UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

PLÁCIDO NILO MUNIZ DO NASCIMENTO

SISTEMA DE SUPERVISÃO DE ELETROPOSTOS BASEADO EM VISÃO COMPUTACIONAL

Recife
2025

PLÁCIDO NILO MUNIZ DO NASCIMENTO

SISTEMA DE SUPERVISÃO DE ELETROPOSTOS BASEADO EM VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em 2025 da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Controle e Automação

Orientador(a): Prof. Dr. Douglas Contente Pimentel Barbosa

Recife
2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Nascimento, Plácido Nilo Muniz do.
Sistema de supervisão de eletropostos baseado em visão computacional / Plácido Nilo Muniz do Nascimento. - Recife, 2025.
77p. : il., tab.

Orientador(a): Douglas Contente Pimentel Barbosa
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia de Controle e Automação - Bacharelado, 2025.
Inclui referências, apêndices.

1. Eletroposto. 2. YOLO. 3. Visão Computacional. 4. veículos elétricos. 5. classificação de veículos. I. Barbosa, Douglas Contente Pimentel . (Orientação). II. Título.

620 CDD (22.ed.)

Plácido Nilo Muniz do Nascimento

SISTEMA DE SUPERVISÃO DE ELETROPOSTOS BASEADO EM VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Controle e Automação

Aprovado em: 11/04/2025.

BANCA EXAMINADORA

_____

Prof. Dr. Douglas Contente Pimentel Barbosa
Universidade Federal de Pernambuco

_____

Prof. Dr. Rafael Cavalcanti Neto
Universidade Federal de Pernambuco

_____

Prof. M.Sc. Valdemar Moreira Cavalcante Junior
Universidade Federal de Pernambuco

AGRADECIMENTOS

Agradeço primeiramente a Deus, que esteve sempre comigo durante toda essa trajetória, me orientando e iluminando. Em seguida, gostaria de agradecer aos meus pais Maria da Assunção e Plácido Jonas, que foram, são e sempre serão minha base. Espero que possa retribuir todo o esforço que eles realizaram para que eu pudesse chegar aqui. E toda minha família, meus tios e primos.

Gostaria de agradecer também a minha namorada Iasmim Mesquita, que alegrava meus dias mais difíceis e que sempre me apoiou em todas as minhas decisões. Aos amigos que fiz nessa caminhada, Matheus Borges, Gabriel Xavier, Anthony Duarte, Ramatis Ferreira, Bernardo Rehn, João Pedro, Flávio Lopes, Vinícius de Alcântara, João Carlos, Paulo Moneta, João Guedes, Felipe Gouveia, Fernando Xavier, Lucas Gabriel, Hallason Matias, a lista é longa, mas ninguém chega no topo sozinho.

Agradeço, de forma especial, à empresa júnior Watt Consultoria, onde tive a oportunidade de crescer significativamente, especialmente no meu desenvolvimento pessoal. Foi nesse ambiente que conheci Alexandre Motta e Samuel Leal, que aceitaram o desafio de empreender comigo na criação de uma startup.

Expresso também minha sincera gratidão ao meu orientador, professor Douglas Contente, por sua orientação essencial não apenas neste trabalho, mas ao longo de toda a minha trajetória acadêmica. Estendo meus agradecimentos aos professores Luiz Henrique, Geraldo Maia e Pedro Rosas, por me proporcionarem a valiosa oportunidade de integrar o projeto de pesquisa do Laboratório de Armazenamento e Mobilidade, onde pude aprofundar meus conhecimentos e vivenciar experiências enriquecedoras.

Obrigado!

"Algumas pessoas querem que algo aconteça, outras desejam que aconteça, outras fazem acontecer." – Michael Jordan.

RESUMO

Com a rápida evolução da inteligência artificial no mundo nos últimos anos, surgiram diversos projetos e estudos utilizando visão computacional com ótimos resultados. Quase que simultaneamente ocorreu um aumento na quantidade de veículos elétricos e estações de recarga no mundo. Dessa forma, este estudo propõe o desenvolvimento de um sistema de monitoramento de eletropostos a partir da visão computacional, sendo aplicado ao eletroposto de estudo sediado em Campinas/SP, projetado e construído por meio de um projeto de Pesquisa & Desenvolvimento (P&D). O objetivo principal é elaborar uma solução composta do modelo YOLO para a detecção dos veículos elétricos agregado a um *dashboard,* desenvolvido em uma plataforma *Low-Code,* que se integra ao eletroposto de estudo. O sistema busca obter dados relevantes para ajudar a gestão das estações de recargas, trazendo informações de padrões de consumo, tempos de permanência, entre outros. O trabalho consiste em destrinchar cada etapa do desenvolvimento de maneira de que se torne um modelo replicável para outros eletropostos de forma prática. Como resultado, obteve-se um sistema funcional capaz de detectar veículos em diferentes condições, identificar padrões de uso e fornecer dados relevantes para a gestão dosestações de recarga de veículos elétricos.

Palavras-chave: Eletroposto; YOLO; Visão Computacional; veículos elétricos; classificação de veículos.

ABSTRACT

With the rapid evolution of artificial intelligence in the world in recent years, several projects and studies using computer vision have emerged with excellent results. Almost simultaneously, there has been an increase in the number of electric vehicles and charging stations in the world. Thus, this study proposes the development of an electric charging station monitoring system based on computer vision, to be applied to the electric charging station under study located in Campinas/SP, designed and built through a Research & Development (R&D) project. The main objective is to develop a solution composed of the YOLO model for detecting electric vehicles combined with a dashboard, developed on a Low-Code platform, which is integrated with the electric charging station under study. The search system obtains relevant data to help manage the charging stations, providing information on consumption patterns, dwell times, among others. The work consists of unraveling each stage of the development so that it becomes a replicable model for other electric charging stations in a practical way. As a result, we obtained a functional system capable of detecting vehicles in different conditions, identify usage patterns and provide relevant data for the management of electric vehicle charging stations.

Keywords: Electric charging station; YOLO; Computer Vision; electric vehicles; vehicle classification.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BYD	BYD *Company Limited*
CINASE	Circuito Nacional do Setor Elétrico
CNNs	*Convolutional Neural Networks*
CTG	Centro de Tecnologia e Geociência
EMS	*Energy Management System*
MQTT	*Message Queuing Telemetry Transport*
P&D	Pesquisa e desenvolvimento
SGE	Sistema de Gestão de Energia
VA	Visão Artificial
VE	Veículo Elétrico

LISTA DE SÍMBOLOS

a_j	Ativação do neurônio de índice j
g	Função de ativação
$\sum$	Símbolo de somatório
$w_{i,g}$	Peso da conexão entre o neurônio i e o neurônio g
b	Bias

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento acelerado dos veículos elétricos em todo o mundo (PARODI, 2024), há indícios de que é necessária uma preparação mundial para atender essa demanda. Associado a isso, vários países pretendem extinguir os veículos de fontes não renováveis em alguns anos, como é o caso do Brasil, que pretende realizar esse feito até 2060, conforme o PLS 454/2017 que já foi aprovado pela Comissão de Assuntos Econômicos, o que representa um avanço significativo para a sustentabilidade e traz diversos benefícios, como a redução das emissões de gases de efeito estufa (VACCANI, 2024).

Essa transição, embora promissora, trouxe à tona uma série de questionamentos para os governantes deste país. A inserção de uma frota de carros elétricos nas ruas exige uma infraestrutura robusta que, até o momento, ainda não está totalmente desenvolvida (ALBERY, QUEIROZ, et al., 2024). A sobrecarga da rede elétrica, o possível aumento do valor da energia, a falta de estações de carregamento público e a ausência de uma legislação específica para veículos elétricos são apenas alguns dos desafios que vêm junto com essa nova realidade.

A vinda de grandes marcas de veículos elétricos, como a empresa chinesa BYD Company Limited (BYD) para o Brasil, movimentou vários órgãos e empresas para a implementação de estações com carregadores para veículos elétricos (VEs). Segundo dados da Associação Brasileira do Veículo Elétrico (ABVE), o Brasil teve um crescimento anual em 2024 de 179% (CESAR, 2024) na infraestrutura de estações de recarga, atingindo 10.662 eletropostos (estação de carregamento de veículos elétricos) no país, o que ultrapassa a estimativa que o Brasil alcançaria a marca de 10.000 eletropostos (ABVE, 2024).

Entretanto, os primeiros modelos de estações de recarga operavam de maneira manual ou com agendamentos fixos, resultando em baixa eficiência na distribuição dos carregadores e dificuldade no atendimento de veículos não programados (MEDEIROS et al., 2022). Para atender à crescente demanda, os estacionamentos precisam modernizar seus sistemas de gestão. No entanto, desafios persistem, como a imprevisibilidade na chegada dos veículos, exigindo soluções mais eficientes para otimizar a ocupação e evitar congestionamentos (TRANSPOQUIP, 2025).

Atualmente, algumas abordagens vêm sendo exploradas para aprimorar esse gerenciamento. A Smart Parking Systems desenvolveu um modelo baseado em sensores subterrâneos, que fornecem dados sobre a disponibilidade das vagas em tempo real (SMART, 2025). Além disso, estudos como o de (CAVENAGLI e MENDES, 2020), propõem o uso de visão computacional para monitoramento de estacionamentos convencionais, embora essa tecnologia ainda não tenha sido amplamente aplicada a eletropostos.

Paralelo a esses avanços, a inteligência artificial tem se consolidado como uma ferramenta essencial em diversos setores, incluindo a mobilidade elétrica (SILVA, 2023). Um dos subcampos da inteligência artificial é a visão computacional, que se destaca como uma tecnologia promissora para a interpretação e análise de imagens captadas por câmeras, possibilitando o desenvolvimento de sistemas inteligentes de monitoramento (LEAL, 2023). O potencial dessa área pode ser evidenciado por uma pesquisa conduzida pela empresa *Grand View Research*, a qual mostrou que o mercado global de visão computacional foi estimado em USD 19,82 bilhões em 2024, com projeção de crescimento anual de 19,8% entre 2025 e 2030 (GRAND VIEW RESEARCH, 2024).

Diante disso, este trabalho propõe desenvolver uma aplicação que integre esses dois cenários: as estações de recarga e a tecnologia de visão computacional. Ou seja, elaborar um sistema de supervisão de eletropostos a partir da visão computacional, fornecendo em tempo real informações relevantes, como disponibilidade de vagas, padrões de uso e gestão do tempo, para auxiliar os gestores das estações a tomarem decisões mais assertivas. Este trabalho é fruto de um projeto de Pesquisa & Desenvolvimento (P&D), intitulado “Eletropostos Integrados à Tecnologia Nacional de Baterias (Chumbo-Carbono) e Sistemas Fotovoltaicos”.

1.1 Objetivos

Desenvolver um sistema de monitoramento de eletropostos a partir de visão computacional aplicado ao eletroposto de estudo.

1.1.1 Objetivos Gerais

Integrar a visão computacional a um sistema de supervisão de eletropostos com o objetivo de obter dados e informações úteis, como a relação entre o tempo de carregamento dos veículos e o tempo total de permanência estacionados e de padrões de consumo ao longo do tempo, visando melhorar a gestão da estação.

1.1.2 Objetivos Específicos

Para atender os objetivos gerais é necessário atender aos seguintes objetivos específicos:

- Elaborar o programa de visão computacional voltado para eletropostos.
- Construir o sistema supervisório para transformar os dados extraídos do programa de visão computacional (VC) em informações relevantes.
- Integrar o sistema supervisório ao programa de visão computacional e conectar ambos as câmeras.

1.2 Organização do Trabalho

A estrutura desse estudo constitui-se em 6 partes. A priori, apresenta-se um embasamento teórico no Capítulo 2, que traz informações sobre o contexto dos eletropostos no Brasil e como são realizados os monitoramentos destes. Ainda nessa seção é abordada a teoria básica de visão computacional necessária para entender o projeto e as pesquisas que já foram realizadas sobre o tema, como modelos de classificação e rastreamentos de veículos.

Em seguida, tem-se a metodologia no Capítulo 3, que descreve o planejamento e a execução do projeto, apresentando os motivos das escolhas de tecnologias e métodos.

Em sequência, o Capítulo 4 apresenta a fase prática do projeto, que mostra em detalhes como foram construídos o programa de visão computacional, o supervisório

e a conexão entre eles a partir de um estudo de caso, de maneira que o projeto possa ser replicável.

Assim, tem-se no Capítulo 5 os resultados e discussões, que elencam suas principais implicações e mostram os pontos positivos e negativos sobre o sistema de monitoramento das estações de veículos elétricos.

Por fim, o Capítulo 6, finaliza este trabalho com as conclusões, bem como sugestões para contribuições futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são abordados os conceitos fundamentais que sustentam o desenvolvimento do sistema proposto, incluindo a visão computacional, redes neurais e convolucionais, o funcionamento do servidor MQTT e o contexto atual dos eletropostos. São apresentados princípios essenciais para alcançar os objetivos estabelecidos e garantir que a aplicação funcione da maneira planejada.

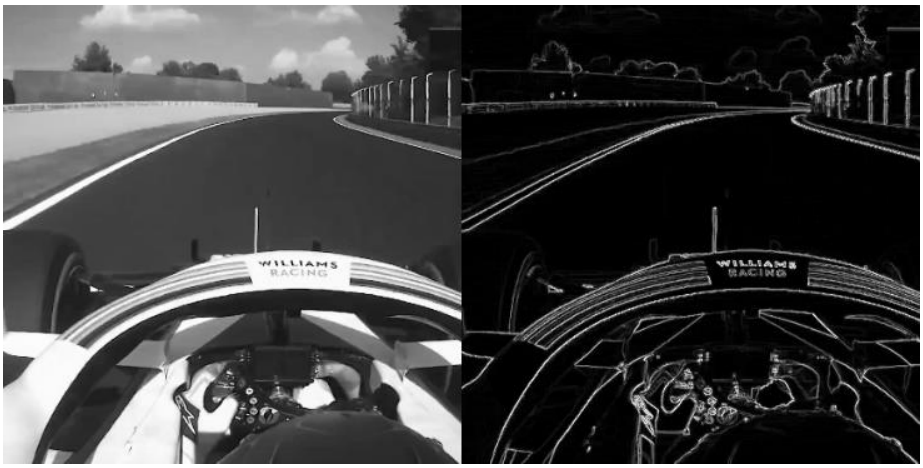
2.1 Visão Computacional

Com o avanço tecnológico em alta velocidade nas últimas décadas, existem câmeras instaladas em muitos lugares, até mesmo dentro das residências (DIAS e DEL VECHIO, 2019). A visão computacional é uma ferramenta que vem ganhando muitas aplicações em diversas áreas ao redor do mundo (OLIVEIRA e MELO, 2024). De acordo com o trabalho (FORSYTH e PONCE, 2012), a visão computacional é uma área da computação que se estuda o processo de formação e interpretação de imagens, utilizando os pixels mesclados com as várias imagens de referência, realizando a segmentação de grupos de pixels, reconhecimentos de objetos e geração de dados. Tudo isso baseado num conhecimento aprofundado no processo físico de construção de imagens a partir de câmeras.

Na perspectiva de (SZELISKI, 2010), a visão computacional é o processo de descrever o mundo que os humanos veem em uma ou mais imagens, de maneira a reconstruir propriedades como forma, iluminação, cor e distribuições. Segundo (FORSYTH e PONCE, 2012), este conceito começou a ser citado a partir de 1960, mas suas principais conquistas e avanços significativos ocorreram nos últimos 25 anos, principalmente devido ao desenvolvimento das redes neurais convolucionais (*Convolutional Neural Networks*- CNNs), o aprimoramento das técnicas de *Machine Learning* (Aprendizado por máquinas) e também com a melhoria na performance dos processos computacionais. Tudo isso facilitou o desenvolvimento de sistemas que antes eram bastante complexos, de forma muito mais prática e rápida, com um custo de implementação bastante reduzido (FORSYTH e PONCE, 2012).

A fim de compreender bem o conceito de visão computacional, também é importante saber relacionar e diferenciar os conceitos de processamento de imagens e computação gráfica. O processamento de imagens é a ação de transformar uma imagem em uma nova representação, a partir de algoritmos que utilizam diversas técnicas matemáticas e estatísticas que tratam os dados daquela imagem (MENESES e DE ALMEIDA, 2012). Como pode ser observado no exemplo da Figura 1, em que se utiliza a técnica detecção de contorno ou bordas em imagens, que consiste em encontrar regiões onde há uma mudança abrupta na intensidade dos pixels, através da análise do gradiente, que mede a variação da intensidade luminosa entre pixels vizinhos, como por exemplo, os algoritmos de *Canny e Sobel* (ANTONELLO, 2018).

Figura 1 – Exemplo da técnica de detecção de contornos.



Fonte: (SANTOS, 2020).

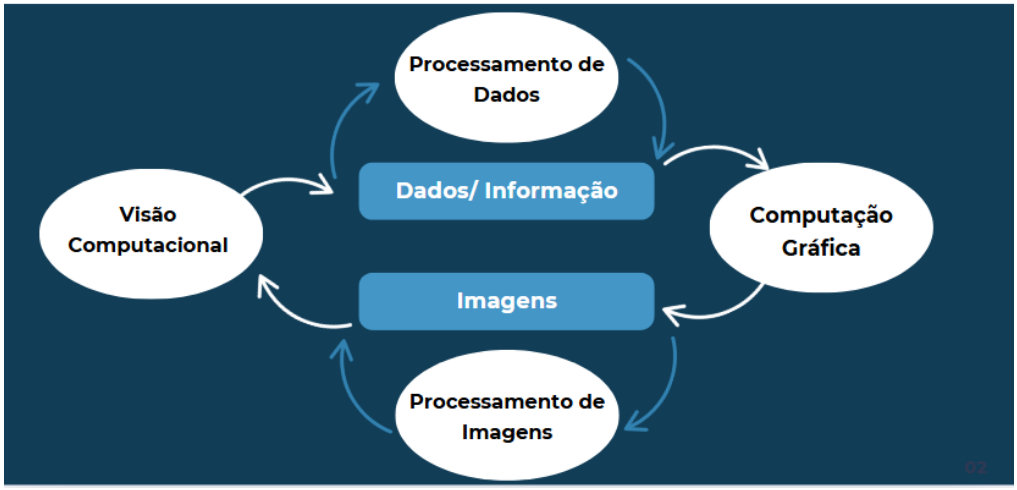
Porém, isto se distingue da computação gráfica, área da computação que foca no processo de gerar imagens a partir de algoritmos através dos computadores (SAÚDE, 2019). Isso é exemplificado na Figura 2, com a imagem do jogador de futebol criada computacionalmente para um jogo digital.

Figura 2 – Exemplo de imagem criada por computação gráfica.

Fonte: (NOVAIS, 2022).

Agregando esses dois conceitos com o de visão computacional (VC) e adicionando o processamento de dados, tem-se o sistema da Figura 3, em que é possível perceber que a VC recebe uma imagem e gera dados, enquanto o processamento de imagens recebe e gera imagens. Já a computação gráfica recebe dados e constrói imagens. São esses processos relacionados que permitem a captura e reconhecimento de vídeos em tempo real, por exemplo.

Figura 3 – Sistema da relação entre os conceitos.



Fonte: (ALBUQUERQUE, 2000).

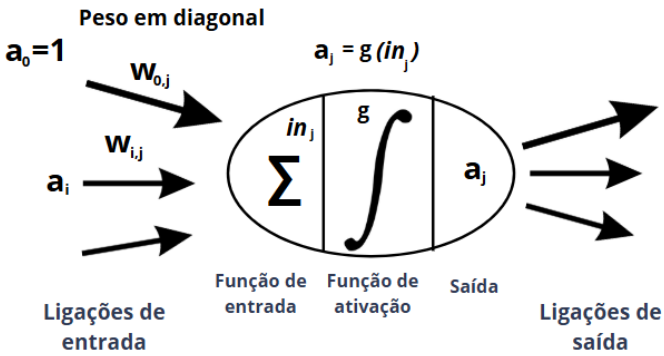
2.2 Redes Neurais Artificiais

Uma rede neural é um processador inspirado no comportamento dos neurônios, que trabalha de forma paralela e é constituída de uma unidade de processamento simples, cuja tarefa é armazenar e usar o conhecimento adquirido no processo de aprendizagem (HAYKIN, 1999). Na Figura 4, pode-se observar o primeiro modelo matemático que representava um neurônio, desenvolvido por McCulloch e Pitts em 1943 (LEMES, 2020). O modelo é representado pela Equação (2.1):

$$a_j = \mathrm{g} \sum_{i=0}^{n} (w_{i,g} * a_i) \tag{2.1}$$

Nessa equação, a_j corresponde à saída do neurônio *j,* enquanto a_i representa as entradas recebidas por esse neurônio. Os termos $w_{i,g}$ são os pesos sinápticos associados a cada entrada i, que indicam a importância relativa de cada entrada para o neurônio. O resultado do sistema equivale ao somatório dos pesos multiplicados pelas variáveis de entradas e multiplicado pela função de ativação $\mathrm{g}()$, função esta que decide se o neurônio vai ser ativado ou não, a Sigmoid e a ReLu são alguns exemplos dessas funções (ZHANG e WOODLAND, 2015).

Figura 4 – Modelo matemático do neurônio



Fonte: Adaptado de (RUSSEL e NORVIG, 2013)

Atualmente, pode-se representar uma rede neural simples em três partes: a primeira formada pela camada de entrada, que recebe os dados; a segunda composta de uma ou mais camadas ocultas, em que ocorre o processamento por meio de funções de ativação; e a terceira constituída de uma camada de saída que gera os resultados (FLECK, TAVARES, *et al.*, 2016). Vale ressaltar que as camadas podem ter diversos neurônios a depender da arquitetura da rede. Dessa forma, cada variável de entrada é atribuída a um peso, cujo valor é gerado aleatoriamente, passando pela função de ativação e retornando uma saída. A partir desse momento ocorre o processo conhecido como *backpropagation* (retropropagação), responsável por fazer a rede aprender (TISSOT, CAMARGO e POZO, 2012). Nesse método, ocorre a propagação reversa, em que o valor calculado na saída da rede é analisado pela função de perda (*Loss Function*), que calcula o erro entre a resposta real e a calculada pela rede. Esse erro é enviado novamente a rede para a modificação dos pesos, possibilitando-a aumentar sua taxa de acertos. Nota-se que este último processo se assemelha bastante aos sistemas de controle de malha fechada, em que o erro é essencial para o ajuste da malha (OGATA, 2010).

Ao longo do tempo, as redes neurais ficaram bem mais complexas, utilizando múltiplas camadas, diferentes formas de conexão, realizando computação distribuída, tolerando diferentes tipos de entradas, entre outras. Além disso, vem sendo utilizadas em diversos cenários como análise de dados, reconhecimento por voz e processamento de linguagem natural. Ainda que outras redes, como as redes bayesianas, também podem realizar esses feitos, as redes neurais artificiais são as mais populares e eficazes em aprendizado do sistema (RUSSEL e NORVIG, 2013).

2.3 Redes Neurais Convolucionais

As CNNs são uma categoria de rede neural cuja entrada de dados é composta principalmente por imagens, dessa forma ela se concentra em ter a arquitetura que melhor se adapta a necessidade de lidar com esse tipo específico de dados. Ao contrário das redes neurais padrões, os neurônios dentro das camadas se conectam apenas a uma pequena região da camada anterior, o que evita um gasto computacional enorme (O'SHEA e NASH, 2015).

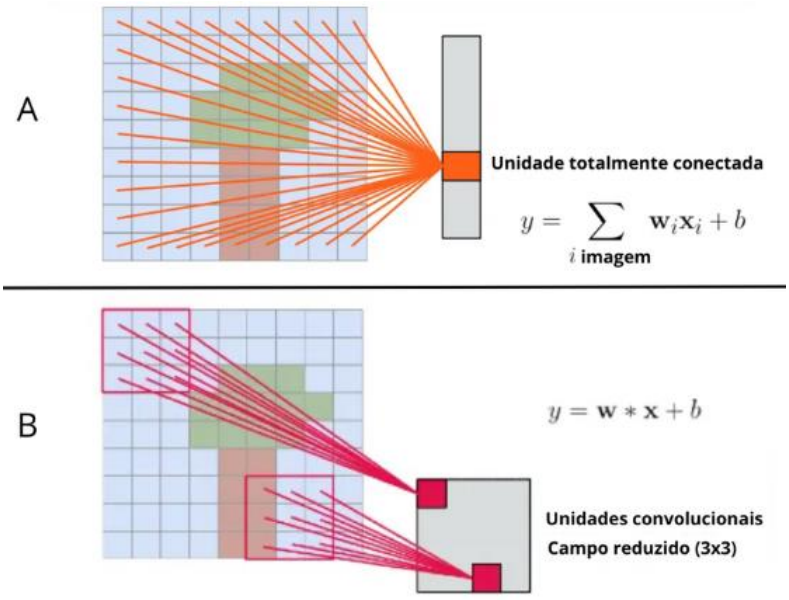
A CNN utiliza conceitos como *kernel*, que é um núcleo de pequena dimensão que passa por toda a entrada, realizando um cálculo que irá representar o pixel por onde ele passou. A Figura 5 mostra a diferença entre uma rede neural padrão e uma convolucional, em que o quadrado rosa da parte B da Figura 5 representa o *kernel*. A Equação (2.2) representa a parte A da figura 5, no qual o cálculo é realizado utilizando um somatório de todos os *pixels* (x_i) por todos os pesos (w_i), que o tornaria inviável computacionalmente para uma imagem com muitos *pixels*.

$$y = \sum_{i \in imagem}^{n} w_i x_i + b \tag{2.2}$$

Já a Equação (2.3) representa a parte B da Figura 5, no qual o peso (w) se mantém contante e é realizado a operação de convolução, representada pelo símbolo " $*$ " , a cada deslocamento do kernel (x), que nesse caso é uma janela 3 por 3. Note ainda, que "b" é o viés (bias), um termo extra que ajuda a ajustar a saída do neurônio.

$$y = w * x + b \tag{2.3}$$

Figura 5 – Comparação entre uma rede neural totalmente conectada (A) e uma rede neural convolucional (B).



Fonte: Adaptado de (HUYNH, 2022).

O YOLO (*You Only Look Once* – Você só olha uma vez) é um dos modelos de detecção de objetos que utiliza como base uma rede neural convolucional, desenvolvido em 2015 por Joseph Redmon e Ali Farhadi. O YOLO é baseado na rede convolucional DarkNet (JIANG, ERGU, *et al.*, 2021), que difere das outras técnicas de detecção por utilizar a abordagem de passagem única (*Singl*e pass), isso quer dizer que basta uma única verificação da imagem que ele já envia para a rede neural. Por esse motivo ele é denominado como "*You Only Look Once*" (Você só olha uma vez) (ALVES, 2020).

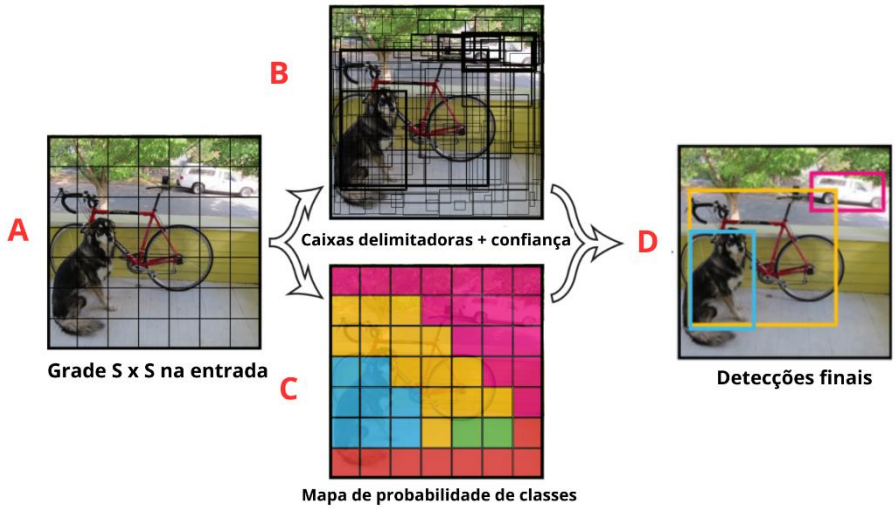
Segundo (DIWAN, ANIRUDH e TEMBHURNE, 2022) um dos principais motivos do YOLO ficar tão popular entre os entusiastas da área é o seu foco na velocidade da detecção com uma precisão razoável, pois mesmo perdendo para o modelo Fast-RCNN em precisão, com uma taxa percentual de 63,4 para 70, ele consegue ser 300 vezes mais rápido. O trabalho (TIAN, YANG, *et al.*, 2019) complementa que a razão dessa velocidade surpreendente pode estar relacionada ao fato de que o YOLO transforma o problema de detecção em um problema de regressão. Pois, em vez de analisar diferentes regiões da imagem para identificar em que local os objetos podem estar e depois classificá-los, ele divide a imagem em várias pequenas partes e para cada uma delas prevê diretamente o tipo de objeto, sua localização e demais informações, sem separar em duas etapas como as abordagens tradicionais.

Ao longo dos anos, o YOLO teve diversas versões com melhorias significativas. Entre elas, a sua segunda versão adicionou o conceito de caixas de âncora, que são retângulos de tamanhos pré-definidos que auxiliam as caixas delimitadoras previstas a se ajustarem mais precisamente às dimensões esperadas dos objetos identificados e *clusters* (agrupamentos) de dimensões. Já o YOLOV4 lançado em 2020, tonou-se o mais rápido e preciso da época (BOCHKOVSKIY, WANG e MARK, 2020), adicionando a função de perda e de detecção sem ancoras, o que tornou sua arquitetura ainda mais simples e direta (DERRENGER, 2023).

Em 2022, a empresa chinesa Meituan disponibilizou a versão 6 do modelo, implementando em seus robôs autônomos de entregas de delivery (DERRENGER, 2023). Este fato demonstra o imenso potencial dessa tecnologia em aplicações reais, especialmente com o YOLOV8 criado pela Ultralystics HUB, uma das versões mais populares, devido a sua maior flexibilidade e precisão aprimorada, com um desempenho melhor em atributos gerais. Neste presente momento, a versão mais

atual é o YOLOV11 com diversas inovações e recursos para as mais variadas aplicações (DERRENGER, 2023).

Figura 6 – Etapas do modelo de detecção de objetos do YOLO: (A) divisão da imagem em grade, (B) predição das caixas delimitadoras e confiança, (C) mapa de probabilidade de classes, e (D) detecções finais.



Fonte: Adaptado de (REDMON, DIVVALA, *et al.*, 2016).

O funcionamento do YOLO, considerando-o na sua 4ª versão, consiste em dividir a imagem em S por S células, pequenas partes da imagem, cada célula possui caixas delimitadoras também conhecidas como *bounding box*, cuja função é analisar e extrair informações de uma parte da célula. A caixa delimitadora é representada por um retângulo que contém a localização do objeto, que pode ter mais de uma caixa para um mesmo objeto. Cada *bounding box* possui um nível de confiança ilustrado pela espessura do traço do retângulo, como pode-se observar na parte B da Figura 6, que quanto maior a espessura, maior a certeza daquele objeto (PIEMONTEZ, 2022).

Além do grau de confiança, também é calculada uma probabilidade da possível classe daquele objeto, a classe é a classificação do tipo de objeto, por exemplo: uma pessoa, um carro, uma maçã, entre outros. Na parte C da Figura 6, mostra-se os possíveis objetos de cada caixa destacados em cores diferentes. Para fazer essa classificação geralmente é comum utilizar *o dataset* COCO (*Common Objects in Context* – Objetos comuns em contexto): conjunto de dados de detecção, segmentação e legenda de objetos em grande escala. O COCO é um dos *datasets*

mais utilizados para fazer testes e validações, contando com 80 categorias de objetos (LIN, MAIRE, et al., 2014).

Outra informação importante gerada pela caixa delimitadora é a localização, que se constitui da posição central do objeto, sua altura e sua largura, informações que foram essenciais para o desenvolvimento deste trabalho.

Na parte da B da Figura 6, são ilustradas inúmeras caixas delimitadoras que foram geradas pelo algoritmo, porém, várias delas possuem uma baixa probabilidade de conter um objeto, não enquadram nenhum objeto ou possuem uma área compartilhada com o mesmo objeto. Para eliminar essas caixas irrelevantes, o YOLO aplica a técnica de Supressão Não Máxima (*Non-Maximum Suppression*), que descarta as *bounding boxes* redundantes e mantém apenas as mais relevantes, conforme se observa na parte D da Figura 6, que mostra o resultado final do modelo.

As âncoras são introduzidas a partir do YOLOV2, que facilitam a adaptação de objetos de tamanhos variados (PIEMONTEZ, 2022). Mesmo que esse conceito tenha sido fundamental para o YOLO nas primeiras versões, ele foi modificado a partir da quarta versão para modelos detectores de objetos sem âncoras (*anchorless*), baseados no FCOS: *Fully Convolutional One-stage Object Detection* (Detecção de Objetos Totalmente Convolucional de Uma Etapa), obtendo assim um desempenho superior aos modelos com âncoras (CHANDRA e RATH, 2022).

Em resumo, o sistema do YOLO funciona ao passar uma rede convolucional uma única vez na imagem, prevendo as caixas delimitadoras e a probabilidade das suas respectivas classes. A cada nova imagem no frame a rede é repassada, tornando o processo muito rápido, ao ponto de se conseguir processar um streaming de vídeo em tempo real com menos de 25 milissegundos de latência e alcançando no mínimo 45 frames por segundo, utilizando um *hardware* de desempenho moderado e (REDMON, DIVVALA, et al., 2016).

Este funcionamento do YOLO é exemplificado pela Figura 7, que mostra o tratamento da imagem ajustando seu tamanho, executando a rede neural convolucional e por último aplicando a técnica de supressão não máxima.

Figura 7 – Etapas do processo de detecção do YOLO: redimensionamento da imagem, execução da rede convolucional e supressão de não-máximos.

Fonte: Adaptado de (REDMON, DIVVALA, et al., 2016)

2.3.1 Transferência de aprendizado

No contexto de aprimoramento do desempenho de redes neurais convolucionais, destaca-se o conceito da transferência de aprendizado, popularmente conhecido como *transfer learning*. Ele é definido como uma técnica de *machine learning*, em que um modelo pré-treinado em uma determinada aplicação é modificado para atuar em outra situação (BARZNJI, 2020). Desta forma, é essencial entender bem o modelo utilizado para se fazer as alterações necessárias e ter sucesso na nova tarefa.

Esse conceito é amplamente utilizado em diversas áreas de *machine learning*, mas também especificamente em visão computacional, pois se obteve bastante sucesso. Um desses casos é o do artigo (MONTALBO, 2020), que mostra a transferência de aprendizado do modelo YOLOv4-Tiny para detecção de glioma, meningioma e tumores cerebrais da hipófise em ressonâncias magnéticas com altas taxas de sucesso.

2.4 Contexto dos Eletropostos

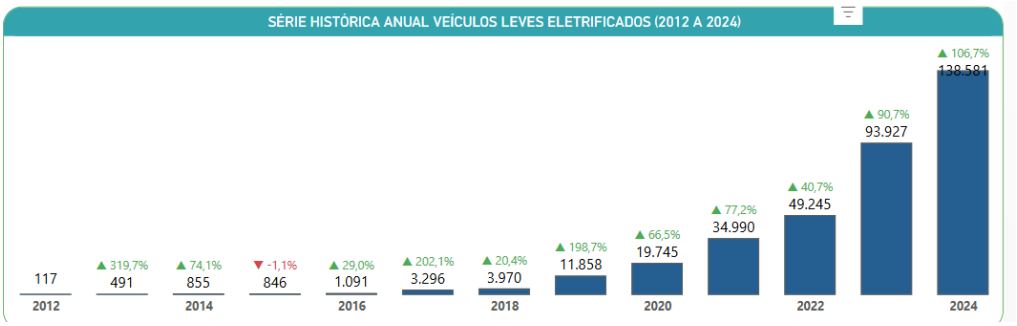
As mudanças climáticas são vistas cada vez mais como ameaça real à humanidade, resultadas principalmente das elevadas emissões de gases de efeito estufa (GEE) produzidos pela humanidade (ANNASWAMY, JOHANSSON e PAPPAS, 2024). O sexto relatório de avaliação do Painel Intergovernamental de Mudanças

Climáticas (IPCC, 2023) enfatiza que os veículos elétricos são aliados essenciais para redução das emissões de GEE no transporte terrestre ao longo de seu ciclo de vida, principalmente quando a energia utilizada para carrega-los provém de fontes renováveis, como a energia solar.

Nesse contexto, os investimentos nesses veículos estão crescendo fortemente no mundo inteiro: no Brasil, foram emplacados 51.296 veículos leves eletrificados nos primeiros 4 meses de 2024, representando um aumento de 162% em comparação ao mesmo período de 2023. Esses dados revelam uma mudança expressiva no cenário da mobilidade elétrica em apenas um ano, o que mostra a rápida adesão da população brasileira (ABVE, 2024).

Entretanto, o Brasil enfrenta um grave problema de infraestrutura para atender todos esses veículos. Por exemplo, ao analisar os dados da Figura 8 disponibilizados pela ABVE, em novembro de 2024 foram registrados 359.012 veículos eletrificados vendidos e 10.622 eletropostos, estações de recargas de VEs, ativos no país. Ou seja, tem-se em média 33 veículos para 1 eletroposto. Neste contexto, é necessário um investimento para a instalação de novas estações de recargas para atender os novos usuários (ALBERY , QUEIROZ, *et al.*, 2024), principalmente porque a maioria desses eletropostos estão concentrados na região Sul e Sudeste do país (ABVE, 2024).

Figura 8 – Gráfico da evolução de veículos elétricos no Brasil.



Fonte: (ABVE, 2024).

2.4.1 Estações de Recarga

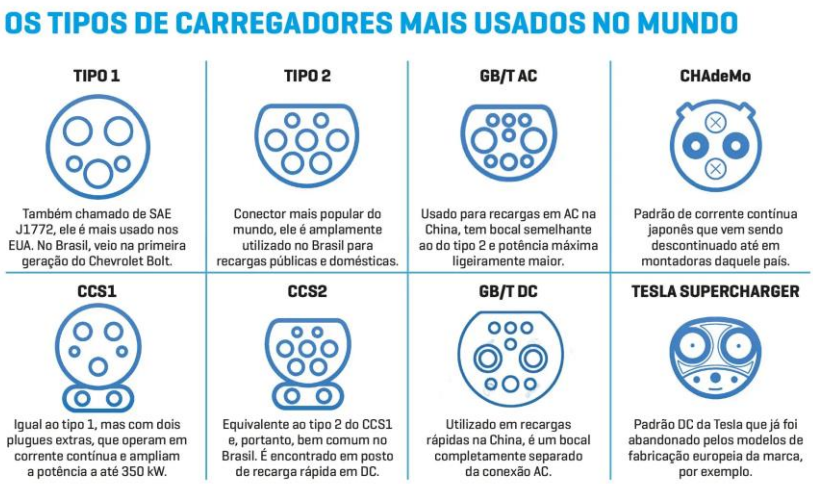
Existem dois tipos principais de padrões de sistemas de carregadores de veículo elétrico (VE): a norma SAE J1772 e IEC 61851. Onde a primeira é utilizada mais na

Europa e a segunda mais America. Neste trabalho irar focar-se na IEC 61851 conforme especificado pelo fabricante do carregador. Nessa norma há 4 modos diferentes de sistemas de carregadores de VE, os modos 1 e 2 se caracterizam por um carregamento via corrente alternada implementado em ambientes residenciais, que se diferenciam por possuírem uma limitação de corrente de 16A e 32A respectivamente (CHEN, ZHANG, et al., 2020). Adicionalmente, o modo 2 é preferível por ter uma proteção contra choques e incêndios elétricos.

O sistema de modo 3 e 4 são focados em locais públicos ou comerciais, com uma capacidade de potência bastante elevada quando comparada aos modos anteriores, pois pode chegar até 50 kW no modo 3 e até acima de 100 kW no modo 4, nesse caso é necessário um circuito e tomadas para fornecimento de energia exclusivo para os seus fornecimentos de energia. A principal diferença entre os dois tipos está na potência de saída. No modo 3, a operação é feita com corrente alternada (CA), em que o próprio carro realiza a conversão de energia de CA para corrente contínua (CC). Já no modo 4, utiliza-se diretamente a CC, o próprio carregador responsável pela conversão (CHEN, ZHANG, et al., 2020).

Além de carregadores diferentes, também existem tipos variados de *plugs* (conectores), que foram criados para atender as especificidades de cada marca, tipo de corrente, diferentes sistemas de comunicação e região, em que cada padrão de plugs domina um mercado, separando-os em América do Norte, Europa, China e Japão (SANTOS, 2024). A Figura 9 ilustra as peculiaridades de cada tipo de conector, assim como o local onde eles são mais utilizados.

Figura 9 – Tipos de *plugs.*



Fonte: (RODRIGUEZ, 2024).

Entretanto, a diversidade de conectores gera uma falta de padronização que afeta diretamente o usuário do veículo, pois os órgãos públicos e as empresas sempre irão escolher um ou mais padrão de *plugs,* correndo o risco de gerar uma incompatibilidade entre os VEs e os carregadores para alguns tipos de usuários. Entretanto, existem algumas iniciativas, como a da organização CharIN (*Charging Interface Initiative*) para implementar padrões globais de conectores de carregamento. Essa iniciativa facilitaria a vida dos usuários, pois poderiam escolher qualquer estação para carregar seus veículos, independentemente do fabricante ou localização geográfica (FRANTZ, 2023).

2.5 Message Queuing Telemetry Transport

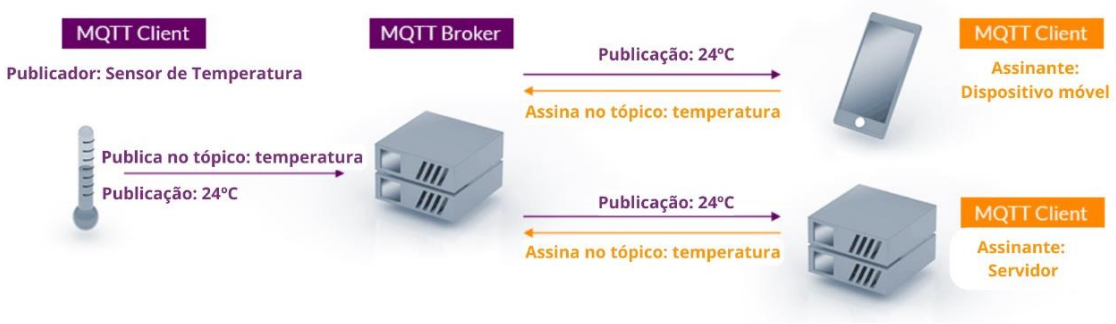
O MQTT, *Message Queue Telemetry Transport* (Transporte de Filas de Mensagem de Telemetria) é um protocolo de comunicação desenvolvido através de uma parceria entre engenheiros da IBM e da Eurotech em 1999 (SONI e MAKWANA, 2017). Originalmente, esse protocolo surgiu para a telemetria entre sensores em oleodutos e satélites, dessa maneira ele precisava ser um protocolo que operasse com poucos recursos computacionais, ser confiável e ter uma baixa largura de banda. Assim, por causa das suas características, ele passou a ser muito utilizado em sistemas de internet das coisas (IOT), sendo implementado por muitas empresas,

como o Facebook e a Amazon, que o adotou como protocolo de comunicação em um de seus sistemas (TORRES, ROCHA e DE SOUZA, 2016)

Seu funcionamento é estruturado em uma arquitetura de publicador/assinante (*publisher/subscriber),* ilustrado na Figura 10. Um dispositivo publicador, por exemplo, publica uma mensagem em um determinado tópico e todos os assinantes que estiverem inscritos previamente nesse tópico irão receber aquela mensagem. Muito semelhante ao sistema do aplicativo X (antigo Twitter), em que para visualizar as mensagens de um conta privada é necessário segui-la.

Entretanto, os dispositivos não se comunicam diretamente, pois todas as mensagens passam por um centralizador chamado "*broker*", responsável por gerenciar as mensagens e os tópicos, este último funciona como um identificador que permite a organização e o roteamento das mensagens entre os publicadores e os assinantes. Adicionalmente, um mesmo equipamento pode ser tanto um cliente quando um servidor.

Figura 10 – Arquitetura MQTT



Fonte: (ORG, 2024)

Assim, este protocolo se destaca pela facilidade de implementação, compatibilidade com diversos sistemas, alta escalabilidade, eficiência e segurança, utilizando protocolos de autenticação modernos (FARIAS e SILVA, 2024). Todos esses fatores tornam o MQTT um excelente candidato a projetos que não necessitam de uma complexidade elevada, como protótipos e testes. Todavia, este protocolo também apresenta algumas desvantagens como a dependência do *broker*, ou seja, caso ocorra algum problema com o *broker* toda a comunicação é interrompida.

3 METODOLOGIA

Neste capítulo, é enfatizado inicialmente a ideação do projeto, apresentando as principais características do eletroposto que serviu como base para este estudo. Em seguida, são abordadas as principais ferramentas, recursos e metodologias utilizadas, incluindo a linguagem de programação, os modelos de detecção, o sistema supervisório e o protocolo de comunicação, destacando as razões que fundamentaram as escolhas desses itens.

Esta parte conta com um teor um pouco mais teórico e servirá como base para o Capítulo 4, em que é mostrado como todos esses elementos foram implementados na prática.

3.1 Estudo de Caso do Eletroposto

O foco deste trabalho é o desenvolvimento de um sistema de monitoramento para eletropostos, visando extrair dados que possibilitam uma melhor gestão da estação, através da visão computacional. Dessa forma, optou-se por utilizar como caso de estudo o eletroposto sediado em Campinas/SP, projetado e construído por meio de um projeto de Pesquisa & Desenvolvimento (P&D), intitulado “Eletropostos Integrados à Tecnologia Nacional de Baterias (Chumbo-Carbono) e Sistemas Fotovoltaicos”. Realizado por integrantes do Laboratório de Armazenamento e Mobilidade (LAM) da UFPE, em parceria com a CPFL Energia, o Instituto de Tecnologia Edson Mororó Moura (ITEMM) e a Agência Nacional de Energia Elétrica (ANEEL).

A escolha desse projeto como base para o presente estudo deve-se à participação ativa do autor nesta pesquisa, o que possibilitou uma maior facilidade nas realizações dos testes deste trabalho.

O eletroposto deste estudo, apresentado na Figura 11, é uma estação de recarga sustentável constituída de um sistema armazenamento de energia por baterias, um sistema fotovoltaico e dois carregadores rápidos, em que o carregador 1 possui dois plugs CCS2 (60kW) e um plug CA tipo 2 (22kW), e o carregador 2 possui um plug CCS2 (60kW), um plug Chademo (60kW) e um plug CA tipo 2 (22kW) (MAIA

JR, , et al., 2024). Assim, a estação consegue suprir 6 carros carregando ao mesmo tempo.

Figura 11 – Eletroposto de estudo (Campinas/SP).

Fonte: (MAIA JR, , et al., 2024).

Esta estação foi projetada baseada no conceito microrrede: “Um grupo de cargas interconectadas e recursos energéticos distribuídos dentro de limites elétricos bem definidos que atuam como uma única entidade controlável em relação à rede“ (DANLEY, 2019), em que possui um Sistema de Gestão de Energia (SGE), popularmente conhecido como *Energy Management System* (EMS), responsável pelo gerenciamento de todos os equipamentos conectados a essa rede, de maneira a otimizar o uso da energia, minimizando os impactos causados à rede de distribuição pelo uso expressivo de veículos elétricos, além de trazer um maior custo benefício a própria loja onde se encontra o eletroposto.

O eletroposto foi inaugurado oficialmente e aberto ao público no dia 20/10/2023. Desde então, foi realizada a operação assistida até o mês de agosto de 2024, etapa importante para se corrigir pequenos problemas que houveram nesse tempo. Desta forma, o desenvolvimento e as análises deste trabalho foram realizados a partir de outubro de 2024, período em que a estação de recarga estava funcionando ativamente com uma demanda relativamente alta de veículos em diversos horários. A

execução das detecções com diferentes carros e em períodos variados foi essencial para o aprimoramento do sistema, tornando-o mais preciso em diversas situações.

A Figura 12 ilustra as seis câmeras ativas presentes no eletroposto, destacando as 2 câmeras que foram utilizadas para esse estudo. Elas são do modelo NVD1408P da Intelbras com o recurso Intelbras Cloud que permite a conexão com a internet via *gateway* IOT2050 da Siemens com chip 4G implementado na estação que fornece internet para todos os dispositivos da microrrede.

Figura 12 – Câmeras do eletroposto.

Fonte: Autor (2025).

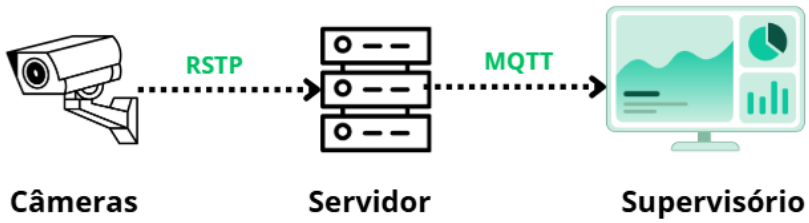
Essas câmeras foram projetadas e instaladas antes da ideia deste presente trabalho, portanto, seu foco era apenas de monitoramento voltado para segurança e não especificamente destinada a detecção de veículos. Fato este que demonstrou ser um desafio interessante para essa pesquisa, pois permitiu várias soluções criativas para alcançar o objetivo esperado, um tópico que será mais evidenciado nas próximas seções.

O acesso as câmeras se deram a partir do aplicativo para celular da Intelbras: ISIC LITE. Este programa, além de transmitir em tempo real, possui uma memória interna para gravação dos vídeos, facilitando os testes de detecção para serem realizados em diferentes períodos.

3.2 Arquitetura do Sistema

Entendida a parte física da estação da recarga e o monitoramento das câmeras, nesta seção será demonstrado a disposição de todo o sistema. Ilustrada pela Figura 13, a arquitetura do sistema é composta pelas câmeras que enviam a imagem em tempo real, através do protocolo RSTP (*Real Time Streaming Protocol* – Protocolo de Transmissão em Tempo Real) para o servidor que pode ser desde um computador tradicional local ou até mesmo uma Raspberry PI. Em seguida, o servidor irá tratar os dados e realizará a detecção utilizando o YOLOV8, a partir de um programa desenvolvido na linguagem de programação *Python*, que será abordado mais adiante. Por fim, este último envia as informações para o supervisório pelo protocolo MQTT, que irá realizar os últimos ajustes para poder apresentar as informações desejadas.

Figura 13 – Arquitetura Inicial.



Fonte: Autor (2025).

Em relação a comunicação, foi utilizado o protocolo RSTP por ser nativo nas câmeras e pela compatibilidade com bibliotecas do *python*, garantindo uma comunicação mais direta. Já a opção do protocolo MQTT foi sustentada principalmente por sua facilidade de implementação entre o supervisório e o *python*, possibilitando uma fácil integração a outros sistemas numa possível expansão do projeto. Além disso, esse protocolo é leve, simples e eficiente (ORG, 2024), se encaixando bastante nesta versão inicial do trabalho.

O desenvolvimento do *script* de detecção foi realizado em *python,* por ter sido uma linguagem bastante utilizada por trabalhos na área de detecção de veículos

usando o OpenCV e YOLO como bibliotecas base, em projetos como (V. ASSUMPÇÃO, DE SOUZA, *et al.*, 2024), (LUCAS, 2022) e (VAZ, 2015).

O supervisório pensado para este trabalho foi baseado no Node-RED, uma ferramenta de desenvolvimento *low-code*, que permite a criação de aplicações tanto por programadores, quanto por não programadores, com mínimo esforço na escrita de código, em uma linguagem de programação específica na instalação e na configuração de ambientes (WASZKOWSKI, 2019). A escolha do Node-Red se deu pelo fato de já ter sido utilizado como supervisório na pesquisa do projeto P&D que precedeu este trabalho, o que facilitou a continuidade e o desenvolvimento do projeto. A plataforma permite diversas integrações com diferentes protocolos e sistemas, possuindo também uma ferramenta de *dashboard* próprio, no qual é possível criar interfaces gráficas que atendem as demandas deste trabalho de forma prática, eficiente e sem o custo de uma licença (NODE-RED, 2025).

Além disso, o node-red está sendo integrado em diversas aplicações reais em conjunto com empresas referências no mercado, como por exemplo o caso da Siemens (THARWAT, 2023).

O modelo de detecção escolhido para esse projeto foi o YOLO, como já mencionado anteriormente. Esta escolha se deve, sobretudo, por ser um dos modelos que mais vem se destacando em projetos de visão computacional. Com uma comunidade bastante ativa, o YOLO está sendo utilizado em soluções de grandes empresas como Meituan, que implementou essas aplicações em produtos comerciais, o que reforça sua confiabilidade (DERRENGER, 2023). Tudo isso, somado as suas vantagens explicadas na seção 2.3, contribuiu para a seleção deste modelo, que entre todos os outros recursos escolhidos, é um dos principais do projeto.

A versão utilizada foi a YOLOV8, com base no artigo “Comparative Analysis of YOLOv8 and YOLOv10 in Vehicle Detection: Performance Metrics and Model Efficacy” (SUNDARESAN , AL RABBANI, et al., 2024), que conclui uma equivalência entre os dois modelos na detecção dos veículos mais comuns, porém com uma leve vantagem para o YOLOV8, especificamente na identificação de carros, que é mais adequado a este estudo. A variante mais nova do modelo, a YOLOV11, não foi considerada por ser uma versão muito recente, apresentando algumas instabilidades.

Todo o *script* (código) em *python* e do supervisório foram desenvolvidos e executados em um computador com processador Intel(R) Core(TM) i7-8550U CPU 1.80GHz, com 1992 Mhz, com 4 Núcleos e com 8 processadores lógicos baseado em arquitetura de 64 bits e com 12 GB de memória RAM instalada. Não foi utilizada uma GPU no *script* de detecção para realizar os testes em uma máquina de baixo poder computacional, tendo em vista que esse sistema seria focado para dispositivos de baixo custo e baixa capacidade de processamento como uma *Raspberry Pi,* por exemplo. Vale ressaltar que foram testados todos os modelos do YOLOV8, todavia, houve uma preferência por utilizar os menores modelos do YOLO, do modelo médio (YOLOv8m) ou inferior, como se pode observar na tabela 1.

Tabela 1 – Características dos modelos do YOLOV8

Modelo	**Tamanho (pixels)**	**mAPval 50-95**	**Velocidade CPU (ms)**	**Parâmetros (M)**	**FLOPs (B)**
YOLOv8n	640	37.3	80.4	3.2	8.7
YOLOv8s	640	44.9	128.4	11.2	28.6
YOLOv8m	640	50.2	234.7	25.9	78.9
YOLOv8l	640	52.9	375.2	43.7	165.2
YOLOv8x	640	53.9	479.1	68.2	257.8

Fonte: Adaptado de (DERRENGER, 2023).

A Tabela 1 apresenta as principais características dos modelos da família YOLOv8. Todos os modelos operam com imagens de entrada de 640x640 *pixels*, porém diferem significativamente em termos de precisão média (mAPval), velocidade de processamento e complexidade computacional. À medida que os modelos aumentam de tamanho, do YOLOv8s até o YOLOv8x, há uma melhora gradual na acurácia da detecção (mAPval), ao custo de um maior número de parâmetros e operações computacionais (FLOPs). Ou seja, modelos maiores tendem a oferecer melhor desempenho na detecção de objetos, mas demandam maior capacidade de *hardware* e apresentam tempos de inferência mais altos.

4 DESENVOLVIMENTO DA APLICAÇÃO

Apresentados todos os recursos e ferramentas utilizadas no projeto, é detalhado agora o desenvolvimento do trabalho na prática. A priori, tentou-se comunicar com as câmeras via protocolo RSTP em tempo real. Entretanto, por questões burocráticas, em decorrência da transferência do gerenciamento do eletroposto para a loja gestora, o acesso direto as câmeras seria inviável. Desta forma, optou-se pela utilização do aplicativo ISIC LITE que foi disponibilizado para os testes, o que foi um ponto positivo, pois com ele pode-se acessar gravações realizadas em dias passados.

A priori, foi utilizado o *software BlueStacks* que permite que aplicativos Androids sejam executados no Windows, já que o ISIC LITE é um programa de *smartphones*. Assim, pôde-se observar o funcionamento do eletroposto em diferentes condições: durante o dia e a noite, com diferentes quantidades de vagas ocupadas, carros com diferentes tamanhos e cores, etc. Tudo isso influenciou na criação do *script* de detecção e processamento do YOLOV8.

Em seguida, gravou-se essas situações com o programa OBS Studio para depois passarem pelo código do *python*, nele são realizados diversos testes com diferentes tamanhos de modelos.

4.1 Desenvolvimento do código inicial

A princípio, foi realizado um código base, em que todo o desenvolvimento do *script* final foi feito a partir dele. Basicamente, representava uma aplicação com o objetivo de detectar os objetos do *dataset* COCO a partir do YOLOV8. Assim, com a assistência de um monitor auxiliar o código monitora a tela deste monitor que continha as gravações do eletroposto e executa a detecção de acordo com o pseudocódigo da Figura 14.

Figura 14 – Pseudocódigo inicial.

```
#Início do programa
#1. Inicializar o ambiente:
Importa biblioteca mss
Importa biblioteca numpy
Importa biblioteca opencv
Importa biblioteca ultralytics

Monitora a imagem usando a biblioteca *mss* com as dimensões do monitor
auxiliar.
Configura o modelo YOLO carregando os pesos e parâmetros necessários.

#2. Entrar no loop de captura:
While not interrupted:
      Captura em tempo real
      #Formata o frame para ser compatível com a biblioteca OpenCV
      Converte o objeto retornado pela biblioteca mss em um array NumPy.
      Converte o frame de BGRA para BGR.

      #Processa a imagem com o modelo YOLO:
      Executa o modelo YOLO no frame fornecido.
      For detections in detections:
            Retorna as detecções identificadas.
            Retorna as coordenadas das bounding boxes.
            Retorna as classes dos objetos detectados.
            Retorna a confiança associada a cada detecção.
            #Usa o método `plot` da biblioteca ultralytics para exibir as
      bounding boxes, classes e valores de confiança diretamente no frame
            Desenha as detecções no frame.

      #3. Exibir os resultados:
      Mostrar o frame processado com as detecções desenhadas em uma nova
      janela.

#5. Finalizar
If usuário pressionou a tecla de saída “q”:
      Encerra o While

End While
Libera captura de tela.
Libera janelas abertas.
#Fim do programa
```

Fonte: Autor (2025).

Inicialmente, o código define o monitoramento da imagem do monitor auxiliar e carrega o modelo do YOLOV8. Para os testes, foram utilizados os cinco modelos disponíveis: *nano, small, medium, large e extra large*. Em seguida, o programa entra no laço de repetição (*while*), em que começa capturando a imagem em tempo real, cujo frame é convertido em um vetor da biblioteca *numpy* e transformado em uma representação de cores BGR (*Blue, Green and Red*), ambas as transformações da imagem para tornar a captura compatível com a biblioteca OpenCV.

Em seguida, este frame passa pelo modelo do YOLOV8 e entra num laço de repetição (*for*), onde são realizas as detecções a princípio sem restrições, retornando as coordenadas das *bounding box*, as classes dos objetos detectados e a confiança dos mesmos. Assim, o programa se repete até que o usuário queira encerrá-lo, para isso, basta ele apertar a letra “q”, o que interromperá o laço *while*, liberará a captura, fechará as janelas abertas e encerrará o programa.

O código, a partir da execução do modelo do YOLO, retorna em cada interação alguns parâmetros chaves do desempenho do modelo, são eles: objetos detectados, tempo de pré-processamento, tempo de pós-processamento, tempo de inferência e a resolução da imagem processada em pixels. O tempo de pré-processamento é o tempo em que a função realiza os ajustes necessários na imagem antes de enviá-la ao modelo, como redimensionamento, normalização, ou conversão para o formato tensor. O tempo de inferência é o tempo gasto na execução da rede neural para identificar e classificar os objetos presentes na imagem e o tempo de pós-processamento equivale ao tempo gasto de filtrar caixas delimitadoras redundantes usando *Non-Maximum Suppression* e organizar os resultados finais, como classes e coordenadas das detecções.

Com o código base funcionando, começou-se o processo de refinamento e adaptações feitas no código. Como o objetivo principal era desenvolver um sistema que conseguisse identificar o status de disponibilidade da vaga e enviar essa informação via protocolo MQTT, foi necessário alterar o código nesses 3 itens:

1. Delimitação de vagas: Segregar e identificar quais vagas estão sendo ocupadas, a partir de sua posição.
2. Detectar apenas carros: Realizar um filtro para que o sistema não detecte todos os objetos, diminuindo o tempo de processamento.
3. Comunicação MQTT: Enviar a informação dos status da vaga para o broker MQTT

Assim, no capítulo de resultados será apresentado e discutido o código final com todos os aprimoramentos.

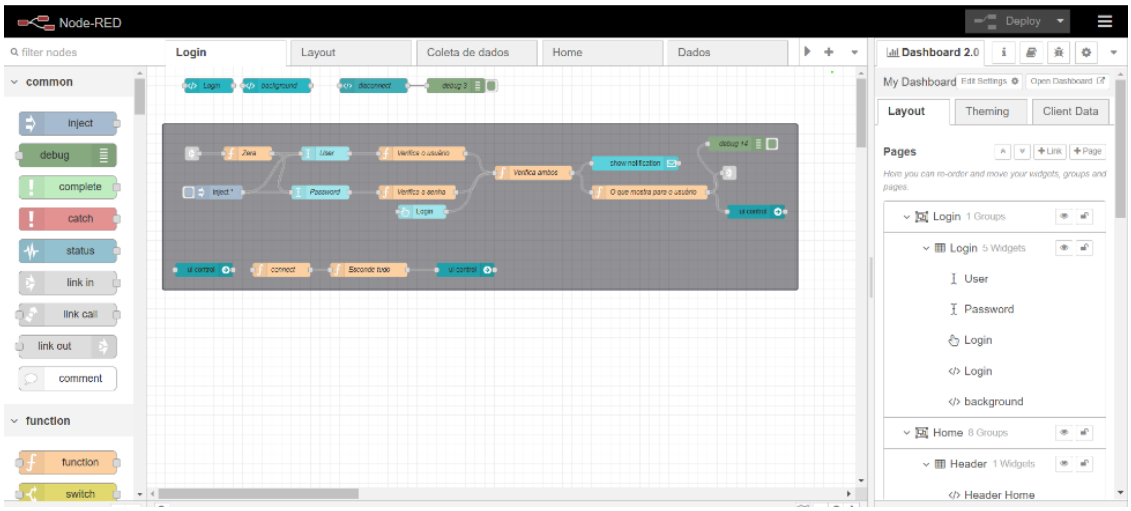
4.2 Desenvolvimento do *Dashboard* de Supervisão

A elaboração do *dashboard* é uma etapa crucial para o projeto, pois é nele que será apresentado ao usuário a disponibilidade das vagas e os demais dados coletados. Assim, será detalhada nessa seção todas as etapas de seu desenvolvimento.

4.2.1 Desenvolvimento com o Node-Red

Para construir esse *dashboard*, foi escolhida como ferramenta o Node-Red, por motivos já evidenciados anteriormente. Assim, foi utilizada a biblioteca *Dashboard* 2.0, desenvolvida e mantida pela comunidade do *Flowfuse*, comunidade que está bastante conhecida por elevar o Node-Red a um nível mais profissional e eficiente para ser utilizado em aplicações reais, o que foi vantajoso para este trabalho. A seguir, é mostrado um exemplo de imagem de como ficou a implementação da tela de login do *dashboard*. Assim como em todas as telas, o desenvolvimento foi realizado com os blocos pré-definidos da biblioteca em conjunto com a personalização direta a partir da trinca: CSS, HTML e Java*Script*; conforme pode ser visto na Figura 15.

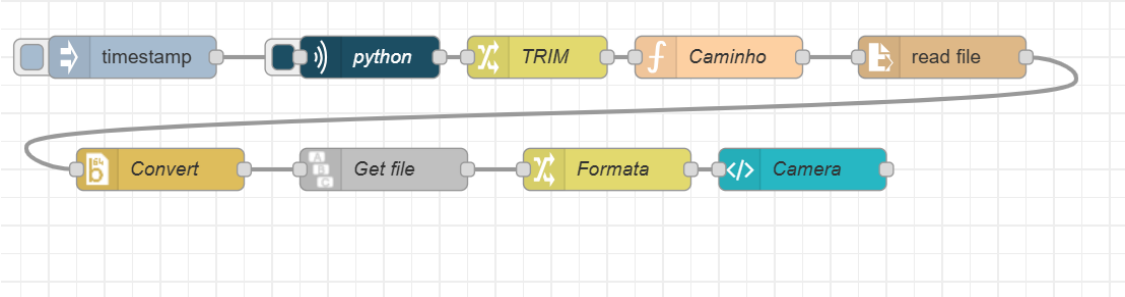
Figura 15 – Blocos do Node-Red.



Fonte: Autor (2025).

O desenvolvimento da tela de monitoramento diferenciou-se das demais, pois nela foi feita uma integração direta com o *script* de *python*, no qual o fluxo do Node-red executa o código através do nó *pythonshell* da biblioteca “node-red-contrib-*python*shell” e lê as imagens geradas. Em seguida, o código as formata e as envia para o nó *template* que é responsável por exibi-las na tela, processo este demostrado pela Figura 16.

Figura 16 – Fluxo da integração do *python* com Node-Red.



Fonte: Autor (2025).

4.2.2 Desenvolvimento do Layout

O sistema de monitoramento proposto por esse trabalho tem como objetivo uma interface intuitiva e eficaz, consoante com diretrizes das teorias da experiência do Usuário (UX) e o *design* de interface (UI), focando em colocar o usuário/operador no centro da aplicação, que também obteve ótimos resultados em aplicações industriais (KAMIZI, 2021). A interface foi planejada para ser executada em um monitor de computador ou uma televisão pequena, mas graças ao Node-Red é possível visualizá-lo até em celulares ou tablets devido a sua responsividade.

Por não ser um sistema bastante complexo, seu desenvolvimento teve como base duas das dez heurísticas que o cientista da computação e designer de interação Jakob Nielsen definiu como os princípios para avaliar a usabilidade de interfaces do usuário (MOMA, 2017). Assim, foram utilizados os conceitos do design minimalista e a consistência de padrões. O primeiro, para manter o equilíbrio entre a simplicidade e a eficácia, fundamental para garantir que o sistema cumpra seu objetivo, tendo em vista que um design limpo e sem elementos desnecessários deve, ao mesmo tempo,

fornecer informações visuais suficientes para que o sistema tenha uma boa usabilidade (KRUG, 2014). E o segundo, com a finalidade de manter um mesmo padrão nas interações, com um padrão de identidade claro e em toda a aplicação, pois desta forma o usuário consegue entender e presumir o comportamento do sistema (MOMA, 2017).

Além disso, a construção do *layout* da interface foi altamente influenciada pela pesquisa do estudo do trabalho de conclusão de curso do engenheiro de controle e automação Alexandre Motta (MENEZES, 2024), em que foi mostrado por meio de entrevistas com estudantes e profissionais da área, quais as cores e diagramação mais adequadas para sistemas supervisórios industriais. Desta forma, foi padronizada para a aplicação a cor cinza como cor de fundo, na qual foi a tonalidade mais escolhida na pesquisa (MENEZES, 2024). A cor secundária escolhida foi a laranja por causa da paleta de cores do LAM.

A aplicação foi dividida em *login*, tela principal de monitoramento, página de dados e a página de visualização do modelo, todas com um *header* fixo. A primeira tela é a de login que foi estruturada como uma página inicial simples, mas que mostra o eletroposto e a identidade do LAM. Conforme ilustrado na Figura 17.

Figura 17 – Página de Login.

Fonte: Autor (2025).

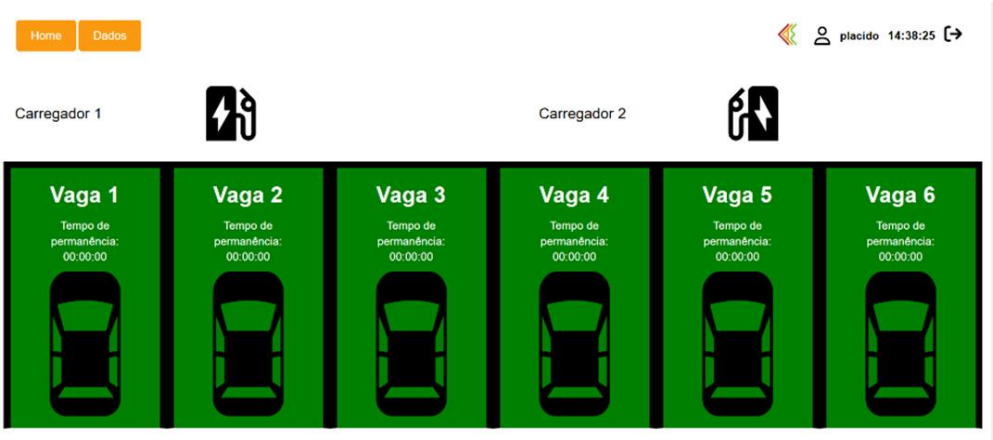
Assim, ao fazer o *login* o usuário entra na página de monitoramento e no canto superior se encontra o cabeçalho, também conhecido como *header,* presente na tela

de dados e na tela de visualização. Este foi diagramado com os botões de navegação *home* e dados inseridos no canto esquerdo superior, o título da página centralizado e em seu canto superior direito é exibido a logo do LAM e as informações do sistema: data, hora, usuário conectado, conexão e a opção de *logout;* estruturado conforme a pesquisa já mencionada anteriormente e ilustrado na Figura 18. A página principal de monitoramento, exibida pela Figura 19, foi projetada com o seu conteúdo principal diagramado no centro da tela, localização esta que é consenso nas aplicações de monitoramento (MENEZES, 2024).

Figura 18 – *Header.*

Fonte: Autor (2025).

Figura 19 – Página de monitoramento.



Fonte: Autor (2025).

Na Figura 19, é apresentada uma interface intuitiva que monitora em tempo real a ocupação das vagas do estacionamento de maneira clara e direta ao usuário. As vagas foram divididas de 1 a 6, as 3 primeiras são associadas ao carregador 1 e o restante ao carregador 2. As suas numerações foram baseadas de acordo com a Figura 20, em que as vagas 5 e 6 são apenas ilustrativas, pois da maneira como foram instaladas as câmeras no eletroposto não é possível capturar com clareza estas vagas.

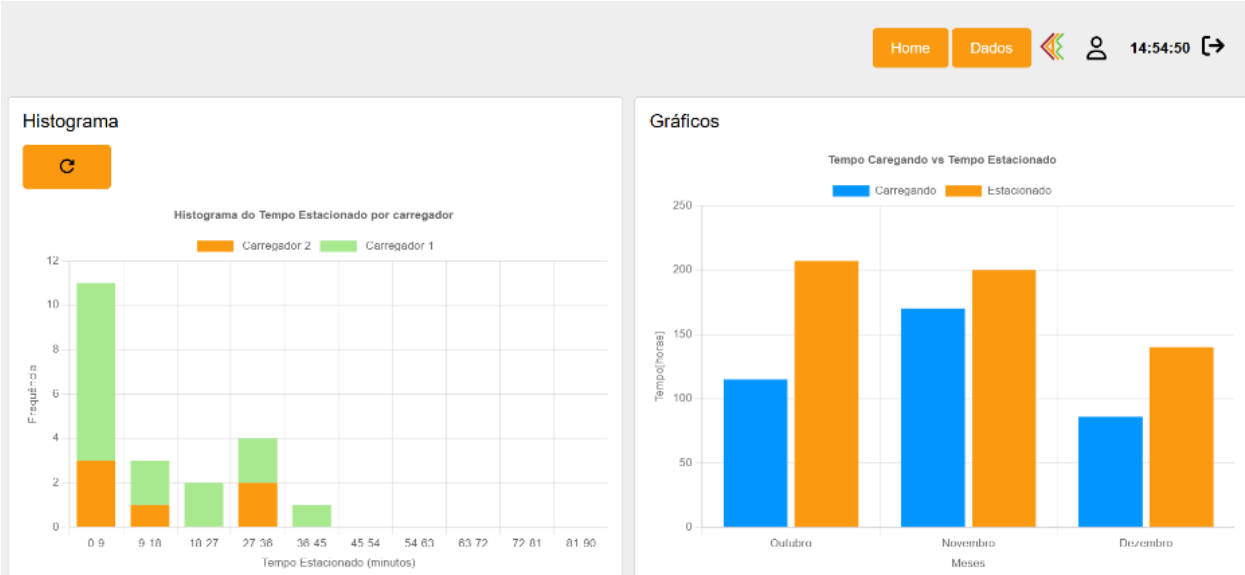
Figura 20 – Identificação e numeração das vagas capturadas pelas câmeras de monitoramento do eletroposto.

Fonte: Autor (2025).

Cada vaga é representada por um *card*, que exibe as informações de disponibilidade da vaga e o tempo de permanência do veículo estacionado, evidenciado com a cor verde para livre e vermelho pra ocupado, cores estas que geralmente são padrão em sistemas similares.

Já a página de dados, a qual possui as informações e estatísticas do monitoramento, foi elaborada com o mesmo *header* da página principal, adicionando gráficos e tabelas que contém os principais dados, destacados de forma clara ao usuário. O *card* a esquerda é constituído por um histograma que apresenta a distribuição de frequência dos tempos de permanência dos carros por cada carregador. Essa visualização possibilita ao usuário identificar os intervalos, em minutos, mais comuns em que os veículos permanecem estacionados, com um botão no lado superior esquerdo para atualizar os dados. Este histograma permite identificar com precisão os tempos mais frequentes, o que pode ser comparado com o real tempo de carregamento dos veículos, conforme se pode observar na Figura 21.

Figura 21 – Página de dados com visualização gráfica do tempo estacionado por carregador e comparação entre tempo de carregamento e tempo estacionado.



Fonte: Autor (2025).

É importante ressaltar que essa aplicação não foi implementada de maneira contínua, pois, para isso, seria necessário um *hardware* operando por 24 horas ou algum serviço de nuvem como a AWS. Dessa forma, os dados foram gerados a partir de alguns vídeos gravados pontualmente do eletroposto pelo aplicativo ISIC Lite, procurando coletar dados de várias situações diferentes. Logo, não são dados contínuos e não representam a atual situação da estação de recarga, mas são suficientes para validar este trabalho.

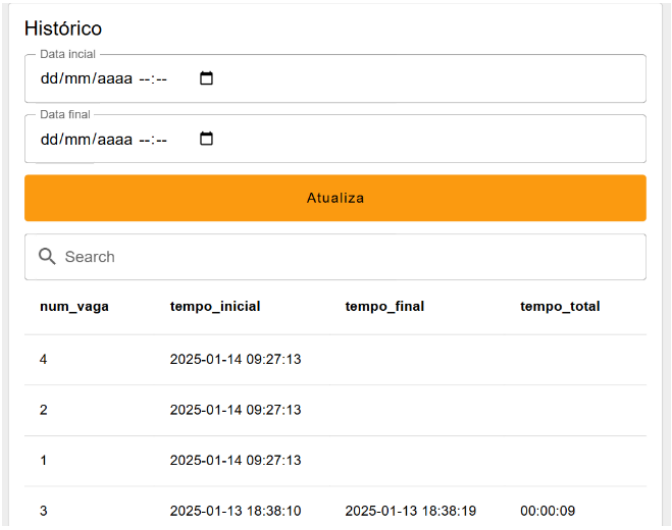
Ao lado direito do histograma na Figura 21, é exibido um *card* com o gráfico do tempo de carregamento total em relação ao tempo estacionado total, ambos considerando os dois carregadores em cada mês de monitoramento. Assim como o histograma, esses dados foram gerados a partir dos vídeos gravados para o teste.

Logo abaixo dos gráficos anteriores, observa-se mais um *card* contendo a tabela dos dados históricos contidos no banco de dados, como demonstrado na Figura 22. Nela é possível filtrar uma determinada data inicial e data final para se obter as informações do horário de permanência, o seu respectivo tempo total e o número da vaga, o que facilita a análise de padrões e tendências ao longo de períodos específicos. Essa funcionalidade permite realizar uma análise dos horários mais frequentes, proporcionando aos gestores uma visão clara dos períodos de maior

demanda. Além disso, a tabela também facilita a identificação de possíveis inconsistências ou picos inesperados de ocupação, permitindo a implementação de estratégias de melhoria contínua, como a necessidade de expansão da infraestrutura para atender a um volume maior de usuários.

Também foi posicionado um botão para a atualização da tabela e um campo para pesquisar algum dado específico da tabela. Nota-se que os 3 *cards* foram construídos de forma padronizada e minimalista, consoante com os princípios adotados em todo o *layout*.

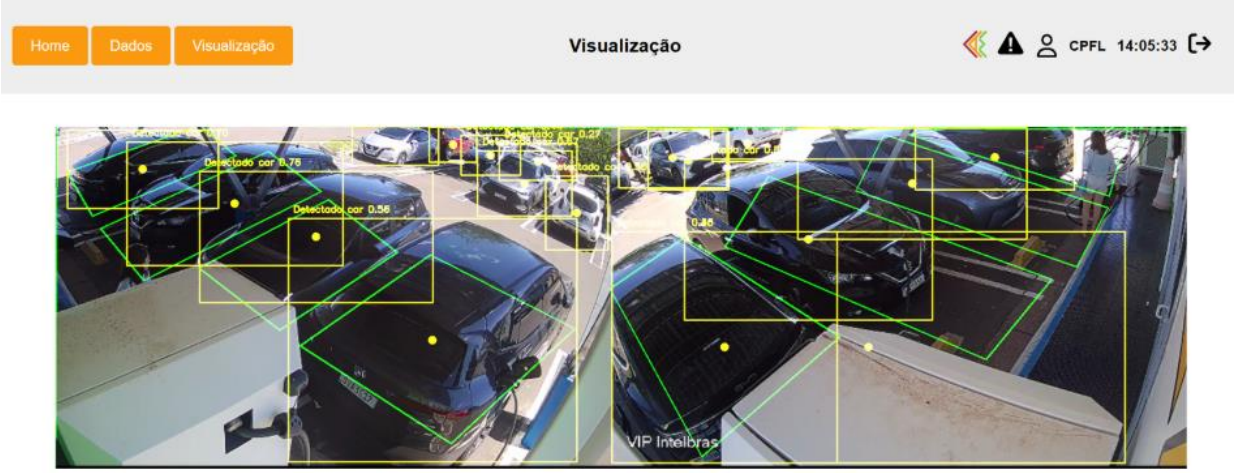
Figura 22 – Tabela do histórico de dados.



Fonte: Autor (2025).

Por último, tem-se a página de visualização em tempo real da detecção do modelo do YOLOv8, na qual é exibida o *output* do código do *python* com as marcações feitas no *frame,* como as *bounding box* e a separação das vagas, como ilustrado na Figura 23.

Figura 23 – Página de visualização do sistema de detecção de objetos utilizando YOLOv8.



Fonte: Autor (2025).

4.2.3 Coleta de Dados e Tratamento

A comunicação entre o *script python* e o Node-Red foi realizada através do broker mosquitto instalado no próprio computador, onde estão os demais sistemas. O broker foi configurado com o *ip* do *localhost* na porta 1884. Entretanto, nota-se que comumente é utilizado a porta 1883 como padrão para o protocolo MQTT, porém, essa porta já estava em uso por outro serviço. Ademais, foi inserido o login e senha para uma maior segurança na conexão, com a QoS (Qualidade do serviço) de valor 2, o tópico escolhido foi o “YOLOV8PN”. Desta forma, implementou-se essas configurações tanto no código *python,* quanto no Node-Red. Estabelecida a comunicação, definiu-se como padrão de mensagem um vetor de valores lógicos de tamanho 6, em que cada elemento representa a disponibilidade da vaga e sua posição no vetor o número da mesma.

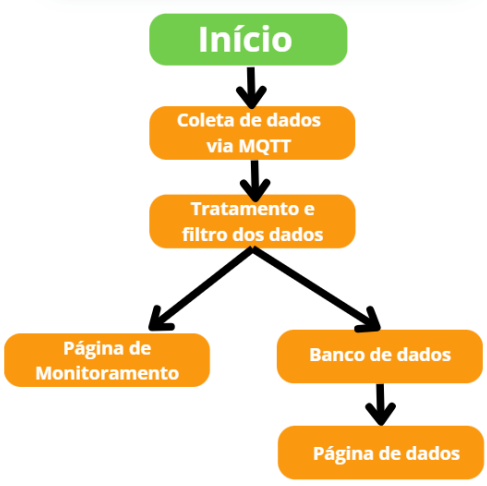
Quando o *script* envia o vetor para o Node-Red pelo mqtt, o Node-Red coleta, converte os dados no formato adequado e envia para um filtro de tempo. Filtro este que é implementado devido à algumas imprecisões de reconhecimento e às vezes o código em *python* não reconhece o carro durante poucos segundos, principalmente quando o modelo é *small* ou nano. Assim, o filtro serve apenas para mudar o status da vaga se o valor do elemento permanecer o mesmo durante 3 segundos, mitigando os erros de detecção.

Após o filtro de tempo, os dados são enviados para a página principal do *dashboard* e são apresentados como tempo de permanência e status da vaga. Paralelamente, eles também são enviados para o banco de dados relacional e de código aberto: Mysql. Foi criada uma tabela de monitoramento formada pelas colunas de tempo inicial, tempo final, número da vaga e tempo total.

Assim todos os gráficos e tabelas da página de dados são alimentados por uma única tabela no banco de dados, mantida na mesma máquina local que os demais *softwares*. Essa escolha inicial de usar um banco de dados local visa a facilidade de implementação, mas permite uma migração futura para a nuvem sem grandes obstáculos, como a *AWS* por exemplo.

Todo esse processo é realizado de forma contínua e paralela, graças ao processamento do Node-Red, que permite a execução simultânea de nós. Para uma melhor compreensão do fluxo natural do código foi ilustrado no fluxograma na figura 24.

Figura 24 – Fluxograma do Node-Red.

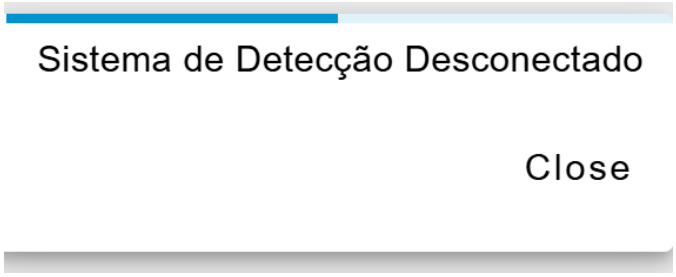


Fonte: Autor (2025).

Além disso, foi implementado no *header* a informação do status da conectividade entre o *dashboard* e a aplicação do *python*. A situação da conexão é obtida através do MQTT pela inscrição do Node-Red no tópico '$SYS/broker/clients/connected' do broker Mosquitto, que publica o número de clientes conectados sempre que algum

cliente se conecta ou desconecta. Dessa forma, quando a aplicação do *python* ou o próprio Node-Red se desconecta do *broker* por algum motivo, o *pop-up* demostrado na figura 25 aparece ao usuário e o ícone de conexão muda a depender da situação, conforme colocado pela figura 26.

Figura 25 – Pop-up.



Fonte: Autor (2025).

Figura 26 – Ícones de conexão.

Fonte: Autor (2025).

4.2.4 Banco de Dados

A definição de um banco de dados adequado é essencial para o projeto, pois ele guardará os dados que serão transformados em informações relevantes apresentadas pelo *dashboard*.

Assim, foi considerada a demanda de carregamento do eletroposto com base no projeto P&D mencionada no item 3.11 com uma média de 350 carregamentos por mês, o que resultaria um valor um pouco maior de dados armazenados no banco

durante o mês. Embora essa seja uma carga relativamente pequena para um banco de dados, optou-se por utilizar o Mysql como o sistema de gerenciamento de banco de dados, considerando a possiblidade do sistema escalar caso seja implementado em diversos eletropostos. Desta forma, foi escolhida a ferramenta MySQL Workbench para ser a tecnologia de banco, por simplificar a manutenção e criação das tabelas, com suporte a diagramas ERD (Entidade-Relação Diagrama) e por ser um *software* gratuito com o banco de dados *open source* mais popular do mundo segundo (TOLEDO e DE SOUZA, 2018).

O banco de dados foi construído como uma única tabela contendo as colunas: tempo inicial, tempo final, tempo total, id e o número da vaga. Nele foi inserido um *trigger* para que toda vez que determinada linha é atualizada com o tempo final do carregamento, seja automaticamente calculado o tempo total.

Os dados foram inseridos no banco através dos testes de validação do *dashboard.* Assim, tanto a tabela exibida quanto o histograma contêm os dados provenientes do código em *python*. Já os dados do gráfico do *card* a direita na Figura 21 foram adicionados manualmente para se ter uma ideia de visualização, pois para ter de fato esse gráfico funcional seria necessário a integração deste projeto com o sistema dos carregadores, algo totalmente possível, mas que não foi o objetivo do trabalho.

4.2.5 Broker MQTT

Para realizar a comunicação entre o *script python* e o *dashboard* de Node-Red foi adotado o *broker* MQTT do *software* Mosquitto. Essa escolha foi motivada pela vasta utilização do Mosquitto em diversos projetos da área, possuindo diversas funcionalidades como o gerenciamento de conexão e desconexão, o baixo consumo de recursos do sistema e também por ser um *broker* de código aberto (EMQX, 2023), tornando-o uma escolha adequada para um projeto não tão complexo no quesito de comunicação. Neste caso, trabalhou-se com a versão 3.1.1 do MQTT, versão esta, que apesar de mais antiga ainda é amplamente utilizada nos projetos de automação.

5 RESULTADOS

Para o projeto do estudo de caso do monitoramento de vagas do eletroposto, foram elaboradas 3 atividades principais: O desenvolvimento da aplicação de Visão Computacional, a comunicação entre os sistemas a partir do protocolo MQTT e o *dashboard* de monitoramento, assim como a integração completa entre esses 3 eixos tornando a aplicação funcional.

5.1 Obstáculos e Soluções

Nesta etapa, são mostrados os obstáculos que surgiram ao refinar o código e suas respectivas soluções.

5.1.1 Sobreposição de Veículos

Ao trabalhar com apenas uma das câmeras disponíveis, observou-se o primeiro obstáculo: Um veículo maior cobria outro veículo menor. Este fato atrapalha bastante a detecção do YOLO, como é possível observar na Figura 27, que ilustra os dois carros sendo reconhecidos em um primeiro momento e depois apenas a detecção carro maior.

Figura 27 – Sobreposição na detecção.

Fonte: Autor (2025).

Para solucionar isso foi necessário trabalhar com as duas câmeras em simultâneo, a partir da gravação do aplicativo ISIC LITE, de forma que ao menos uma das câmeras iria visualizar o carro mais claramente, facilitando a detecção do YOLO. Essa estratégia adiciona ao código final a lógica “ou”. Onde após a segmentação das vagas, o status da disponibilidade da vaga seria alterado para nível lógico alto caso um carro fosse detectado naquela vaga, por qualquer uma das câmeras.

Este processo foi viável porque foi realizado capturando a tela do monitor que contém a gravação do estacionamento, mostrando as 2 telas. Entretanto, caso fosse utilizado o protocolo RSTP diretamente, como previsto inicialmente, não seria tão simples, já que ele apenas permite transmitir apenas uma câmera por vez. Assim, seria necessário pensar em outra estratégia, como por exemplo, executar duas detecções ou dois códigos em paralelo.

A partir de então, começou-se a trabalhar com as duas câmeras, associada a lógica “ou”, combinação esta que aumentou a eficiência do sistema, principalmente para os modelos *small* e nano.

5.1.2 Múltiplas Detecções

Ao corrigir a sobreposição dos veículos, surgiu uma nova problemática: o código realizava inúmeras detecções dos veículos próximos ao eletroposto por causa da grande frota de carros estacionados na loja onde está localizado o eletroposto. Isso tornava o processamento bem mais lento, principalmente com os modelos mais pesados que já são mais lentos por natureza. A grande quantidade de inferências é mostrada claramente na Figura 28, em que o sistema reconhece os carros que estão estacionados na frente e do lado do eletroposto.

Figura 28 – Múltiplas detecções.

Fonte: Autor (2025).

Para quantizar o quanto isso de fato interfere no tempo do sistema, foram realizadas duas interações com imagens diferentes, com a finalidade de comparar as informações fornecidas pelo código a cada interação. Na primeira interação, ajustou-se o código para monitorar apenas uma região especifica da tela, onde tinha poucos carros. O segundo caso foi realizado considerando toda a imagem disponível. Assim, em cada situação o código retorna 3 parâmetros: tempo de pré-processamento, tempo de inferência e tempo de pós processamento. Na tabela 2 abaixo, pode-se observar que o segundo caso teve uma inferência de 111ms a mais que o primeiro caso, mesmo tendo um tempo de pré-processamento menor, e teve 3ms a mais no tempo de pós processamento. Como o código final ainda se comunicaria e enviaria mensagens via MQTT, o sistema ficaria ainda mais lento. Portanto, foi necessário inserir na imagem uma região de interesse (ROI) para a inferência ser realizada apenas nos veículos que estão nas vagas.

Tabela 2 - Informações das detecções.

Interação	Objetos detectados	Tempo pré-processamento	Tempo de Inferência	Tempo pós-processamento
1	7 carros	3,8 ms	815,6 ms	2,0 ms
2	17 carros, 1 moto e 1 caminhão	3,0 ms	926,6 ms	5,0 ms

Fonte: Autor (2025).

Entretanto, ao se realizar alguns testes para encontrar uma região de interesse adequada foi verificado que ela não pode ser muito restritiva. Isso ocorre porque o modelo YOLO leva em consideração o contexto ao redor do objeto para realizar as detecções. Assim, ao limitar muito essa região a efetividade do sistema abaixava. Então, pelo método de tentativa e erro, encontrou-se uma região intermediária que reduz a detecção de carros estacionados fora das vagas, sem ser muito restritiva ao ponto de comprometer a performance do sistema.

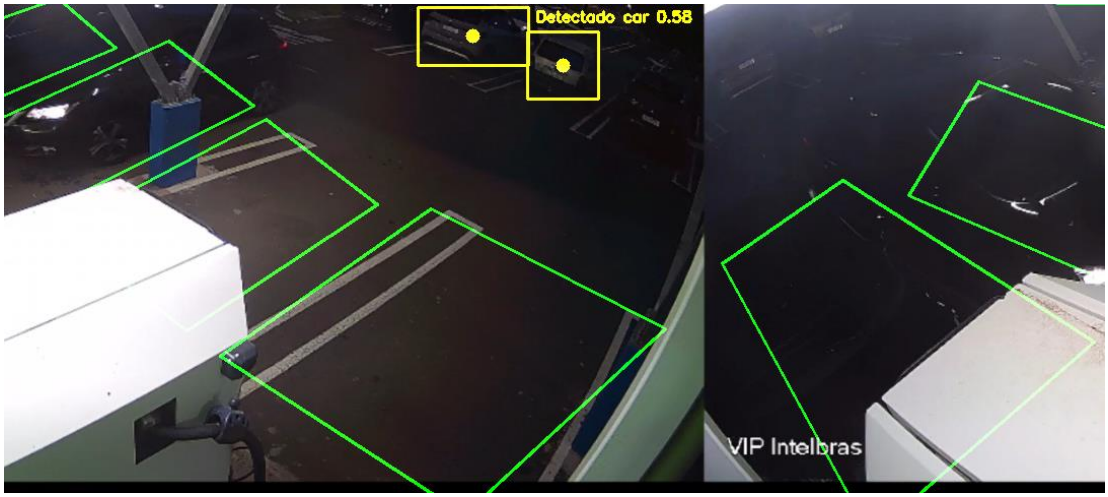
Além disso, também foi aplicado um filtro de classes, para que após a detecção dos objetos pelo YOLO seja passado para o restante do código apenas as classes de carros. Desta forma, consegue-se diminuir ainda mais o tempo total de execução do programa.

5.2 Detecções Durante a Noite

Outra situação em que houve baixa precisão na detecção foi no período da noite, principalmente com os carros que possuíam a cor preta. A Figura 29, mostra uma falha na detecção dos carros nas duas câmeras, utilizando o modelo *medium* do YOLO. Isto revela a problemática desse tipo de situação, agravada pela qualidade de captura das câmeras, que as vezes piora devido à instabilidade da conexão com a internet.

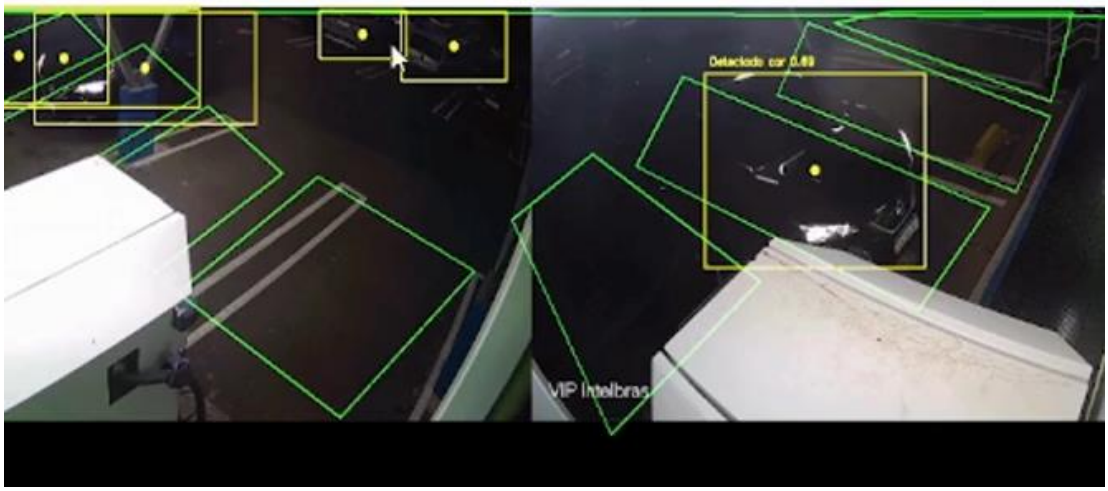
Esse foi o pior caso encontrado durante os testes, e também por isso que o filtro de tempo aplicado ao Node-Red é tão importante, pois mesmo que o sistema pare de detectar e volte a detectar no próximo *frame*, o tempo de permanência não é zerado, evitando erros no tempo de estacionamento. Assim, para esse caso foi preciso alterar o tamanho do modelo de *medium* para *large*. Dessa forma, o modelo conseguiu realizar as detecções com sucesso, como pode-se observar na Figura 30.

Figura 29 – Falha na detecção durante à noite, utilizando o tamanho *medium* do YOLOv8.



Fonte: Autor (2025).

Figura 30 – Sucesso na detecção durante à noite, utilizando o tamanho *large* do YOLOv8.



Fonte: Autor (2025).

5.3 Código Final em *Python*

Figura 31 – Pseudocódigo final.

```
#1. Inicializa o ambiente:
Importa as bibliotecas paho.mqtt.client,mss, numpy, opencv, os e ultralytics.

#2. Configuração dos parâmetros do MQTT:
Configuração do broker.
Definição do tópico.
Definição das funções de callback. #Funções para conexão e desconexão
#3. Outras configurações
Define as posições dos vértices da área de interesse.
Define as posições dos vértices da área de cada vaga.
Inicia o vetor das vagas e o vetor auxiliar.
Monitora a imagem usando a biblioteca *mss* com as dimensões do monitor auxiliar.
Configura o modelo YOLO carregando os pesos e parâmetros necessários.
Inicia a quantidade de frames como 0
#4. Entrar no loop de captura:
While not interrupted:

      Começa a captura em tempo real.
      #Formata o frame para ser compatível com a biblioteca OpenCV
      Converte o objeto retornado pela biblioteca mss em um array NumPy.
      Converte o frame de BGRA para BGR.
      Cria uma máscara preta do tamanho do frame.
      Preenche a máscara com a área de interesse.
      Aplica a máscara ao frame.
      Executa o modelo YOLO no frame fornecido.
      Desenha as áreas das vagas no frame.
      Se vagas for diferente de vagas auxiliar:
            Publica vagas auxiliar no tópico.
            Vagas será igual a vagas auxiliar.
      Zera todos os elementos do vetor vagas auxiliar

      #Processa a imagem com o modelo YOLO:
      For detections in detections:
            Se a classe da detecção for carro:
                  Coleta as informações da detecção: nome, confiança e as
posições da bounding box
                  Calcula o centro da bounding box.
                  Desenha a bounding box, o centro e a descrição.
                  Verifica se o centro está dentro da área de alguma vaga
      End for
 #5. Exibir os resultados:
 Desenhar a área de interesse no frame completo.
 Combina o frame mascarado com o frame original.
 Mostra os frames juntos.
```

```
 Salva os frames em imagens jpg, mantendo somente 2 imagens na pasta
 Atualiza o número do frame
#6. Finaliza
If usuário pressionou a tecla de saída “q”:
      Encerra o While

End While
Libera captura de tela.
Libera janelas abertas.
#Fim do programa
```

Fonte: Autor (2025).

Na Figura 31, é apresentado o pseudocódigo final depois de todas as modificações realizadas perante os desafios encontrados. O código completo será anexado no apêndice deste trabalho.

A primeira otimização adicionada ao código final frente ao código inicial apresentado no capítulo 4, foi a inserção do protocolo MQTT para enviar o vetor com os status da vaga ao *dashboard*, juntamente com algumas medidas de reconexão, prevenindo problemas de rede. Depois, foi feita a aplicação de uma máscara no frame para restringir a área de detecção e mitigar o problema das múltiplas detecções. A localização desta área foi projetada de maneira empírica após os testes.

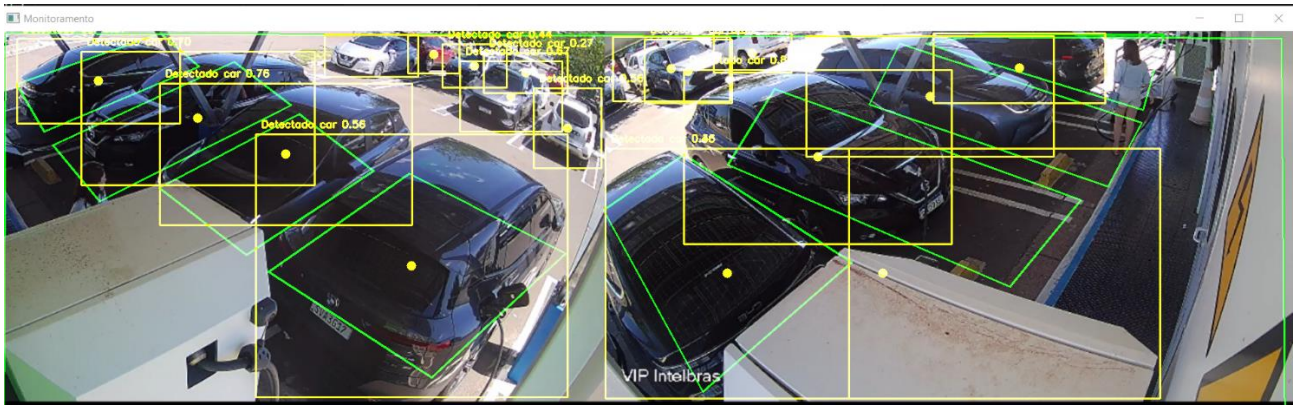
Não foi possível eliminar toda área indesejada, pois conforme explicado anteriormente o modelo do YOLO precisa do contexto da região para ter uma melhor precisão. Assim, também foi implementado um filtro de classes para otimizar o processamento do modelo, evitando que ele gaste tempo detectando outros objetos. Por fim, definiu-se empiricamente os vértices das áreas das vagas em formato de polígono para a verificar se o centro da *bounding box* do veículo está dentro da área delimitada, utilizando a lógica “ou”, para que independente da câmera que detectasse o código, seja enviado o status da vaga. Tais áreas são ilustradas na Figura 32, representadas pela cor verde e as *bounding box* representadas pela cor amarela.

Uma prática comum ao procurar os melhores parâmetros dos modelos de visão computacional, é alterar os valores da confiança para não obter falsos positivos (OLIVEIRA, 2024). Entretanto, não foi adicionado nenhum tipo de filtro para o valor da confiança, ou seja, qualquer valor de confiança em que o modelo detectasse um carro

seria considerado. Isso foi feito pois, durante os testes em situações específicas, quando era à noite, por exemplo, verificou-se que os modelos *smalls* e nano não possuíam uma precisão satisfatória quando tentavam detectar veículos pretos, como já mencionado anteriormente. Assim, não seria necessário colocar um valor mínimo de confiança, porque já foi colocado um filtro de classe.

A Figura 32 exemplifica uma situação em que é possível ver todos carros sendo detectados e os centros de suas respectivas *bounding box* se encontram perfeitamente dentro das áreas correspondentes às vagas.

Figura 32 – Visualização das detecções do modelo.



Fonte: Autor (2025).

O item 5 do pseudocódigo é a etapa em que o código gera e exibe uma tela simples com as detecções dos frames e as vagas destacadas. Ainda nessa parte, foi implementada uma lógica para salvar os *frames* no formato “jpg” em uma pasta local, onde o número de cada imagem é salvo dinamicamente, sempre sobrescrevendo os arquivos para deixar apenas duas imagens na pasta. Isso foi feito para que o fluxo do Node-Red da página de visualização conseguisse ler e exibir as imagens na tela, como se fosse o vídeo em tempo real. Caso houvesse acesso total as câmeras do eletroposto, este processo seria necessário pois, poderia conectar diretamente as câmeras no Node-Red via RSTP.

Ao final do código está presente uma das maneiras do usuário interromper o processo apertando a tecla “q”, em cima da tela gerada. Ao fazer isso, as imagens geradas na pasta são excluídas e as janelas são fechadas. O código utilizou os

modelos pré-treinados do YOLO dos tamanhos nano*, small*, *medium* e *large* para os testes, verificou-se que para a maioria das situações os modelos *small* e nano são suficientes para essa aplicação. Somente no caso do turno da noite com veículos de cor preta que os modelos de tamanho *medium* e *large* possuem uma melhor precisão.

Assim, com essas modificações foi possível finalizar o *script*, minimizando os problemas de detecção causados principalmente pelo posicionamento das câmeras.

5.4 Testes

Para a validação do código de *python* integrado com todo o sistema, foram realizados diversos testes em situações diferentes para entender como o sistema se comportava e fazer os ajustes necessários.

Com esse intuito, foram gravados diversos vídeos pelo ISIC LITE, através do OBS Studio para análise do modelo, variando a ocupação das vagas, horários e tipos de veículos, entre outros cenários. A fim de garantir a reprodutibilidade dos resultados, as gravações eram feitas sempre na mesma configuração de tela do monitor do computador usado para este trabalho, por isso todas as áreas das vagas e a máscara do frame foram dimensionadas manualmente para esta máquina específica.

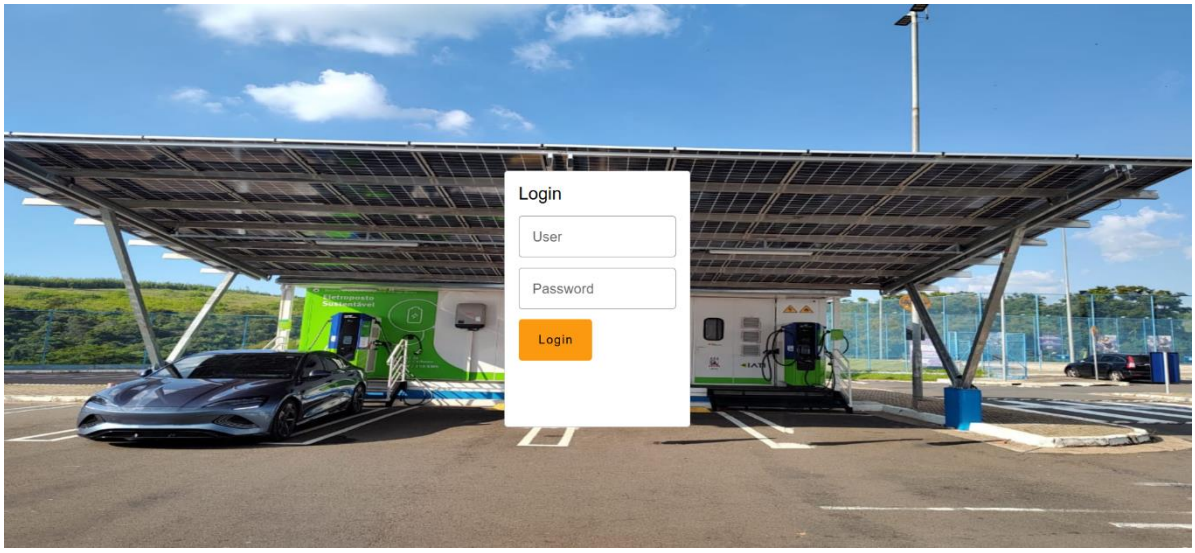
5.5 Aplicação Completa Final

Após a integração dos sistemas, resultou-se em uma aplicação funcional em que foram testadas diversas gravações em diferentes situações e conseguiu-se detectar os carros pela aplicação em *python,* que por sua vez enviou com sucesso para o *broker* MQTT, de onde o *dashboard* de Node-Red captou os dados e apresentou de forma satisfatória, como era esperado. Uma demonstração da aplicação pode ser visualizada a partir de um vídeo no Youtube, cujo *link* está presente no Apêndice A deste trabalho.

Em relação as telas do *dashboard,* não houve mudanças em relação ao que foi apresentado previamente, apenas será ilustrado na Figura 33 a nova tela de login e a tela de monitoramento na Figura 34, exemplificando uma situação em que 4 carros

foram detectados. A Figura 33 mostra uma pequena alteração na tela de *login,* devido a alguns *feedbacks,* em relação a centralização do *card* de entrada.

Figura 33 – Tela de *Login* Modificada.



Fonte: Autor (2025).

Figura 34 – Tela de Monitoramento com 4 carros carregando;

Fonte: Autor (2025).

5.5.1 Erros

Mesmo após as modificações realizadas e mitigação dos erros apresentados, o sistema ainda apresentou alguns problemas de detecções, principalmente quando a internet do eletroposto apresentava instabilidades, reduzindo a resolução das imagens *e* resultando numa pixelização das mesmas. Na Figura 35, por exemplo, pode-se observar que na câmera da esquerda o primeiro carro não foi detectado, mas foi detectado pela câmera da direita, que devido a lógica “ou”, os dados seriam enviados corretamente. Já na câmera da direita na última vaga, obtém-se duas detecções para o mesmo carro, o que não necessariamente atrapalha de forma significativa o sistema, mas se por acaso apenas houver um carro estacionado e duas detecções, em que uma delas possui seu centro na outra vaga, isto pode causar um problema de um falso positivo, que pode ser mitigado com modelos de tamanhos maiores, a partir do *médium.*

Figura 35 – Erros.

Fonte: Autor (2025).

6 CONCLUSÕES E PROPOSTAS DE CONTINUIDADE

Este trabalho apresentou uma aplicação funcional do monitoramento de estacionamentos a partir de visão computacional através do modelo do YOLOV8 e utilizou como caso de estudo o eletroposto da pesquisa P&D, integrando o sistema de detecção com o supervisório de maneira eficiente, a partir das gravações feitas pelas câmeras da loja com o uso do protocolo MQTT.

Assim, conclui-se que a abordagem por visão computacional traz diversos benefícios para o monitoramento de eletropostos. O sistema desenvolvido demonstrou eficiência na detecção de veículos elétricos em diferentes situações, incluindo períodos noturnos e cenários com sobreposição de veículos. Adicionalmente, os dados coletados ao longo do monitoramento permitem uma previsão da relação entre o tempo de carregamento dos veículos e o tempo total de permanência estacionados. A comparação entre essas métricas pode revelar possíveis gargalos na utilização dos eletropostos, como veículos que permanecem ocupando as vagas por um período significativamente maior do que o necessário para a recarga, impactando a eficiência da estação. Os gráficos gerados pelo sistema possibilitam a identificação de padrões dos usuários ao longo do tempo, permitindo aos gestores prever horários de maior demanda, otimizar a ocupação das vagas e até mesmo avaliar a necessidade de expansão da infraestrutura. A análise estatística dessas informações pode ser utilizada para a tomada de decisões estratégicas, como a implementação de sistemas de tarifação diferenciada ou incentivos para reduzir tempos ociosos. Além disso, a substituição de vários sensores de presença por um sistema de câmeras e servidores reduziria o custo operacional.

Outro ponto é a facilidade de implementação, já que é comum as estações de recargas terem câmeras de segurança. Com a ressalva de que o sistema possa ser previamente projetado para que a instalação das câmeras seja feita em posição estratégicas, no geral em um local superior que forneça boa visão das vagas. E caso seja necessário, é válido adicionar fontes de iluminação adequadas para minimizar imprecisões.

Entretanto, a precisão da aplicação varia, entre outros fatores, com o tamanho do modelo, que está relacionado com a velocidade da detecção e com o

processamento do *hardware* que está instalado. Desta forma, é importante achar um equilíbrio entre esses fatores para a aplicação ser eficaz.

Como discutido anteriormente, o sistema não foi implementando na prática devido à falta de recursos para colocar um servidor e um monitor no local operando 24h por dia ou continuamente. Além de não ter sido possível obter o acesso completo as câmeras, por causa de algumas restrições da pesquisa. Apesar disso, o trabalho atingiu seus objetivos principais: a elaboração de um programa de visão computacional voltado para eletropostos, a construção de um sistema supervisório para transformar os dados extraídos do programa de visão computacional em informações e a integração desses dois sistemas em uma aplicação funcional.

Caso tivesse sido implementado, as mudanças principais seriam a configuração do novo servidor e um redimensionamento da nova de tela do monitor. Vale ressaltar que este último processo poderia ser facilmente automatizado, caso as câmeras fossem instaladas numa posição superior às vagas de forma padronizada. Com essa condição, os ajustes necessários para instalar o sistema em outro eletroposto seria bem mais rápido e fácil.

A continuidade deste projeto pode, inicialmente, seguir para uma implementação nos próprios eletropostos da UFPE, visto que os mesmos já foram validados e estão funcionais desde a entrega do presente trabalho. Para isso, seria fundamental um planejamento prévio dos locais de instalação das câmeras, conforme as nuances levantadas por este projeto. Também é viável e enriqueceria o projeto a adição da integração com internet, para que seja possível acessar o *dashboard* de qualquer lugar do mundo.

Outra linha de pesquisa seria um estudo mais voltado para a coleta e análise dos dados obtidos em um determinado tempo de monitoramento, o que possibilitaria a identificação de padrões, como a diferença entre o tempo de permanência frente ao tempo carregando, horários com uma frequência alta de carregamento, quais carregadores são mais acionados, entre outros. Além de se adicionar outras variáveis para análise.

Com as devidas melhorias e o adicional de novas funcionalidades, o projeto apresenta um grande potencial para ser comercializável, podendo se transformar em uma solução robusta para ajudar as empresas que pretendem ter um gerenciamento

mais efetivo de suas estações de recarga, atendendo à crescente demanda por infraestrutura inteligente na mobilidade elétrica.

O projeto P&D mencionado neste trabalho foi classificado em 1º lugar no Prêmio OSE no Circuito Nacional do Setor Elétrico (CINASE) 2024 (IATI, 2024).

REFERÊNCIAS

ABVE. **ABVE**, 2024. Disponível em: <https://abve.org.br/bi-geral/>. Acesso em: 17 Novembro 2024.

ALBERY , Lucas P. et al. DESAFIOS E OBJETIVOS NA IMPLEMENTAÇÃO DE MODELOS AUTOMOBILISTICOS ELETRICOS NO BRASIL. **Contemporânea - Contemporary Journal**, Caruaru, 29 Janeiro 2024. 3-6.

ALBUQUERQUE, Márcio P. D. Visão por Computador (conceito básicos). **MESONPI**, 2000. Disponível em: <https://mesonpiold.cbpf.br/cat/pdsi/visao/index.html>. Acesso em: 18 Novembro 2024.

ALVES, Gabriel. Detecção de Objetos com YOLO – Uma abordagem moderna. **IA Expert Academy**, 2020. Disponível em: <https://iaexpert.academy/2020/10/13/deteccao-de-objetos-com-yolo-uma-abordagem-moderna/>. Acesso em: 15 Novembro 2024.

ANNASWAMY, A. M. ; JOHANSSON, K. H.; PAPPAS, G. Control for Societal-Scale Challenges. **IEEE Control Systems Magazine**, v. 44, p. 30-32, Julho 2024.

ANTONELLO, Ricardo. **Introdução a Visão Computacional com Python e OpenCV**. Universidade Federal de Santa Catarina. [S.l.], p. 11-14. 2018.

BARZNJI, Hemn M. K. **Transfer Learning as New Field in Machine Learning**. Sulaimani University. [S.l.], p. 1-5. 2020.

BOCHKOVSKIY, Alexey; WANG, Chien-Yao; MARK, Hong-Yuan L. YOLOv4: Optimal Speed and Accuracy of Object Detection. **arXiv preprint arXiv:** 2004.10934, 2020. Disponível em: <https://arxiv.org/abs/2004.10934>. Acesso em: 16 Novembro 2024.

CAVENAGLI, Willian M.; MENDES, Caroline M. **Detecção de Vagas em Estacionamento Usando Visão Computacional e Redes Neurais Convolucionais**. COMPUTER ON THE BEACH. Balneário Camboriú: [s.n.]. 2020. p. 65.

CESAR, Julio. Brasil teve crescimento anual de 179% na infraestrutura de carregamento. **INSIDE EVS UOL**, 2024. Disponível em: <https://insideevs.uol.com.br/news/736410/brasil-infraestrutura-recarga-crescimento/>. Acesso em: 10 Novembro 2024.

CHANDRA, Prakash; RATH, Sovit. **Learn OpenCV**, 2022. Disponível em: <https://learnopencv.com/fcos-anchor-free-object-detection-explained/>. Acesso em: 16 Novembro 2024.

CHEN, Tianjin et al. A Review on Electric Vehicle Charging Infrastructure Development in the UK. **JOURNAL OF MODERN POWER SYSTEMSAND CLEAN ENERGY**, v. 8, n. 2, p. 193 - 196, Março 2020.

DANLEY, Douglas R. Defining a Microgrid Using IEEE 2030.7. **Business & Technology** , Novembro 2019.

DERRENGER, Paula. ultralytics, 2023. Disponível em: <https://docs.ultralytics.com/pt>. Acesso em: 11 Novembro 2024.

DIAS, Robinson A.; DEL VECHIO, Gustavo H. Transformações tecnológicas ao longo da história: impacto e perspectivas futuras. **Revista Interface Tecnológica**, v. 14, p. p. 32-43, junho 2019.

DIWAN, Tausif; ANIRUDH, G.; TEMBHURNE, Jitendra. Object detection using YOLO: challenges, architectural. **Multimedia Tools and Applications**, Agosto 2022.

EMQX. Mosquitto MQTT Broker: Pros/Cons, Tutorial, and Alternative. **emqx**, 2023. Disponível em: <https://www.emqx.com/en/blog/mosquitto-mqtt-broker-pros-cons-tutorial-and-modern-alternatives>. Acesso em: 22 Janeiro 2025.

FARIAS, Thalison B. M.; SILVA, Paulo H. L. **SMART TASK CONTROL: UM SISTEMA DE AGENDAMENTO E EXECUÇÃO DE TAREFAS.** Universidade Federal Rural do Semiárido - UFERSA. Mossoró, p. 5-7. 2024.

FLECK, Leandro et al. REDES NEURAIS ARTIFICIAIS: PRINCÍPIOS BÁSICOS. **Revista Eletrônica Científica Inovação e Tecnologia**, v. 1, n. 13, p. 47-57, 2016.

FORSYTH, David A.; PONCE, Jean. COMPUTER VISION A MODERN APPROACH. In: PONCE, Jean; FORSYTH, David.A **COMPUTER VISION A MODERN APPROACH**. 2. ed. [S.l.]: Pearson, 2012.

FRANTZ, ALBERTO R. **DESAFIOS ATUAIS DO MERCADO DE CARROS ELÉTRICOS**. UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL. Porto Alegre. 2023.

HAYKIN, Simon. Redes Neurais: Princípios e Prática. 2ª. ed. [S.l.]: Prentice Hall Inc, 1999. p. 27-29.

HUYNH, Nghi. Convolutional Neural Networks for Image Recognition. **Medium**, 2022. Disponível em: <https://medium.com/@nghihuynh_37300/convolutional-neural-networks-for-image-recognition-7148a19f981f>. Acesso em: 17 Novembro 2024.

IATI. IATI é Destaque no CINASE 2024: Inovação e Sustentabilidade Premiada em Projetos de PDI. **IATI**, 2024. Disponível em: <https://www.iati.org.br/pt_br/noticias/iati-e-destaque-no-cinase-2024-inovacao-e-sustentabilidade-premiada-em-projetos-de-pdi/>. Acesso em: 11 Março 2024.

IPCC. **MUDANÇA DO CLIMA 2023: Relatório Síntese**. Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change. Geneva, p. 1-34. 2023.

JIANG, Peiyuan et al. **A review of YOLO algorithm developments**. THE 8TH INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY AND QUANTITATIVE MANAGEMENT (ITQM 2020 & 2021). [S.l.]: Elsevier B.V. 2021. p. 1066-1073.

KAMIZI, Siti A. **UI/UX of Human-Machine Interface for Industrial Application:** Review and Preleminary Design. FCSIT UNIMAS FYP Symposium. Kota Samarahan: Universiti Malaysia Sarawak (UNIMAS). 2021. p. 28-30.

KRUG, Steve. Don't Make Me Think. In: KRUG, Steve **Don't Make Me Think**. 3ª. ed. [S.l.]: New Riders, 2014. Cap. 10, p. 152-153.

LEAL, DANILO M. **DETECÇÃO E RASTREAMENTO DE OBJETOS EM VÍDEO VIA REDE NEURAL CONVOLUCIONAL (CNN): YOLO E DEEPSORT APLICADOS PARA CONTAR VEÍCULOS E ESTIMAR SUAS VELOCIDADES MÉDIAS A PARTIR DE REFERENCIAL FIXO**. INSTITUTO FEDERAL DO ESPÍRITO SANTO. Serra, p. 68-72. 2023.

LEMES, Nelson H. T. Neurônio de McCulloch-Pitts. **Nelson Henrique Teixeira Lemes**, 2020. Disponível em: <https://pessoas.unifal-mg.edu.br/nelsonlemes/neuronio-de-mcculloch-pitts/>. Acesso em: 19 Novembro 2024.

LIN, Tsung-Yi et al. **Microsoft COCO:** Common Objects in Context. Computer Vision – ECCV 2014: 13th European Conference. Zurich: [s.n.]. 2014. p. 740-755.

LUCAS, Jean D. S. C. **Reconhecimento de Imagens: uso do Método yolo no reconhecimento de placas de Trânsito**. Universidade Federal do Pampa. Bagé, p. 19-20. 2022.

MAIA JR, Geraldo L et al. EV Smart-Charging Strategy for Power Management in Distribution Grid with High Penetration of Distributed Generation. **Energies**, v. 17, n. 21, 2024. ISSN 5394.

MENESES, Paulo Roberto; ALMEIDA, Tati de (org.). Introdução ao processamento de imagens de sensoriamento remoto. Brasília: UnB, 2012

MENEZES, Alexandre M. D. **APLICAÇÕES DE DESIGN EM SISTEMAS DE AUTOMAÇÃO: ANÁLISE DE BOAS PRÁTICAS E RECURSOS PARA INTERFACES VOLTADAS À EXPERIÊNCIA DO USUÁRIO**. Universidade Federal de Pernambuco. Recife, p. 29-35. 2024.

MOMA, Gabriel. 10 heurísticas de Nielsen para o design de interface. **Medium**, 2017. Disponível em: <https://brasil.uxdesign.cc/10-heur%C3%ADsticas-de-nielsen-para-o-design-de-interface-58d782821840>. Acesso em: 12 dez. 2024.

MONTALBO, Francis J. P. A Computer-Aided Diagnosis of Brain Tumors Using a Fine-Tuned YOLO-based Model with Transfer Learning. **KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS**, Filipinas, v. 14, n. 12, Dezembro 2020.

NODE-RED. Node-RED: Low-code programming for event-driven applications. **nodered**, 2025. Disponível em: <https://nodered.org/>. Acesso em: 28 Janeiro 2025.

NOVAIS, Guilherme. FIFA 23: veja evolução gráfica de Neymar nos jogos. **Ge Globo**, 2022. Disponível em: <https://ge.globo.com/esports/fifa/noticia/2022/09/28/fifa-23-veja-evolucao-grafica-de-neymar-nos-jogos-da-franquia.ghtml>. Acesso em: 10 Novembro 2024.

O'SHEA, Keiron ; NASH, Ryan. An Introduction to Convolutional Neural Networks, 2 Dezembro 2015. 1-3.

OGATA, Katsuhiko. Engenharia de controle moderno. 5. ed. São Paulo: Pearson Prentice Hall, 2010.

OLIVEIRA, Bruno V. N. D.; MELO, Filipe T. D. FUNDAMENTOS DA VISÃO COMPUTACIONAL: ARCABOUÇO TEÓRICO DO RECONHECIMENTO ARTIFICIAL DE IMAGENS E VÍDEOS. **Revista Humanidades & Inovação**, 17 Junho 2024. 321-323.

OLIVEIRA, MATHEUS D. S. **DETECÇÃO DE EPI'S COM FERRAMENTA DE VISÃO COMPUTACIONAL**. UNIVERSIDADE FEDERAL DO PARÁ CAMPUS UNIVESITÁRIO DE TUCURUÍ. Tucuruí, p. 40-50. 2024.

ORG, MQTT. MQTT: The Standard for IoT Messaging. **MQTT**, 2024. Disponível em: <https://mqtt.org/>. Acesso em: 20 Novembro 2024.

PARODI, Alessandro. Global EV sales up 21% in July as China records biggest jump of 2024, Rho Motion says. **Reuters**, 2024. Disponível em: <https://www.reuters.com/business/autos-transportation/global-ev-sales-up-21-july-china-records-biggest-jump-2024-rho-motion-says-2024-08-12/>. Acesso em: 10 Novembro 2024.

PIEMONTEZ, Rafael A. **Visão Computacional**, 2022. Disponível em: <https://visaocomputacional.com.br/yolo-para-deteccao-de-objetos-visao-geral/>. Acesso em: 16 Novembro 2024.

REDMON, Joseph et al. **You Only Look Once:** Unified, Real-Time Object Detection. Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.]: [s.n.]. 2016.

RODRIGUEZ, Henrique. Como carregar um carro elétrico? Veja tipos de tomadas e carregadores. **Quatro Rodas**, 2024. Disponível em: <https://quatrorodas.abril.com.br/carros-eletricos/como-carregar-um-carro-eletrico>. Acesso em: 17 Novembro 2024.

RUSSEL, Stuart; NORVIG, Peter. Inteligência Artificial. In: RUSSEL, Stuart; NORVIG, Peter **Inteligência Artificial**. 3ª. ed. Rio de Janeiro: Elsevier Editora Ltda, 2013. Cap. 18, p. 842-844.

SANTOS, CAIO C. L. D. **CONTROLE E SUPERVISÃO DE UMA ESTAÇÃO DE RECARGA DE VES: APLICAÇÃO DO PROTOCOLO OCPP 1.6**. Universidade Federal de Pernambuco. Recife, p. 34-42. 2024.

SANTOS, Elton F. D. Detecção de Bordas. **Grupo OpenCV BR**, 2020. Disponível em: <https://grupo-opencv-br.github.io/posts/cap1/>. Acesso em: 10 Novembro 2024.

SAÚDE, André V. Computação Gráfica e Processamento de Imagens. Londrina: Editora e Distribuidora Educacional S.A., 2019. Cap. 1.

GRAND VIEW RESEARCH. Computer vision market size, share & trends analysis report by component (hardware, software), by product (smart camera-based, PC-based), by application, by vertical, by region, and segment forecasts, 2025 - 2030. San Francisco, 2024. Disponível em: <https://www.grandviewresearch.com/industry-analysis/computer-vision-market>. Acesso em: 11 Março 2025.

SILVA, Rodolfo A. V. D. **Inteligência Artificial aplicada à Infraestrutura de Carregamentos para Veículos Elétricos**. Universidade do Minho. [S.l.], p. 1-10. 2023.

SMART. Gestão de estações de carregamento. **smartparkingsystems**, 2025. Disponível em: <https://smartparkingsystems.com/pt-br/gestao-de-estacoes-de-carregamento/>. Acesso em: 1 Março 2025.

SONI, Dipa; MAKWANA, Ashwin. **A SURVEY ON MQTT:** A PROTOCOL OF INTERNET OF THINGS(IOT). INTERNATCONFERENCE ON TELECOMMUNICATION, POWER ANALYSIS AND COMPUTING TECHNIQUES (ICTPACT - 2017). Chennai: [s.n.]. 2017.

SUNDARESAN , Athulya G. et al. Comparative Analysis of YOLOv8 and YOLOv10 in Vehicle Detection: Performance Metrics and Model Efficacy. **Vehicles**, v. 6, p. 1364-1382, 2024.

SZELISKI, Richard. Computer Vision: Algorithms and Applications. [S.l.]: Springer, 2010. Cap. 1, p. 3-10.

THARWAT, Ahmed. Siemens TIA Portal PLC Dashboard using Node-RED: A Step-by-Step Tutorial. **solisplc**, 2023. Disponível em: <https://www.solisplc.com/tutorials/siemens-tia-portal-plc-dashboard-using-node-red-a-step-by-step-tutorial>. Acesso em: 28 jan. 2025.

TIAN, Yunong et al. Apple detection during different growth stages in orchards. **Computers and Electronics in Agriculture**, v. 157, p. 417-426, 2019.

TISSOT, Hegler C.; CAMARGO, Luiz C.; POZO, Aurora T. **Treinamento de redes neurais feedforward:** comparativo dos algoritmos backpropagation e differential evolution. Brazilian Conference on Intelligent Systems. [S.l.]: [s.n.]. 2012.

TOLEDO , Marcello D. S.; DE SOUZA, David Luiz C. SISTEMAS GERENCIAIS DE BANCO DE DADOS: UM ESTUDO DO MYSQL. **REVELA**, n. 23, p. 380-387, 2018.

TORRES, Andrei BB; ROCHA, Atslands R.; DE SOUZA, José Neuman. **Análise de desempenho de brokers mqtt em sistema de baixo custo**. Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance). [S.l.]: SBC. 2016. p. 2804-2815.

TRANSPOQUIP. Estacionamentos inteligentes: as principais tendências que estão transformando a mobilidade urbana. **TranspoQuip**, 2025. Disponível em: <https://transpoquip.com.br/estacionamentos-inteligentes-as-principais-tendencias-que-estao-transformando-a-mobilidade-urbana/>. Acesso em: 1 Março 2025.

V. ASSUMPÇÃO, Hudson Guilherme et al. **Estacionamento inteligente:** uma comparação entre sensores ultrassônicos e visão computaciona. XLII SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS. Belém: [s.n.]. 2024.

VACCANI, June T. Estratégias e impactos da descarbonização para o desenvolvimento sustentável. **Instituto de Energia e Ambiente, Universidade de São Paulo**, 2024. Disponível em: <https://www.iee.usp.br/noticia/estrategias-e-impactos-da-descarbonizacao-para-o-desenvolvimento-sustentavel/>. Acesso em: 5 Fevereiro 2025.

VAZ, David D. S. **Sistema de visão inteligente de baixo custo para parque de estacionamento**. Dissertação de Mestrado - Universidade do Algarve. [S.l.], p. 16-17. 2015.

WASZKOWSKI, Robert. Low-code platform for automating business processes in manufacturing. **IFAC-PapersOnLine**, 2019. 376-381.

ZHANG, Chao; WOODLAND, Philip C. **Parameterised sigmoid and ReLU hidden activation functions for DNN acoustic modelling**. INTERSPEECH 2015. Dresden: International Speech Communication Association. 2015. p. 3224-3228.

APÊNDICES

APÊNDICE A – *LINK* DO VÍDEO DA APLICAÇÃO

Link para o vídeo da demonstração da aplicação no Youtube:

https://youtu.be/_6maawIKuW8

APÊNDICE B – Código Final do Python

```
from ultralytics import YOLO
import cv2
import mss
import numpy as np
import paho.mqtt.client as mqtt
import time
import os

# Configurações do broker MQTT
broker_address = "127.0.0.1"  # Endereço local do broker MQTT
broker_port = 1884  # Porta do broker MQTT
topic = "YOLOV8PN"  # Tópico para envio de dados

# Credenciais de autenticação
username = "placido"
password = "nilo"

# Callback para verificar a conexão ao broker MQTT
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("*Conectado ao broker MQTT com sucesso!")
    else:
```

```
        print(f"*Falha na conexão com o broker. Código de retorno: {rc}")

# Callback para tratar desconexão e tentativa de reconexão
def on_disconnect(client, userdata, rc):
    print("*Desconectado do broker MQTT. Tentando reconectar...")
    while True:
        try:
            client.reconnect()
            print("*Reconectado ao broker MQTT com sucesso!")
            break
        except:
            print("*Falha na reconexão. Tentando novamente em 5 segundos...")
            time.sleep(5)

# Criação do cliente MQTT
client = mqtt.Client()
client.username_pw_set(username, password)
client.on_connect = on_connect
client.on_disconnect = on_disconnect

# Tentativa de conexão ao broker MQTT
print("*Tentando conectar ao broker...")
client.connect(broker_address, broker_port, 120)
client.loop_start()

# Definição das áreas de interesse (ROI - Região de Interesse)
vertices = np.array([[7, 3], [1894, 10], [1899, 600], [4, 600]], np.int32)

# Definição das áreas de estacionamento
poligonais_vagas = [
    np.array([[236, 3], [293, 52], [64, 148], [23, 78]], np.int32),
    np.array([[345, 45], [429, 109], [146, 251], [77, 168]], np.int32),
    np.array([[441, 122], [551, 206], [363, 330], [234, 231]], np.int32),
    np.array([[604, 210], [835, 328], [637, 511], [396, 355]], np.int32),
```

```
    np.array([[891, 264], [1010, 182], [1258, 341], [1038, 532]], np.int32),
    np.array([[1144, 85], [1597, 249], [1492, 377], [1075, 197]], np.int32)
]
```

```
# Inicialização das variáveis de vagas
total_vagas = 6
vagas = [0] * total_vagas
vagas_aux = [0] * total_vagas

# Configuração para captura de tela
monitor = {"top": 100, "left": -2200, "width": 1920, "height": 1080}
sct = mss.mss()

# Carregar o modelo YOLO pré-treinado
model = YOLO("yolov8s.pt").to('cpu')

# Contador de frames
count = 0

# Loop principal para detecção
while True:
    screen = np.array(sct.grab(monitor))
    frame = cv2.cvtColor(screen, cv2.COLOR_BGRA2BGR)

    # Criar máscara preta e aplicar ROI
    mask = np.zeros_like(frame)
    cv2.fillPoly(mask, [vertices], (255, 255, 255))
    masked_frame = cv2.bitwise_and(frame, mask)

    # Executar modelo YOLO
    detections = model(masked_frame, verbose=False)[0]

    # Desenhar áreas de estacionamento
    for poli_vaga in poligonais_vagas:
```

```
    cv2.polylines(frame, [poli_vaga], isClosed=True, color=(0, 255, 0), thickness=2)

# Atualizar status das vagas e enviar para MQTT
if vagas != vagas_aux:
    client.publish(topic, str(vagas_aux))
    vagas = vagas_aux.copy()
vagas_aux = [0] * total_vagas

# Processar detecções
for box in detections.boxes:
    if int(box.cls.item()) == 2:  # Verifica se a detecção é um carro
        xmin, ymin, xmax, ymax = map(int, box.data.tolist()[0][:4])
        centro = ((xmin + xmax) // 2, (ymin + ymax) // 2)

        cv2.rectangle(masked_frame, (xmin, ymin), (xmax, ymax), (0, 255, 255), 2)
        cv2.circle(masked_frame, centro, 7, (0, 255, 255), -1)

        # Verifica se o carro está dentro de alguma vaga
        for i, poli_vaga in enumerate(poligonais_vagas):
            if cv2.pointPolygonTest(poli_vaga, centro, False) > 0:
                vagas_aux[i] = 1

# Exibir imagem
frame_with_detections = cv2.bitwise_or(frame, masked_frame)
cv2.imshow("Monitoramento", frame_with_detections)

# Salvar frames
caminho_base = 'c:/Users/Windows 10/Documents/TCC/testes/'
cv2.imwrite(f'{caminho_base}{count}.jpg', frame_with_detections)
old_file = f'{caminho_base}{count-2}.jpg'
if os.path.exists(old_file):
    os.remove(old_file)

count += 1
```

```
    print(count)

    # Encerrar loop ao pressionar 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cv2.destroyAllWindows()
```