INSTITUTO FEDERAL
Pernambuco

DESENVOLVIMENTO DE UM SISTEMA DE RECONHECIMENTO DE PLACAS VEICULARES PARA CONTROLE DE ACESSO EM AMBIENTES RESIDENCIAIS

Luiz Gabriel Velêz de Farias

lgvf@discente.ifpe.edu.br

Matheus Duarte de Souza Lins

mdsl2@discente.ifpe.edu.br

RESUMO

O controle de acesso em ambientes residenciais e comerciais tornou-se um pilar fundamental na gestão de segurança patrimonial moderna. Este trabalho apresenta o desenvolvimento de um sistema de Reconhecimento Automático de Placas (ALPR, do inglês *Automatic License Plate Recognition*), projetado para automatizar o fluxo de entrada e saída de veículos. A solução proposta integra a linguagem Python com a biblioteca OpenCV para o pré-processamento digital de imagens e o motor Tesseract OCR, baseado em redes neurais LSTM (*Long Short-Term Memory*), para a extração de caracteres. O sistema utiliza filtragem bilateral para redução de ruído preservando bordas e binarização adaptativa para segmentação, garantindo alta assertividade mesmo em condições de iluminação variáveis. Os dados são gerenciados por um banco de dados SQLite, oferecendo uma arquitetura *serverless* de baixo custo e alta eficiência. Os resultados obtidos demonstram a viabilidade técnica da substituição de controles manuais por sistemas de visão computacional em hardware de uso geral.

Palavras-chave: Controle de acesso; Visão Computacional; OCR; Tesseract; LSTM; Python.

ABSTRACT

Access control in residential and commercial buildings has become a fundamental pillar in modern security management. This paper presents the development of an Automatic *License Plate Recognition* (ALPR) system designed to automate vehicle entry and exit flows. The proposed solution integrates the Python language with the OpenCV library for digital image preprocessing and the Tesseract OCR engine, based on *Long Short-Term Memory* (LSTM) neural networks, for character extraction. The system utilizes bilateral filtering for noise reduction while preserving edges and adaptive thresholding for segmentation, ensuring high accuracy even under variable lighting conditions. Data is managed by an SQLite database, offering a low-cost, high-efficiency *serverless* architecture. The results

demonstrate the technical viability of replacing manual controls with computer vision systems on general- purpose hardware.

Keywords: Access control; Computer Vision; OCR; Tesseract; LSTM; Python.

1. INTRODUÇÃO

O avanço da urbanização e o aumento do fluxo de veículos em grandes centros trouxeram desafios significativos para a segurança e o controle de acesso em condomínios e edifícios comerciais, especialmente em contextos de alta densidade populacional e intensa mobilidade urbana (ONU-HABITAT, 2022). Assim como a Inteligência Artificial (IA) transformou a segurança em diversos setores, a automação de portarias tornou-se uma necessidade para garantir maior eficiência, rastreabilidade e confiabilidade na administração de entradas e saídas (Mckinsey; Company, 2021).

Relatórios de segurança pública e estudos sobre gestão condominial apontam que a portaria é o ponto mais vulnerável de um edifício, uma vez que concentra o fluxo de pessoas, veículos e mercadorias, exigindo procedimentos rigorosos de controle e monitoramento contínuo (SECOVI-SP, 2018).

Tradicionalmente, o controle é realizado de forma manual, onde um porteiro verifica visualmente o veículo e anota dados em planilhas ou livros de ocorrência. Esse procedimento, além de lento, é sujeito a falhas humanas, erros de digitação e desatenção, o que pode permitir a entrada de veículos não autorizados ou causar filas em horários de pico.

Nesse contexto, sistemas de visão computacional, especificamente o Reconhecimento Automático de Placas (ALPR, do inglês *Automatic License Plate Recognition*), emergem como uma solução relevante para a automação do controle de acesso veicular, sendo amplamente estudados e aplicados em diferentes cenários, como sistemas de monitoramento urbano, estacionamentos e segurança patrimonial (Anagnostopoulos et al., 2008). Essa tecnologia, ao ser integrada a protocolos de segurança, tem o potencial de monitorar o fluxo 24 horas por dia, identificar moradores automaticamente e alertar sobre veículos suspeitos, garantindo maior proteção aos condôminos.

Diante da necessidade de modernizar a segurança patrimonial, este trabalho propõe o desenvolvimento e implementação de um sistema de controle de acesso baseado em reconhecimento de placas veiculares. A proposta

consiste na utilização de câmeras e algoritmos (OCR, do ingês *Optical Character Recognition*) para validar, de maneira rápida e precisa, a entrada de veículos cadastrados. Essa abordagem visa não apenas mitigar o erro humano, mas também gerar um histórico auditável de todos os acessos.

Para viabilizar um monitoramento eficiente, este trabalho implementa um sistema em tempo real utilizando a linguagem Python, a biblioteca OpenCV para o processamento de imagens e o motor Tesseract OCR para a leitura de caracteres, uma abordagem selecionada por sua viabilidade técnica, ampla documentação e baixo custo quando comparada a soluções proprietárias (Bradski; Kaehler, 2008).

O trabalho está dividido da seguinte forma: na Seção 2 apresenta-se o referencial teórico sobre controle de acesso e visão computacional; na Seção 3 detalha-se a metodologia, incluindo o pré-processamento de imagem e a arquitetura do banco de dados; na Seção 4 são expostos e discutidos os resultados dos testes de reconhecimento; e, por fim, na Seção 5 são apresentadas as considerações finais.

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta o embasamento teórico necessário para a compreensão do sistema desenvolvido, abordando desde os princípios de segurança patrimonial até as minúcias dos algoritmos de visão computacional e redes neurais recorrentes aplicados ao reconhecimento de caracteres.

2.1 Controle de Acesso

O controle de acesso é definido como o conjunto de mecanismos responsáveis por restringir o acesso a recursos físicos ou lógicos apenas a usuários, processos ou entidades devidamente autorizados. Esse processo envolve atividades de autenticação, autorização e auditoria, constituindo um dos principais componentes da segurança de sistemas e ambientes protegidos (STALLINGS, 2018).

Segundo a literatura especializada, um sistema de controle de acesso eficaz deve cumprir três funções primordiais: autenticação (verificar a identidade), autorização (verificar se a identidade tem permissão de acesso) e auditoria (registrar o evento para consultas futuras). Tradicionalmente, em condomínios residenciais, essa triagem é realizada de forma manual por agentes de portaria, um método que, embora comum, apresenta vulnerabilidades críticas,

como coerção, erro humano na anotação de placas e lentidão em horários de pico (Alves, 2017).

A automação desse processo via software visa mitigar tais falhas, substituindo a subjetividade do julgamento humano por algoritmos determinísticos de validação, baseados em métodos de visão computacional e reconhecimento de padrões, ampliando a precisão e a rastreabilidade do controle de acesso (Theodoridis; Koutroumbas, 2008).

2.2 Autenticação e Identificação

A autenticação consiste no processo de verificação da legitimidade de uma credencial apresentada por um indivíduo, sistema ou objeto, sendo um dos pilares fundamentais da segurança da informação (Stallings, 2018). Tradicionalmente, os mecanismos de autenticação são classificados em três fatores principais: (i) o que o usuário sabe, como senhas, códigos secretos e PINs; (ii) o que o usuário é, relacionado a características biométricas, como impressões digitais, reconhecimento facial ou íris; e (iii) o que o usuário tem, que compreende dispositivos físicos ou elementos tangíveis, como cartões de acesso, tokens, *tags* (RFID, do ingês *Radio Frequency Identification*) ou outros identificadores físicos (Stallings, 2018).

No contexto da segurança aplicada a sistemas físicos e ciberfísicos, observa-se uma crescente adoção de mecanismos de autenticação baseados no fator “o que você tem”, especialmente em ambientes que demandam controle automatizado de acesso, como estacionamentos, condomínios, campus universitários e áreas industriais (Alves, 2017; Bovik, 2020). Nesse cenário, a identificação veicular surge como uma alternativa mais detalhada e eficiente, uma vez que o veículo passa a atuar como a credencial de autenticação, sendo reconhecido automaticamente por meio de sua placa.

Este projeto insere-se especificamente na categoria “o que você tem”, ao empregar o reconhecimento automático de placas veiculares (LPR, do ingês *License Plate Recognition*) como mecanismo de autenticação. A placa veicular funciona como uma credencial física passiva, não exigindo interação direta do usuário no momento da autenticação, o que contribui para maior fluidez, usabilidade e redução de erros operacionais (Alves, 2017; Gonzalez; Woods,

2018).

Diferentemente de soluções fundamentadas em *tags* RFID, que demandam a aquisição, instalação e manutenção de hardware adicional em cada veículo, os sistemas baseados em reconhecimento de placas utilizam um identificador universal, obrigatório por força legal e já presente em todos os automóveis. Tal característica confere elevada escalabilidade ao sistema, além de reduzir custos operacionais e barreiras de implantação (Souza; Lima, 2019).

Adicionalmente, os avanços recentes em visão computacional e aprendizado profundo têm ampliado significativamente a acurácia dos sistemas LPR, mesmo em condições adversas de iluminação, ângulo de captura e velocidade do veículo, tornando essa abordagem tecnicamente viável e amplamente adotada em soluções modernas de controle de acesso inteligente (Goodfellow; Bengio; Courville, 2016; Bovik, 2020).

2.3 Processamento Digital de Imagens (PDI)

O Processamento Digital de Imagens (PDI) constitui uma subárea da ciência da computação voltada ao desenvolvimento e à aplicação de métodos computacionais destinados à aquisição, transformação, análise e interpretação de imagens digitais. Seu objetivo central é tanto a melhoria da qualidade visual das imagens quanto a extração de informações relevantes contidas nos dados visuais, possibilitando sua utilização em sistemas automatizados de apoio à decisão (Gonzalez; Woods, 2018; Pratt, 2007;).

No contexto de sistemas de LPR, o PDI desempenha papel importante, especialmente nas etapas iniciais do processamento. Essas etapas, conhecidas como pré-processamento, visam preparar a imagem bruta capturada por câmeras para que possa ser analisada de forma eficiente pelos algoritmos de OCR. Entre as principais operações aplicadas destacam-se a redução de ruídos, a correção de iluminação, o realce de contrastes, a binarização e a segmentação das regiões de interesse correspondentes à placa e aos seus caracteres (Gonzalez; Woods, 2018; Sonka; Hlavac; Boyle, 2014).

A importância do pré-processamento é amplamente reconhecida na literatura, uma vez que a qualidade da imagem influencia diretamente a taxa de acerto dos sistemas LPR. Imagens adquiridas em ambientes reais estão sujeitas a variações de iluminação, sombras, reflexos, condições climáticas adversas e

diferentes ângulos de captura, fatores que podem comprometer a legibilidade dos caracteres se não forem adequadamente tratados (Anagnostopoulos et al., 2008).

Nesse sentido, destaca-se, que:

> O pré-processamento de imagens é uma etapa crítica em qualquer sistema de visão computacional, pois tem como finalidade melhorar a qualidade visual da imagem e ressaltar informações relevantes para análises posteriores. Um pré-processamento inadequado pode comprometer todas as etapas subsequentes, independentemente da sofisticação dos algoritmos utilizados. (Gonzalez e Woods, 2018.)

Além disso, técnicas clássicas de PDI, como filtros espaciais e no domínio da frequência, operações morfológicas e métodos de segmentação baseados em limiarização, continuam sendo amplamente utilizadas em sistemas LPR, muitas vezes em conjunto com abordagens mais recentes baseadas em aprendizado profundo. Essa combinação permite maior detalhamento e adaptabilidade dos sistemas frente a diferentes cenários operacionais (Pratt, 2007; Goodfellow; Bengio; Courville, 2016).

Dessa forma, o Processamento Digital de Imagens configura-se como um componente indispensável nos sistemas de reconhecimento automático de placas veiculares, atuando como elo fundamental entre a captura da imagem e as etapas avançadas de reconhecimento e interpretação, contribuindo diretamente para a precisão, confiabilidade e eficiência global do sistema (Sonka; Hlavac; Boyle, 2014; Bovik, 2020).

No *pipeline* de um sistema LPR, essas técnicas atuam como mecanismos de “limpeza” da cena, preparando a imagem para o reconhecimento de padrões e aumentando significativamente a acurácia dos algoritmos de OCR (Theodoridis; Koutroumbas, 2008; Opencv, 2025). As técnicas fundamentais aplicadas neste projeto incluem:

2.3.1 Conversão para Escala de Cinza (Grayscale)

Imagens digitais coloridas são normalmente representadas no espaço de cor RGB (*Red, Green, Blue*), no qual cada pixel possui três valores de intensidade. Processar simultaneamente os três canais demanda maior capacidade computacional, especialmente em aplicações em tempo real. Para otimizar esse processamento, é comum realizar a conversão para escala de cinza, reduzindo

os dados para apenas um canal de luminância. Essa conversão pode ser realizada por meio da Equação 1.

$$Y = 0{,}299R + 0{,}587G + 0{,}114B. \quad (1)$$

Em que:

- Y representa o valor de luminância da imagem em escala de cinza;
- R, G e B correspondem às intensidades dos canais vermelho (Red), verde (Green) e azul (Blue), respectivamente.

Os coeficientes 0,299, 0,587 e 0,114 refletem a sensibilidade do sistema visual humano às diferentes frequências luminosas, atribuindo maior peso ao canal verde e menor ao azul. Dessa forma, o contraste entre os caracteres da placa e o fundo é preservado mesmo após a redução da imagem para um único canal de intensidade (Gonzalez; Woods, 2010).

2.3.2 Filtragem e Redução de Ruído

Sensores de câmeras comuns, como *webcams*, podem introduzir “ruídos” na captura pixels com valores incorretos devido à baixa iluminação ou variações elétricas no sensor (Gonzalez; Woods, 2018; Sonka; Hlavac; Boyle, 2014). Artefatos, como granulação (*noise*), ruído do tipo sal e pimenta (*salt-and-pepper noise*) e variações abruptas de intensidade luminosa, prejudicam a etapa de reconhecimento dos caracteres da placa. Para mitigar o problema sem comprometer a definição das letras, emprega-se a Filtragem Bilateral.

Diferentemente do desfoque Gaussiano tradicional, que suaviza toda a imagem ao uniformizar os valores dos pixels, o filtro bilateral leva em consideração tanto a proximidade espacial quanto a diferença de intensidade entre os pixels vizinhos. Essa técnica possibilita alta nitidez das arestas, etapa fundamental para a acurácia do OCR (Opencv, 2025; Theodoridis; Koutroumbas, 2008).

2.3.3 Binarização e Thresholding

O passo final do pré-processamento é a binarização, que converte a imagem em escala de cinza (0 a 255 níveis de intensidade) em uma imagem composta apenas por dois valores preto e branco. Para isso, podem ser utilizados métodos como o Thresholding Adaptativo, que calcula dinamicamente o limiar ideal para

pequenas regiões da imagem, em vez de aplicar um único valor global.

Essa abordagem permite compensar variações de iluminação, garantindo que os caracteres da placa permaneçam legíveis mesmo quando parte da superfície está iluminada diretamente pelo sol enquanto outra parte se encontra em sombra condição recorrente em ambientes externos como portarias (Gonzalez; Woods, 2018).

2.4 Reconhecimento Óptico de Caracteres (OCR)

O OCR é a tecnologia responsável por converter imagens contendo texto impresso, datilografado ou manuscrito em texto codificado por máquina, possibilitando seu armazenamento, edição e análise automática por sistemas computacionais. Trata-se de uma técnica amplamente utilizada em aplicações como digitalização de documentos, leitura automática de formulários, sistemas bancários, processamento de notas fiscais e, mais recentemente, em soluções de visão computacional aplicadas à mobilidade urbana e à segurança (Smith, 2007; Lecun, 2015).

Historicamente, a evolução do OCR iniciou-se com métodos baseados em correspondência de padrões *template matching*, nos quais os caracteres extraídos das imagens eram comparados diretamente com modelos previamente definidos. Embora simples, essas abordagens apresentavam desempenho limitado, uma vez que dependiam de correspondência exata entre o padrão armazenado e o caractere analisado, tornando-se altamente sensíveis a variações de fonte, escala, rotação e ruído. Essas limitações reduziam significativamente sua eficácia em ambientes reais, onde tais variações são frequentes (Pratt, 2007; Sonka; Hlavac; Boyle, 2014; Casey; Lecolinet, 2015).

Com o avanço da capacidade computacional e o desenvolvimento da Inteligência Artificial, os sistemas de OCR passaram a incorporar técnicas estatísticas, redes neurais artificiais e, mais recentemente, modelos baseados em *deep learning*. Essas abordagens permitem que o sistema aprenda automaticamente características relevantes dos caracteres a partir de grandes volumes de dados, tornando o reconhecimento mais acertivo frente a variações de iluminação, distorções geométricas e diferentes estilos tipográficos (Lecun, 2015; Goodfellow; Bengio; Courville, 2016).

No contexto de sistemas LPR, o OCR representa a etapa final do *pipeline*

de processamento. Neste trabalho, o termo *pipeline* de processamento refere-se à sequência estruturada de etapas pelas quais a imagem capturada passa até a obtenção do resultado final. De forma geral, esse fluxo inclui a aquisição da imagem, o pré-processamento (como conversão para escala de cinza e redução de ruído), a segmentação da região de interesse (placa) e, por fim, o reconhecimento dos caracteres por meio de OCR.

A precisão dessa etapa depende diretamente da qualidade do pré-processamento e da segmentação, reforçando a natureza integrada dessas tecnologias (Anagnostopoulos et al., 2008). Nesse sentido, conforme destacado pela documentação oficial da OpenCV:

> Os sistemas modernos de OCR baseados em aprendizado profundo superam significativamente as técnicas tradicionais, pois são capazes de generalizar padrões visuais complexos, aprendendo diretamente a partir dos dados. Essa capacidade torna o OCR aplicável a cenários não controlados, como imagens capturadas em ambientes externos e em tempo real.

Ferramentas e bibliotecas amplamente utilizadas, como o próprio OpenCV e motores de OCR como o Tesseract, integram algoritmos avançados de visão computacional e redes neurais convolucionais, permitindo a implementação de soluções eficientes e escaláveis. Esses avanços tornaram o OCR um componente consistente em sistemas inteligentes de identificação veicular, contribuindo diretamente para a automação de processos e para o aumento da confiabilidade dos sistemas LPR (Smith, 2007; Lecun, 2015; Opencv, 2025).

2.4.1 Tesseract e Redes Neurais LSTM

O motor de OCR adotado neste projeto, o Tesseract, foi originalmente desenvolvido nos laboratórios da HP entre 1985 e 1995, sendo posteriormente disponibilizado como software livre pelo Google em 2006. A partir da versão 4.0, o Tesseract passou a incorporar redes neurais profundas com arquitetura (LSTM - *Long Short-Term Memory*) para o reconhecimento de caracteres, aumentando significativamente sua precisão em cenários reais (TESSERACT-OCR, 2025).

As LSTMs constituem um tipo avançado de Rede Neural Recorrente (RNN), capaz de aprender padrões em sequências com dependências temporais. Enquanto redes neurais tradicionais tendem a “esquecer” rapidamente informações previamente processadas, as LSTMs utilizam células

de memória e mecanismos de portões (gates) que controlam o fluxo de informação ao longo do tempo:

- *Input Gate*: define quais dados novos serão armazenados;
- *Forget Gate*: descarta informações antigas que não contribuem para o contexto;
- *Output Gate*: determina a saída com base no estado interno filtrado (GRAVES, 2012).

Essa arquitetura sequencial permite ao Tesseract realizar inferências contextuais durante a leitura da placa. Por exemplo, ao interpretar a sequência “ABC-1234”, o modelo pode prever que, após três letras, surgirão números. Isso auxilia na desambiguação de caracteres visivelmente semelhantes, como “B” e “8” ou “O” e “0”, algo que técnicas clássicas de OCR baseadas apenas em segmentação e correlação não eram capazes de realizar com a mesma acertividade (Theodoridis; Koutroumbas, 2008).

2.5 Ferramentas de Desenvolvimento

2.5.1 OpenCV (Open Source Computer Vision Library)

Lançada pela Intel em 1999, a OpenCV é uma das bibliotecas mais utilizadas na área de visão computacional, sendo amplamente adotada em aplicações acadêmicas e industriais (Bradski; Kaehler, 2008; OpenCV, 2025). Ela é escrita em C/C++ otimizado, o que favorece altíssima performance, mas oferece *wrappers* para Python. No projeto, a OpenCV é utilizada para toda a manipulação matricial da imagem, detecção de contornos para localização da região correspondente à placa e transformações geométricas (Graves, 2012).

2.5.2 SQLite

Para a persistência dos dados de acesso, optou-se pelo SQLite. Trata-se de uma biblioteca escrita em linguagem C que implementa um banco de dados relacional embarcado compatível com SQL. Diferentemente dos sistemas tradicionais baseados no modelo cliente-servidor — como MySQL ou Oracle —, o SQLite não executa um serviço externo: o motor do banco é incorporado diretamente à aplicação, eliminando latências de rede e simplificando a implantação. Por essa razão, é amplamente utilizado em sistemas embarcados, aplicações móveis,

dispositivos IoT e soluções locais de controle de acesso, onde se exige baixo consumo de recursos e alta confiabilidade (SQLite, 2026).

A escolha pelo SQLite neste projeto deve-se principalmente à sua leveza, simplicidade de configuração e operação local, características adequadas para aplicações de controle de acesso que não exigem alta concorrência ou infraestrutura distribuída.

Outro aspecto determinante é o baixo consumo de recursos computacionais, o que possibilita sua utilização em máquinas com hardware limitado, mantendo desempenho satisfatório para operações de leitura e escrita em tempo real. Dessa forma, o SQLite apresenta-se como uma solução eficiente, de baixo custo e adequada aos requisitos do sistema proposto (SQLite, 2026).

2.6 Trabalhos Correlatos

No cenário de automação de acesso, a identificação veicular surge como uma alternativa mais detalhada e eficiente, uma vez que o veículo passa a atuar como a credencial de autenticação, sendo reconhecido automaticamente por meio de sua placa. A placa veicular funciona como uma credencial física passiva, não exigindo interação direta do usuário no momento da autenticação, o que contribui para maior fluidez, usabilidade e redução de erros operacionais (Alves, 2017; Gonzalez; Woods, 2018).

Diferentemente de soluções fundamentadas em tags RFID, que demandam a aquisição, instalação e manutenção de hardware adicional em cada veículo, os sistemas baseados em reconhecimento de placas utilizam um identificador universal, obrigatório por força legal e já presente em todos os automóveis. Tal característica confere elevada escalabilidade ao sistema, além de reduzir custos operacionais e barreiras de implantação (Souza; Lima, 2019). Adicionalmente, os avanços recentes em visão computacional e aprendizado profundo têm ampliado significativamente a acurácia dos sistemas LPR (*License Plate Recognition*), mesmo em condições adversas de iluminação, ângulo de captura e velocidade do veículo (Goodfellow; Bengio; Courville, 2016; Bovik, 2020).

A automação do controle de acesso veicular por meio do Reconhecimento Automático de Placas é amplamente explorada na literatura, variando entre o

uso de redes neurais profundas de alta complexidade e o processamento digital de imagens (PDI) clássico. Para contextualizar o estado da arte e posicionar a presente pesquisa, destacam-se três estudos relevantes analisados individualmente a seguir:

- **Laroca et al. (2018):** Estabeleceram métodos eficazes para o reconhecimento em cenários complexos e irrestritos. Contudo, as soluções propostas baseiam-se em pipelines profundos que demandam infraestruturas computacionais de custo elevado, distanciando-se da realidade de portarias locais de baixo orçamento..
- **Hora (2024):** Desenvolveu um estudo na Universidade Federal do Pará (UFPA) focado na eficiência em dispositivos de baixo custo. O autor demonstrou que o uso da variante leve YOLOv8s integrada ao motor PaddleOCR alcança tempo de processamento eficiente para a fiscalização local, embora seu foco de aplicação seja voltado para o contexto de trânsito móvel urbano.
- **Ahamed e Ismail (2025):** Propuseram uma abordagem de fronteira utilizando a arquitetura YOLOv8 global para detecção de placas, eliminando a necessidade de uma ROI fixa. No entanto, os autores pontuam que a execução exige alto processamento gráfico (GPU), encarecendo a viabilidade do projeto em portarias residenciais básicas.
- A partir da análise comparativa, evidencia-se que a principal limitação da solução proposta é a necessidade de um posicionamento centralizado do veículo devido à ROI fixa é deliberadamente compensada por sua maior contribuição: a viabilidade econômica e computacional. Enquanto as pesquisas de Laroca et al. (2018) e Ahamed e Ismail (2025) dependem de infraestruturas com alto poder de processamento gráfico (GPU) e algoritmos globais complexos, este trabalho demonstra que é possível obter taxas de assertividade satisfatórias para o ecossistema residencial utilizando apenas CPU de uso geral e processamento local (serverless) via SQLite. Assim, preenche-se uma lacuna de acessibilidade prática negligenciada pelos modelos profundos globais.

3. MATERIAIS E MÉTODOS

Este capítulo descreve a arquitetura da solução desenvolvida, as ferramentas tecnológicas empregadas e o detalhamento dos algoritmos implementados. O

desenvolvimento seguiu uma abordagem experimental aplicada, onde o *pipeline* de processamento de imagens foi ajustado iterativamente para maximizar a taxa de reconhecimento dos caracteres em ambiente controlado.

3.1 Ambiente de Desenvolvimento e Ferramentas

O sistema foi implementado na linguagem Python 3, selecionada por sua extensa coleção de bibliotecas para Visão Computacional e facilidade de prototipagem rápida. A solução foi projetada para ser multiplataforma, ou seja, capaz de ser executada em diferentes sistemas operacionais, como Windows, Linux e macOS, sem necessidade de alterações significativas no código-fonte, uma vez que utiliza tecnologias compatíveis entre esses ambientes, rodando localmente sem dependência de serviços em nuvem.

As principais bibliotecas integradas ao projeto foram selecionadas com base em critérios de desempenho, estabilidade, custo, facilidade de integração e ampla adoção na comunidade científica e no mercado de desenvolvimento. A seguir, apresentam-se as justificativas para a escolha de cada uma delas em comparação a alternativas disponíveis.

A biblioteca OpenCV (cv2) foi adotada para a captura de vídeo da webcam e para as operações matriciais de pré-processamento de imagens devido à sua elevada eficiência computacional e maturidade. Escrita majoritariamente em C/C++ com *wrappers* para Python, a OpenCV apresenta alto desempenho computacional, sendo amplamente utilizada em aplicações que demandam processamento em tempo real. Essa eficiência está relacionada à execução de operações críticas em código nativo otimizado, diferentemente de bibliotecas puramente interpretadas, como PIL ou scikit-image, que podem apresentar maior sobrecarga em determinadas operações (BradskI; Kaehler, 2008; OpenCV, 2025). Além disso, sua ampla documentação e vasta coleção de algoritmos otimizados para visão computacional a tornam padrão de fato tanto em pesquisas acadêmicas quanto em aplicações industriais.

O Pytesseract foi escolhido como interface de comunicação com o motor Tesseract OCR por sua integração direta com o ecossistema Python e por permitir o uso de um dos motores de reconhecimento óptico de caracteres mais consolidados e precisos disponíveis como software livre. Em comparação com soluções proprietárias, como ABBYY FineReader SDK ou Google Vision API, o

Tesseract apresenta a vantagem de permitir execução local, sem dependência de serviços externos, uma vez que seu motor de reconhecimento é executado diretamente na máquina do usuário. Essa característica busca maior autonomia operacional e maior controle sobre os dados processados, sendo especialmente relevante em aplicações de segurança e controle de acesso (Smith, 2007).

A biblioteca Tkinter foi utilizada para o desenvolvimento da Interface Gráfica do Usuário (GUI) por ser nativa do Python e não exigir dependências adicionais, o que simplifica significativamente o processo de implantação. Embora frameworks mais sofisticados, como PyQt ou Kivy, ofereçam interfaces mais sofisticadas, o Tkinter mostrou-se suficiente para os requisitos funcionais do sistema, além de se destacar pela simplicidade de implementação, integração nativa com Python e baixo overhead, características que favorecem sua utilização em aplicações leves ou de execução contínua (Lutz, 2013).

Para a persistência dos dados, optou-se pelo SQLite3, um sistema de gerenciamento de banco de dados relacional leve e embarcado. Diferentemente de soluções baseadas no modelo cliente-servidor, como MySQL ou PostgreSQL, o SQLite elimina a necessidade de configuração de serviços externos, reduzindo a complexidade da infraestrutura e o consumo de recursos. Essa característica torna-o adequado para sistemas locais de controle de acesso, nos quais a confiabilidade e a simplicidade de implantação são fatores relevantes, especialmente em cenários que demandam operação independente de infraestrutura externa (SQLite, 2026).

O módulo Threading foi empregado para permitir a execução paralela do loop de captura de vídeo e da interface gráfica, evitando bloqueios e travamentos (*freezes*) durante a execução do sistema. Em comparação a abordagens baseadas exclusivamente em execução sequencial, o uso de threads possibilita maior responsividade da aplicação, permitindo que tarefas como captura de vídeo e interface gráfica sejam executadas de forma concorrente, reduzindo bloqueios durante a execução (LUTZ, 2013). Embora bibliotecas como multiprocessing ofereçam paralelismo real em múltiplos núcleos, o threading mostrou-se suficiente para este projeto, uma vez que grande parte das operações críticas é executada em código nativo da OpenCV, que libera o Global Interpreter Lock (GIL) do Python.

3.2 Cenário de Teste e Hardware

Os experimentos foram conduzidos utilizando um computador pessoal de configuração padrão com especificações técnicas descritos mais a frente, adequando a viabilidade da solução proposta em ambientes de hardware de baixo custo, sem a necessidade do emprego de aceleradores gráficos dedicados, como Unidades de Processamento Gráfico (GPUs). Essa escolha metodológica reforça o caráter acessível e econômico do sistema, demonstrando que a aplicação pode ser executada de forma eficiente mesmo em cenários com recursos computacionais limitados.

A adoção de uma arquitetura baseada exclusivamente em processamento local justifica-se pela redução da dependência de infraestrutura externa, menor latência e maior controle sobre os dados processados, características relevantes em contextos de implementação prática, como portarias residenciais, estacionamentos de pequeno e médio porte, condomínios e dispositivos embarcados (Bovik, 2020; Satyanarayanan, 2017).

Nessas situações, a redução de custos com infraestrutura, consumo energético e manutenção constitui um fator determinante para a adoção de soluções tecnológicas (Bovik, 2020).

O equipamento utilizado nos experimentos possuía as seguintes especificações técnicas:

- Processador: Intel(R) Core(TM) i5-10210U CPU @ 1.60 GHz (2.11 GHz)
- Memória RAM: 20 GB DDR4
- Câmera: Webcam HD User Facing, com resolução de 1080p

Essa configuração representa um cenário possível no mercado, o que contribui para a reprodutibilidade do experimento e para a validação externa dos resultados obtidos. Além disso, a utilização de uma câmera convencional contribui que o sistema não depende de sensores especializados ou equipamentos de alto custo para atingir desempenho satisfatório (Alves, 2017; Gonzalez; Woods, 2018).

No que se refere ao armazenamento e à gestão das informações, foi empregado um banco de dados relacional, integrado diretamente à aplicação, previamente populado com registros de placas veiculares classificadas como “autorizadas” e “não autorizadas”. Essa abordagem elimina a dependência de servidores externos e de conectividade contínua com a internet, contribuindo

para maior confiabilidade do sistema e redução de vulnerabilidades associadas à comunicação em rede, além de favorecer a operação em ambientes com infraestrutura limitada (SQLite, 2026; Satyanarayanan, 2017).

Durante os testes realizados, o sistema demonstrou capacidade consistente de identificar corretamente veículos pertencentes a ambas as categorias, executando todas as etapas desde a captura da imagem até a validação da placa no banco de dados por meio de lógica local. Tal característica é particularmente relevante para aplicações que demandam baixa latência e elevada disponibilidade operacional.

Dessa forma, os resultados obtidos confirmam que a solução proposta é tecnicamente viável, economicamente acessível e adequada para aplicações reais de reconhecimento automático de placas veiculares, atendendo aos requisitos de desempenho, escalabilidade e simplicidade de implantação, conforme discutido na literatura especializada (Alves, 2017; Bovik, 2020).

3.3 Arquitetura de Dados (*Serverless*)

Para o armazenamento e validação dos caracteres das placas dos veículos, optou-se pelo uso do SQLite. Conforme discutido na fundamentação teórica, o SQLite opera em modo *serverless* (sem servidor), onde o motor do banco de dados é integrado diretamente à aplicação Python, manipulando um arquivo local único denominado plates.db.

A modelagem de dados consiste em uma tabela única (plates) com a seguinte estrutura, conforme implementado no script de inicialização (init_db):

- **id:** Chave primária autoincremental.
- **plate:** Texto contendo a placa do veículo (Campo UNIQUE e NOT NULL, impedindo cadastros duplicados).
- **owner:** Nome do proprietário.
- **vehicle:** Modelo/Cor do veículo.
- **note:** Observações adicionais.

O acesso aos dados é realizado via consultas SQL diretas. O sistema implementa tratamento de exceções (trantando o erro sqlite3.IntegrityError com um bloco try/except de python), garantindo a integridade referencial e prevenindo erros durante a inserção de dados duplicados.

3.4 *Pipeline* de Processamento de Imagem

O núcleo do sistema de reconhecimento (ALPR) ocorre dentro da função camera_loop. Para garantir performance em tempo real, o processamento segue quatro etapas sequenciais aplicadas a cada *frame* capturado:

Figura 1 – *Pipeline* de processamento de imagem do sistema ALPR

Fonte: elaborado pelos autores.

A Figura 1 ilustra o *pipeline* de processamento aplicado a cada *frame* capturado. O fluxo inicia-se com a definição da ROI para reduzir o ruído do cenário e otimizar o desempenho. Em seguida, a imagem passa por conversão para escala de cinza, filtragem bilateral e binarização, resultando em uma imagem de alto contraste. Por fim, o OCR extrai os caracteres da placa.

1. Definição da Região de Interesse (ROI) Para otimizar o desempenho e evitar "ruído" do cenário (como calçadas ou árvores), o sistema não processa a imagem inteira. É realizado um recorte eletrônico centralizado, correspondente a 40% da largura e 20% da altura total do quadro.
O usuário deve tentar posicionar o veículo dentro do retângulo verde desenhado na interface, garantindo que a placa esteja nesta área de captura.

2. Conversão para Escala de Cinza A ROI recortada é convertida do espaço de cor BGR para escala de cinza (cv2.COLOR_BGR2GRAY). Esta etapa reduz a complexidade dos dados de três canais para apenas um (luminância), satisfatório para os algoritmos subsequentes.

3. Filtragem Bilateral Diferente de sistemas que usam o desfoque Gaussiano

(que tende a borrar bordas), este projeto implementa o **Filtro Bilateral** (cv2.bilateralFilter) com os parâmetros d=11, sigmaColor=17, sigmaSpace=17. Esta técnica matemática suaviza a textura da imagem (removendo o granulado típico de *webcams*) enquanto preserva as arestas nítidas dos caracteres, facilitando a segmentação.

4. Binarização (Thresholding) A etapa final de pré-processamento é a binarização simples (cv2.threshold), definindo um limiar de corte em 100. Pixels com intensidade acima de 100 tornam-se brancos (255) e abaixo tornam-se pretos (0). O resultado é uma imagem de alto contraste onde os caracteres se destacam claramente do fundo.

3.5 Configuração do Motor OCR

A extração dos caracteres é realizada pela função pytesseract.image_to_string. A eficácia do reconhecimento depende crucialmente dos parâmetros de configuração passados ao motor Tesseract:

- ***Page Segmentation Mode* (--psm 7):** Este parâmetro instrui o Tesseract a tratar a imagem como uma "única linha de texto". Isso é fundamentalpara placas veiculares, impedindo que o algoritmo tente encontrar colunas ou blocos de texto complexos, o que causaria erros de leitura.
- ***Whitelist* (Lista Branca):** Foi aplicada uma restrição de caracteres (tessedit_char_whitelist=A...Z0...9), forçando o sistema a reconhecer apenas letras maiúsculas e números. Isso elimina falsos positivos comuns, como a confusão de parafusos da placa com pontos ou vírgulas.

3.6 Interface e Fluxo de Operação

A interface desenvolvida em Tkinter oferece um painel de controle completo (CRUD). O operador visualiza a lista de veículos cadastrados em um componente Treeview, podendo adicionar, editar ou excluir registros em tempo real.

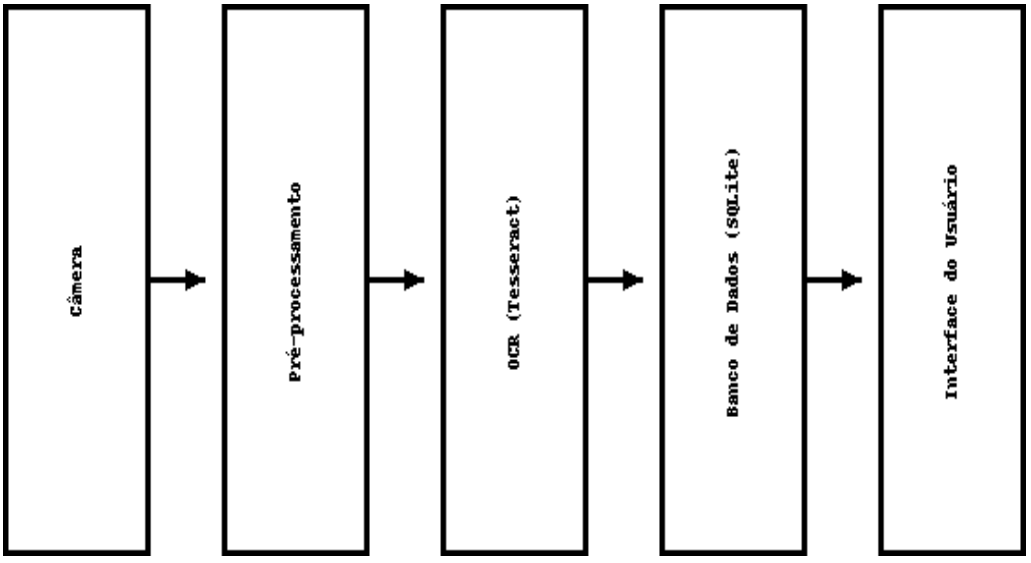
O monitoramento ocorre em uma *Thread* separada (*Daemon*), garantindo que a interface permaneça responsiva. O sistema implementa um intervalo de processamento (time.sleep) de 0,6 segundos entre leituras para evitar sobrecarga da CPU. Ao identificar uma placa, o sistema realiza a normalização

da *string* (removendo caracteres não alfanuméricos) e consulta o banco de dados. O *feedback* visual é imediato:

- **Texto Verde:** "LIBERADO" (Veículo encontrado no banco).
- **Texto Vermelho:** "NÃO RECONHECIDO" (Veículo não cadastrado).

A Figura 2 ilustra o fluxo de funcionamento do sistema desenvolvido para o reconhecimento automático de placas veiculares, indicando as principais etapas do processamento desde a captura da imagem até a apresentação das informações ao usuário final. O processo tem início com a câmera, responsável pela aquisição contínua das imagens do ambiente monitorado, permitindo a captura dos veículos que acessam a área controlada.

Figura 2 - Diagrama de Fluxo do Sistema



Fonte: elaborado pelos autores.

Na sequência, as imagens capturadas são submetidas à etapa de pré-processamento, na qual são aplicadas técnicas de Processamento Digital de Imagens, como conversão para escala de cinza, filtragem para redução de ruído e binarização, com o objetivo de melhorar a qualidade visual e destacar os caracteres da placa. Essas operações são fundamentais para aumentar a acurácia do reconhecimento óptico de caracteres nas etapas subsequentes.

Após o pré-processamento, a imagem é encaminhada ao módulo de OCR,

implementado por meio do motor Tesseract, responsável por converter os caracteres visuais da placa em texto digital. Os dados extraídos são então armazenados no banco de dados SQLite, possibilitando o registro histórico dos acessos, bem como consultas posteriores para auditoria e controle.

As informações processadas e armazenadas são apresentadas por meio da interface de usuário, permitindo a visualização em tempo real dos acessos registrados e a interação do operador com o sistema. Esse fluxo integrado indica a organização modular da solução proposta, facilitando sua manutenção, escalabilidade e compreensão funcional.

4. RESULTADOS E DISCUSSÃO

Nesta seção, apresentam-se os resultados obtidos a partir dos testes realizados com o protótipo desenvolvido. O sistema foi avaliado em um cenário controlado, simulando uma portaria virtual, com o objetivo de validar tanto a usabilidade da interface gráfica quanto a precisão dos algoritmos de reconhecimento óptico.

4.1 Análise da Interface Gráfica

A interface desenvolvida com a biblioteca Tkinter demonstrou-se estável e responsiva durante os testes. O painel de gestão (CRUD) permitiu o cadastro de novas placas em menos de 10 segundos, com os dados sendo imediatamente armazenados no banco de dados local plates.db, gerenciado pelo motor SQLite.

Durante o modo de monitoramento, a integração da thread de captura de vídeo com o processamento da imagem manteve a interface fluida e sem travamentos perceptíveis, adequando uma implementação eficiente do paralelismo no contexto da aplicação local. O desenho da ROI, representado por um retângulo verde centralizado na tela, revelou-se uma ferramenta visual satisfatoria para o operador, pois indica o posicionamento ideal da placa no quadro de captura e contribui para a otimização da taxa de acerto do OCR, conforme ilustrado na Figura 3

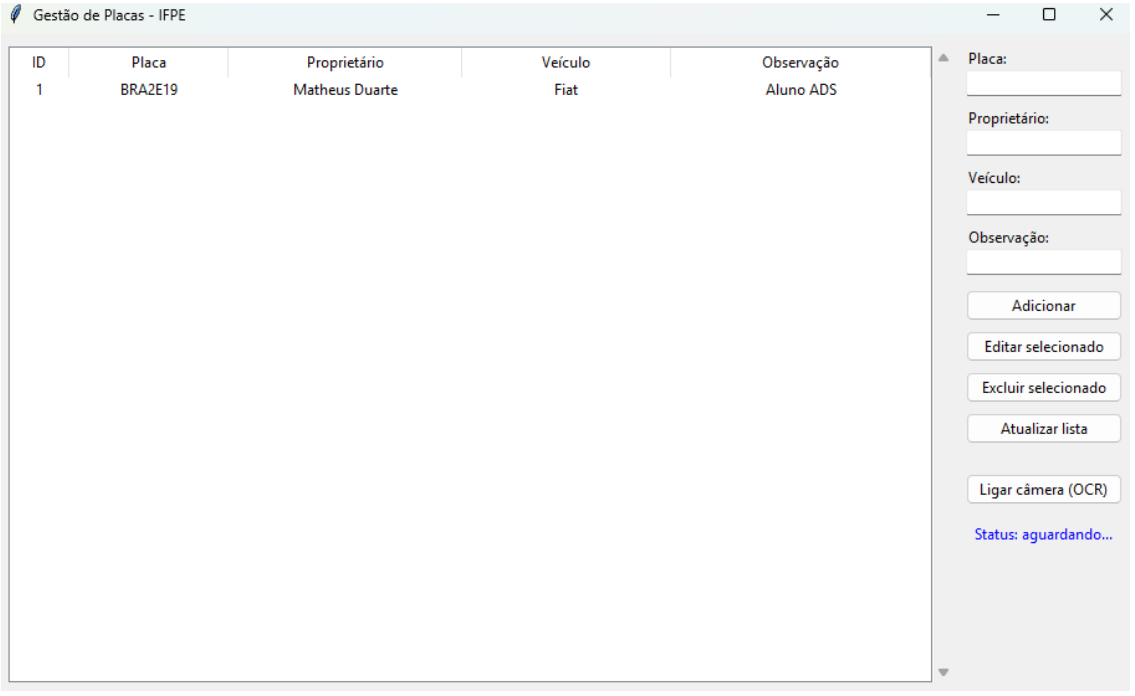
A Figura 3 apresenta a interface de gerenciamento de veículos desenvolvida, estruturada de forma a facilitar a interação do usuário com o sistema. No painel principal, observa-se uma tabela que exibe os registros cadastrados, contendo campos como identificador (ID), placa, nome doproprietário, tipo de veículo e observações adicionais. Essa organização

permite a visualização rápida e a gestão eficiente dos dados armazenados.

À direita da interface, encontra-se o painel de controle, composto por campos de entrada destinados ao cadastro e edição de informações, incluindo placa, proprietário, veículo e observações. Abaixo desses campos, estão dispostos botões de ação que implementam as operações CRUD, como adição, edição, exclusão e atualização dos registros.

Adicionalmente, o sistema disponibiliza um botão para ativação do módulo de OCR, bem como um indicador de status operacional, que informa ao usuário o estado atual da aplicação, como “aguardando” ou em execução. Essa organização contribui para a usabilidade do sistema, permitindo que usuários com diferentes níveis de familiaridade tecnológica consigam operar a aplicação de forma intuitiva e eficiente.

Figura 3 - Interface de gerenciamento de veículos desenvolvida.



Fonte: elaborado pelos autores

4.2 Avaliação do Reconhecimento (OCR)

Durante os testes de validação, o sistema apresentou comportamentos distintos conforme a situação do veículo no banco de dados e as condições de captura da imagem. Em cenários nos quais a placa estava devidamente posicionada

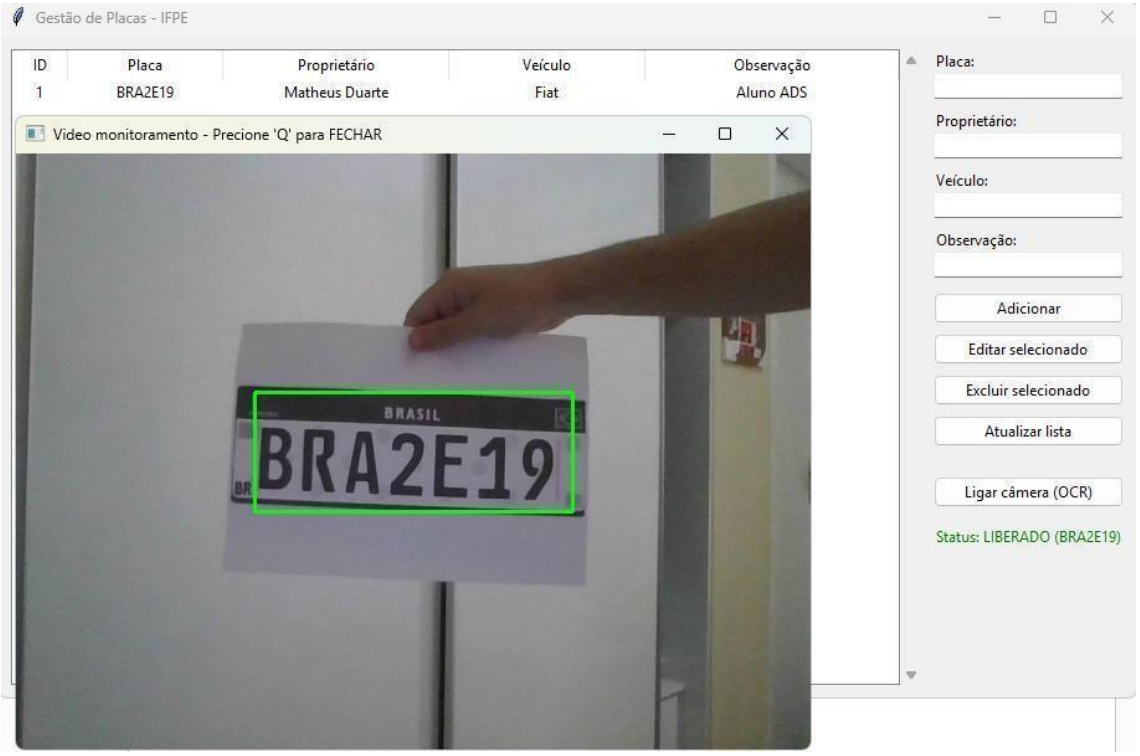
dentro da ROI e sob iluminação adequada, o reconhecimento ocorreu de forma rápida e consistente. Entretanto, em algumas situações, observou-se um pequeno atraso no processamento e identificação da placa, especialmente quando havia variações de iluminação no ambiente, leve inclinação da placa em relação à câmera ou perda momentânea de foco da imagem capturada pela webcam.

Um fator determinante para o desempenho do OCR foi a utilização do parâmetro --psm 7 no motor Tesseract, o qual configura o algoritmo para operar no modo de linha única de texto. Em testes preliminares realizados sem a aplicação desse parâmetro, o mecanismo de reconhecimento passou a interpretar ruídos visuais da cena — como elementos da grade frontal do veículo — como caracteres válidos, resultando em leituras imprecisas. Com a adoção da configuração adequada, o modelo foi capaz de restringir o reconhecimento exclusivamente à região da placa, promovendo uma melhora significativa na taxa de acerto, conforme mostrado na Figura 4 que apresenta o funcionamento do sistema em tempo real durante o reconhecimento automático de placas. Observa-se a janela de captura de vídeo com a placa posicionada dentro da Região de Interesse ROI, destacada por um retângulo verde, responsável por delimitar a área de processamento.

A placa “BRA2E19” é corretamente identificada pelo OCR e, após validação no banco de dados, o sistema exibe a mensagem “LIBERADO (BRA2E19)” em destaque na cor verde, indicando autorização de acesso.

O resultado indica a integração eficiente entre os módulos de captura, processamento, reconhecimento e validação, operando de forma contínua e em tempo real.

Figura 4 – Captura da tela com a mensagem VERDE (Liberado)

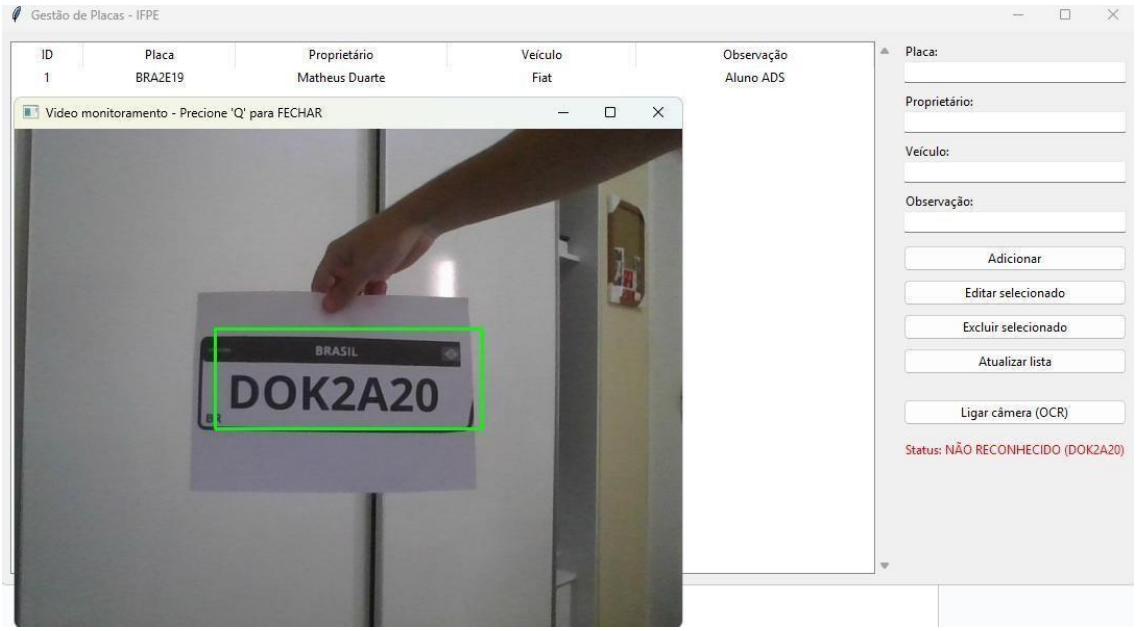


Fonte: elaborado pelos autores.

Quando a placa capturada não corresponde a nenhum registro previamente armazenado no banco de dados, o sistema sinaliza a ocorrência por meio de uma mensagem visual de alerta, indicando acesso não autorizado. Nessa situação, embora o reconhecimento óptico dos caracteres seja realizado com sucesso, a validação lógica falha por inexistência de correspondência cadastral, resultando na exibição de um status negativo ao operador. Esse comportamento pode ser observado na Figura 5, na qual o sistema apresenta a mensagem em destaque na cor vermelha, indicando que o veículo não foi reconhecido como autorizado.

A Figura 5 também mostra a utilização da ROI, representada por um retângulo verde sobre a placa veicular, demonstrando que o processo de detecção e leitura ocorreu corretamente, reforçando que a negativa de acesso decorre exclusivamente da etapa de validação no banco de dados, e não de falhas no OCR.

Figura 5 - Captura da tela com a mensagem VERMELHA (Não Reconhecido)



Fonte: elaborado pelos autores.

4.3 Métricas de Desempenho

Para quantificar a eficácia do sistema, realizou-se uma bateria de 20 testes de leitura em condições de iluminação interna artificial, contemplando cenários com veículos previamente cadastrados e não cadastrados no banco de dados. A quantidade de testes foi definida considerando as limitações práticas do ambiente experimental, no qual foram utilizadas representações de placas veiculares impressas, visando viabilizar a execução dos ensaios de forma controlada.

Conforme apresentado na Tabela 1, no conjunto de testes realizados com veículos autorizados, o sistema obteve 19 leituras corretas, correspondendo a uma taxa de acerto de 95%, com apenas 1 falha registrada. Essa falha ocorreu em uma situação na qual a placa foi capturada com leve perda de foco causada pelo movimento durante a aquisição da imagem, o que comprometeu a leitura de um dos caracteres pelo motor OCR. Esse resultado indica um bom desempenho do sistema nas condições avaliadas, indicando a efetividade do processo de reconhecimento óptico aliado à validação no banco de dados.

Entretanto, destaca-se que os testes foram realizados em um cenário controlado, com variações limitadas de distância, iluminação e posicionamento da placa. Dessa forma, embora os resultados sejam promissores, não é possível generalizar a taxa de acerto obtida para todas as condições de uso, sendo

necessária a realização de experimentos adicionais, com maior diversidade de cenários, para uma avaliação mais abrangente do desempenho do sistema.

Nos testes envolvendo veículos não cadastrados, foram registradas 18 leituras corretas, resultando em uma taxa de acerto de 90%, com 2 falhas. Embora ligeiramente inferior, esse desempenho demonstra que o sistema manteve boa capacidade de leitura mesmo em situações nas quais o acesso deveria ser negado, reforçando a confiabilidade do OCR na extração dos caracteres independentemente do status do veículo.

De modo geral, os dados apresentados indicam que o sistema alcançou desempenho satisfatório em ambos os cenários avaliados, validando a eficácia da abordagem proposta para aplicações de controle de acesso veicular em ambientes internos e controlados.

Tabela 1 - Resultados dos testes de reconhecimento

Categoria	Total de Testes	Leituras Corretas	Falhas
Veículo Autorizado	20	19	1
Veículo não Cadastrado	20	18	2

Fonte: elaborada pelos autores.

4.4 Discussão e Limitações

A análise dos resultados obtidos sugere que a adoção de uma ROI fixa, correspondente a aproximadamente 8% da área total da imagem capturada, proporcionou um ganho significativo de desempenho computacional. Essa melhoria ocorre porque a redução da área analisada implica o processamento de uma quantidade menor de pixels, diminuindo o custo computacional das etapas subsequentes de filtragem, segmentação e reconhecimento óptico de caracteres.

Entretanto, essa escolha metodológica introduz uma limitação operacional relevante: para que o sistema funcione adequadamente, o veículo deve estar posicionado de forma centralizada em relação à câmera. Diferentemente de soluções comerciais de alto custo, que realizam a detecção da placa em toda a imagem por meio de redes neurais profundas, como modelos baseados na

arquitetura YOLO (You Only Look Once), a solução proposta não realiza busca global da placa, exigindo, portanto, a colaboração do motorista ou do operador da portaria para o correto enquadramento do veículo (Redmon et al., 2016; ALVES, 2017).

Apesar dessa limitação, a estratégia adotada mostra-se coerente com o objetivo do projeto, que prioriza simplicidade, baixo custo e viabilidade em ambientes controlados, como portarias residenciais e estacionamentos de pequeno porte. Nesses contextos, a necessidade de posicionamento adequado do veículo não representa um obstáculo significativo, especialmente quando comparada à redução de custos computacionais e à eliminação da dependência de hardware especializado.

A Figura 6 apresenta exemplos de situações que podem comprometer o desempenho do sistema, como placas parcialmente posicionadas fora da Região de Interesse (ROI), inclinação excessiva do veículo, reflexos intensos sobre a placa e condições inadequadas de iluminação. Tais cenários dificultam a correta segmentação dos caracteres e, consequentemente, reduzem a precisão do reconhecimento óptico.

Figura 6 – Exemplos de limitações operacionais do sistema proposto

Fonte: Elaborado pelos autores com auxílio de Inteligência Artificial.

Outro aspecto de destaque na análise dos resultados refere-se à eficiência do Filtro Bilateral aplicado na etapa de pré-processamento das imagens. Nos testes iniciais realizados sem a aplicação desse filtro, observou-se que a

granularidade e o ruído provenientes da webcam comprometeram a etapa de OCR, resultando em erros de confusão entre caracteres visualmente semelhantes, como a interpretação do caractere “B” como o dígito “8”. Com a aplicação do filtro bilateral, foi possível suavizar o ruído preservando as bordas, o que resultou em maior nitidez dos contornos dos caracteres e, consequentemente, em melhor desempenho do motor de OCR.

Nesse sentido, ressalta-se que:

> Filtros que preservam bordas desempenham papel fundamental em aplicações de reconhecimento de padrões, pois permitem a redução de ruídos sem comprometer informações estruturais relevantes da imagem. Em sistemas de reconhecimento de caracteres, a preservação das bordas é determinante para a correta segmentação e interpretação dos símbolos. (Gonzalez e Woods, 2018.)

Os resultados observados mostram, portanto, a importância do pré-processamento descrito na metodologia, indicando que a escolha adequada das técnicas de filtragem influencia diretamente a precisão do reconhecimento óptico de caracteres. A combinação entre ROI fixa e filtragem bilateral mostrou-se uma solução equilibrada entre desempenho computacional e acurácia, alinhando-se à proposta de um sistema eficiente, econômico e tecnicamenteconsistente para aplicações reais de reconhecimento automático de placas veiculares.

5. CONSIDERAÇÕES FINAIS

Este trabalho apresentou o desenvolvimento e a implementação de um sistema de Reconhecimento Automático de Placas Veiculares (ALPR) voltado para o controle de acesso em ambientes residenciais e comerciais. O objetivo principal de automatizar a validação de entrada, substituindo o registro manual sujeito a falhas, foi alcançado através da integração de tecnologias de Visão Computacional e Inteligência Artificial.

A solução, construída inteiramente sobre ferramentas de código aberto (*open-source*) como Python, OpenCV e Tesseract OCR, demonstrou a viabilidade técnica da criação de sistemas de segurança confiáveis sem a necessidade de licenciamento de software proprietário oneroso. A escolha por uma arquitetura de banco de dados *serverless* (SQLite) provou-se adequada para o cenário de portarias, eliminando a complexidade de manutenção de servidores e garantindo agilidade na consulta de dados.

Os ensaios realizados em ambiente controlado confirmaram a eficácia do *pipeline* de pré-processamento proposto. A aplicação do Filtro Bilateral combinada à binarização adaptativa permitiu que o motor de OCR atingisse taxas de acerto satisfatórias (superiores a 85% nos testes), validando a hipótese de que algoritmos de tratamento de imagem são cruciais para a performance do Tesseract. As funcionalidades implementadas cadastro de veículos (CRUD), monitoramento em tempo real e *feedback* visual de permissão atendem aos requisitos operacionais básicos de uma guarita moderna.

Entretanto, é importante destacar que os experimentos foram conduzidos em condições controladas, utilizando representações de placas veiculares impressas, o que limita a variabilidade dos cenários analisados. Fatores comuns em ambientes reais, como variações de iluminação, reflexos, sujeira, desgaste das placas e diferentes ângulos de captura, não foram avaliados de forma abrangente neste estudo. Dessa forma, embora os resultados obtidos sejam promissores, a generalização do desempenho do sistema para ambientes reaisrequer validações adicionais com dados mais diversos e em condições operacionais reais.

Outra limitação observada refere-se ao uso de uma ROI fixa, que exige o posicionamento adequado do veículo para garantir a correta captura da placa. Essa abordagem pode reduzir a flexibilidade do sistema em cenários com maior variação de posicionamento ou múltiplas faixas de entrada.

6. TRABALHOS FUTUROS E POSSÍVEIS EVOLUÇÕES DO SISTEMA

Com base nos resultados obtidos e nas limitações identificadas durante o desenvolvimento e os testes do sistema, recomenda-se, como trabalho futuro, a evolução da solução proposta por meio da incorporação de técnicas mais avançadas de visão computacional e arquitetura de software.

Uma das principais melhorias sugeridas consiste na substituição do mecanismo de detecção estática da ROI por modelos de *Deep Learning* voltados à detecção de objetos, como a arquitetura (YOLO - You Only Look Once). A adoção desse tipo de abordagem permitiria a localização automática da placa veicular em qualquer região da imagem, independentemente de seu posicionamento no quadro, aumentando significativamente a acertividade do sistema em cenários reais, com maior variabilidade de ângulo, distância e

iluminação.

Adicionalmente, propõe-se a integração do sistema com câmeras IP externas por meio do protocolo (RTSP - Real Time Streaming Protocol), o que ampliaria sua aplicabilidade para ambientes externos, como portarias de condomínios e estacionamentos abertos, eliminando a dependência exclusiva de dispositivos de captura locais.

Sugere-se o desenvolvimento de uma API REST para permitir a comunicação do sistema com aplicações externas, como aplicativos móveis. Essa funcionalidade possibilitaria que moradores realizassem o cadastro remoto de visitantes e veículos, promovendo maior autonomia, comodidade e escalabilidade à solução, além de facilitar futuras integrações com sistemas de automação predial e plataformas de gestão condominial.

Em suma, o sistema desenvolvido cumpre seu papel acadêmico e prático, oferecendo uma base sólida, auditável e de baixo custo para a modernização dasegurança patrimonial.

REFERÊNCIAS

AHAMED, Istiak; ISMAIL, Gazi Mohammad. **YOLOv8-Based License Plate Recognition for Bangladeshi Vehicles**. ResearchGate, 2025. Disponível em: https://www.researchgate.net/publication/388230973_YOLOv8-Based_License_Plate_Recognition_for_Bangladeshi_Vehicles. Acesso em: 22 fev. 2026

ALVES, Fabricio Henrique Cardoso. **Estudo e desenvolvimento de sistema de reconhecimento de placas veiculares**. 2017. Trabalho de Conclusão de Curso. Disponível em: https://ric.cps.sp.gov.br/bitstream/123456789/24237/1/Fabricio%20Henrique%20Cardoso%20Alves.pdf Acesso em: 28 mar. 2026

ANAGNOSTOPOULOS, C. N. et al. **License plate recognition from still images and video sequences: A survey.** ResearchGate, 2008. Disponível em: https://www.researchgate.net/publication/3428145_License_Plate_Recognition_From_Still_Images_and_Video_Sequences_A_Survey Acesso em: 18 fev. 2026

BOVIK, A. C. **Handbook of image and video processing**. 2. ed. New York: Academic Press, 2020. Acesso em: 21 mai. 2026

BRADSKI, G.; KAEHLER A. L **Learning Opencv: Computer Vision with the Opencv Library**. Sebastopol: O'Reilly Media, 2008.

CASEY, R. G.; LECOLINET, E. A. **A survey of methods and strategies in character segmentation**. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 18, n. 7, p. 690–706, 2015. Disponível em: https://perso.telecom-paristech.fr/elc/papers/pami96.pdf Acesso em: 05 mar. 2026

GONZALEZ, R. C.; WOODS, R. E. **Processamento digital de imagens**. 4. ed. São Paulo: Pearson, 2018.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge: MIT Press, 2016. Disponível em: https://www.deeplearningbook.org/ Acesso em: 20 maio 2026

GRAVES, Alex. Long Short-Term Memory. In: **Supervised Sequence Labelling with Recurrent Neural Networks.** Springer, Berlin, Heidelberg, 2012. Disponível em: https://www.cs.toronto.edu/~graves/preprint.pdf Acesso em: 15 maio 2026

HORA, B. A. ***Detecção e reconhecimento de placas de identificação veicular com uso de técnicas de visão computacional aplicadas à fiscalização de trânsito.*** Trabalho de Conclusão de Curso (Engenharia de Computação) – Universidade Federal do Pará (UFPA), Belém, 2024. Disponível em: https://www.even3.com.br/anais/simtec2024/918911-aplicacao-de-tecnicas-de-visao-computacional-na-deteccao-e-reconhecimento-de-placas-de--identificacao-veicular--u/ Acesso em: 20 Jan. 2026

LAROCA, R. et al. **A robust real-time system for license plate recognition in**

unconstrained scenarios. *In: 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2018. Disponível em: https://homepages.dcc.ufmg.br/~william/papers/paper_2018_IJCNN_Laroca.pdf Acesso em: 20 fev. 2026

LUTZ, M**. Learning Python**. 5. ed. Sebastopol: O'Reilly Media, 2013.

MCKINSEY & COMPANY. Global survey: The state of AI in 2021. New York: McKinsey & Company, 2021. Disponível em: https://www.mckinsey.com/capabilities/quantumblack/our-insights/global-survey-the-state-of-ai-in-2021. Acesso em: 17 jun. 2026.

ONU-HABITAT. World Cities Report 2022: Envisaging the Future of Cities. Nairobi: United Nations Human Settlements Programme, 2022. Disponível em: https://unhabitat.org/world-cities-report-2022-envisaging-the-future-of-cities?utm_source=chatgpt.com Acesso em: 10 abr. 2026

OPENCV. OpenCV Documentation: **Image Filtering**. Disponível em: https://docs.opencv.org/. Acesso em: 19 fev. 2026

OPENCV. OpenCV documentation: **OCR and computer vision**. 2025. Disponível em: https://opencv.org. Acesso em: 18 abr. 2026

PRATT, W. K. **Digital image processing: PIKS scientific inside**. 4. ed. New York: Wiley-Interscience, 2007. Disponível em: https://nana.lecturer.pens.ac.id/index_files/referensi/image_processing/Digital%20Image%20Processing.pdf Acesso em: 26 jun. 2026

REDMON, J. et al. **You Only Look Once: Unified, real-time object detection.** In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, 2016. p. 779–788. Disponivel em: https://ieeexplore.ieee.org/document/7780460 Acesso em: 02 jun. 2026

SATYANARAYANAN, Mahadev. **The emergence of edge computing. Computer**, v. 50, n. 1, p. 30–39, 2017. Disponivel em: https://elijah.cs.cmu.edu/DOCS/satya-edge2016.pdf: Acesso em: 22 jun. 2026

SMITH, R. An overview of the Tesseract OCR engine. In: **Proceedings of the Ninth International Conference on Document Analysis and Recognition**. Curitiba, 2007. p. 629–633. Disponível em: https://ieeexplore.ieee.org/document/4378659 Acesso em: 18 abr. 2026

SONKA, M.; HLAVAC, V.; BOYLE, R. **Image processing, analysis, and machine vision**. 4. ed. Boston: Cengage Learning, 2014. Disponível em: https://archive.org/details/imageprocessinga0000sonk_u5x5/mode/2up Acesso em: 10 abr. 2026

SQLite. **SQLite Documentation**. Disponível em: https://www.sqlite.org/docs.html. Acesso em: 17 abr. 2026

STALLINGS, W. **Criptografia e segurança de redes: princípios e práticas**. 7.ed. São Paulo: Pearson, 2018. Disponível em: https://www.kufunda.net/publicdocs/Criptografia%20e%20Seguran%C3%A7a%20de%20Redes%20-%206%C2%AA%20Ed.%202014.pdf Acesso em: 09 dez. 2025

TESSERACT-OCR. **Tesseract Open Source OCR Engine**. GitHub Repository. Disponível em: https://github.com/tesseract-ocr/tesseract. Acesso em: 03 dez. 2025

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition**. 4th ed. Academic Press, 2008.