revocação = 0,857; na prática, `conf` entre 0,30 e 0,40 é preferível para reduzir perdas de detecção sem aumentar falsos positivos de forma significativa. A Figura 29 ilustra o teste de um frame com o modelo treinado.

Figura 29 – Exemplo de teste do modelo com vídeo.



Fonte: Elaborado pelo autor.

4.4.3 Resultado com fluxo RTSP

O funcionamento do sistema com fluxo de vídeo em tempo real via RTSP foi validado qualitativamente em ambiente operacional, com foco na verificação do *pipeline* integrado sob condições reais de uso. Diferentemente dos testes quantitativos realizados anteriormente com imagens estáticas e vídeos pré-gravados, esta avaliação priorizou a observação do comportamento dinâmico do sistema.

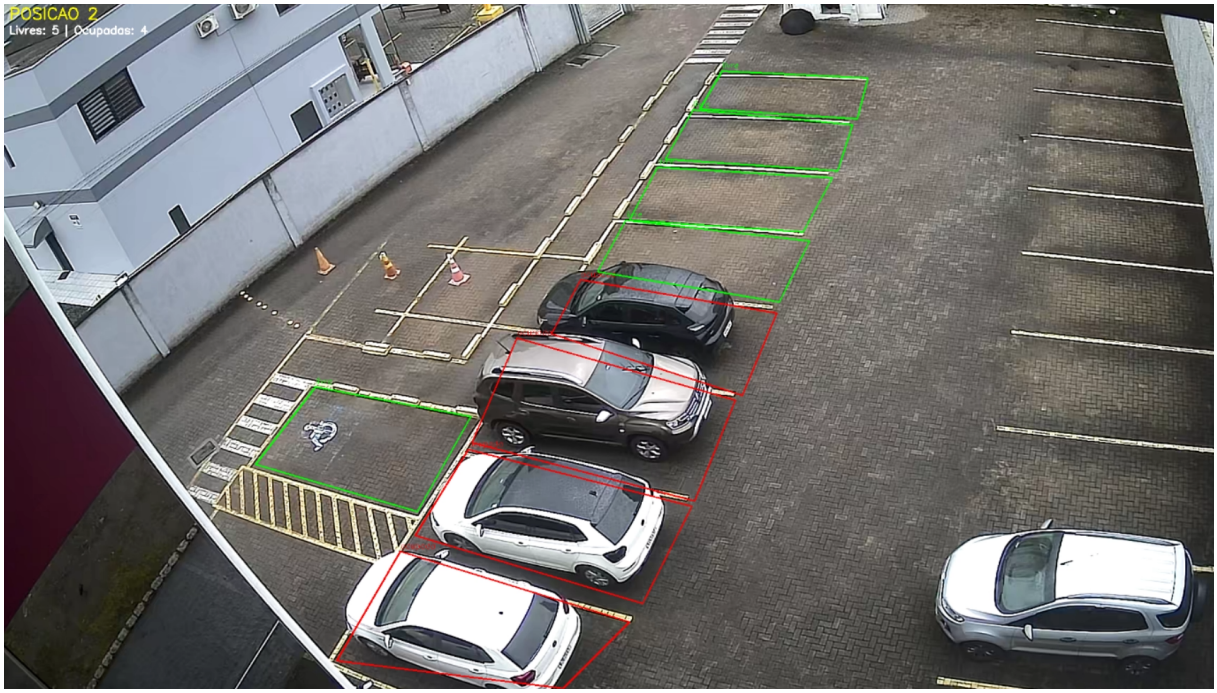
Durante os testes, observou-se que o sistema responde adequadamente às mudanças no cenário do estacionamento, detectando corretamente o estado das vagas quando o fluxo de vídeo atinge a estabilização. As regiões de interesse previamente mapeadas mantiveram-se consistentes com o enquadramento da câmera, mesmo após múltiplos ciclos de reposicionamento da PTZ, demonstrando o funcionamento adequado durante a operação contínua. As Figuras 30 e 31 ilustram o funcionamento do sistema proposto.

Figura 30 – Sistema em funcionamento na posição 1.



Fonte: Elaborado pelo autor.

Figura 31 – Sistema em funcionamento na posição 2.



Fonte: Elaborado pelo autor.

4.5 ANÁLISE DOS ERROS

A avaliação do sistema apresenta padrões recorrentes de erros que podem ser categorizados conforme a natureza da falha e as condições operacionais. Esta análise busca identificar as causas fundamentais dos falsos positivos e falsos negativos observados durante os testes.

4.5.1 Erros em Imagens Estáticas

A análise revelou que ângulos de captura muito laterais geram deformações geométricas nos veículos, dificultando sua identificação pelo modelo, como pode ser visto na Figura 32. Simultaneamente, algumas condições do piso, como asfalto escurecido por sombras intensas ou superfícies molhadas pela chuva, reduziram significativamente o contraste entre os veículos e o fundo, resultando em falsos positivos, especialmente para carros com tonalidades escuras. A Figura 33 ilustra a situação.

Figura 32 – Falso negativo pelo ângulo da vaga.



Fonte: Adaptado de (ESTACIONAMENTOS, 2025).

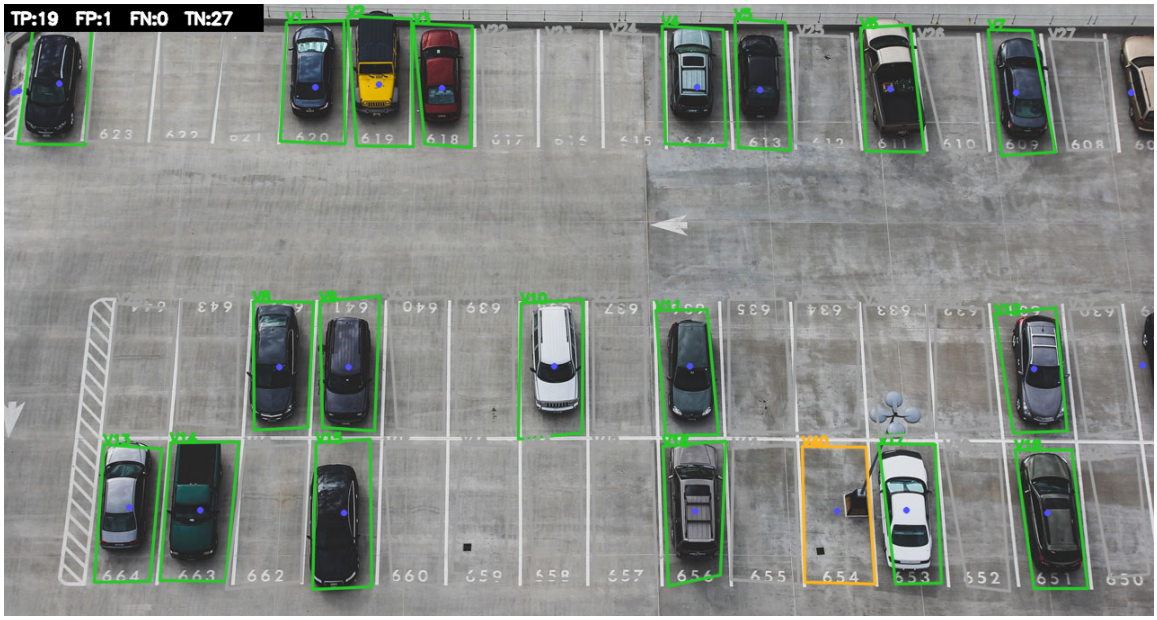
Obstruções fixas no ambiente, como postes de iluminação localizados dentro dos limites das vagas, foram incorretamente interpretadas como componentes veiculares, gerando falsos positivos; a Figura 34 ilustra essa situação. Adicionalmente, veículos com coloração semelhante à do asfalto apresentaram taxas de detecção reduzidas; a Figura 35 apresenta essa situação, na qual ocorre um falso negativo.

Figura 33 – Falso positivo pela cor da superfície.



Fonte: Adaptado de (GETTY IMAGES, 2025).

Figura 34 – Falso positivo pela cor da superfície.

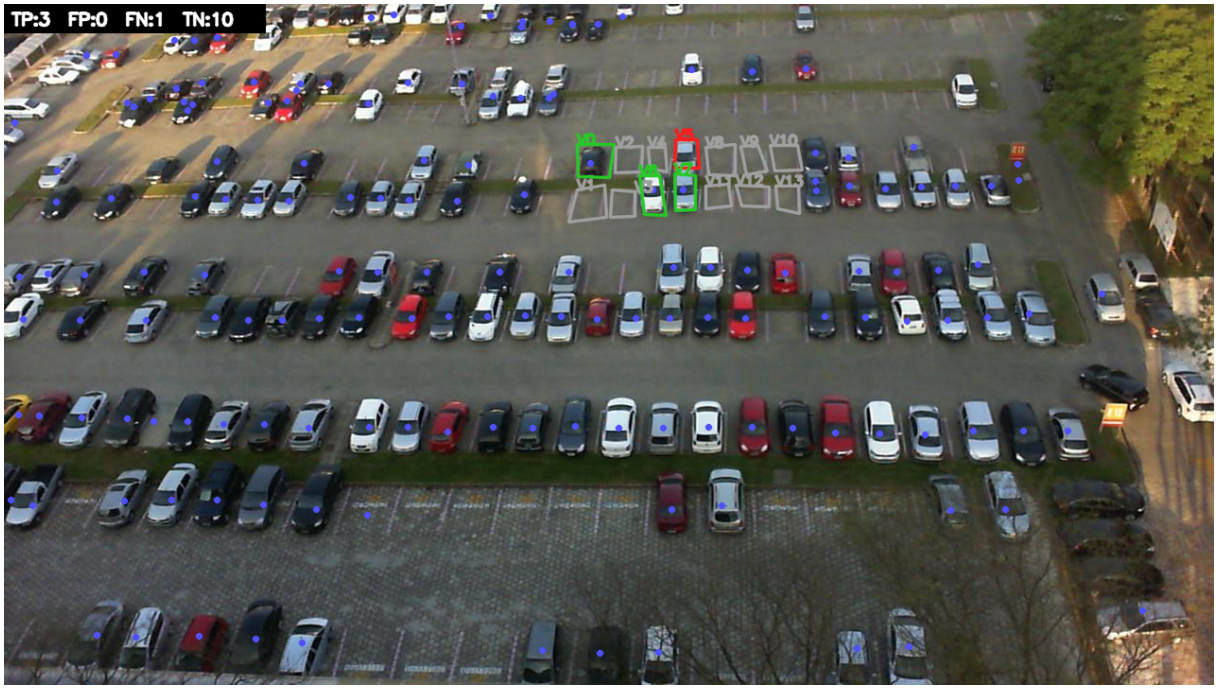


Fonte: Adaptado de (TRANSPORTES, 2025).

4.5.2 Erros em vídeos

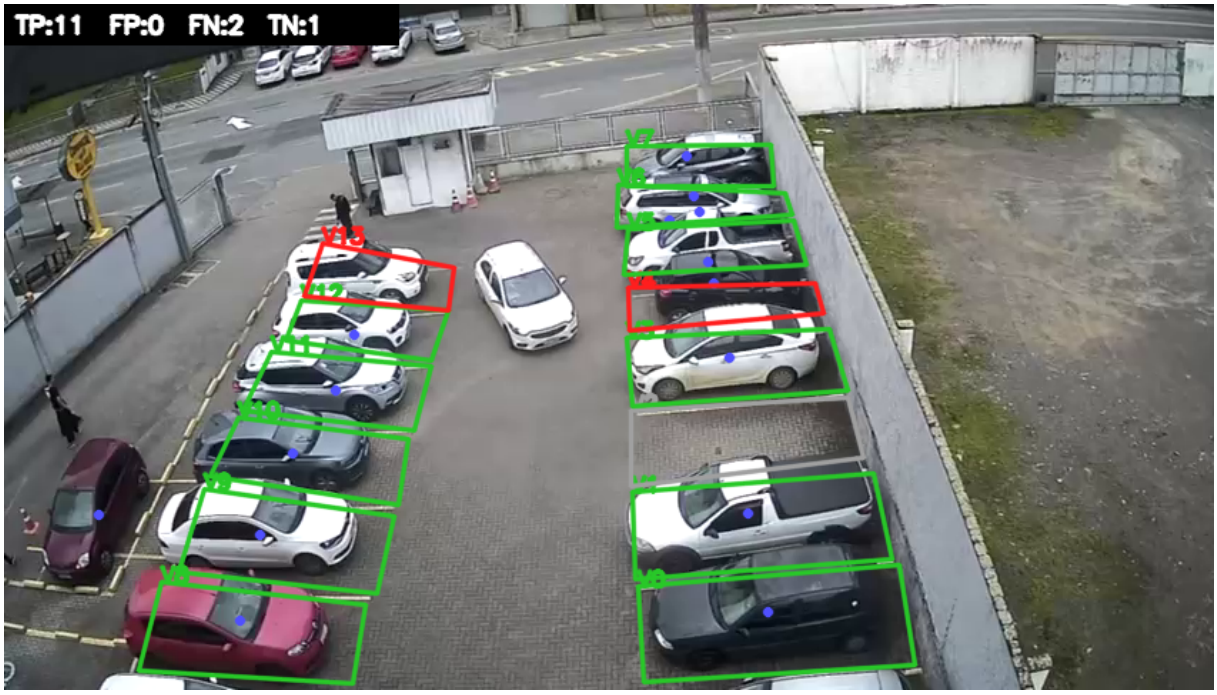
A demarcação incorreta das vagas mostrou-se mais crítica em vídeo, onde pequenos desvios no posicionamento das ROIs resultaram em inconsistências temporais nas detecções. Problemas como vagas mais distantes da câmera em um posicionamento mais lateral também geram erros, conforme pode ser constatado na Figura 36. Adicionalmente, bordas de vagas com iluminação reduzida ou de cores diferentes, como mostrado na Figura 37, apresentaram falsos negativos.

Figura 35 – Falso negativo pela cor do carro.



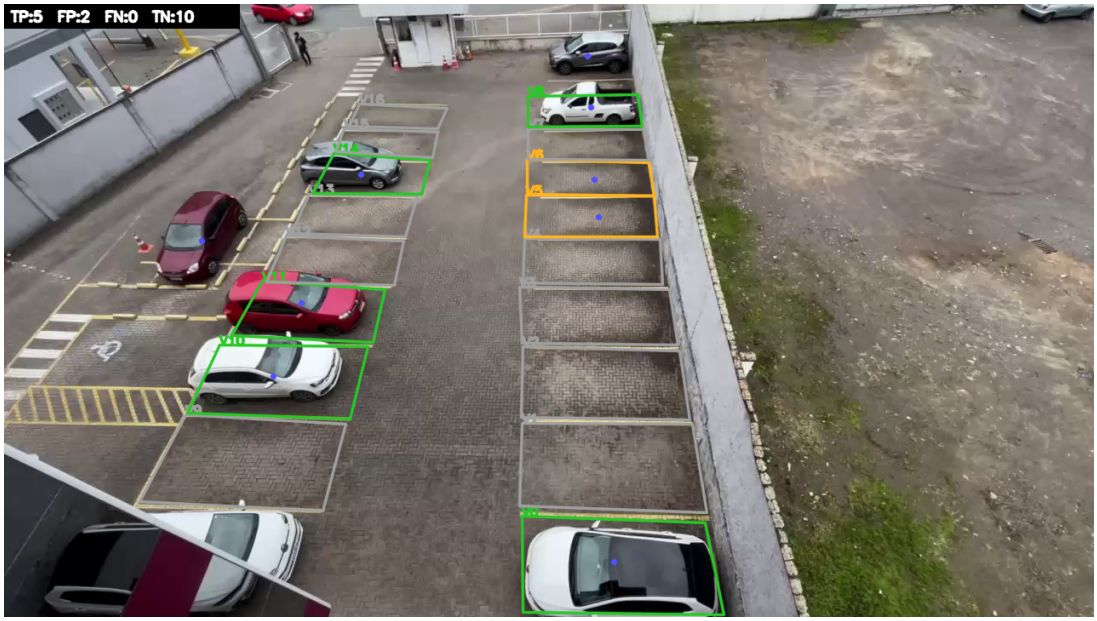
Fonte: Adaptado de (ALMEIDA, P. *et al.*, 2015).

Figura 36 – Falso negativo pela ROI e posição.



Fonte: Elaborado pelo autor.

Figura 37 – Falso positivo pela borda da vaga com possível sujeira.



Fonte: Elaborado pelo autor.

4.5.3 Erros no fluxo RTSP

A operação com fluxo RTSP em tempo real revelou desafios significativos relacionados à estabilidade da conexão, particularmente em ambiente universitário, onde os testes foram conduzidos via *hotspot* de celular. A instabilidade da rede Wi-Fi resultou em perda intermitente de quadros e oscilações na qualidade do vídeo, ocasionando falhas pontuais no processo de inferência, como pode ser visto na Figura 38.

Figura 38 – Falha no fluxo RTSP.



Fonte: Elaborado pelo autor.

4.5.4 Limitações

O sistema desenvolvido apresentou restrições associadas ao equipamento utilizado. A câmera PTZ empregada dispunha apenas de controles de *pan* e *tilt*, sem recurso de zoom óptico, o que limitou a flexibilidade de adaptação do campo de visão. Além disso, verificaram-se limitações tanto na implementação do protocolo ONVIF, que dificultou a criação e acionamento de *presets*, quanto na API da Tuya, que impediu a automação completa de determinadas configurações disponíveis apenas pelo aplicativo móvel.

No ambiente de testes, localizado no estacionamento universitário, a funcionalidade de varredura da PTZ não foi efetivamente necessária, uma vez que o campo de visão da câmera já abrangia toda a área sem necessidade de reposicionamento — caracterizando uma simulação parcial das condições reais. Observou-se também que a operação via Wi-Fi, sobretudo em configuração de *hotspot* móvel, mostrou-se pouco adequada para cenários críticos, evidenciando a importância do uso de câmeras com conectividade Ethernet, sensores de posicionamento precisos e implementação completa do protocolo ONVIF Perfil S para garantir operação confiável.

5 CONCLUSÃO

O presente trabalho demonstrou a viabilidade de utilizar câmeras PTZ, em conjunto com técnicas de visão computacional, para monitoramento de vagas em estacionamentos. O sistema desenvolvido foi capaz de mapear regiões de interesse, realizar detecção de veículos em tempo real e disponibilizar o estado de ocupação por meio de uma API REST. A integração entre a Tuya OpenAPI e o fluxo RTSP viabilizou a automação parcial do controle da câmera, permitindo a varredura de diferentes posições e a captura de informações relevantes. Apesar das limitações observadas, os resultados obtidos em imagens, vídeos e no fluxo em tempo real confirmaram o potencial da solução proposta como alternativa a sistemas tradicionais baseados em sensores.

A análise de erros evidenciou que fatores como a ausência de zoom óptico, a restrição do critério do centro da *bounding box* e a operação dependente de conectividade Wi-Fi impactaram o desempenho do sistema. Além disso, a simulação em ambiente de testes controlado não explorou completamente o potencial da câmera PTZ em cenários reais de grande escala. Essas limitações indicam a necessidade de melhorias no *hardware*, como câmeras com suporte completo ao protocolo ONVIF, conectividade via Ethernet e maior precisão no reposicionamento, bem como ajustes nos algoritmos de decisão de ocupação para reduzir falsos positivos e falsos negativos.

Apesar dos desafios, conclui-se que trabalho contribui como prova de conceito para o uso de visão computacional aplicada à mobilidade urbana, demonstrando uma solução escalável e economicamente mais acessível em comparação a sensores individuais por vaga. Para trabalhos futuros, recomenda-se explorar câmeras PTZ mais robustas, incorporar técnicas de rastreamento temporal para aumentar a acurácia das detecções, além de integrar os resultados a plataformas de *smart cities*, permitindo que motoristas tenham acesso em tempo real à disponibilidade de vagas. Tais avanços podem ampliar o impacto da solução, tornando-a aplicável em cenários reais e colaborando para a redução de congestionamentos e emissões em áreas urbanas.

REFERÊNCIAS

ABESE. **Conheça os principais tipos de câmeras de segurança**. Acesso em: 18 ago. 2025. 2025. Disponível em: https://www.abese.org.br/conteudos/conheca-os-principais-tipos-de-cameras-de-seguranca.

ALKHEDER, Sharaf A.; RAJAB, Murad M. Al; ALZOUBI, Khalid. Parking problems in Abu Dhabi, UAE toward an intelligent parking management system "ADIP: Abu Dhabi Intelligent Parking". **Alexandria Engineering Journal**, v. 55, n. 3, p. 2679–2687, 2016.

ALMEIDA, P.; OLIVEIRA, L. S.; SILVA JR, E.; BRITTO JR, A.; KOERICH, A. PKLot - A robust dataset for parking lot classification. **Expert Systems with Applications**, Elsevier, v. 42, n. 11, p. 4937–4949, 2015.

ALMEIDA, Paulo R.L. de; OLIVEIRA, Luiz S.; JR., Alceu S. Britto; JR., Eunelson J. Silva; KOERICH, Alessandro L. PKLot – A robust dataset for parking lot classification. **Expert Systems with Applications**, v. 42, n. 11, p. 4937–4949, 2015.

APRENDA CFTV. **Qual a distância de visualização de uma câmera de segurança (CFTV)**. Acesso em: 01 set. 2025. 2025. Disponível em: https://aprendacftv.com/distancia-de-visualizacao-de-camera-de-cftv/.

AVIGILON. **Guia completo dos diferentes tipos de câmeras CCTV: o que é**. Acesso em: 18 ago. 2025. 2025. Disponível em: https://www.avigilon.com/br/blog/types-of-cctv-cameras.

BRADSKI, Gary; KAEHLER, Adrian. **Learning OpenCV: Computer Vision with the OpenCV Library**. Sebastopol, CA: O'Reilly, 2008. ISBN 978-0596516130.

DAVANI, Sina G.; AL-HADRUSI, Musab; SARHAN, Nabil J. An Autonomous System for Efficient Control of PTZ Cameras. **ACM Transactions on Autonomous and Adaptive Systems**, ACM, v. 16, n. 2, p. 1–22, jun. 2021.

ESTACIONAMENTOS, Olimpark. **Administradora de Estacionamentos para Festas e Eventos**. Acesso em: 10 out. 2025. 2025. Disponível em: https://www.olimpark.com.br/administradora-de-estacionamentos-para-festas-e-eventos/.

GETTY IMAGES, iStock by. **Vista aérea de estacionamentos**. Acesso em: 10 out. 2025. 2025. Disponível em: https://www.istockphoto.com/br/search/more-like-this/825076454?assettype=image.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. [*S.l.*]: MIT Press, 2016. http://www.deeplearningbook.org.

GOOGLE RESEARCH. **Colaboratory: Jupyter notebooks in the cloud**. [*S.l.*: *s.n.*], 2025. Acesso em: 15 set. 2025. Disponível em: https://research.google.com/colaboratory/.

HAMPSHIRE, Robert C.; SHOUP, Donald. What Share of Traffic is Cruising for Parking? **Journal of Transport Economics and Policy**, v. 52, n. 3, p. 184–201, 2018.

HEIMBERGER, Markus; HORGAN, Jonathan; HUGHES, Ciarán; MCDONALD, John; YOGAMANI, Senthil Kumar. Computer vision in automated parking systems: Design, implementation and challenges. **CoRR**, abs/2104.12537, 2021. arXiv: 2104.12537.

ISARSOFT GMBH. **What is Pan-Tilt-Zoom (PTZ)?** Knowledge Hub article; published 2025-06-04. Jun. 2025. Disponível em: https://www.isarsoft.com/knowledge-hub/ptz. Acesso em: 30 ago. 2025.

JABBAR, Waheb A.; TIEW, Lu Yi; SHAH, Nadiah Y. Ali. Internet of things enabled parking management system using long range wide area network for smart city. **Internet of Things and Cyber-Physical Systems**, v. 4, p. 82–98, 2024.

JOCHER, Glenn; QIU, Jing. **Ultralytics YOLO11**. [*S.l.*: *s.n.*], 2024. Disponível em: https://github.com/ultralytics/ultralytics.

KALANTAR, Reza. **Receptive Fields in Deep Convolutional Networks**. [*S.l.*: *s.n.*], 2023. https://medium.com/@rekalantar/receptive-fields-in-deep-convolutional-networks-43871d2ef2e9. Acesso em: 14 set. 2025.

KINTRONICS. **How the IP Camera Works**. Acesso em 13 de agosto de 2025. 2022. Disponível em: https://kintronics.com/how-the-ip-camera-works/.

KUMAR, Kamlesh; SINGH, Vijander; RAJA, Linesh; BHAGIRATH, Swami Nisha. A Review of Parking Slot Types and their Detection Techniques for Smart Cities. **Smart Cities**, v. 6, n. 5, p. 2639–2660, 2023.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015.

LEE, Chang-Hsing; WEN, Ming-Gang; HAN, Chin-Chuan; KOU, Da-Cheng. An automatic monitoring approach for unsupervised parking lots in outdoors. *In*: PROCEEDINGS 39th Annual 2005 International Carnahan Conference on Security Technology. [*S.l.*: *s.n.*], 2005. P. 271–274.

MARTÍN NIETO, Rafael; GARCÍA-MARTÍN, Álvaro; HAUPTMANN, Alexander G.; MARTÍNEZ, José M. Automatic Vacant Parking Places Management System Using Multicamera Vehicle Detection. **IEEE Transactions on Intelligent Transportation Systems**, v. 20, n. 3, p. 1069–1080, 2019.

MICHELONI, Christian; RINNER, Bernhard; FORESTI, Gian Luca. Video Analysis in Pan-Tilt-Zoom Camera Networks. **IEEE Signal Processing Magazine**, v. 27, n. 5, p. 87–90, set. 2010.

MOKOSMART. **Magnetic Parking Sensor**. [*S.l.*: *s.n.*], 2022. https://www.mokosmart.com/pt/magnetic-parking-sensor/. Acesso em: 139 abr. 2025.

NITHYA, R.; PRIYA, V.; SATHIYA KUMAR, C.; DHEEBA, J.; CHANDRAPRABHA, K. A Smart Parking System: An IoT Based Computer Vision Approach for Free Parking Spot Detection Using Faster R-CNN with YOLOv3 Method. **Wireless Personal Communications**, v. 125, p. 3205–3225, 2022.

ONVIF. **ONVIF - Profiles, Add-ons and Specifications**. [*S.l.*: *s.n.*], 2025. Acesso em: 06 jun. 2025. Disponível em: https://www.onvif.org/profiles-add-ons-specifications/.

ONVIF. **Profiles, Add-ons and Specifications**. [*S.l.*: *s.n.*], 2025. https://www.onvif.org/profiles-add-ons-specifications/. Acesso em: 13 abr. 2025.

PARKSOL. **TOF Sensor Solution**. [*S.l.*: *s.n.*], 2025. https://parksol.lt/solutions/tof/. Acesso em: 26 abr. 2025.

POPOVIC, Gradimirka; ARSIC, Nebojsa; JAKSIC, Branimir; GARA, Boris; PETROVIC, Mile. Overview, Characteristics and Advantages of IP Camera Video Surveillance Systems Compared to Systems with other Kinds of Camera. **International Journal of Engineering Science and Innovative Technology (IJESIT)**, v. 2, n. 5, p. 356–362, 2013.

PROJECT. **Car Park Detection Dataset**. [*S.l.*]: Roboflow, mar. 2023. https://universe.roboflow.com/project-btwlz/car-park-detection-lq0z4. visited on 2025-09-20. Disponível em: https://universe.roboflow.com/project-btwlz/car-park-detection-lq0z4.

REDMON, Joseph; DIVVALA, Santosh Kumar; GIRSHICK, Ross B.; FARHADI, Ali. You Only Look Once: Unified, Real-Time Object Detection. **CoRR**, abs/1506.02640, 2015. arXiv: 1506.02640.

REYNOLDS, Anh. **Convolutional Neural Networks (CNNs)**. 2019. Disponível em: https://anhreynolds.com/blogs/cnn.html. Acesso em: 14 set. 2025.

SAFWAN, Hussain Md.; MAHIM, Mahbub Islam. Enhancing Wide-Angle Image Using Narrow-Angle View of the Same Scene. **arXiv preprint arXiv:2504.09455**, 2025. Accessed: 2025-08-30.

SANTA CATARINA, UFSC - Universidade Federal de. **UFSC - Universidade Federal de Santa Catarina - Campus Blumenau**. [*S.l.*: *s.n.*], 2025. Acesso em: 22/09/2025. Disponível em: https://ufsc.br/campi/.

SANTIN, Francele; MARIO, José; GRZYBOWSKI, J. M. V.; SILVA, Roberto. Application of Neural Network Ensembles to the Problem of Estimating Riparian Buffer Width as a Function of Desired Filtering Properties. *In*: PROCEEDINGS of the 1st International Congress of Management, Technology and Innovation. Erechim, Brazil: [*s.n.*], set. 2015.

SHAO, Yunfeng; CHEN, Pengzhen; CAO, Tongtong. A Grid Projection Method Based on Ultrasonic Sensor for Parking Space Detection. *In*: 2018 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). [*S.l.*: *s.n.*], 2018. P. 3378–3381.

SZELISKI, Richard. **Computer Vision: Algorithms and Applications**. London: Springer, 2010. ISBN 978-1848829343.

TRANSPORTES, Planalto. **O que é Self-Parking?** Acesso em: 10 out. 2025. 2025. Disponível em: https://institucional.planalto.com.br/o-que-e-selfie-parking/.

TUYA INC. **PTZ Control: Cloud Services API Reference**. [*S.l.*: *s.n.*], 2023. https://developer.tuya.com/en/docs/cloud/32373785cc?id=Kbsm7f4wke0yc. Endpoint: POST /v1.0/cameras/{device_id}/configs/ptz. Acesso em: 30 ago. 2025.

ULTRALYTICS. **Pontuação F1: definição, fórmula e aplicações**. Acesso em: 10 out. 2025. 2025. Disponível em: https://www.ultralytics.com/pt/glossary/f1-score.

UNITED NATIONS. **68% of the world population projected to live in urban areas by 2050, says UN**. [*S.l.*: *s.n.*], 2018. https://www.un.org/uk/desa/68-world-population-projected-live-urban-areas-2050-says-un. Acesso em: 13 abr. 2025.

VIGNE, Nancy G. La; LOWRY, Samantha S.; MARKMAN, Joshua A.; DWYER, Allison M. **Evaluating the Use of Public Surveillance Cameras for Crime Control and Prevention**. Washington, D.C., 2011. Disponível em: https://www.researchgate.net/profile/Nancy-La-Vigne/publication/280089845_Evaluating_the_Use_of_Public_Surveillance_Cameras_for_Crime_Control_and_Prevention/links/55a7ccbe08ae5aa1579dbaa8/Evaluating-the-Use-of-Public-Surveillance-Cameras-for-Crime-Control-and-Prevention.pdf.

VINA, Abirami. **Ultralytics YOLOv8 for smarter parking management systems**. [*S.l.*: *s.n.*], 2024. https://www.ultralytics.com/blog/ultralytics-yolov8-for-smarter-parking-management-systems. Acesso em: 15 set. 2025.

WEG. **Linha IP – câmeras de segurança eletrônica – WCAM**. Acesso em: 19 ago. 2025. 2025. Disponível em: https://www.weg.net/weghome/produtos/seguranca-eletronica/seguranca-cameras/linha-ip/.

WEG HOME. **Câmeras WEG Home**. Página de catálogo com especificações e materiais para download. WEG S.A. 2025. Disponível em: https://www.weg.net/weghome/produtos/smart-home/cameras/. Acesso em: 16 set. 2025.

WEG S.A. **Linha WCAM**. Jaraguá do Sul: [*s.n.*], 2025. Brochura institucional, 32 páginas. Disponível em: https://static.weg.net/medias/downloadcenter/hc5/h59/WEG-WCAM-brochure-50131922-pt.pdf. Acesso em: 3 ago. 2025.

YANI, Muhamad; IRAWAN, Budhi; SETININGSIH, Casi. Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. *In*: JOURNAL of Physics: Conference Series. [*S.l.*]: IOP Publishing, 2019. v. 1201, p. 012052.

ZHANG, Zusheng; HE, Xianmang; HUANG, Jinwang; YUAN, Huaqiang. Parking Detection Using Combined Magnetic Sensor and Pulsed Coherent Radar. **IEEE Internet of Things Journal**, v. 9, n. 18, p. 17210–17219, 2022.