Centro Federal de Educação Tecnológica de Minas Gerais
Campus Divinópolis
Graduação em Engenharia Mecatrônica

Bernardo Michel Slailati

DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO DE PLACAS VEICULARES PARA CONTROLE DE ACESSO EM CONDOMÍNIOS

Divinópolis
2021

Bernardo Michel Slailati

DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO DE PLACAS VEICULARES PARA CONTROLE DE ACESSO EM CONDOMÍNIOS

Monografia de Trabalho de Conclusão de Curso apresentada à comissão avaliadora do curso de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Computação e Controle.

Orientador: Prof. Dr. Thiago Magela Rodrigues Dias

Divinópolis
2021

Bernardo Michel Slailati

DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO DE PLACAS VEICULARES PARA CONTROLE DE ACESSO EM CONDOMÍNIOS

Monografia de Trabalho de Conclusão de Curso apresentada à comissão avaliadora do curso de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Computação e Controle.

Comissão Avaliadora:

Prof. Raulivan Rodrigo da Silva
CEFET/MG *Campus* V

Prof. Dr. Thiago Magela Rodrigues Dias
CEFET/MG *Campus* V

Prof. Dr. Christian Gonçalves Herrera
CEFET/MG *Campus* V

Divinópolis
2021

Dedico este trabalho a todos aqueles que de alguma forma contribuíram para a minha formação ao longo dessa árdua jornada.

Agradecimentos

Agradeço,

primeiramente a todos os meus familiares, que estiveram ao meu lado me dando forças para seguir meus passos durante cada etapa a ser enfrentada. Principalmente minha mãe, Silvana, minha tia, Soraya e, em especial, minha avó, Ivone, os principais exemplos de dedicação, superação e coragem que tive na vida. Saibam que, onde eu estiver, levarei vocês sempre comigo no coração. Também agradeço muito à minha namorada, Lorena, por me apoiar tanto e torcer sempre por mim, aos meus amigos, por me incentivarem em diversos momentos, colegas de faculdade que, sem dúvida, foram fundamentais inúmeras vezes nesta trajetória, funcionários e professores da instituição CEFETMG - *Campus V* pelo trabalho árduo executado de formação e bem estar dos alunos, ao longo de todos esses anos.

Descobrir consiste em olhar para o que todo mundo está vendo e pensar uma coisa diferente.

Roger Von Oech

Resumo

O projeto desenvolvido consiste de um sistema embarcado de reconhecimento de placas veiculares para condomínios verticais (prédios, os chamados "condomínios de edifícios") e horizontais (também conhecidos como "condomínios residenciais"), associado ao controle de acessos de veículos nesses ambientes. No Brasil, muito desses locais não possuem tecnologias que contribuem com o serviço de monitoramento, devido principalmente ao alto custo de implantação e manutenção. Busca-se então, com a aplicação desse sistema, aumentar a segurança para moradores e funcionários, através do registro detalhado do fluxo de veículos no local, em tempo real. Para isso, são fornecidos ao funcionário responsável pelo controle de acessos, informações importantes sobre o veículo e seu respectivo proprietário, após serem identificados de forma correta os caracteres da placa, permitindo conceder de forma segura a permissão de entrada ao indivíduo no ambiente. A estrutura completa do sistema elaborado, consiste basicamente de um *hardware* próprio para aquisição das imagens e processamento do programa criado, algoritmos de tratamento digital e identificação de caracteres em imagens, interface gráfica intuitiva para garantir uma boa experiência por parte do usuário e abstração de códigos para manipular banco de dados, visando gerenciar cadastros de moradores, veículos e registros de entradas. Dessa forma, esse projeto engloba, principalmente, duas das principais áreas da Engenharia Mecatrônica: Controle e Computação. Os componentes físicos a serem utilizados no *hardware* são um módulo de câmera para a captura de imagens e um computador de tamanho reduzido para processamento do *software* e condicionamento dos sinais, advindos de periféricos a serem relacionados. Resulta-se deste projeto, um sistema eficaz que promove uma solução para o problema relacionado, apresentando um mecanismo de identificação funcional de placas e caracteres, atrelado a um custo acessível de produto final, aumentando a segurança em áreas condominiais a médio e longo prazo.

Palavras-chave: Processamento Digital de Imagens, Identificação de Placas Veiculares, Reconhecimento de Caracteres, Controle de Acesso Veicular.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos e Notação

GPU	Graphics Processing Unit (unidade de processamento gráfico)
MIT	Massachusetts Institute of Technology (Instituto de Tecnologia de Massachusetts)
RGB	Red, Green and Blue (vermelho, verde e azul)
CNN	Convolutional Neural Network (rede neural de convolução)
$UART$	Universal Asynchrounous Receiver/Transmiter (receptor/transmissor assíncrono universal)
USB	Universal Serial Bus (porta universal serial)
RAM	Random Access Memory (memória de acesso randômico)
$HDMI$	High-Definition Multimedia Interface (interface multimídia de alta resolução)
SD	Secure Digital (segurança digital)
CSI	Camera Serial Interface (interface serial de câmera)
DSI	Display Serial Interface (interface serial de tela)
BLE	Bluetooth Low Energy (Bluetooth de baixo consumo energético)
SSH	Secure Shell (protocolo de execução de comandos em um computador remoto através da rede)
VNC	Virtual Network Computing (protocolo de internet que permite a visualização de interfaces gráficas remotas através de uma conexão segura)
OCR	Optical Character Recognition (reconhecimento ótico de caracteres)
TIF	Tagged Image Format (formato de imagem por marcação)
GUI	Graphical User Interface (interface gráfica do usuário)

Capítulo 1

Introdução

O tema segurança vem sendo discutido cada vez mais em todo o mundo e influencia diretamente o cotidiano das populações de diversos países. No quesito violência urbana, atualmente o Brasil se apresenta como o quarto país no mundo onde as pessoas se sentem mais inseguras, dentre 142 nações analisadas (GALLUP, 2018). Nesse ranking, o Brasil só fica atrás de Gabão, Afeganistão e Venezuela, países que enfrentam crises sociais e econômicas graves. Nota-se que esse percentual aumentou de forma significativa nos últimos dois anos, sendo que em 2015 o Brasil não estava sequer entre os dez mais inseguros.

Neste contexto, em 2018, apenas no estado de São Paulo, o número de roubos e furtos a condomínios cresceu 56% se comparado ao ano anterior, segundo pesquisa da Secretaria da Segurança Pública (SSP) e Lei de Acesso à Informação (G1, 2018). Foram registrados 1.300 crimes (roubos e furtos) realizados em prédios entre janeiro e abril deste ano e 832 no mesmo período do ano anterior. Com isso, constata-se a necessidade de se controlar o acesso de indivíduos nesses tipos de locais, evitando assim a entrada de desconhecidos e inibindo crimes como os citados anteriormente em ambientes residenciais.

Logo, considerando os avanços tecnológicos atuais, técnicas de visão computacional surgem como um excelente mecanismo de auxílio para soluções de problemas como o citado anteriormente. Segundo BRADSKI; KAEHLER (2008), a visão computacional é a transformação de dados de uma câmera estática ou de vídeo em uma decisão ou nova representação, sendo extremamente aplicável em projetos de mecanismos autônomos que necessitam realizar escolhas baseadas na análise de imagens. Com isso, essa tecnologia acaba por se relacionar com o tema segurança de forma incontestável, se mostrando como uma potente ferramenta a ser utilizada em sistemas avançados de vigilância.

Tendo em vista o quadro citado acima, fica evidente a necessidade de soluções que auxiliem e promovam o aumento da segurança, sendo esta a principal motivação para a proposta abordada neste projeto. O sistema a ser desenvolvido se mostra como uma interface computacional de fornecimento de informações, sobre os moradores e seus veículos, a

partir da identificação das respectivas placas veiculares que solicitam a entrada no local. A interação usuário-máquina será conferida por meio de um *software* a ser instalado em computadores, se comunicando de forma direta a uma câmera e outros periféricos (monitor, teclado e mouse, por exemplo), que concederão o controle das funcionalidades por parte do usuário. O esquemático da utilização deste sistema, pode ser visto na Figura 1.1.

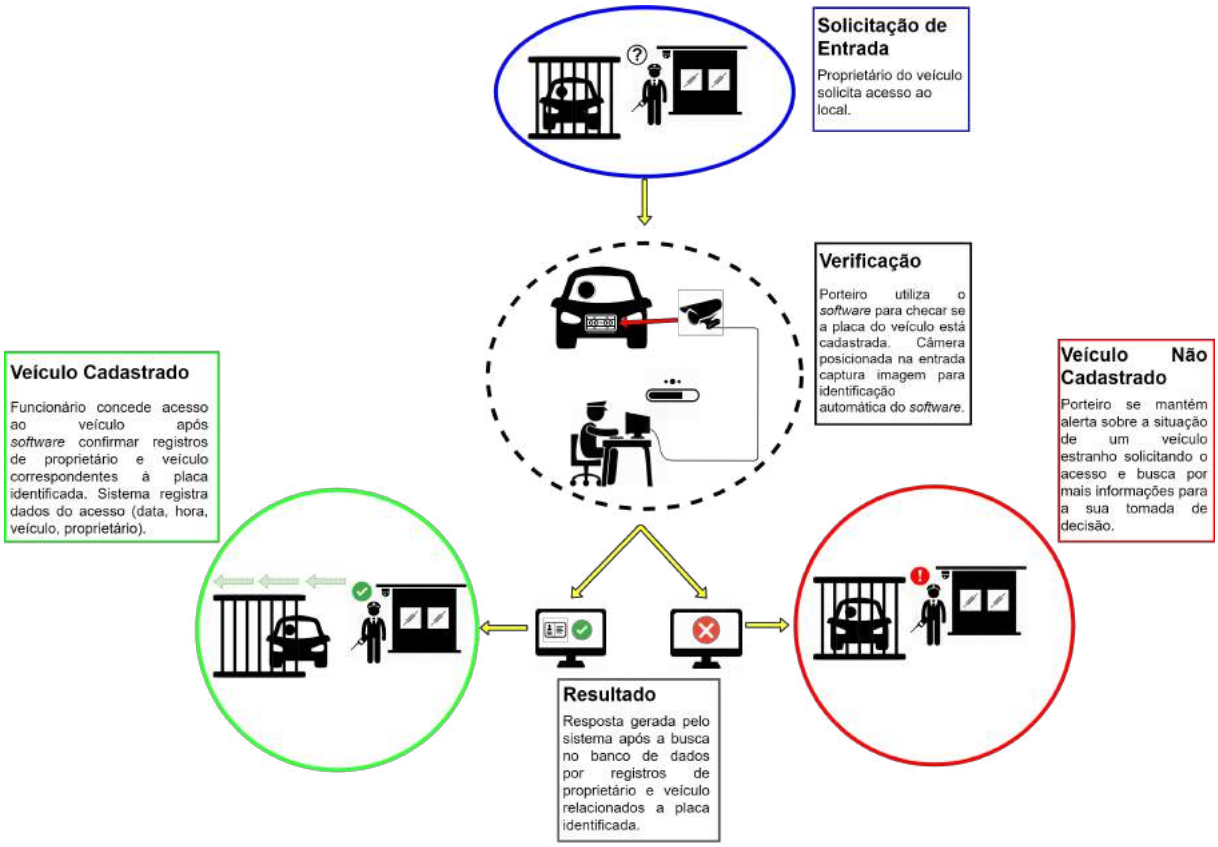


Figura 1.1: Demonstrativo de funcionamento do sistema

O intuito será garantir uma melhor tomada de decisão por parte do funcionário relacionado à portaria do condomínio, além de um registro completo da entrada de veículos no local, tendo em vista que todas as verificações serão armazenadas computacionalmente, contribuindo assim com o aumento da segurança no ambiente.

1.1 Definição do Problema

Como citado anteriormente, os níveis de insegurança no Brasil vem aumentando muito ao longo dos anos. Pesquisas apontam resultados alarmantes sobre o quão inseguro o brasileiro se sente morando no país. Furtos e roubos em condomínios, mais especificamente, são alguns dos índices que influenciam diretamente nesses números. Um fato preponderante para tal situação, está relacionado a falta de controle do acesso a esses ambientes. Sendo assim, essa problemática se apresenta como a escolhida para ser tratada neste

trabalho.

1.2 Motivação

Após ficar evidente a necessidade de soluções que auxiliem e promovam o aumento da segurança em ambientes residenciais no cenário atual, esta se torna a principal motivação para o tema abordado neste trabalho. Além disso, sistemas completos como o proposto, podem ser encontrados hoje para compra a preços relativamente altos - em torno de algumas dezenas de milhares de reais -, sendo este também um incetivo para o desenvolvimento de um projeto que, futuramente, possa gerar uma versão de produto final de valor reduzido, se comparado aos semelhantes disponíveis no mercado atualmente.

Ademais, por abordar conceitos atuais da engenharia, as tecnologias a serem utilizadas neste trabalho se mostram condizentes com o grande desenvolvimento tecnológico visto ao redor do mundo nos últimos anos. Logo, o projeto pode ser visto como uma abordagem com grande potencial mercadológico, fornecendo também ao aluno percepções importantes, tanto na área acadêmica quanto profissional, sobre áreas que se mostram de grande interesse para o mesmo.

1.3 Objetivos do Trabalho

O projeto proposto tem como objetivo principal fornecer ao funcionário responsável pelo controle de acesso em portarias de condomínios verticais e horizontais, as características do veículo posicionado na entrada do ambiente, relacionando o mesmo ao seu respectivo proprietário. Almeja-se a criação de um sistema eficiente, que apresente resultados semelhantes aos da literatura baseada.

Busca-se realizar o procedimento de identificação de placas veiculares de forma rápida e eficaz, apresentando as informações consideradas relevantes em uma interface computacional amigável a ser utilizada pelo usuário do sistema. Vale ressaltar que a ferramenta a ser criada será voltada apenas para contribuir com uma melhor tomada de decisão por parte do funcionário, por isso todas as funcionalidades do sistema poderão ser realizadas independentemente do resultado dos algoritmos de identificação de placas. Portanto, a fim de se obter tais resultados, o projeto será divido para atender os seguintes objetivos específicos:

- Pesquisar viabilidade de mercado e estudo de caso para a situação a ser abordada (segurança em ambientes condominiais), com o intuito de adquirir maiores conhecimentos sobre o problema a ser resolvido;

- Realizar estudos sobre as técnicas existentes para identificação de placas veiculares (formas retangulares específicas) e caracteres próprios contidos em imagens;
- Desenvolver protótipo físico eficaz para embarque dos componentes de *hardware*;
- Desenvolver algoritmos de reconhecimento das placas, identificação dos caracteres, interface para controle das funcionalidades do sistema e gerenciamento e estruturação dos bancos de dados;
- Executar testes visando obter a porcentagem de acertos dos algoritmos de reconhecimento e realizar prováveis correções, a fim de conceder maior precisão e exatidão aos resultados;
- Desenvolver *software* integrado que caracterize o sistema, relacionando os algoritmos de identificação e gerenciamento de bancos de dados à interface gráfica a ser disponibilizada para o usuário.

1.4 Estado da Arte

O aumento acelerado da capacidade de processamento computacional se mostra como o principal precursor da expansão de trabalhos utilizando técnicas de visão computacional. Nos últimos dez anos, algumas das áreas que se destacam por receberem pesquisas envolvendo esses métodos são: biomédica, automotiva, robótica, agricultura e industrial. Alguns exemplos de estudos recentes realizados nessas áreas, podem ser vistos em KIL *et al.* (2017), TSAI *et al.* (2011), KUMAR *et al.* (2018), SANTHI *et al.* (2017) e JIANG (2016), respectivamente.

No Brasil, a FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) contribui com o avanço em pesquisas desse tipo, dedicando um seção exclusiva de sua biblioteca virtual para informar sobre bolsas disponíveis com o tema relacionado. Atualmente, se destaca uma que, em apoio com a Microsoft Brasil, fornece um valor que pode chegar a R$4.5 milhões de reais ao longo de quatro anos - com início em 2018 - voltados para propostas de soluções em mobilidade urbana, infraestrutura e segurança pública e industrial, aliando visão computacional e Inteligência Artificial (FAPESP, 2017).

Aplicada na identificação de placas veiculares, a visão computacional apresenta como estado atual, a elaboração de projetos como os descritos em LIN; LIN; LIU (2018), FENG; LI; PANG (2018) e BJÖRKLUND *et al.* (2019). Em LIN; LIN; LIU (2018), um eficiente sistema de reconhecimento é desenvolvido, adotando a metodologia de rede neural de convolução para aprimorar o reconhecimento de caracteres principalmente em imagens desfocadas e obscuras, resultando na precisão de 99.2% de acertos (desempenho superior se comparado a sistemas tradicionais de reconhecimento de placas). Já em FENG; LI;

PANG (2018), também utiliza-se um sistema que conta com uma rede neural treinada para identificação dos caracteres obtidos, porém esta segue a tipologia de retropropagação e possui um índice de 99% de identificações corretas das placas (conjunto de testes de cem amostras, onde veículos se encontravam em diferentes cenários). Por fim, o projeto visto em BJÖRKLUND *et al.* (2019) se assemelha ao desenvolvido em FENG; LI; PANG (2018), por utilizar a mesma tipologia de rede neural para reconhecimento de caracteres, mas com um nível de complexidade maior: outra rede neural convolucional treinada é aplicada para identificação dos contornos das placas.

Comercialmente, em território nacional, se destaca o desenvolvimento de projetos da *startup* SVA Tech. Essa empresa emergente está entre as três melhores *startups* no mundo na categoria "*Computer Vision*", segundo "*Ranking 100 Open Startups 2018*", que consiste da avaliação anual das *startups* mais atraentes de acordo com o mercado corporativo brasileiro (DIGITAL SECURITY, 2018). As soluções tecnológicas propostas pela mesma, se baseiam em inteligência artificial e *deep learning* para monitoramento inteligente de segurança e controle de acesso veicular. Além disso, um sistema como o abordado neste trabalho que se mostra disponível no mercado brasileiro é o *GV-LPR*. O mesmo é descrito como um sistema de controle com reconhecimento automático de placas de veículos (conjunto *software* e câmera), e, segundo dados do fabricante (GEOVISION, 2019), apresenta uma taxa de reconhecimento de 99% em melhor desempenho, velocidade de reconhecimento menor que 0.2 segundos, armazenamento de até dois milhões de imagens de placas, função de contagem de veículos que contribui para o controle do tráfego no estacionamento local, entre outras características relevantes.

1.5 Metodologia

Nesta seção, são descritos os principais marcos metodológicos a serem contemplados durante a realização deste projeto, conforme detalhado na Figura 1.2.

O início da realização deste projeto, abrangeu a compreensão completa da problemática tratada, sendo esta de extrema importância durante a primeira parte do trabalho. Buscou-se informações importantes acerca do assunto em pesquisas, índices, estudos e reportagens sobre o tema segurança em geral e aplicado no âmbito residencial de condomínios verticais e horizontais, foco da solução proposta. Essa, ao ser definida pelo autor, passou por uma revisão de literatura, sendo o segundo passo dado no processo de desenvolvimento do projeto. Para isso, a visão computacional foi contextualizada historicamente, desde sua origem até o momento presente, sendo abrangidos os principais métodos utilizados em cada época, as circunstâncias as quais estão sendo relacionadas atualmente e outras considerações relevantes sobre essa tecnologia, foco principal do tema em questão.

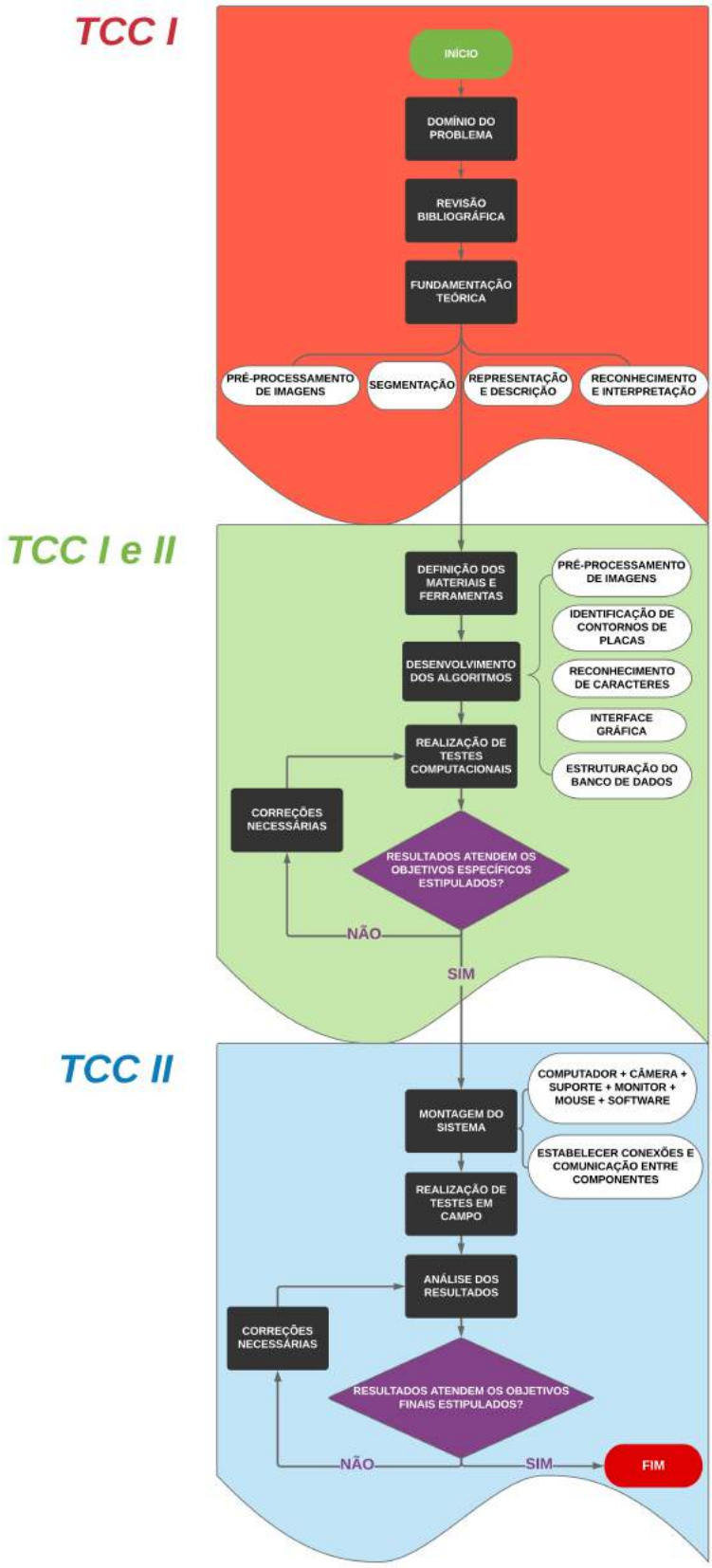


Figura 1.2: Fluxograma metodológico aplicado ao projeto contendo as atividades a serem feitas ao longo das disciplinas de TCCI e TCCII.

Em seguida, fundamentou-se teoricamente os principais conceitos metodológicos abrangidos, a fim de validar os algoritmos aplicados durante todo o desenvolvimento do trabalho. Juntos, esses conhecimentos forneceram ao projeto uma base sólida para a criação de sistemas como o proposto, englobando as bases científica e tecnológica necessárias. Alguns dos subtópicos descritos foram: tipos de caracterização de imagens digitais, filtros digitais para aprimoramento visual, algoritmos de detecção de bordas e contornos, dentre outros.

Feito isso, o foco se voltou para a definição dos componentes físicos destinados ao *hardware*, que processam o *software* integrado de reconhecimento e interface gráfica, constituindo a estrutura do sistema embarcado. Alguns dos recursos escolhidos, que caracterizam sistemas como o proposto, são: um computador compacto de baixo custo (modelo *Raspberry Pi*), módulo de câmera específico, suporte personalizado para posicionamento de componentes, *mouse*, teclado e monitor, direcionados ao interfaceamento com o usuário. A comunicação estabelecida entre todo o sistema, garante a funcionalidade em campo dos algoritmos computacionais, podendo ser agregados outros elementos mecânicos e eletrônicos com a evolução do projeto.

A próxima etapa elaborada, consistiu de uma das principais para o funcionamento efetivo do sistema: a concepção dos algoritmos computacionais. Para isso, foram escolhidas a linguagem de programação *Python* - devido a grande disponibilidade de conteúdos relacionados - e, principalmente, as bibliotecas *OpenCV*, *Tesseract* e *Tkinter*, voltadas para aplicações de visão computacional, reconhecimento de caracteres em imagens e criação de interface gráfica, respectivamente. Foram fundamentais o desenvolvimento de algoritmos de pré-processamento de imagens digitais, identificação de contornos de placas em imagens de veículos em diferentes cenários, reconhecimento de caracteres, estruturação e manipulação de banco de dados e criação de interface gráfica, para interação do usuário com o sistema.

Por fim, ao serem determinados os algoritmos e componentes do sistema embarcado, após a realização de testes oportunos, a montagem do *hardware* foi estabelecida em seu formato final. Ao ser aplicado em um ambiente propício, visto como ideal para uso, analisou-se seu comportamento e os resultados gerados, sendo necessárias algumas modificações posteriores para otimização de seu desempenho. Almejou-se e alcançou-se, com isso, o atendimento dos objetivos previstos inicialmente, aferindo respostas com um índice de acertos considerado razoável para a solução proposta. Na Figura 1.2 a seguir, ilustra-se o fluxograma referente as atividades desempenhadas pelo autor durante as disciplinas de Trabalho de Conclusão de Curso I e II.

Capítulo 2

Fundamentos

No início deste capítulo, a tecnologia aplicada no trabalho é contextualizada historicamente, apresentando as principais características de seu progresso ao longo dos anos. Logo em seguida, descreve-se a metodologia aplicada no projeto, detalhando e ilustrando as etapas a serem concluídas durante o andamento do mesmo. Na sequência, são fundamentados teoricamente os conceitos considerados importantes relacionados ao trabalho de maneira geral.

2.1 Revisão de Literatura

A técnica de visão computacional consiste da extração, análise e compreensão automáticas de informações úteis a partir de uma única imagem ou sequência de imagens, envolvendo o desenvolvimento de uma base teórica e algorítmica, para alcançar uma compreensão visual automática (BMVA, 2019). Os primeiros estudos dessa técnica se deram em meados dos anos 60, com o objetivo de identificar estruturas de objetos tridimensionais através do método de extração de bordas contidas em cenários bidimensionais, com ROBERTS (1965), comumente aceito como o pai da visão computacional. Posteriormente, ainda neste período, essa metodologia recebeu outros trabalhos relacionados importantes, como os de DAVIS (1975) e BARROW *et al.* (1978).

De acordo com documentos oficiais, em 1966, Marvin Minsky, co-fundador do laboratório de inteligência do *MIT* (Instituto de Tecnologia de Massachusetts), pediu a um de seus alunos de graduação, Gerald Jay Sussman, que "passasse o verão ligando uma câmera a um computador, com o intuito de que o computador conseguisse, por fim, descrever o que viu", citado por BODEN (2008), segundo SZELISKI (2010), sendo mais um dos primeiros indícios de pesquisa na área. Outro marco ocorrido também nesta época, foi o trabalho feito por FISCHLER; ELSCHLAGER (1973), descrito como a análise de arranjos elásticos de partes de estruturas pictóricas (Figura 2.1). Segundo os próprios autores, este método "oferece uma combinação de esquema de descrição e métrica de decisão

geral, intuitivamente satisfatória e que levou a resultados experimentais promissores".

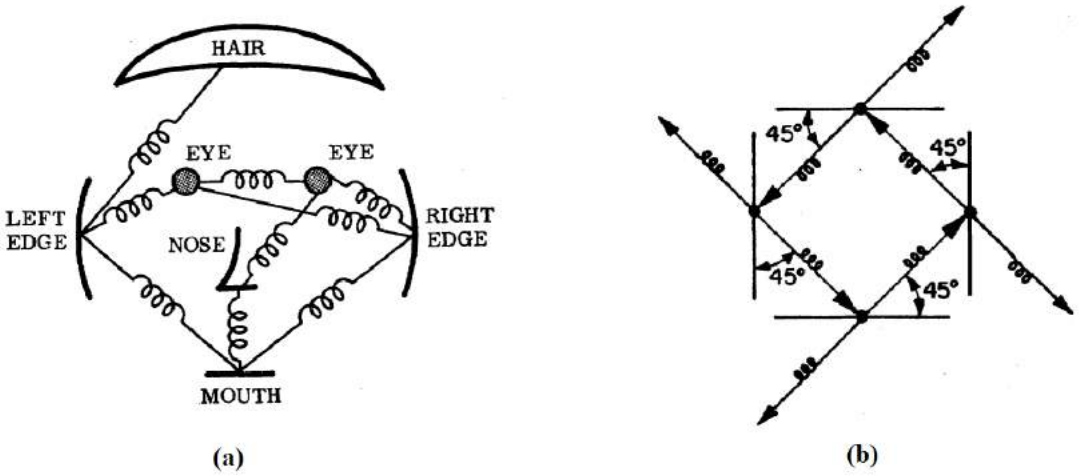


Figura 2.1: Descrição por arranjos elásticos das referências de uma face (a) e um quadrado (b) (Fonte: Fischler e Elschlager, 1973).

Nos anos 80, o foco dos estudos nessa área se voltaram para técnicas matemáticas mais sofisticadas, a fim de realizar análises quantitativas de imagens e cenas (SZELISKI, 2010). Um processo amplamente aplicado na época foi o de pirâmides de imagem, que "constituem um mecanismo bastante utilizado para realizar a decomposição de uma imagem em diferentes níveis de resolução"(VAQUERO, 2004). Esse método pode ser visto em diversos trabalhos dessa época, como os de ROSENFELD (1980), ROSENFELD (1984) e BURT; ADELSON (1983)). Pesquisas voltadas para a otimização da detecção de bordas e contornos em imagens digitais também marcaram esse período, como JOHN (1986) e NALWA; BINFORD (1986), sendo muito utilizados até os dias de hoje. Além disso, o processamento de imagens tridimensionais sofreu um avanço considerável, com trabalhos como os de BESL; JAIN (1985) e FAUGERAS; HEBERT (1986), continuando a crescer significativamente ao longo dos anos, devido, principalmente, ao aumento da capacidade de processamento computacional das máquinas.

Na década de 90, a interação com computação gráfica - especialmente no setor multidisciplinar de modelagem e renderização baseada em imagens, vistos em BEARDSLEY; TORR; ZISSERMAN (1996) e DEBEVEC; TAYLOR; MALIK (1996) -, representou o campo de maior desenvolvimento dentro da visão computacional, segundo SZELISKI (2010). No final deste período, iniciou-se o projeto de uma biblioteca computacional multiplataforma revolucionária para a área, livre para utilização tanto no meio acadêmico quanto comercial, a *OpenCV*. Voltada para o desenvolvimento de aplicativos gerais, hoje a biblioteca integra mais de 2.500 algoritmos otimizados, que incluem um conjunto abrangente de técnicas de visão computacional e de aprendizado de máquina, clássicos

e de última geração (OPENCV, 2019). O projeto iniciou-se em 1999 pela Intel, sendo lançado em 2000 a versão *alpha* e em 2006 a primeira versão oficial do programa (Figura 2.2). No momento, a mesma se encontra disponível na versão 4.1.0.

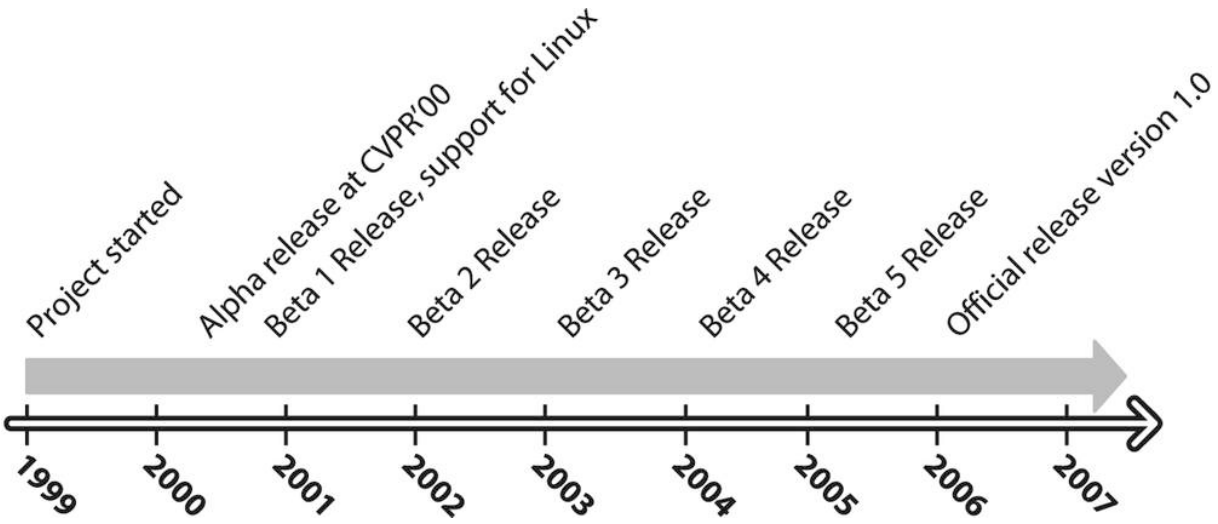


Figura 2.2: Linha do tempo da biblioteca OpenCV, do início do projeto até o lançamento da primeira versão oficial (Fonte: Kaehler e Bradski, 2008).

A tendência final, que agora domina grande parte das pesquisas de reconhecimento visual, é a aplicação de sofisticadas técnicas de aprendizado de máquina para problemas relacionados a esse campo (SZELISKI, 2010), devido a introdução da unidade de processamento gráfico ao mercado, juntamente a grande quantidade de informações disponíveis na Internet. A GPU normalmente transforma números em imagens, porém pode realizar a tarefa reversa, processando imagens e as convertendo em um conjunto números (FUNG; MANN, 2008). Esse procedimento é constantemente aplicado a visão computacional de alto desempenho atual, se relacionando muitas vezes a algoritmos de redes neurais e *deep learning*, que consiste de uma classe de técnicas de aprendizado de máquina que são usadas na extração de padrões em conjuntos de dados. NISHANI; ÇIÇO (2017) se destacam nesse aspecto, apresentando o estudo de três diretivas para otimizar o desempenho da estimativa de poses humanas através de CNN e *deep learning*.

2.2 Fundamentação Teórica

2.2.1 Imagem Digital

Uma imagem digital (ou computacional) consiste, segundo Nixon e Aguado (2002), de uma matriz bidimensional de *pixels*, sendo este sempre o menor ponto que forma qualquer imagem deste tipo. O valor de cada *pixel* é definido de acordo com o brilho correspondente a região de uma determinada cena, de forma diretamente proporcional. Portanto, uma

imagem digital pode ser definida comumente como um conjunto de $M \times N$ *pixels* de resolução *n-bits*. Os valores de M e N representam as quantidades de *pixels* das linhas e colunas da matriz, respectivamente, e n estabelece os limites dos valores de brilho, de 0 a $2^n - 1$.

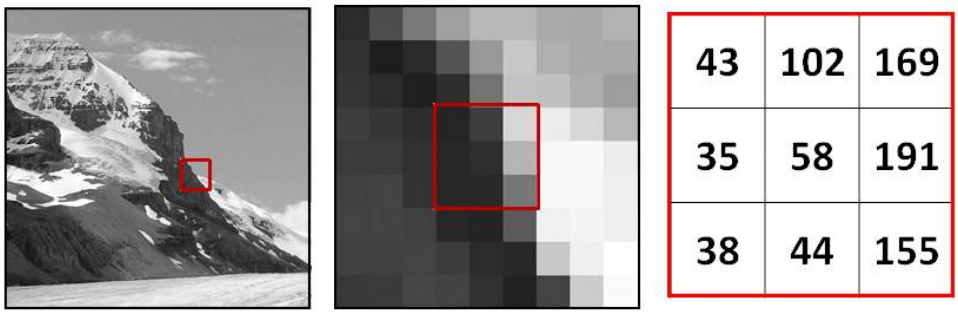


Figura 2.3: Representação dos valores dos *pixels* de uma imagem digital recortada de tamanho 3x3 e resolução 8-bits (Fonte: Humboldt State University[2]).

Para a conversão de uma cena real em uma imagem digitalizada, são obrigatoriamente necessários dois processos: a aquisição da imagem e sua posterior digitalização (MARQUES FILHO; NETO, 1999), como apresentado na Figura 2.4. A aquisição da imagem consiste da transformação de uma cena tridimensional (a e b) em uma imagem analógica bidimensional (e), através de um equipamento específico (c), como uma câmera fotográfica ou de vídeo. O sinal analógico obtido como saída do dispositivo (d), deve ser submetido a uma discretização espacial e em amplitude, que caracterizam a digitalização, para fornecer o formato de imagem desejável ao processamento computacional relacionado. "Do ponto de vista eletrônico, a digitalização consiste em uma conversão analógico-digital na qual o número de amostras do sinal contínuo por unidade de tempo indica a taxa de amostragem e o número de bits do conversor A/D utilizado determina o número de tons de cinza resultantes na imagem digitalizada.", (MARQUES FILHO; NETO, 1999, pg. 22).

2.2.2 Escala Cinza

Imagens monocromáticas - ou em escala cinza - são aquelas descritas matematicamente por uma função $f(x, y)$ de uma banda de frequência única, que, segundo MARQUES FILHO; NETO (1999), representa a intensidade luminosa a partir de valores proporcionais ao brilho em um ponto de coordenadas espaciais qualquer (x, y). Essa função corresponde, ainda segundo os autores, ao produto entre a iluminância $i(x,y)$ (quantidade de luz que incide diretamente sobre o objeto) e refletância $r(x,y)$ (fração de luz incidente que o objeto

[2]Disponível em: <http://gsp.humboldt.edu/OLM_2016/Courses/GSP_216_Online/lesson3-1/raster-models.html>. Acesso em: 02 mai. 2019.

[4]Disponível em: <https://fabiorogeriosj.com.br/2016/09/25/Sensores-e-aquisicao-de-imagens/>. Acesso em: 02 mai. 2019.

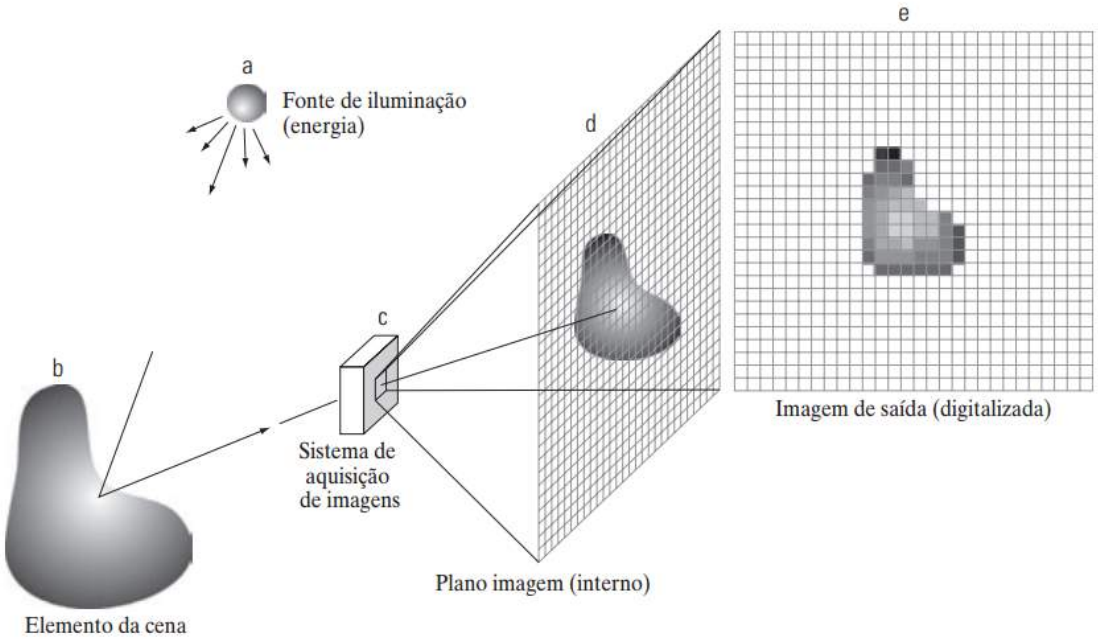


Figura 2.4: Exemplo de um processo completo de digitalização de imagem (Fonte: Fábio Rogério SJ[4].)

vai transmitir ou refletir ao ponto (x,y)) próprias do objeto contido no cenário analisado, exemplificado na Figura 2.5.

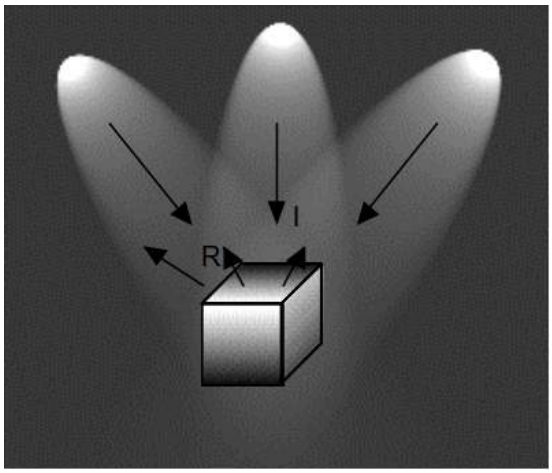


Figura 2.5: Componentes de iluminância (I) e refletância (R) em uma imagem (Fonte: Marques Neto e Filho, 1999).

Matematicamente, essa expressão pode ser descrita da seguinte forma:

$$f(x, y) = i(x, y) \cdot r(x, y)$$

As condições de validez dessa expressão são $0 < i(x, y) < \infty$ e $0 < r(x, y) < 1$.

Basicamente, esta escala fornece tons variados de cinza dentre a faixa de valores dos *pixels*, limitados pela resolução de n *bits*, variando sempre entre o preto e o branco (sim-

bolizadas pelas grandezas mínima de 0 e máxima de $2^n - 1$, respectivamente). De acordo com o valor de n, a imagem pode apresentar diferentes níveis de nitidez, podendo influenciar assim na análise do conteúdo gráfico. A Figura 2.6 mostra uma mesma cena exibida em diferentes resoluções de escala cinza, sendo notável a diferença na nitidez entre as mesmas.

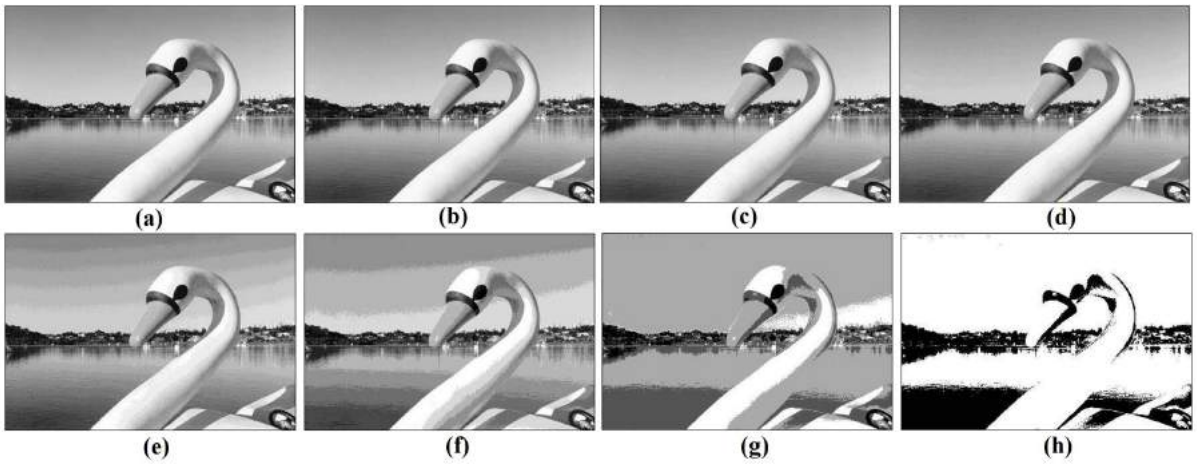


Figura 2.6: Efeito de diferentes resoluções de escala de cinza. Valores de n, em *bits*: (a) 256, (b) 128, (c) 64, (d) 32, (e) 16, (f) 8, (g) 4, e (h) 2 (Fonte: Marques Neto e Filho, 1999, p. 24).

2.2.3 Binarização

Imagens binárias possuem apenas dois valores possíves de "níveis de cinza"para os *pixels*, 0 ou 1, e, portanto, são representadas usando apenas um *bit* de resolução BOVIK (2009). Com isso, a obtenção desse formato pode ser gerado usando operadores lógicos rápidos, que exigem baixo custo de processamento computacional. Utiliza-se esse tipo de imagem, ainda segundo o autor, para abstrair informações importantes de cenas como localização e limites de objetos e presença ou ausência de alguma propriedade específica de imagem. Ao ser exibida digitalmente, essas imagens apresentam apenas duas cores possíveis: preto, valor 0, ou branco, 1. A Figura 2.7 demonstra a aplicação do processo de binarização em uma imagem de gradiente em escala cinza, destacando com ênfase a divisão do cenário em duas porções bem definidas.

2.2.4 Modelo *RGB*

A sigla *RGB* representa, respectivamente, as três cores aditivas primárias: vermelho (*red*), verde (*green*) e azul (*blue*). Imagens digitais no modelo *RGB* apresentam uma estrutura tricromática, composta pela sobreposição de três matrizes de *pixels* que reproduzem cada uma dessas componentes de cor, exemplificada pela Figura 2.8. A justaposição

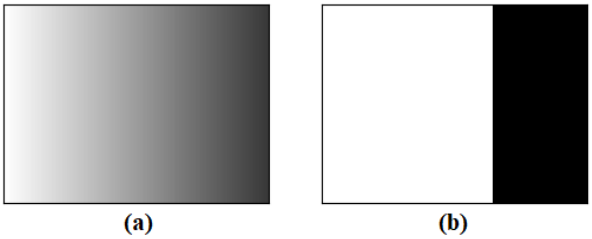


Figura 2.7: Comparação entre a imagem de um gradiente em escala cinza (a) e sua versão binarizada (b).

desses *pixels* propicia visualmente um largo espectro cromático, proporcional a resolução de seus valores. Normalmente, segundo GONZALEZ C. RAFAEL (2000), o termo "imagem totalmente colorida", representa uma imagem no modelo *RGB* com *24-bits* de resolução, garantindo uma variedade de $(2^8)^3$ cores a serem possivelmente representadas, ou 16.777.216.

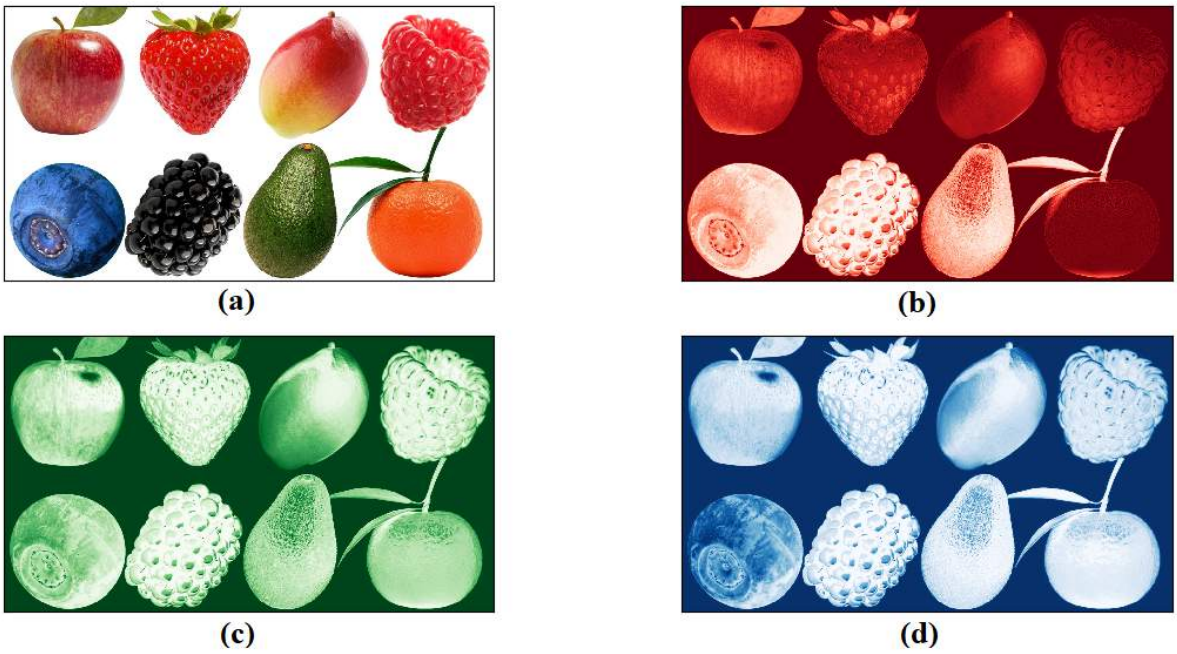


Figura 2.8: Exemplo de imagem em modelo *RGB* (a) e suas respectivas componentes matriciais de cores: (b) *red*, (c) *green* e (d) *blue*.

"O modelo *RGB* é baseado em um sistema de coordenadas cartesianas, que pode ser visto como um cubo onde três de seus vértices são as cores primárias, outros três as cores secundárias, o vértice junto à origem é o preto e o mais afastado da origem corresponde à cor branca", MARQUES FILHO; NETO (1999). A faixa de valores normalizada, utilizada convencionalmente para representação das cores , variam de 0 a 1, como ilustrado na Figura 2.9. O objetivo das imagens nesse formato resume-se em retratar cores variadas

em dispositivos eletrônicos como telas de computador e televisores, câmeras digitais de fotografia ou vídeo, *scanners*, retroprojetores, entre outros.

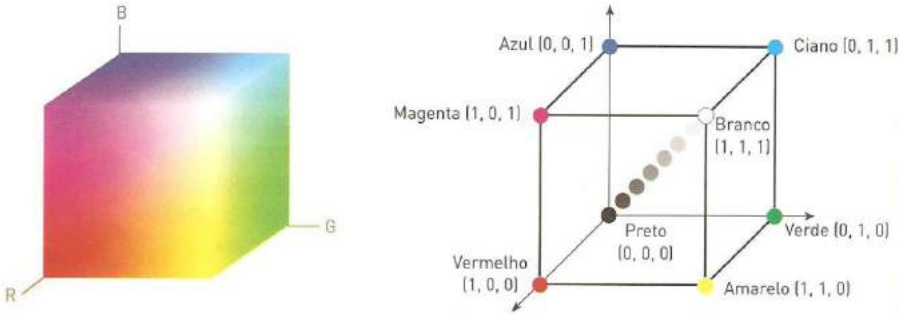


Figura 2.9: Representação do modelo de coordenadas *RGB* (Fonte: João Pedro Ferreira Valente[6]).

2.2.5 Limiarização

"O princípio da limiarização consiste em separar as regiões de uma imagem quando esta apresenta duas classes (o fundo e o objeto)"MARQUES FILHO; NETO (1999). Esse processo, transforma os valores dos *pixels* de uma imagem em escala cinza em brancos, quando os mesmos se apresentam dentro de uma faixa de valores limitada, e os demais em preto, criando por fim uma imagem binarizada. Ainda segundo os autores, a operação de limiarização pode ser descrita matematicamente da seguinte forma:

$$g(x,y) = \begin{cases} 1 & \text{, se } f(x,y) \geq T \\ 0 & \text{, se } f(x,y) < T \end{cases}$$

A função *f(x, y)* representa o valor do *pixel* no ponto de coordenadas *(x, y)* que descreve a imagem no formato matricial em escala cinza, o valor de T corresponde ao valor em tom de cinza do limiar aplicado no processo que limitará as regiões a serem destacadas e *g(x, y)* a imagem limiarizada resultante do processo. Por convenção, os *pixels* de grandezas iguais a 1 correspondem aos "objetos" e aqueles com valores iguais a 0 equivalem o "fundo" (*background*).

Segundo BOVIK (2009), o limiar T a ser utilizado é de importância crítica, pois define a qualidade da abstração de informações obtidas como resultado. Seu valor, como citado por MARQUES FILHO; NETO (1999), pode variar de acordo com parâmetros como o tom de cinza original do ponto analisado (*f(x, y)*), alguma propriedade local deste ponto

[6]Disponível em: <https://sites.google.com/site/aimcjv/modelo-rgb>. Acesso em: 05 mai. 2019.

(*p(x, y)*, como por exemplo a média dos valores dos *pixels* da vizinhança) ou até mesmo as coordenadas espaciais *(x, y)* do *pixel* em foco. Quando T depende apenas de *f(x, y)*, é definido como global; ao vincular-se também com *p(x, y)*, é chamado de local; se, além disso, ser influenciado pelos valores das coordenadas x e y, é descrito como dinâmico ou adaptativo. A Figura 2.10 exemplifica a aplicação deste processo, utilizando valores de T escolhidos arbitrariamente. Nota-se, ao comparar as imagens finais, que o valor de *T=150* é o que proporciona a maior diferenciação entre objeto e fundo, necessária para extração de informações consideradas relevantes na cena.

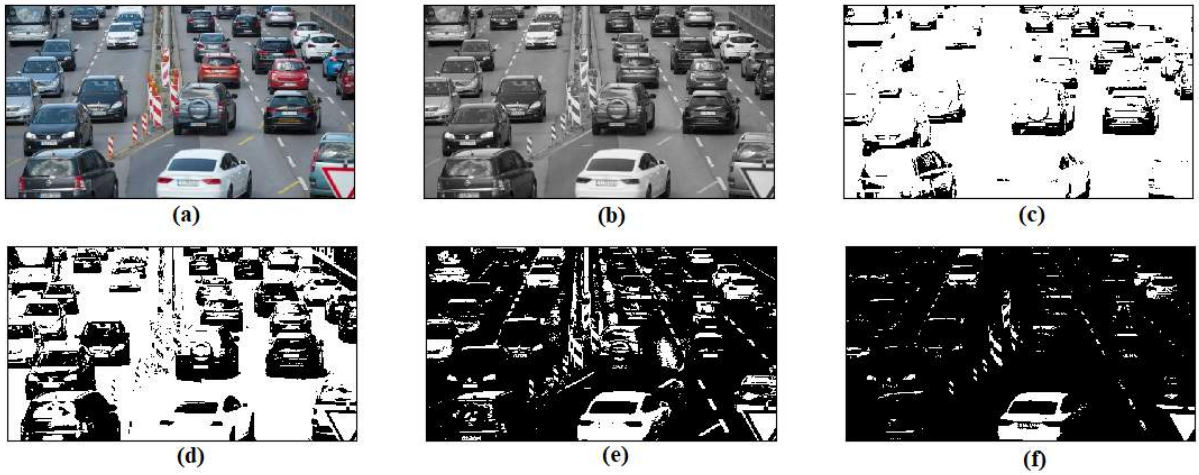


Figura 2.10: Aplicação de limiarização global em uma imagem contendo veículos em uma rodovia. (a) Imagem em formato *RGB*; (b) Imagem em escala cinza; (c) *T=50*; (d) *T=100*; (e) *T=150*; (f) *T=200*.

2.2.6 Filtros Morfológicos

Filtros morfológicos tem a função extrair informações relativas a geometria e topologia de um conjunto desconhecido de dados (imagem), a partir da transformação de outro conjunto totalmente definido, chamado de elemento estruturante, utilizando como princípio básico a morfologia matemática MARQUES FILHO; NETO (1999). Por BOVIK (2009), filtros morfológicos (também chamados de filtros booleanos) são usados para expandir, encolher, suavizar ou eliminar ruídos em objetos de cena e podem ser representados pela seguinte expressão:

$$g(\mathbf{n}) = h[\mathbf{B}f(\mathbf{n})]$$

Desta, a variável f representa a imagem original a ser filtrada, h a operação booleana e $\boldsymbol{B}$ a janela que irá regrar geometricamente a área de aplicação do filtro. A imagem filtrada é definida por $g = h(f)$. Assim, cada *pixel*, denotado por n no domínio da imagem, apresentará um valor $g(n)$ resultante de 0 ou 1, baseado na execução de uma operação

booleana fundamentada pela coleta dos valores dos *pixels* da vizinhança, de acordo com a regra geométrica relacionada a esse conjunto presente na janela.

Dois dos principais filtros utilizados são o filtro de dilatação e erosão, relacionados às operações booleanas *OR* e *AND*, respectivamente. O processo de dilatação expande o tamanho das regiões de primeiro plano, referentes ao objeto, além de suavizar seus limites e remover possíveis lacunas ou orifícios existentes. Já o método de erosão desempenha a função contrária, reduzindo as regiões do primeiro plano e expandindo o plano de fundo, ou regiões com valor 0. Como na dilatação, a erosão também suaviza os limites do objeto, mas de forma diferente, removendo outros objetos de tamanhos considerados pequenos do cenário e deformidades do próprio objeto principal. As Figuras 2.11a e 2.11b, na devida ordem, ilustram a aplicação dos filtros citados anteriormente.

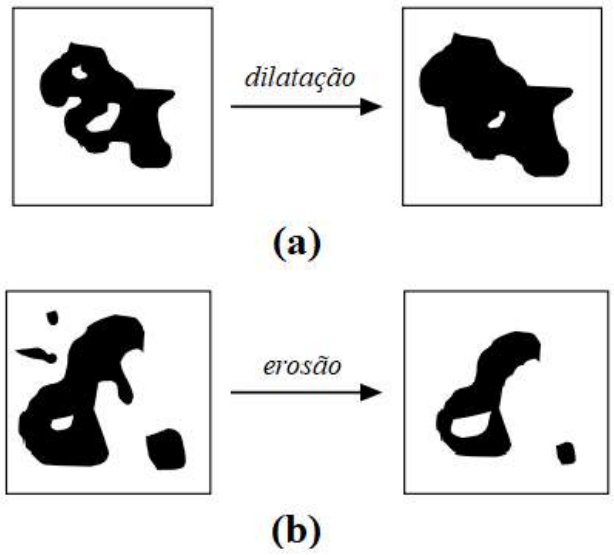


Figura 2.11: Aplicação de filtros morfológicos em imagens binarizadas. (a) Dilatação; (b) Erosão (Fonte: BOVIK, 2009, p. 84 e 85).

2.2.7 Detecção de Bordas

Para GONZALEZ C. RAFAEL (2000), "uma borda é o limite entre duas regiões com propriedades relativamente distintas em nível de cinza". Considerado como o procedimento mais comum na detecção de descontinuidades significantes nos níveis de cinza, as técnicas envolvidas na maioria dos algoritmos voltados para esse propósito se baseiam na computação de um operador local diferencial. De acordo com Bovik (2009), mudanças de intensidade em imagens digitais geralmente correspondem a alterações físicas em alguma propriedade das superfícies dos objetos tridimensionais representados (por exemplo, al-

terações na refletância, descontinuidades de textura, profundidade ou orientação, limites dos objetos) ou mudanças na sua iluminação. Existem três tipos de bordas, classificados de acordo com a aproximação de seus formatos padrões, definidos por: linhas, degraus, rampas e telhados. Cada um desses tipos estão relacionados, respectivamente, à uma função impulso de Dirac na derivada da ordem 0, 1 e 2.

Segundo SENTHILKUMARAN; RAJESH (2009), no artigo *"Edge Detection Techniques for Image Segmentation A Survey of Soft Computing Approaches"*,

> *"Discontinuities in the image intensity can be either Step edge, where the image intensity abruptly changes from one value on one side of the discontinuity to a different value on the opposite side, or Line Edges, where the image intensity abruptly changes value but then returns to the starting value within some short distance. However, Step and Line edges are rare in real images. Because of low frequency components or the smoothing introduced by most sensing devices, sharp discontinuities rarely exist in real signals. Step edges become Ramp Edges and Line Edges become Roof edges, where intensity changes are not instantaneous but occur over a finite distance"(SENTHILKUMARAN; RAJESH, 2009)*[7].

[7]As descontinuidades na intensidade da imagem podem ser bordas degraus, onde a intensidade da imagem muda abruptamente de um valor, em um lado da descontinuidade, para um valor diferente no lado oposto, ou bordas lineares, onde a intensidade da imagem altera abruptamente o valor, mas retorna para o valor inicial dentro de uma distância curta. No entanto, as bordas degrau e linha são raras em imagens reais. Devido aos componentes de baixa frequência ou à suavização introduzida pela maioria dos dispositivos de detecção, descontinuidades acentuadas raramente existem em sinais reais. Bordas degrau tornam-se bordas rampa e bordas do tipo linha tornam-se bordas telhado, onde as mudanças de intensidade não são instantâneas, mas ocorrem em uma distância finita (SENTHILKUMARAN; RAJESH, 2009).

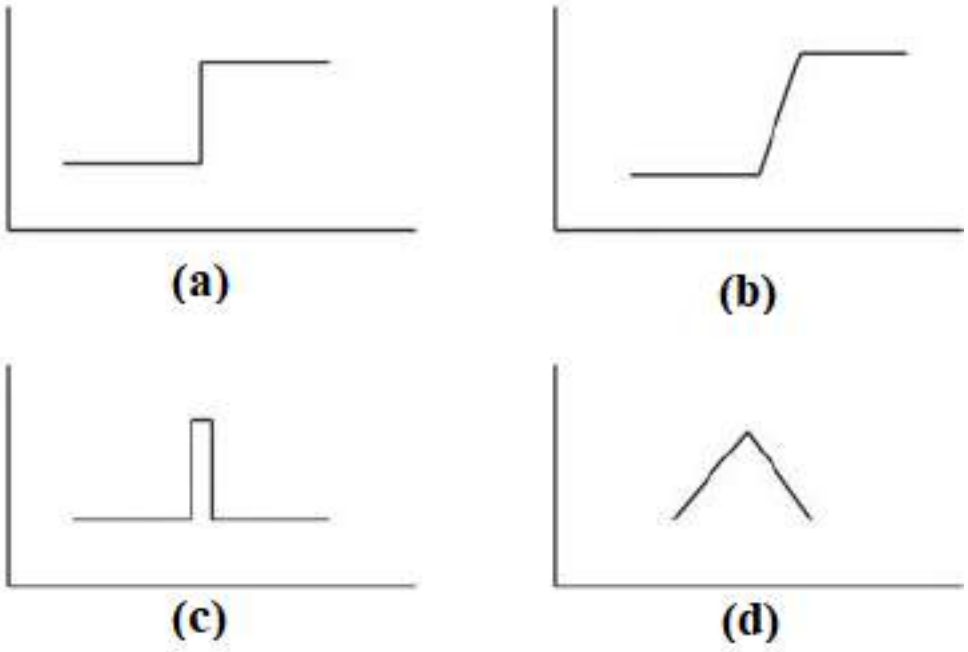


Figura 2.12: Formas dos possíveis tipos de bordas: (a) Degrau; (b) Rampa; (c) Linha; (d) Telhado (Fonte: SENTHILKUMARAN; RAJESH, 2009 , p. 251).

Capítulo 3

Desenvolvimento

O presente capítulo apresenta de forma detalhada as descrições de cada etapa de desenvolvimento do projeto: *hardware* e *software*. Na primeira parte, são detalhadas as escolhas de componentes que compõem o *hardware*, a estrutura da montagem final e como é seu funcionamento na solução proposta. Em seguida, são apresentadas as ferramentas selecionadas para a elaboração do *software* e os algoritmos computacionais e interfaces gráficas vinculadas à aplicação.

3.1 Desenvolvimento do *Hardware*

A princípio, buscou-se estruturar uma versão de *hardware* visando a obtenção de forma rápida, eficaz e prática das imagens das placas veiculares na entrada no ambiente a ser monitorado. O levantamento dos componentes a serem utilizados, se fundamentou em pesquisas, comparações e análises da estrutura dos *hardwares* de sistemas semelhantes, levando sempre em consideração o valor de cada elemento relacionado.

A partir dessas escolhas, desenvolveu-se então o projeto conceitual mostrado na Figura 3.1, onde são integrados um computador de tamanho reduzido e baixo custo a um módulo de câmera compatível, energizado através de uma fonte externa e alocados de forma compacta em uma *case*. O intuito desse conjunto seria garantir a aquisição e armazenamento das imagens, realizar os devidos tratamentos computacionais e enviar os resultados para uma aplicação digital interativa, a ser controlada pelo usuário (através de um *notebook* ou *smartphone*, por exemplo).

Os subtópicos desse capítulo, vistos a seguir, detalham a estrutura de cada elemento selecionado para integrar o *hardware* utilizado no sistema, bem como o por quê de suas escolhas.

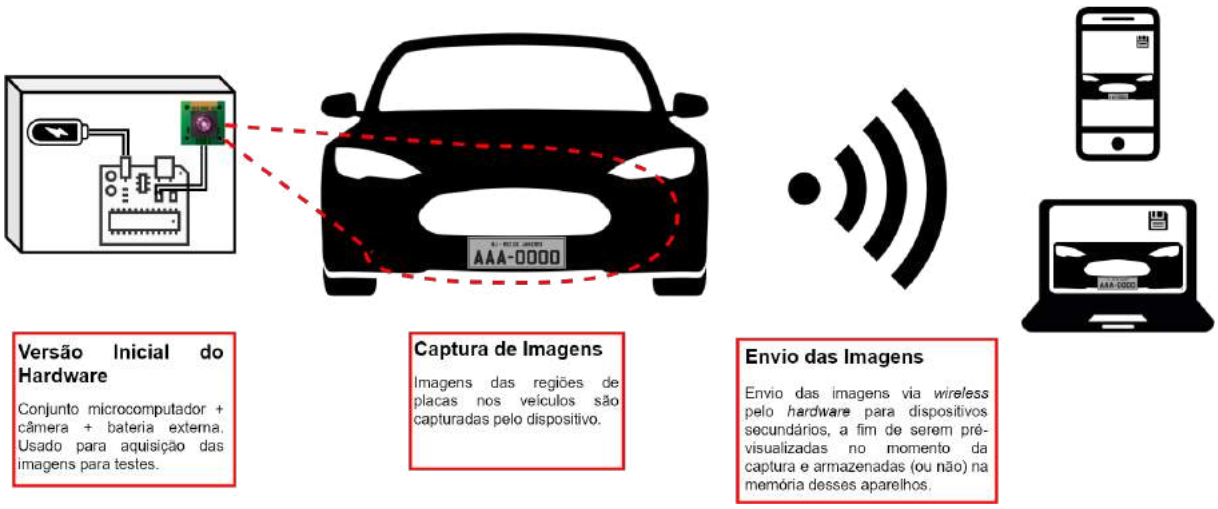


Figura 3.1: Demonstrativo do funcionamento do sistema.

3.1.1 Raspberry Pi 3 Model B

O modelo de computador compacto selecionado para este projeto foi o *Raspberry Pi 3 Model B*, devido ao custo acessível de mercado de acordo com o orçamento previsto inicialmente, aliado a um grande poder de processamento. Este sistema oferece um ambiente de desenvolvimento interativo, já conhecido pelo autor, com uma gama de conteúdo e documentação disponível de forma gratuita. Comumente aplicado em projetos de sistemas embarcados, possui um tamanho reduzido de apenas 85 x 56 x 17 mm. Além disso, caracteriza-se também por sua robustez em campo e se mostrar compatível com sistemas operacionais baseados em *Linux*, considerados rápidos, eficazes e com interfaces amigáveis, como o *Raspbian*, escolhido para este trabalho.

Capaz de acoplar periféricos dos mais diversos tipos como botões, sensores, câmera, display *LCD*, monitor, teclado, *mouse*, dentre outros, fornece a possibilidade de controle dos mesmos através de códigos de programação, sendo extremamente útil se necessário agregar outras funcionalidades ao sistema em projetos futuros. Apresenta compatibilidade com a linguagem *Python*, tecnologia previamente conhecida e estudada pelo autor, que apresenta uma infinidade de bibliotecas disponíveis, dentre elas voltadas para áreas importantes para o trabalho, como visão computacional, criação de interfaces gráficas e gerenciamento de bancos de dados.

A troca de informações com outros dispositivos pode ser feita via *wireless*, por protocolos baseados em *Bluetooth* ou *Wi-Fi*, ou através de cabos utilizando as portas *Serial*, *Ethernet* ou *UART*, por exemplo. Portanto, faz-se possível obter imagens de um módulo de câmera relacionado em aparelhos como *smartphones*, *notebooks* ou computadores, de diferentes formas. Abaixo, segue uma lista contendo algumas das principais características

desse componente:

- Processador Broadcom BCM2837 64bit Quad Core ;
- Clock 1.2 GHz;
- 1 GB RAM;
- Adaptador Wifi 802.11n integrado;
- Bluetooth 4.1 (BLE) integrado;
- Conector 100Base Ethernet;
- Conector de vídeo HDMI;
- Conector de áudio e vídeo;
- Conector Micro SD para armazenar sistema operacional e dados em geral;
- 40 pinos GPIO;
- 4 portas USB 2.0;
- Porta CSI para conexão de módulo externo de câmera compatível;
- Porta DSI para conexão de módulo externo de display compatível;
- Fonte de energia Micro USB comutada (5,1 V e até 2,5 A);
- Dimensões de 85 x 56 x 17mm;

Figura 3.2: Raspberry Pi 3 Model B (Fonte: Amazon[1]).

3.1.2 Câmera

O módulo de câmera escolhido para o *hardware* é o '*Raspberry Pi Camera 5MP Rev 1.3*', que consiste de uma versão criada especialmente para utilização em placas *Raspberry Pi* (compatível com os modelos *2, 3, 3B* e *3B+* e com o sistema operacional *Raspbian*), bastando apenas plugar a câmera ao conector *CSI* (*Camera Serial Interface*) da placa,

[1]Disponível em: <`https://www.amazon.com/Raspberry-Pi-MS-004-00000024-Model-Board/dp/B01LPLPBS8`>. Acesso em: 31 mai. 2019.

especialmente dedicado para este tipo de conexão. Além disso, apresenta um valor considerado baixo e se mostra apta a gerar imagens e vídeos de alta qualidade, devido a capacidade de fornecer uma resolução de até 2592 x 1944 *pixels* para imagens estáticas e suportar gravações em resolução de 1080p a 30 fps. Muito leve, pesando aproximadamente 3 gramas, e medindo apenas 25 x 24 mm, dispõe do sensor OV5647 de 5MP, se mostrando como um componente que supriu as demandas de projeto relacionadas a obtenção e qualidade de imagens, compatibilidade de conexão com a central e viabilidade de acoplamento físico na versão final do sistema.

Figura 3.3: Raspberry Pi Camera 5MP Rev 1.3 (Fonte: Electronics[2]).

3.1.3 *Case*

Com o objetivo de agrupar os elementos eletrônicos de forma segura e compacta, foi necessário efetuar a compra de uma *case*, espécie de caixa protetora, moldada especificamente para agrupar uma placa *Raspberry Pi* e o modelo de módulo de câmera adquirido para o projeto. A principal função desse recurso é proteger os componentes eletrônicos do meio externo e de possíveis intempéries, como quedas ou exposição direta a partículas de água ou poeira. Além disso, também contribui com um melhor manuseio e apresentação visual do *hardware*, possuindo dimensões reduzidas de apenas 100 x 63 x 35 mm.

Visto como ideal e indispensável caso o sistema venha a ser implementado em um ambiente de uso contínuo, apresenta também um suporte rotativo a ser adicionado em sua estrutura. Este, possui fixação por parafusos, possibilitando o acoplamento do *hardware*

[2]Disponível em: <https://www.electronics.com.bd/rasberry-pi-camera-rev-1-3-5mp>. Acesso em: 31 mai. 2019.

em diferentes superfícies de forma estável, além de viabilizar a rotação livre do conjunto, visando obter uma aquisição otimizada de imagens dos veículos em diferentes posições.

Figura 3.4: Case específica para agrupar o *Raspberry Pi 3 Model B* e módulo de câmera *Raspberry Pi Camera Rev 1.3*, com suporte rotativo parafusável (Fonte: Mercado Livre[4]).

3.1.4 Fonte de Alimentação Externa

Para evitar o uso de cabos de alimentação, que se relaciona a dificuldade encontrada em conectá-los em ambiente aberto, utiliza-se uma bateria de lítio recarregável e portátil, do tipo *Power Bank*, de posse prévia do autor para fornecer a energia necessária ao conjunto de elementos do *hardware*.

Essa bateria tem capacidade energética de 10.000 mAh, aplicando em sua saída uma tensão de 5 V e 2,1 A por um conector micro USB, compatível com a alimentação e conexão exigida pelo modelo *Raspberry Pi* utilizado. Ademais, destaca-se pelo dimensionamento

[3]Disponível em: <`https://lista.mercadolivre.com.br/case-raspberry-pi-3-camera-rotativo`>. Acesso em: 03 abr. 2021.

reduzido de 155 x 80 x 10 mm e peso razoável de 190,7 gramas, na média de outros produtos semelhantes, que se enquadram nessa categoria. Como resultado, obteve-se um modo de suprimento energético compacto e eficaz ao sistema inicial, contribuindo para uma melhor manipulação, transporte e ativação do conjunto como um todo.

3.1.5 Montagem e Funcionamento

Ao serem adquiridos os componentes necessários para a construção do *hardware* idealizado, foi preciso realizar configurações específicas visando estabelecer a comunicação entre os dispositivo principal e os periféricos, a fim de obter e armazenar as imagens das placas de carros de forma remota, que iriam compor em um momento inicial a base de testes para os algoritmos de identificação.

Primeiramente, foi feito o *download* do sistema operacional *Raspbian*, baseado em *Linux* e voltado para utilização em ambientes *Raspberry Pi*. Transferiu-se o arquivo para um cartão SD de 16GB a ser acoplado ao *Raspberry Pi 3 Model B* energizado, de posse do autor, e, com isso, foi possível acessar, através de uma interface gráfica intuitiva e bem estruturada, as ferramentas e opções disponibilizadas. Para visualização e controle dessa interface (Figura 3.5), foram conectados ao computador compacto, através das portas *HDMI* e *USB*, um monitor e um *mouse*, respectivamente.

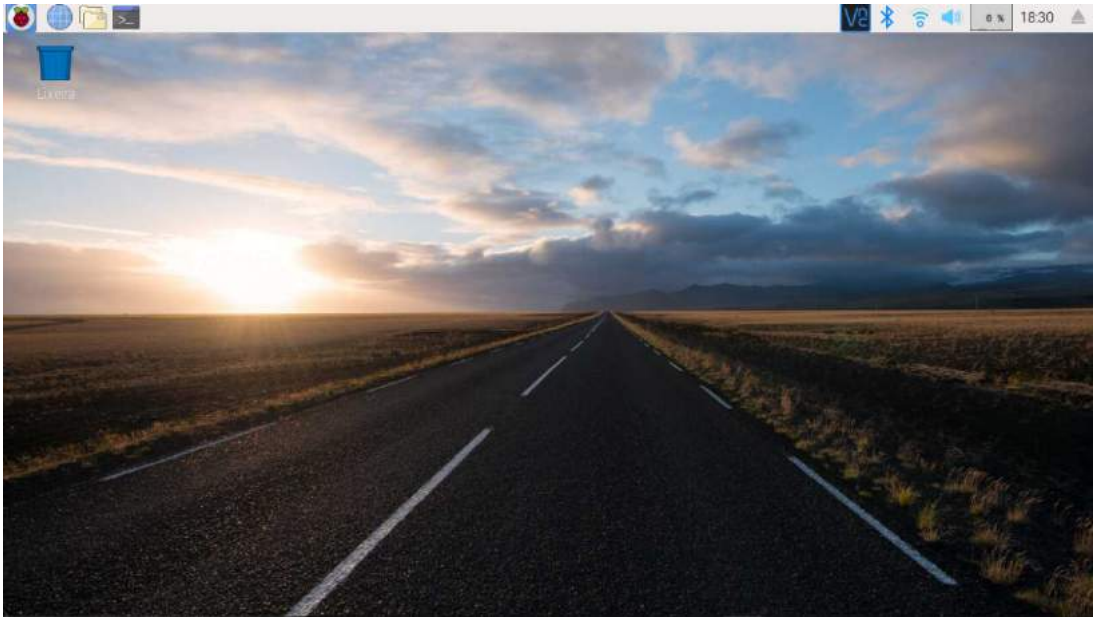


Figura 3.5: Tela inicial do sistema operacional *Raspbian* aplicado ao *Raspberry Pi*.

Após concluído esse processo, precisou-se habilitar as opções de interface '*Camera*', '*SSH*' e '*VNC*', através da seguinte sequência de cliques: '*Applications Menu*' → '*Preferences*' → '*Raspberry Pi Configuration*' → '*Interfaces*'. O intuito de liberar essas opções se baseiam, nesta ordem, em estabelecer a comunicação com um módulo de câmera re-

lacionado, efetivar o acesso remoto as linhas de comando do dispositivo e permitir o compartilhamento do conteúdo gráfico com outra máquina via internet.

Feito isso, foram conectados o *Raspberry Pi* e um *notebook* a uma mesma rede *Wi-Fi*, instalados na segunda máquina os *softwares* para acesso remoto *VNC Viewer* (protocolo *VNC*) e *Adafruit Pi Finder* (protocolo *SSH*) e estabelecido os dois tipos de conexão através de ambos os programas, visando demonstrar possíveis maneiras de controle indireto do sistema, a serem usadas posteriormente ao longo do projeto. Utilizando o *VNC Viewer*, exigiu-se apenas o IP a ser disponibilizado para o *Raspberry Pi* (fornecido pela rede conectada) e preencher corretamente os campos de *login* e senha do mesmo. Já ao utilizar o *Adafruit Pi Finder*, após iniciado, bastou clicar no botão '*Find My Pi!*', aguardar alguns segundos e acessar o terminal de forma remota, clicando no botão '*Terminal*'. Ambos os modos de acesso podem ser vistos na Figura 3.6, mostrada a seguir.

A próxima tarefa consistiu em conectar a câmera ao *Raspberry Pi* e capturar imagens aleatórias, para confirmar as conexões entre os componentes. Para isso, fixou-se o cabo do tipo FLAT da câmera - composto por uma película de plástico plana e flexível - à entrada própria da placa principal. Depois, foram passadas as seguintes instruções ao terminal para captura da primeira imagem e vídeo, respectivamente:

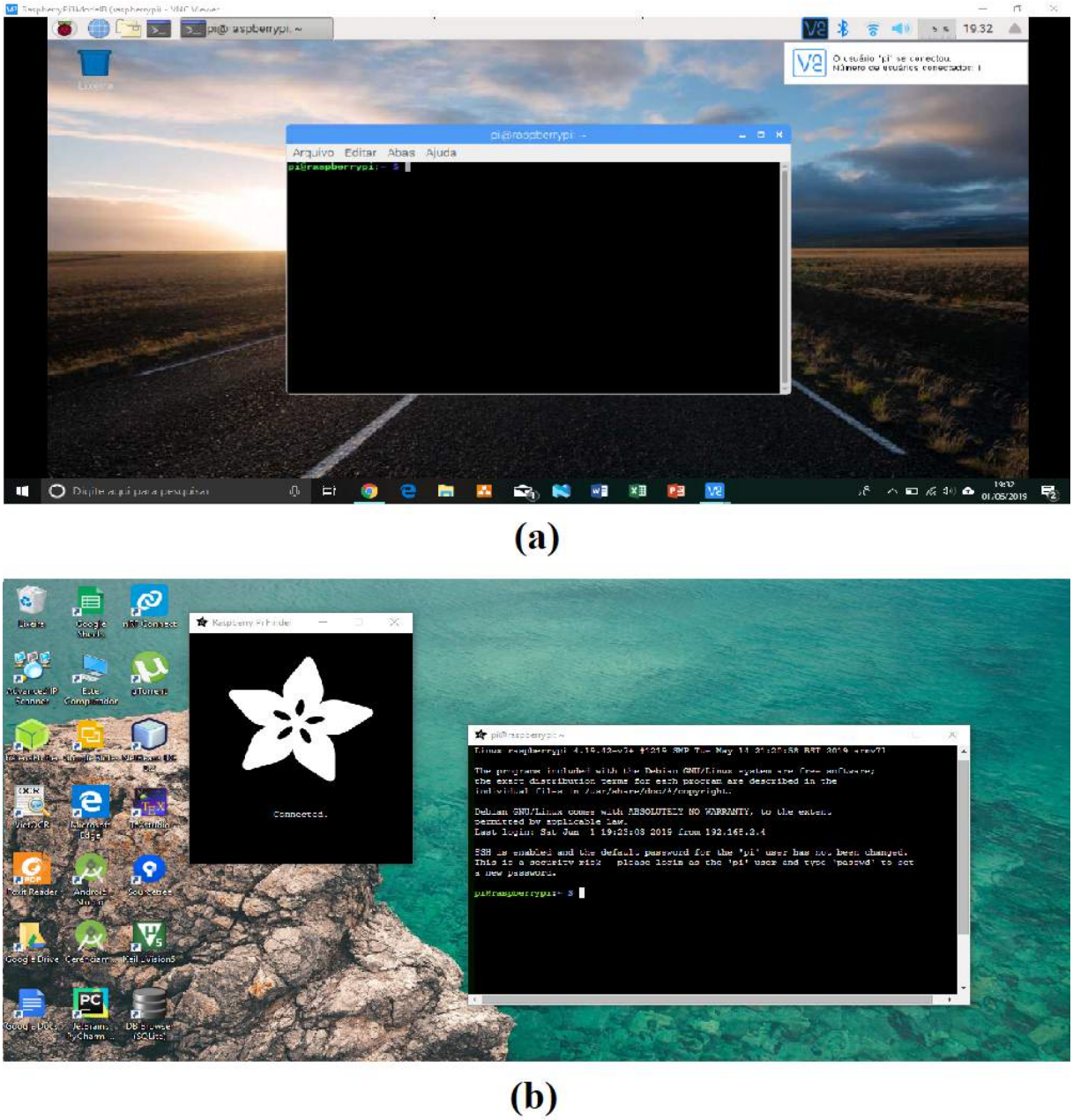
$$raspistill \quad -o \quad minha\_foto.png$$

$$raspivid \quad -o \quad meu\_video.h264 \quad -t \quad 5000$$

Os arquivos foram criados com sucesso, uma imagem do tipo *PNG* e um vídeo de 5 segundos de duração no formato *H264*, vistos na Figura 3.7.

Na sequência, após os conhecimentos adquiridos sobre os modos de comunicação disponíveis, compatibilidade com outros dispositivos e obtenção de imagens pelo módulo de câmera, pesquisou-se na *Play Store*, loja oficial de aplicativos para o sistema operacional *Android*, possíveis aplicativos *mobile* para esse ambiente. Foram buscadas aplicações que proporcionavam o controle indireto das ações de um *Raspberry Pi* por *smartphone*, sendo encontrados vários *softwares* com esse propósito. O aplicativo escolhido foi o *RaspController*, devido, principalmente, a variedade de opções de controle concedidas (inclusive para módulos de câmera), interface intuitiva, compatibilidade com protocolo *SSH* e elevada nota facultada pelos usuários (4,3 de 5, dentre 649 avaliações feitas).

A Figura 3.8 apresenta a interface principal deste programa (a), controle da câmera (b) e acesso ao terminal via *SSH* (c). Destaca-se a opção de controle de câmera, que disponibiliza as ações de configurar a qualidade de imagem, ajustar a claridade local, mostrar em tempo real a visualização da câmera e capturar imagens, que são armazenadas no próprio dispositivo *mobile*, propriedades importantes que facilitarão o uso do sistema projetado em campo aberto.



(a)

(b)

Figura 3.6: Acesso remoto por um *notebook* ao *Raspberry Pi*, feito através dos *softwares*: (a) *VNC Viewer*; (b) *Adafruit Pi Finder*.

Após definidos os elementos descritos anteriormente, partiu-se para aplicação em campo do sistema. O *hardware* montado foi levado ao estacionamento aberto da instituição *CEFETMG - Campus V*, situada em Divinópolis, Minas Gerais, para capturar imagens de placas de carros de diferentes marcas, modelos e cores. Previamente, conectou-se o *Raspberry Pi* e o *smartphone* a mesma rede *Wi-Fi* local, a fim de estabelecer uma posterior conexão remota entre os dispositivos pelo aplicativo *RaspController*. Solicitou-se, aos respectivos proprietários dos veículos, permissão para que fossem obtidas as imagens, e, para gerar um conjunto de figuras diversificado, capturou-se imagens de placas frontais

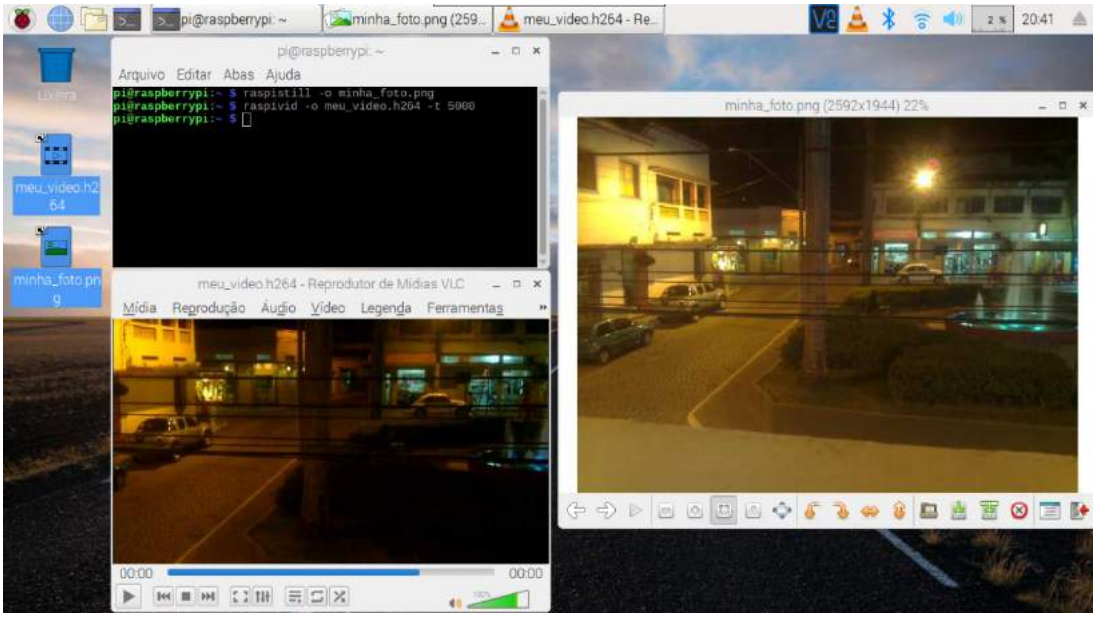


Figura 3.7: Obtenção dos primeiros arquivos de imagem e vídeo pela câmera conectada ao *Raspberry Pi*.

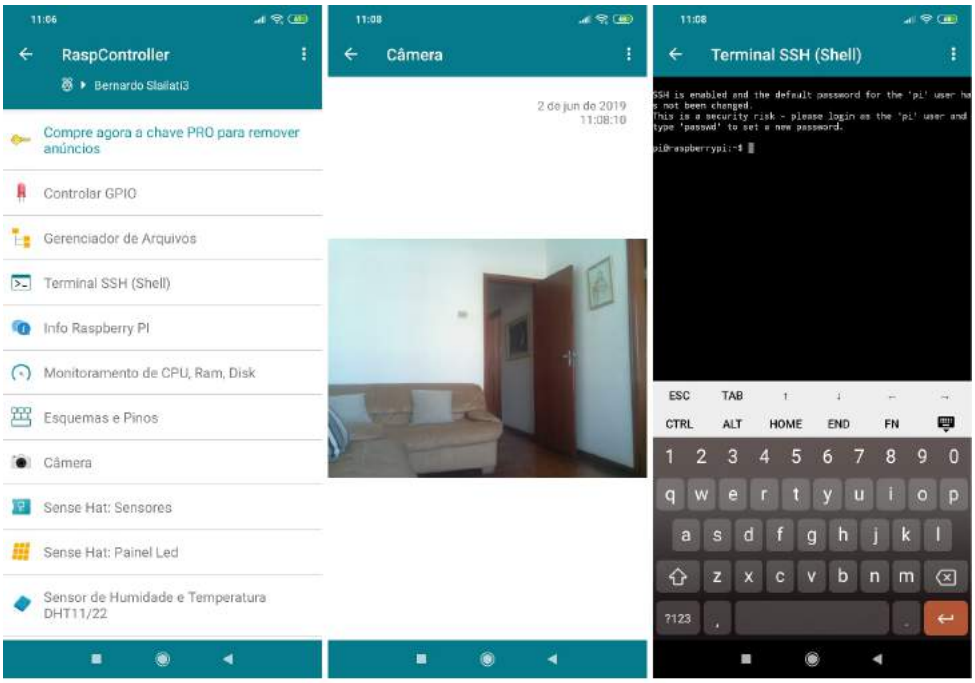


Figura 3.8: Aplicativo *RaspController* em ambiente *mobile* conectado a um *Raspberry Pi*. (a) Interface principal; (b) Opção 'Câmera'; (c) Opção 'Terminal SSH (Shell)'.

e traseiras, em diferentes distâncias da câmera (entre 80 a 100 cm, aproximadamente), a fim de validar a precisão dos algoritmos de detecção das placas e caracteres em diferentes situações. A Figura 3.9 ilustra o procedimento real feito pelo autor, para obtenção das imagens de placas frontais utilizando o *hardware*, onde o *Raspberry Pi* e o módulo

de câmera, agrupados em uma *case*, a uma determinada distância em relação a placa, é alimentado pela bateria portátil do tipo *Power Bank* e controlado por um *smartphone*, com sistema operacional *Android*, contendo o aplicativo *RaspController*.

Figura 3.9: Ilustração do modo de aplicação da primeira versão do sistema.

3.2 Desenvolvimento do *Software*

A idealização da interface inicial do *software*, se baseou em dispor para o usuário as funções definidas como essenciais para o controle do sistema, atreladas a um visual simples e intuitivo. Dentre as opções de funcionalidades supostas, caracterizaram-se: em uma tela principal, a apresentação de informações da placa encontrada, veículo e respectivo proprietário, destaque para os últimos registros de entrada de veículos e acesso a telas secundárias, que conteriam o registro completo de entradas feito até o momento e realização do cadastro de veículos e proprietários. A ilustração vista na Figura 3.10, representa a interface gráfica prevista para o *software* a ser desenvolvido neste projeto.

Com o intuito de descrever os proprietários, como mostrado a cima, além da área voltada para a alocação da foto dos mesmos, foram escolhidos os seguintes atributos: *Nome*, *CPF*, *Telefone*, *Apartamento* e opção de ser ou não visitante. Aos veículos, destinam-se os campos: *Tipo* (carro, moto, caminhão, entre outros), *Cor*, *Marca*, *Modelo* e *Ano*. Na

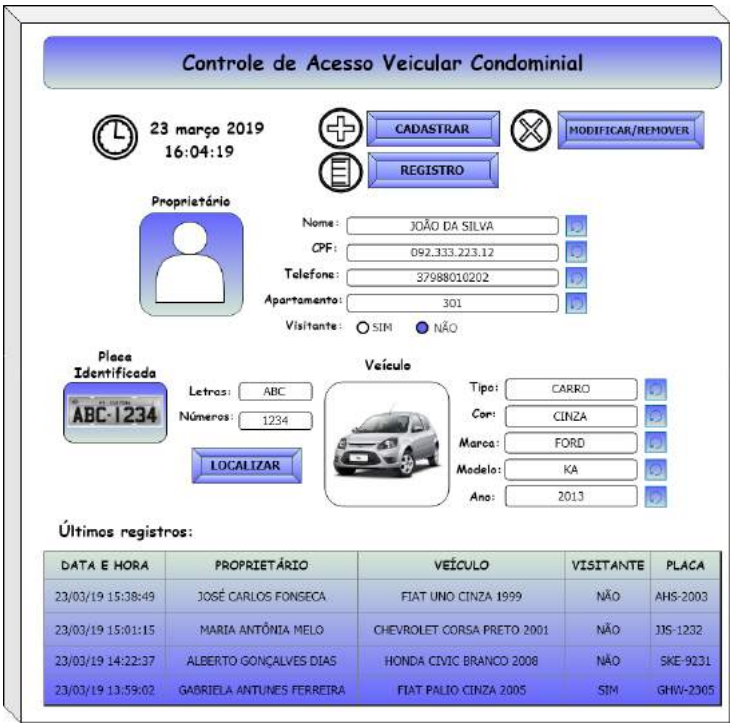


Figura 3.10: Protótipo da interface de tela principal idealizada para a versão inicial do *software*.

região voltada as informações da placa a ser identificada, será posicionado o botão *'LOCALIZAR'*. Esse botão deverá ser clicado pelo usuário para capturar a imagem da placa do veículo que solicita a entrada no local. Dessa forma, a câmera será ligada apenas em determinados momentos, de acordo com a solicitação do usuário, gerando economia de energia para o sistema e evitando um possível processamento desnecessário de dados. Ao ser clicado, a câmera realiza instantaneamente a captura da imagem, o arquivo é salvo no diretório especificado internamente pelo programa e o inicia-se o processo completo de identificação. Como resposta, espera-se o preenchimento correto dos campos de texto *Letras* e *Números*, com os caracteres referentes a placa relacionada. Objetiva-se também completar automaticamente os espaços que descrevem o veículo e o proprietário, se os mesmos estiverem sido cadastrados previamente no sistema e relacionados a placa identificada. Além disso, posicionados à frente de cada campo editável de texto, serão alocados botões de *Atualizar*, simbolizados pela figura de uma seta circular, que, ao serem clicados, atualizarão os dados registrados no banco de dados do atributo associado.

As interfaces secundárias vislumbradas, são ilustradas na Figura 3.11, sendo ambientes destinados a funcionalidades específicas que complementam o desempenho principal do sistema. O atalho de acesso para essas telas serão os botões '*CADASTRO*', *'MODIFICAR/REMOVER*' e '*REGISTRO*', que conduzirão às interfaces mostradas nas Figuras 3.11a, 3.11b e 3.11c, na respectiva ordem. A primeira, conta com elementos designados a realizar o cadastro dos proprietários e seus respectivos veículos, a segunda mostra em

detalhes todos os registros de entradas efetuados pelo *software* e a terceira, disponibiliza a opção de remover ou modificar os cadastros salvos de proprietários e veículos.

Vale ressaltar que o conteúdo apresentado anteriormente consistiu da idealização inicial do *software*, recebendo alterações no desenvolver do projeto que visaram aprimorar e simplificar seu funcionamento.

Figura 3.11: Protótipo das interfaces de telas secundárias idealizadas para a versão inicial do *software*: (a) Cadastro de veículos e proprietários; (b) Detalhes dos registros de entrada de veículos; (c) Modificar/Remover proprietários e veículos.

3.2.1 Python

Vista a necessidade de escolher uma linguagem programação para o desenvolvimento dos algoritmos, optou-se pela linguagem *Python*, após pesquisas e estudos sobre essa e ou-

tras possíveis alternativas. Por se caracterizar como uma linguagem de alto nível simples, robusta e de fácil entendimento, compatível com diferentes sistemas operacionais (*Linux* e *Windows*, por exemplo), possuir bibliotecas bem estruturadas para áreas aplicáveis ao projeto (interfaceamento gráfico, visão computacional, gerenciamento de banco de dados, entre outros) e dispor de uma grande diversidade de conteúdo disponibilizado gratuitamente, a mesma foi escolhida logo no início do trabalho. Outro fator importante para a definição desta linguagem, foi o conhecimento prévio do autor sobre o seu uso e funcionalidades, e, principalmente, a sinergia com a mais famosa biblioteca de visão computacional do mercado, escolhida para ser utilizada neste trabalho: *OpenCV*.

3.2.2 PyCharm

Para auxiliar no momento de criação dos algoritmos, foi utilizado pelo autor o ambiente de desenvolvimento integrado *PyCharm*. Este *software* foi desenvolvido especialmente para a linguagem *Python*, fornecendo módulos como: depurador integrado, refatoração (incluindo ações como renomear, extrair método, introduzir variável ou constante, empurrar para baixo e outros), navegação em projetos e códigos, interpretação de diferentes formatos de arquivo, entre outras funções. Estes, contribuíram consideravelmente com a velocidade da realização das atividades envolvidas no momento da programação, facilitando a criação dos códigos e permitindo uma maneira mais eficaz de trabalho ao desenvolvedor. Vale ressaltar a habilidade em utilizar esse *software*, adquirida anteriormente ao projeto, como sendo também um dos motivos de escolha dessa ferramenta de apoio.

3.2.3 OpenCV

Algoritmos de pré-processamento e reconhecimento de padrões em imagens digitais necessitam métodos consideravelmente complexos de manipulação de matrizes, criados especificamente para aplicações na área de visão computacional. A biblioteca computacional multiplataforma *OpenCV*, contém diversas dessas técnicas, das quais englobam funções de reconhecimento de objetos, filtros de imagem, calibração de câmera, análise estrutural em cenas, rastreamento de objetos em vídeo, interface gráfica simples para interação com o usuário e outros, como citado em sua própria documentação, sendo então escolhida como a base para os algoritmos de identificação a serem desenvolvidos neste projeto.

Compatível com a linguagem *Python*, escolhida para o presente trabalho, e sistemas operacionais como *Linux*, *Android* e *Windows*, é utilizada em diversas aplicações de reconhecimento facial e de objetos, manipulação de vídeos, identificação de movimentos, reconstrução de ambientes tridimensionais, realidade virtual, aprendizado de máquinas,

entre outros. Apresenta sua estrutura em código aberto, com distribuição inteiramente gratuita, além de uma ampla quantidade de projetos disponibilizados livremente em livros, artigos, estudos relacionados e documentação própria. Diversos tutorias também são oferecidos oficialmente, para ajudar no aprendizado dos usuários.

3.2.4 Tesseract

No contexto de identificação dos caracteres de placas veiculares, ao serem pesquisados modos de alcançar computacionalmente esse objetivo, encontrou-se o mecanismo de reconhecimento ótico de caracteres *Tesseract*. O mesmo consiste de um *software* livre, voltado principalmente para projetos nos sistemas operacionais *Linux* e *Windows*, com desenvolvimento patrocinado pelo *Google* desde 2006. Possui suporte para textos *unicode* do tipo UTF-8 - padrão de representação por computadores em 8 *bits* de qualquer modelo de escrita existente -, podendo reconhecer de forma eficaz mais de cem idiomas previamente disponíveis. Seus métodos podem ser acessados por código *Python*, sendo outra característica marcante para a escolha.

Porém, a principal particularidade deste mecanismo cabível ao projeto, é a possibilidade de treinamento pelo usuário, para aprimorar o reconhecimento de caracteres específicos não cadastrados inicialmente. Basicamente, esse treinamento consiste em passar para o mecanismo um conjunto de caracteres em um formato de arquivo específico, contendo as dimensões espaciais de cada caractere definidas numericamente, relacionando os mesmos ao caractere correto que se espera obter após o processo de assimilação. Isso se aplica ao caso, devido a fonte textual utilizada nas placas veiculares brasileiras não existir no banco de dados principal de fontes treinadas para o *Tesseract*. Portanto, ao gerar um conjunto exclusivo de treinamento voltado para a identificação das possíveis letras e números envolvidas nesse contexto, obteve-se o aumento da eficácia dos resultados de reconhecimento.

3.2.5 Tkinter

Visando gerar a interface gráfica do sistema, escolheu-se a biblioteca padrão que acompanha a distribuição oficial do interpretador *Python*, *Tkinter*. Relaciona-se a uma *API* (*Application Programming Interface*) de fácil aprendizado e tem como algumas de suas principais vantagens, a garantia de execução do programa em diferentes sistemas operacionais, sem a necessidade de se instalar bibliotecas extras, diversidade de elementos gráficos disponíveis (botões simples, caixas de texto, botões de escolha e checagem, caixa para listagem de itens, entrada de textos, entre outros) e documentação bem estruturada, com vários tutoriais e referências de qualidade disponíveis voluntariamente.

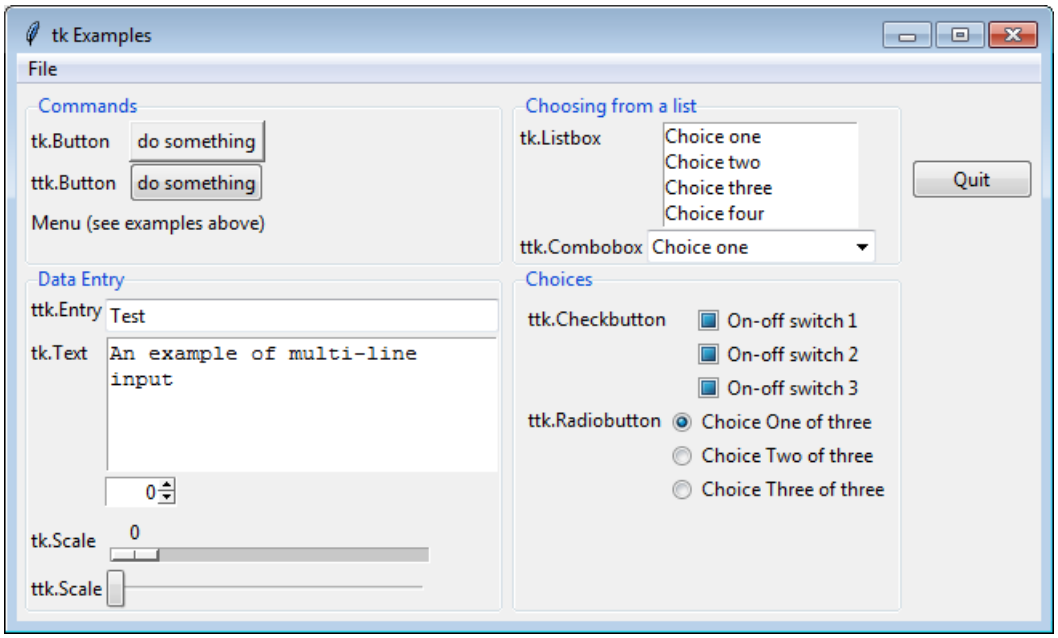


Figura 3.12: Elementos gráficos disponíveis pela biblioteca Tkinter (Fonte: Interactive Python[5]).

3.2.6 Elaboração dos algoritmos computacionais

Para o processo de criação dos códigos computacionais, optou-se por utilizar o *notebook* como máquina de desenvolvimento, devido as ferramentas disponíveis exclusivamente para este modelo, dentre as quais se destaca o ambiente de programação *PyCharm*, descrito anteriormente. Após serem devidamente desenvolvidos e testados na máquina local, os algoritmos foram inseridos no núcleo do *Raspberry Pi*.

Inicialmente, foram instaladas as versões oficiais dos *softwares OpenCV 4.5.1* e *Tesseract 4.1.0* na máquina de desenvolvimento e feitas as devidas configurações para habilitar o uso de suas funcionalidades em linguagem *Python*. Para isso, efetuou-se o *download* dos pacotes '*opencv-python*' e '*pytesseract*', através da própria interface do *PyCharm*, bem como o interpretador *Python* na versão *3.8*. Feito isso, os métodos de cada biblioteca se mostraram disponíveis para o uso em ambiente de codificação.

3.2.6.1 Algoritmos de identificação de placas e caracteres

Antes de começar o desenvolvimento computacional do sistema, obtiveram-se imagens dos mais diversos tipos e resoluções contendo placas brasileiras de carros disponíveis na *internet*. Essas imagens serviram de base para os primeiros testes do algoritmo de pré-processamento criado. Como resultado dos estudos feitos baseados em artigos, trabalhos de conclusão de curso e descritivos de projetos semelhantes, notou-se que, para identificar

[5]Disponível em: <https://interactivepython.org/runestone/static/CS152f17/GUIandEventDrivenProgramming/03_widgets.html>. Acesso em: 02 jun. 2019.

corretamente os contornos de placas veiculares, era necessário aplicar previamente uma sequência de métodos de aperfeiçoamento visual das imagens originalmente coloridas. Dentre as técnicas abordadas, a ordem de aplicação considerada ideal pelo autor, consistiu dos seguintes passos:

Imagem Digital Colorida → Transformação em Escala Cinza → Aplicação dos Filtros Morfológicos de Erosão e posterior Dilatação → Limiarização Adaptativa

O intuito dessa sequência de operações, tem como finalidade estipular um contorno nítido e bem definido em volta das dimensões da placa, que poderão se encontrar em diferentes cenários quando o sistema for aplicado em campo. A Figura 3.13 a seguir, mostra as seguidas aplicações desses processos em uma imagem aleatória, dentre as amostras obtidas para testes.

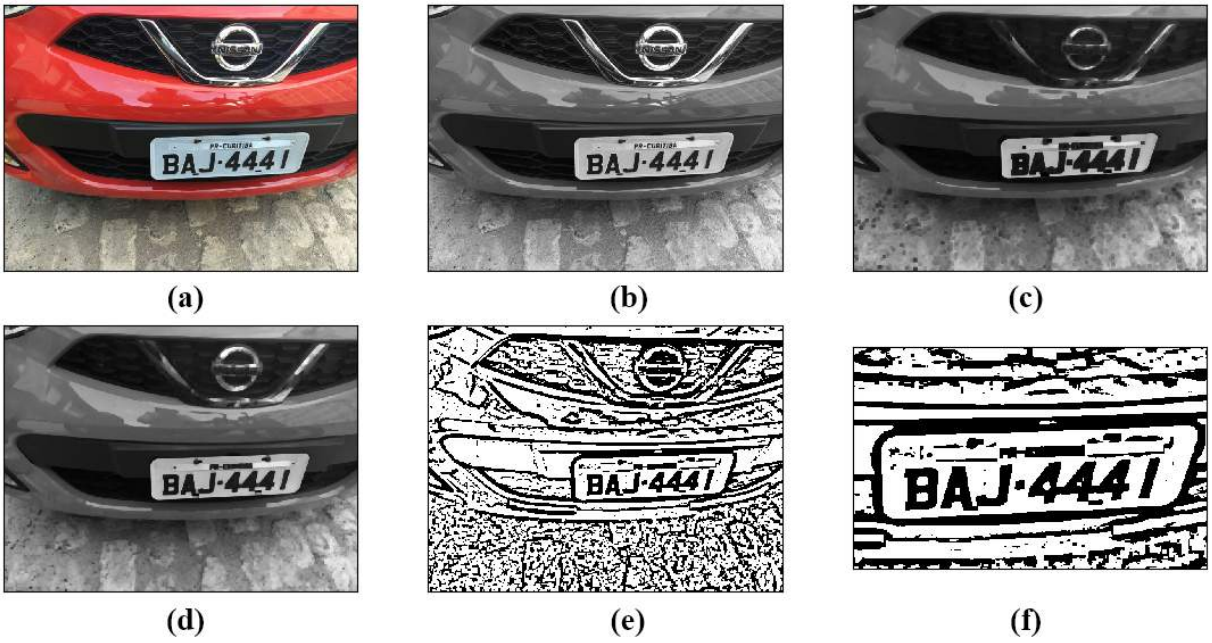


Figura 3.13: Sequência de métodos de pré-processamento considerado adequado pelo autor para a identificação de placas veiculares. (a) Imagem original colorida; (b) Representação em escala cinza; (c) Erosão; (d) Dilatação; (e) Limiarização adaptativa; (f) Zoom da região de interesse da imagem limiarizada.

Após ser aplicado em diferentes figuras e gerado resultados considerados aceitáveis, foi organizado um código em *script Python* que realizava de forma automática essa sucessão de procedimentos (arquivo '*PreProcessamento.py*'), disponíveis na biblioteca *OpenCV*. A entrada aplicada nesse algoritmo, consiste do diretório de um arquivo de imagem colorida no formato *JPG* ou *PNG*, por exemplo, e a saída apresenta a imagem devidamente tratada, como visto na Figura 3.13e.

Finalizada essa etapa, partiu-se para o desenvolvimento do arquivo '*IdentificaPlaca.py*', algoritmo voltado para a busca de possíveis placas em imagens. Para isso, utilizou-se um método específico fornecido pela *OpenCV* para encontrar contornos, sendo esses interpretados como regiões demarcadas por mudanças bruscas nos valores dos *pixels*. Por isso, um dos parâmetros de entrada desse método, é a imagem limiarizada pelo processo anterior e a identificação das possíveis placas baseia-se no cálculo de suas respectivas áreas. Computacionalmente, essas áreas são representadas na grandeza de números de *pixels*. A Figura 3.14 mostra o resultado da execução desse método, sendo destacados os contornos encontrados na imagem limiarizada vista na Figura 3.13e.

Figura 3.14: Aplicação do método de busca de contornos disponibilizado pela biblioteca *OpenCV*.

Devido ao previsto posicionamento do sistema à distâncias padronizadas dos veículos (entre 80 a 100 cm), foi necessário aplicar uma determinada faixa de valores para caracterizar áreas de regiões de placas, relacionando sempre uma margem de erro para englobar casos peculiares.

Em seguida, migrou-se o foco ao desenvolvimento do algoritmo de reconhecimento de caracteres. Testes em imagens de placas previamente cortadas foram feitos, utilizando o comando, disponível pela biblioteca que abstrai as funcionalidades do *Tesseract OCR* para *Python* ('*pytesseract*'), que realiza a ação imediata de transformar os caracteres identificados em uma variável do tipo texto. Como resposta inicial aos experimentos, obtiveram-se uma quantidade ínfima de acertos, e por isso, pesquisou-se também modos de aperfeiçoar a identificação dos caracteres, em específico para o padrão de escrita de placas veiculares brasileiras.

Encontrou-se então uma maneira viável de se realizar tal otimização: através da realização de um treinamento específico baseado em redes neurais do mecanismo de identificação do *Tesseract OCR* que, na versão utilizada e também em outras versões, apresenta esse tipo de suporte, para a fonte utilizada nas placas veiculares brasileiras - fonte *Mandatory*. Para isso, foi utilizado a ferramenta *jTessBoxEditor*, que consiste de um mecanismo de execução de treinamentos para *Tesseract OCR*, baseado no modelo de editor de caixas. Arquivos de caixa pré-formados são necessários para conduzir o treino, e essa ferramenta possibilita gerar esse conteúdo com base na referência dos caracteres da fonte específica almejada, fornecendo edição de dados de caixa de diversos formatos. Basicamente, esse processo consiste em mapear, em coordenadas X e Y, cada possível caractere da fonte escolhida, definindo-os pelo seu tamanho e altura respectivo, como mostra a Figura 3.15. Vale ressaltar que o programa requer o pacote de aplicativos *Java Runtime Environment 8* ou posterior.

Figura 3.15: Mapeamento dos caracteres da fonte *Mandatory* (padrão de placas brasileiras) vindos de uma imagem *TIF* pelo programa *jTessBoxEditor*.

A conclusão desse treinamento gerou um arquivo do tipo '*TRAINEDDATA*', nomeado como *man.traineddata*, inserido na pasta de conteúdos da instalação do *Tesseract OCR*, se tornando assim visível para ser utilizado no *script Python* de identificação de caracteres. A Figura 3.16 mostra a realização da rotina de treinamento efetuada pelo *software*.

Feito isso, iniciou-se o processo de criação do algoritmo de identificação dos caracteres. A primeira etapa, consistiu da aplicação de um pré-processamento específico voltado para segmentar as regiões de interesse: os contornos dos caracteres. A base do algoritmo, disposto no arquivo "PreProcessamentoCaracteres.py", se assemelha ao pré-processamento

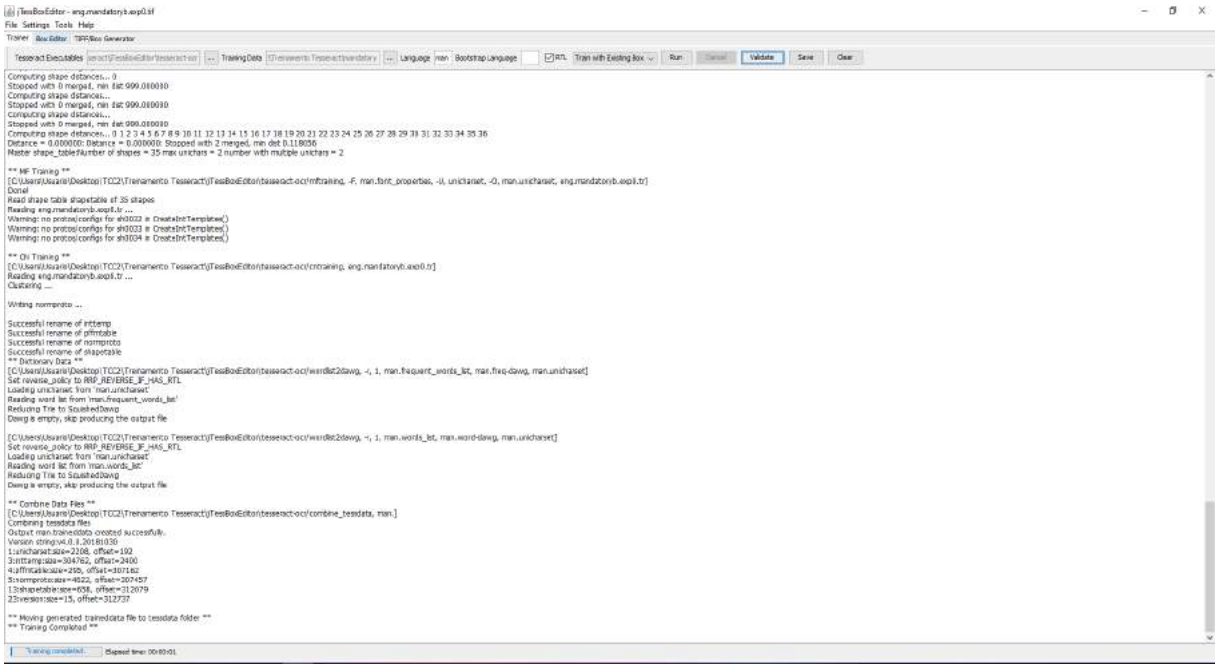


Figura 3.16: Execução do algoritmo de treinamento do *software jTessBoxEditor*, gerando por fim o arquivo *man.traineddata*.

aplicado no momento de busca pela área da placa, porém com algumas diferenças cruciais que garantem a eficácia da ação. Ao se obter a imagem contendo o recorte correto da placa (Figura 3.17a), é necessário, na sequência, isolar a área que abrange apenas os caracteres. Para isso, foi feito um estudo sobre as dimensões do modelo de placas tratado. Com base na Resolução CONTRAN n° 231 de 15/03/2007, Art. 11 - Anexo 1, as placas deverão possuir altura padrão de 13 cm, comprimento de 40 cm e altura de 2 cm para as letras que definem cidade e estado na placa. Além disso, os espaços vazios laterais, superior e inferior, apresentam cerca de 2 cm de altura e largura, respectivamente. Essas dimensões foram utilizadas no código para isolar apenas os caracteres que caracterizam a placa, como mostrado na Figura 3.17b.

Após aplicar o processo de extração da região de interesse, foi vista a necessidade de se corrigir os níveis de brilho da imagem. Percebeu-se que capturas relacionadas a um brilho médio/alto, diminuíam bastante o contraste entre os caracteres e o entorno da placa. Por isso, o próximo passo do algoritmo diminui todos os valores dos *pixels* da imagem, para fazer com que as letras e números (ambos devem apresentar a cor preta) ganhem destaque se comparados ao fundo de cena (tom de cinza claro), sendo exemplificada sua aplicação na Figura 3.17c. Na sequência, são executados os seguintes métodos de tratamento digital de imagens para finalizar o pré-processamento: conversão para a escala cinza (Figura 3.17d) para simplificar a representação matricial da imagem, filtro bilateral, que altera a intensidade de cada *pixel* por uma média ponderada dos valores de intensidade dos *pixels* próximos, visando suavizar regiões, eliminar ruídos e preservar bordas (Figura 3.17e) e

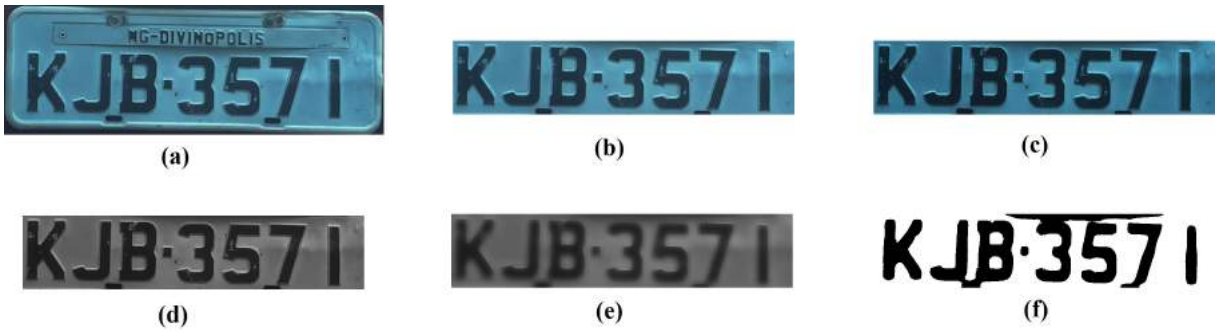


Figura 3.17: Sequência de métodos de pré-processamento considerado adequado pelo autor para a identificação dos caracteres de placas veiculares. (a) Imagem original colorida; (b) Imagem original recortada; (c) Correção de claridade; (d) Representação em escala cinza; (e) Filtro Bilateral; (f) Limiarização.

limiarização convencional, com o intuito de binarizar a imagem por meio da definição de um determinado limite e deixá-la pronta para o processo de identificação dos caracteres através do mecanismo *OCR* associado (Figura 3.17f).

Com o resultado descrito anteriormente pronto, bastou relacioná-lo como parâmetro ao método "image_to_string" da biblioteca "pytesseract", para obter o resultado da identificação dos caracteres da placa. Foram necessárias algumas configurações específicas para otimizar o reconhecimento de acordo com o cenário de busca deste trabalho. A primeira consiste na definição do modo de segmentação de página, escolhendo-se a opção de tratar a imagem como uma única linha de texto (no código, "*–psm 13*"), modo do mecanismo *OCR* definido como o padrão disponível no sistema (no código, "*–oem 3*") e um filtro aplicado para verificação apenas de uma determinada lista de caracteres: de A à Z (maiúsculas), 0 à 9 e o caractere especial "-".

3.2.6.2 Interface Gráfica do Usuário

A última parte do projeto do *software*, constituiu do desenvolvimento da interface gráfica completa e integrada da aplicação, baseada nas funcionalidades previstas inicialmente pelo autor e representadas nas Figuras 3.10 e 3.11. Outros adicionais, vistos como obrigatórios durante o processo, também foram incluídos, e serão descritos posteriormente.

Para tal, criou-se o arquivo '*Interface.py*', que, basicamente, define o tipo, posição e comportamento de todos os elementos visuais da aplicação, distribuídos em diferentes janelas, e os inicializa em uma única tela com área compartilhada. Foi definido pelo autor, a distribuição de telas através de abas, para facilitar a navegação do usuário. Essas, abrangem funcionalidades gerais de inserir, modificar, excluir e listar proprietários e veículos, realizar a ação de identificação e vínculo da placa com o respectivo proprietário (caso o mesmo esteja previamente cadastrado) e exclusão e listagem de registros de entradas, apresentando as informações de data e hora da ação e dados sobre proprietário e

seu veículo.

Inicialmente, realizou-se o processo de codificação da janela principal, que recebe todas as abas da aplicação. Para esse fim, a classe instanciada foi a *Tk*, advinda do kit de ferramentas *GUI Tkinter*. Vinculada a essa classe, foi necessário instanciar uma outra, nomeada *Notebook*, que recebe como parâmetro único a instância de *Tk*. A mesma, fornece a capacidade de adicionar abas à janela principal de forma prática e direta. Na sequência, o foco passou a ser a elaboração de cada uma das abas através do uso da classe *Frame*. Tal classe abstrai métodos para adicionar, organizar e customizar elementos em uma tela em branco. As classes que compõem os elementos interativos utilizadas em todos os *Frames* desse projeto são: *Label* (campo fico de texto), *Entry* (campo editável de texto), *Button* (botão convencional vinculado a ação de clique), *RadioButton* (botão de escolha), *Treeview* (visualização de conteúdos em formato de tabela) e *OptionMenu* (caixa seletora de opções, no padrão *dropbox*).

Dentre as abas criadas, começou-se pela definição da tela inicial do usuário. Esta, pode ser dividida em quatro partes, de acordo com a orientação da página: esquerda, direita, topo e a baixo. À esquerda, apresentam-se os campos de preenchimento de letras e números que integram a placa a ser identificada, juntamente de um botão 'Localizar'. Esse botão, se vincula a ação de efetuar a captura da imagem do veículo que solicita a entrada e executar o processo de identificação e busca pelas informações de proprietário e veículo relacionados. No topo, estão contidos os campos que definem as características do proprietário e à direita as particularidades do veículo. A baixo, são mostrados, em formato de tabela, os cinco últimos registros de entradas efetuadas.

Na segunda aba, são exibidas as funções de cadastro de proprietários e veículos, dispondo os campos necessários à esquerda e direita, respectivamente. Vale ressaltar que, o campo de CPF do proprietário a ser preenchido fornece apenas opções disponíveis na base de dados local da aplicação, sendo atualizada de acordo com as alterações (inserção, modificação ou remoção) na tabela de proprietários.

Já a terceira aba, apresenta ações de alteração e remoção de proprietários e veículos previamente registrados no sistema. O processo se inicia pela pesquisa da inserção a qual se deseja modificar, onde proprietários podem ser buscados pelos filtros 'Apartamento', 'Nome', 'CPF' ou 'ID' e veículos por 'Placa', 'CPF Proprietário' ou 'ID'.

A quarta aba, expõe em detalhes no formato de tabela todos os registros de entradas no ambiente e a quinta, as características gerais dos proprietários e veículos cadastrados, respectivamente.

3.2.6.3 Criação e Gerenciamento do Banco de Dados

Ao perceber a necessidade de se armazenar informações geradas pelo usuário durante a utilização do programa a longo prazo, foi primordial a escolha de um banco de dados a ser implementado na aplicação. Porém vários modelos podem ser abstraídos utilizando a linguagem *Python*, e, por isso, pesquisou-se sobre as possibilidades e suas principais vantagens e desvantagens de uso.

O mecanismo de base de dados selecionado foi o *SQLite*. Este, já vem previamente instalado junto ao núcleo do *Python* de forma gratuita, apresenta código aberto, possui boa documentação e é compatível com os principais *browsers* e sistemas operacionais *desktop* e *mobile* do mercado. Segue o padrão relacional, considerado rápido, leve (com todos os recursos habilitados, o tamanho da biblioteca pode ser inferior a 600 KiB), independente (não apresenta a necessidade de instalação de nenhuma biblioteca externa para uso), de alta confiabilidade e recursos completos. Como a quantidade de dados a ser armazenada localmente dentro da aplicação é considerada pequena, tendo em vista as funcionalidades disponíveis ao usuário e a demanda de inserções a serem feitas durante a vida útil do programa, esta se mostrou como uma ótima escolha. Todos os dados gerados são armazenados em um único arquivo dentro do disco rígido onde o *software* está instalado (SQLITE, 2021).

Após ser escolhido, foram então definidas as estruturas de dados obrigatórias para compôr o *software* em desenvolvimento: *proprietários, veículos e entradas*. Cada uma dessas, passou a ser representada por uma tabela inserida no banco de dados no momento de instalação do programa na máquina do usuário (arquivo nomeado como *gerencial.db*). Os campos determinados para a composição de cada uma das tabelas citadas anteriormente, bem como o formato dos dados de cada um desses, são detalhados logo a baixo, na Figura 3.18.

Figura 3.18: Estruturas de tabelas do banco de dados SQLite: proprietários, veículos e entradas.

Feito isso, foram implementados os arquivos '*BDEntradas.py*', '*BDProprietario.py*' e

'*BDVeiculo.py*', contendo toda a abstração de lógica necessária para a conexão com o banco de dados e criação e acesso das tabelas respectivas, referenciadas na nomenclatura de cada um dos arquivos. Após isso, foram gerados também '*Entrada.py*', '*Proprietario.py*' e '*Veiculo.py*', abstraindo as características das entidades e ações de busca, inserção, alteração e remoção das mesmas no banco.

Capítulo 4

Resultados

O atual capítulo, descreve em detalhes os resultados gerais obtidos no projeto, divididos nas seções *Hardware* e *Software*.

4.1 *Hardware*

Seguindo as diretrizes do projeto, a primeira atividade realizada foi a montagem do *hardware* para aquisição das imagens a serem aplicadas nos testes dos algoritmos de reconhecimento, como descrito na Seção 3.1.5. Os componentes utilizados em sua estrutura foram:

- 01 *Rapsberry Pi 3 Model B*;
- 03 Dissipadores de calor;
- 01 Cartão MicroSD 16GB;
- 01 *Case* para *Raspberry Pi 3* + módulo de câmera;
- 01 Módulo *Raspberry Pi Camera Rev 1.3 5MP*;
- 01 Bateria *Power Bank* 10.000 mAh;

Acoplaram-se manualmente o *Rapsberry Pi* e o módulo de câmera à *case*, após a anterior conexão física estabelecida entre ambos. Feito isso, fixou-se, através do uso de uma fita dupla face, a bateria externa junto a base da *case*, para fornecer, de forma prática, a alimentação necessária ao conjunto através de um conector micro USB. A comunicação, tanto com o *notebook*, quanto com o *smartphone*, através dos respectivos *softwares* de controle e protocolos de comunicação (*VNC* e *SSH*), foram estabelecidas de maneira correta, como o esperado. A Figura 4.1, apresenta o *hardware* já montado e operante.

Capturaram-se inicialmente um total de 27 imagens consideradas adequadas aos testes computacionais de identificação de contornos de placas, e outras 6 para os testes finais de identificação completa de placas e caracteres, vislumbrando um cenário ideal de aplicação do sistema. A Figura 4.2 e 4.3, mostra uma prévia do conjunto de imagens adquiridas

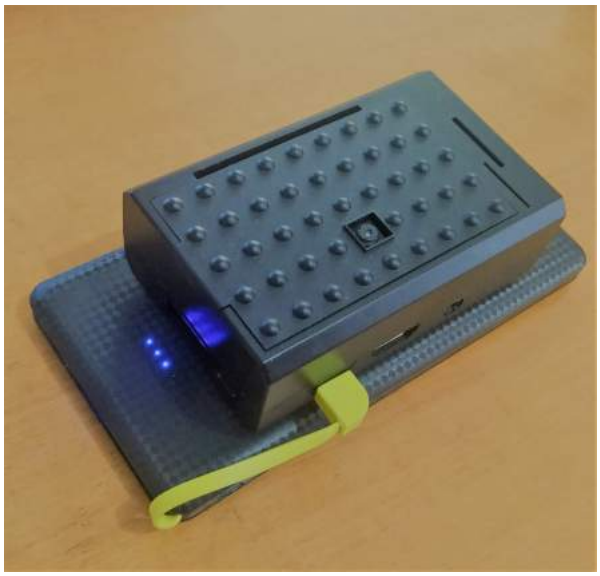


Figura 4.1: Montagem final do *hardware*.

para testes pelo *hardware*, mantendo sempre uma distância aproximada de 80 a 100 cm entre a câmera e o veículo, vinculando às mesmas sempre a resolução máxima de 2592 x 1944 *pixels* fornecida pela câmera em questão.

4.2 *Software*

Primeiramente foram instalados no *notebook*, máquina escolhida para o processo de desenvolvimento, os programas e bibliotecas necessários para criação dos códigos. Todas as ações nessa etapa decorreram conforme o previsto, sendo consultados tutoriais disponibilizados na *internet* para sanar dúvidas gerais, principalmente de instalação e uso.

Ao desenvolver os códigos computacionais, buscou-se previamente definir quais seriam os métodos aplicados no pré-processamento das imagens, como relatado na Seção 3.2.6. Após definida a sequência de técnicas mais adequada, aplicaram-se as mesmas nas imagens obtidas pelo protótipo físico vinculado ao sistema. Os resultados podem ser vistos na Figura 4.4.

As imagens tratadas foram fornecidas como entrada para o próximo algoritmo desenvolvido, referente a técnica de identificação das regiões que contém as placas. Como consequência de sua aplicação no conjunto de imagens visto na Figura 4.4, houve o reconhecimento correto de 19 das 27 regiões de placas que compõem esse conjunto de testes.

Para avaliar com mais detalhes o desempenho geral desse mecanismo, criou-se a Tabela 4.1, que permite analisar minuciosamente as condições da imagem inicial e de sua representação final, após passar pelo método de pré-processamento e posterior tentativa de identificação da placa. As colunas abrangidas nessa tabela, se referem ao nível de luminosidade incidente na região da placa (varia de 0 a 3, onde 0 representa mínima incidência,

Figura 4.2: Imagens obtidas para testes do algoritmo de identificação parcial de placas pela versão do *hardware* desenvolvido.

ambiente muito escuro, e 3 máxima, área extremamente iluminada), cor do veículo (pode gerar um contraste indevido com a placa e dificultar a identificação), porção do veículo em que a placa está contida (frontal ou traseira), identificação realizada ou não com sucesso e área em *pixels* referente ao contorno de placa encontrado (se for, de fato, identificada).

Tabela 4.1: Análise detalhada dos resultados fornecidos pelo algoritmo de identificação de placas.

Arquivo	Luminosidade na Região da Placa	Cor do Veículo	Placa Frontal ou Traseira?	Placa Identificada?	Área do Contorno
001.jpg	3	CINZA	FRONTAL	**SIM**	90.758,0
002.jpg	2	ROXO	FRONTAL	**SIM**	574.184,0
003.jpg	1	ROXO	TRASEIRA	**SIM**	233.154,0
004.jpg	1	PRETO	FRONTAL	**NÃO**	-
005.jpg	2	PRETO	TRASEIRA	**SIM**	216.775,5
006.jpg	3	ROXO	FRONTAL	**SIM**	179.709,5
007.jpg	1	VERMELHO	FRONTAL	**SIM**	305.371,0
008.jpg	3	CINZA	FRONTAL	**SIM**	45.173,5
009.jpg	2	PRETO	TRASEIRA	**SIM**	212.736,0
010.jpg	1	CINZA	TRASEIRA	**SIM**	121.530,5
011.jpg	1	CINZA	FRONTAL	**SIM**	60.001,5
012.jpg	1	VERDE ESCURO	TRASEIRA	**SIM**	288.252,5
013.jpg	2	PRETO	TRASEIRA	**NÃO**	-
014.jpg	3	CINZA	FRONTAL	**NÃO**	-
015.jpg	3	PRETO	FRONTAL	**NÃO**	-
016.jpg	1	PRETO	FRONTAL	**SIM**	317.787,0
017.jpg	1	VERMELHO	TRASEIRA	**SIM**	198.484,5
018.jpg	1	CINZA	FRONTAL	**SIM**	501.887,5
019.jpg	3	CINZA	TRASEIRA	**NÃO**	-
020.jpg	1	CINZA	FRONTAL	**SIM**	169.073,5
021.jpg	3	CINZA	FRONTAL	**NÃO**	-
022jpg	2	VERMELHO	TRASEIRA	**NÃO**	-
023.jpg	1	VERDE	FRONTAL	**SIM**	57.176,0
024.jpg	1	CINZA	FRONTAL	**NÃO**	-
025.jpg	1	CINZA	TRASEIRA	**SIM**	258.225,5
026.jpg	1	CINZA	TRASEIRA	**SIM**	131.639,5
027.jpg	1	CINZA	FRONTAL	**SIM**	58.043,0
TOTAL DE IDENTIFICAÇÕES CORRETAS			**19/27 (70,37%)**		

Figura 4.3: Imagens obtidas para testes de identificação completa de placas e caracteres pela versão do *hardware* desenvolvido.

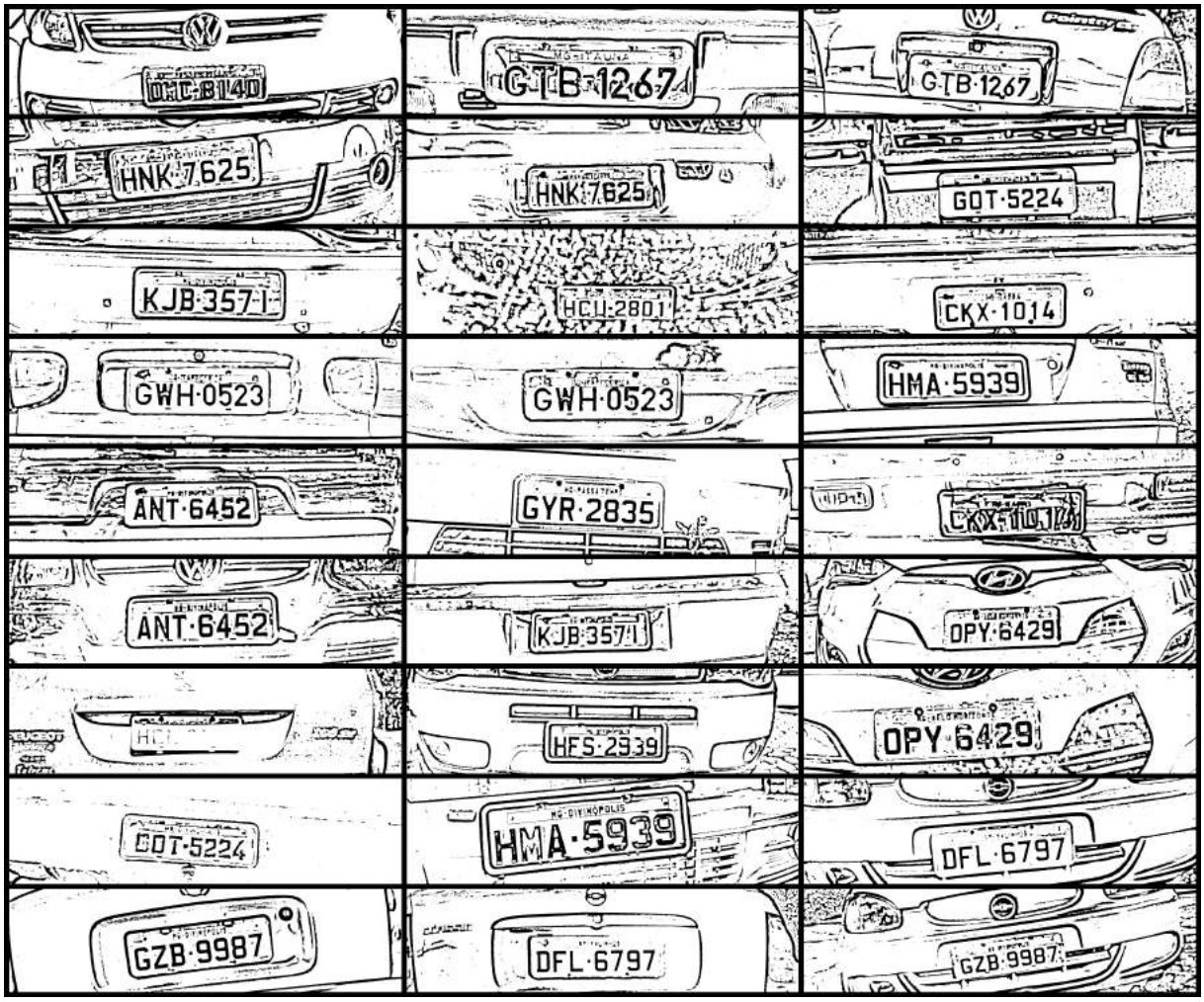


Figura 4.4: Imagens resultantes obtidas após o pré-processamento das imagens originais obtidas para testes.

Vale ressaltar que, dentre as 27 imagens obtidas, 5 delas retrataram intensa luminosidade incidindo diretamente na lente da câmera. Esta condição se mostrou como um obstáculo no momento de identificação dos contornos das placas e demarcação dos perfis dos caracteres, sendo estes casos destacados na Figura 4.5.

Figura 4.5: Resultados do algoritmo de identificação de placas em imagens contendo intensa luminosidade incidente diretamente na câmera.

Na sequência, concluiu-se o desenvolvimento da interface gráfica relacionada as telas do *software*, vinculadas as ações de gerenciamento de um banco de dados local, como apresentado pelo autor nas Figuras 4.6 (aba principal de reconhecimento do veículo e proprietário), 4.7 (cadastro de proprietários e veículos), 4.8 (alterar/remover proprietários e veículos), 4.9 (listar registros de entradas) e 4.10 (listar proprietários e veículos), e descritas com mais detalhes na Seção 3.2.6.2 e 3.2.6.3.

A seguir, são listadas todas as funcionalidades disponibilizadas ao usuário:

- Identificação automática ou inserção manual dos caracteres da placa para efetuar busca na base de dados de veículo e proprietário que solicita a entrada;
- Listagem na aba principal dos últimos 5 registros de entradas efetuados;
- Cadastro de proprietários e veículos;
- Modificação ou remoção de proprietários e veículos;
- Listagem detalhada dos registros de entradas no ambiente;
- Remover registro de entrada (apertar tecla DEL após selecionar o item que se deseja excluir diretamente na tabela);
- Listagem detalhada de todos os proprietários e veículos cadastrados.

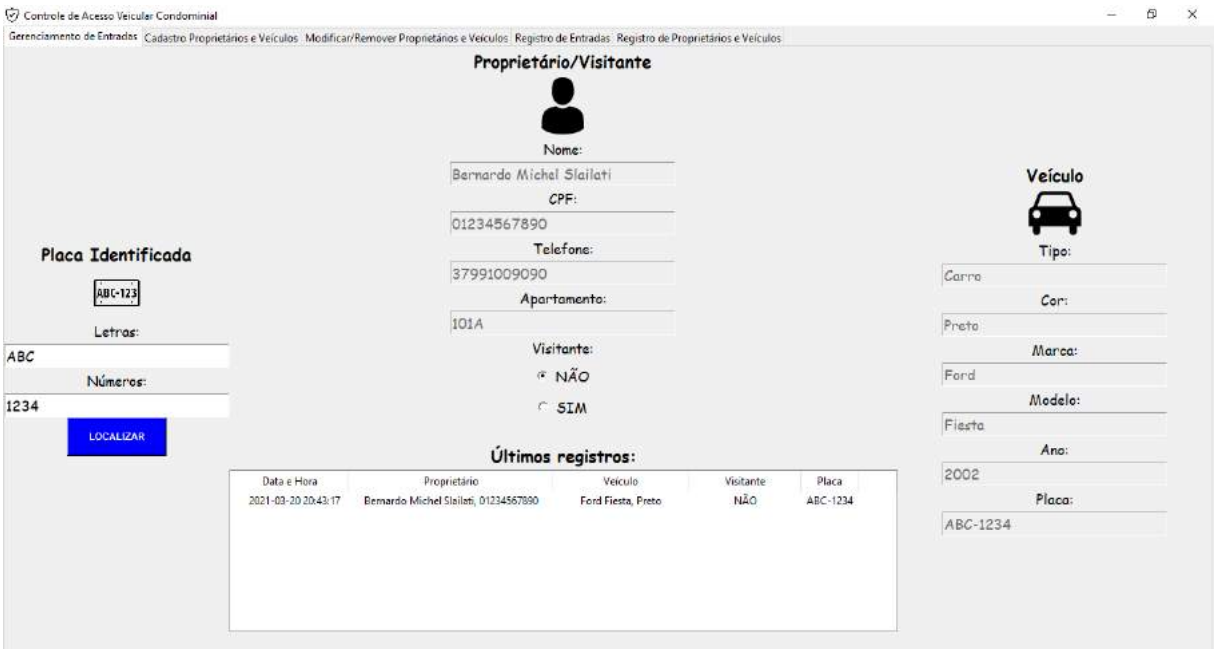


Figura 4.6: Primeira aba: localização da placa, características do proprietário e veículo alvo e últimos registros de entradas.

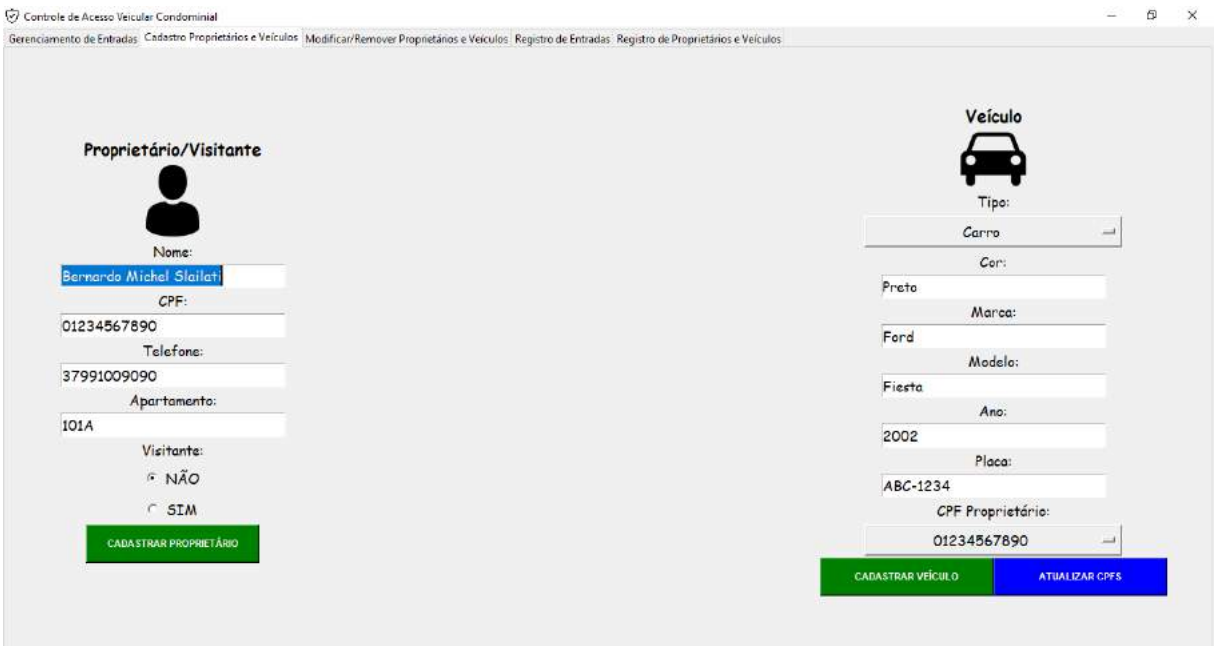


Figura 4.7: Segunda aba: cadastrar proprietários e veículos.

Feito isso, após a conclusão da versão final do *software* e da realização de testes em ambiente de desenvolvimento, instalou-se o programa corretamente no hardware, como mostrado na Figura 4.11.

Por fim, para consolidar a eficácia do algoritmo completo de reconhecimento, foram aplicadas as 6 imagens de veículos consideradas ideais (Figura 4.3) para serem reconhecidas as placas e consequentemente os caracteres, simulando então a utilização do sistema

Figura 4.8: Terceira aba: alterar e remover de proprietários e veículos.

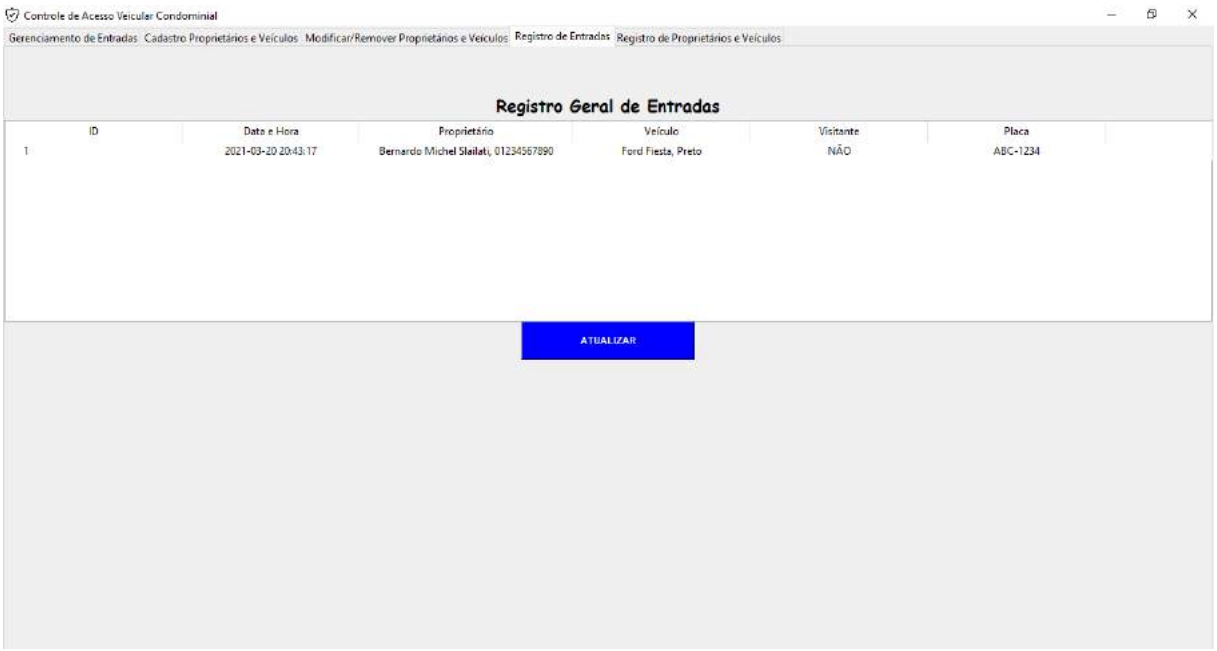


Figura 4.9: Quarta aba: registro completo de entradas.

em uma situação real de aplicação: distância padrão atendida entre câmera e placa (cerca de 80 a 100 cm, vertical e horizontalmente) e posicionamento da câmera alinhado à placa. A sequência de técnicas de reconhecimento aplicadas foi: pré-processamento para encontrar a placa, identificação da placa, pré-processamento para identificar os caracteres e identificação dos caracteres. Como resultado, obtiveram-se as seguintes respostas:

Nota-se que nenhum dos caracteres da placa *PWT-0506* foram identificados no processo de segmentação (Figura 4.12c), devido a um dobramento horizontal ao longo do

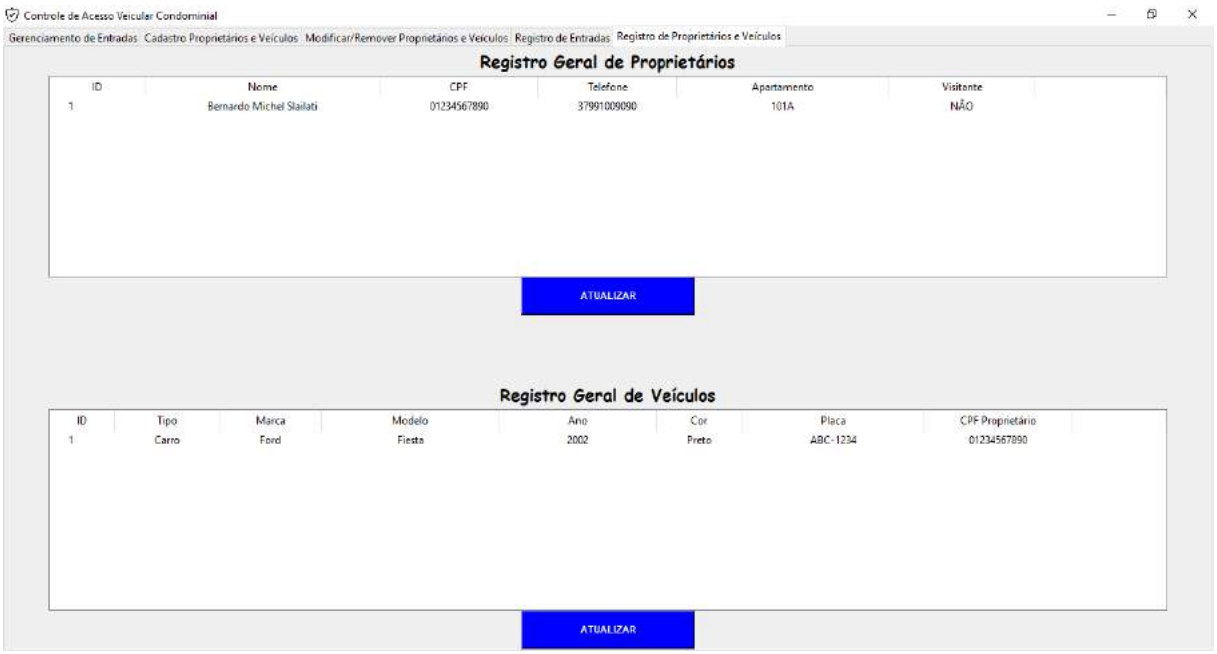


Figura 4.10: Quinta aba: registro completo de proprietários e veículos.

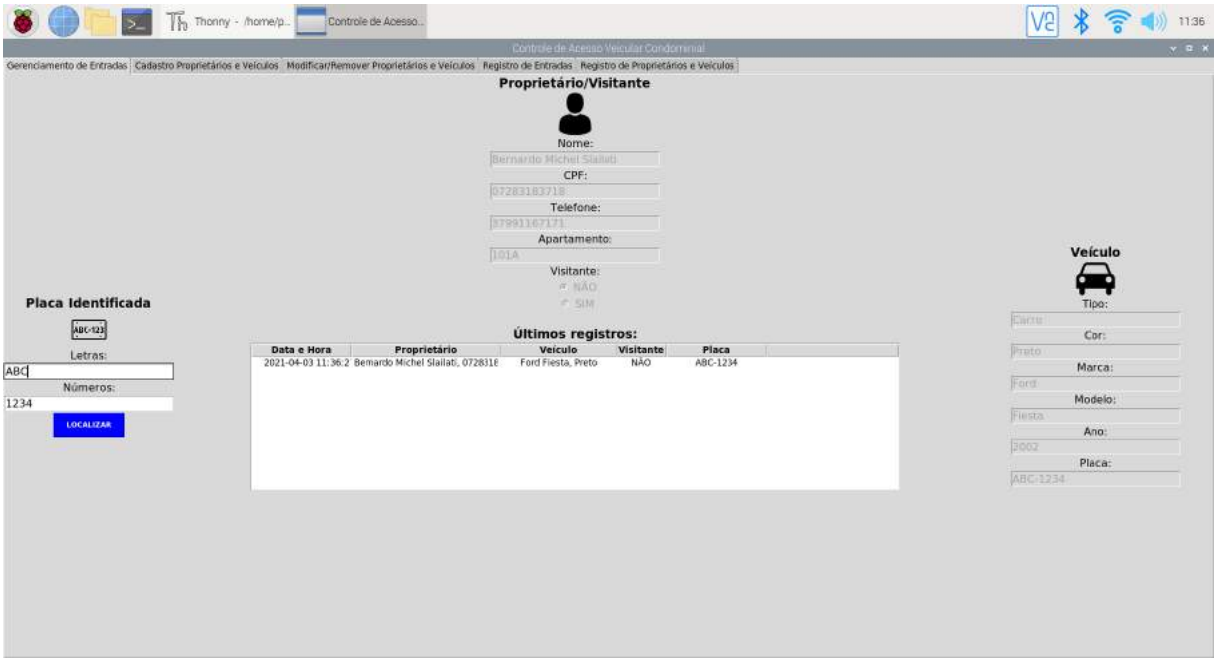


Figura 4.11: *Software* operando corretamente no *hardware*.

centro da placa, que faz com que a porção superior reflita a luz incidente de forma diferente da parte inferior, resultando no pré-processamento incorreto dos algoritmos de identificação. Já as placas *GXO-0991* e *QUL-0097*, apresentaram 1 de seus 7 caracteres incorretos: *9* (encontrado *8*) e *Q* (encontrado *L*), respectivamente. Analisando as respostas do pré-processamento nessas placas, Figura 4.12a e Figura 4.12b, pode-se perceber que o erro ocorreu devido a sombra envolvida nesses caracteres terem feito com que suas proporções, no momento de aplicação do desfoque, se alterassem a ponto de interferir

Tabela 4.2: Análise detalhada dos resultados fornecidos pelo algoritmo de identificação de caracteres.

Caracteres Reais da Placa	**Caracteres Identificados**
GXO-0991	GXO-0981
PYW-4509	PYW-4509
DDY-0710	DDY-0710
QUL-0097	LUL-0097
HMK-0506	HMK-0506
PWT-0506	-
TOTAL DE IDENTIFICAÇÕES CORRETAS	**36/42 (78,57%)**

diretamente no reconhecimento correto das letras e números. Isso mostra o quanto o processo de isolamento da região de interesse, remoção de ruídos e destaque de contornos importantes é crucial para a efetividade dos resultados.

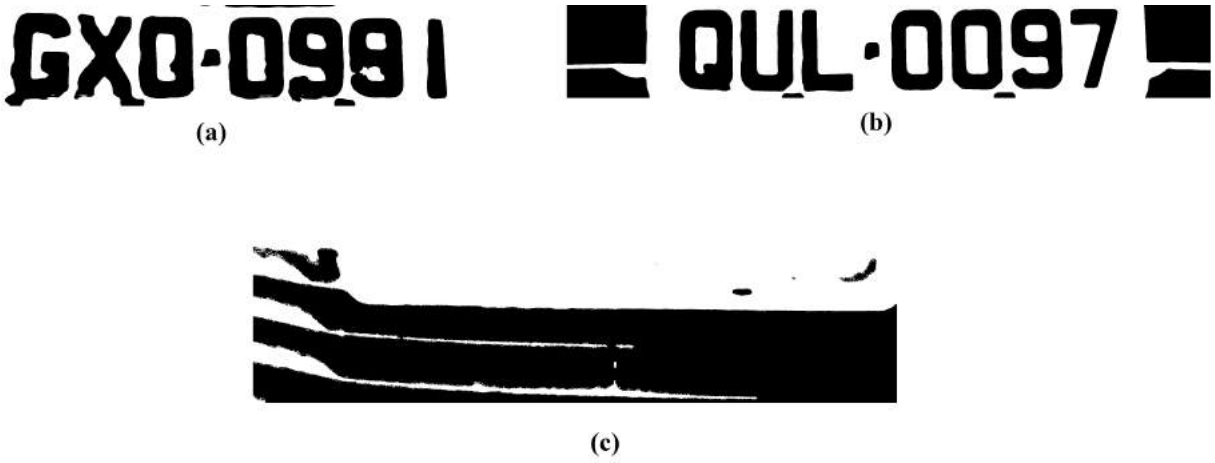


Figura 4.12: Placas identificadas com caracteres errados: .

Capítulo 5

Considerações Finais

A seguir, serão citadas as etapas concluídas do projeto, relacionadas com os objetivos específicos estipulados inicialmente neste trabalho - vistos na seção 1.3. Concerne-se posteriormente, sobre o que de fato foi efetuado, quais metas foram concluídas parcialmente e por quê, e futuras otimizações que poderão ser incluídas neste projeto.

5.1 Conclusões

Como base deste trabalho, contextualizou-se a problemática a ser tratada baseando em pesquisas gerais e específicas sobre o tema segurança, com a intenção de definir os objetivos a serem cumpridos pelo sistema proposto. Após serem devidamente estipulados, realizaram-se estudos sobre a tecnologia que fundamenta a maioria dos algoritmos do sistema almejado: a visão computacional. Foram realizadas a revisão de literatura sobre este tema, juntamente a fundamentação teórica de métodos específicos desta área, utilizados ao longo do desenvolvimento do projeto.

A prototipação do *hardware* foi efetuada com sucesso, ao serem escolhidos e adquiridos os componentes que, após estudos, pesquisas e comparativos entre sistemas semelhantes, se mostraram obrigatórios para concepção do mesmo. Porém, notou-se que ainda poderão ser acoplados outros elementos estruturais para conceder ao sistema melhores condições de uso em uma próxima versão (sensor de presença, sensor de luminosidade, *LEDs*, elementos mecânicos estruturais para otimizar o *design*, etc.). As imagens aplicadas em todos os testes feitos, foram capturadas com sucesso pelo conjunto, caracterizando uma parte importante dos resultados iniciais esperados. O valor total aproximado dos itens que compõem por completo o *hardware*, são mostrados na Tabela 5.1, se mostrando condizentes com o objetivo inicial do autor de gerar uma solução relativamente barata, se comparado as demais tecnologias semelhantes presentes atualmente no mercado.

As ferramentas computacionais a serem utilizadas durante o processo de desenvolvimento dos códigos foram convenientemente escolhidas, baseando-se em características

Tabela 5.1: Estimativa de valores aproximados dos componentes de *hardware* (com base em orçamento feito na data de 25/03/2021).

Item	Quantidade	Preço
Raspberry Pi 3 Model B + Dissipadores + Cartão SD 16GB	1	R$ 330
Suporte + Case	1	R$ 150
Módulo Câmera Raspberry Pi 3 Model B	1	R$ 45
Bateria Power Bank 10.000mAH	1	R$ 70
Teclado com fio	1	R$ 40
Mouse com fio	1	R$ 30
Monitor 18.5"	1	R$ 500
Cabo HDMI $10 \sim 20m$	1	R$ $40 \sim 60$
TOTAL: R$ 1335 ~ 1355		

como funcionalidades disponíveis, compatibilidade de recursos, conteúdo documental fornecido e experiência de uso do autor. Com isso, iniciou-se a criação dos algoritmos que iriam estruturar o futuro *software* do sistema, sendo estes devidamente testados, de acordo com o âmbito voltado para a análise de seus desempenhos. Versões funcionais dos códigos de interface gráfica e algoritmos de pré-processamento, identificação dos contornos das placas e reconhecimento de caracteres em imagens foram desenvolvidos e seus resultados propriamente analisados.

Conceitos envolvendo a criação e manipulação de bancos de dados locais foram aprendidos e integrados ao *software* corretamente, viabilizando assim todas as funcionalidades previstas para a execução do programa. Essa abstração, juntamente a todos os algoritmos de identificação citados no parágrafo anterior, geraram como resultado um *software* totalmente funcional e que supre as demandas de utilidades estipuladas no projeto para o usuário final.

Assim, os objetivos específicos concluídos, de acordo com a Seção 1.3, são mostrados a seguir:

- ✓ Pesquisar viabilidade de mercado e estudo de caso para a situação a ser abordada (segurança em ambientes condominiais), com o intuito de adquirir maiores conhecimentos sobre o problema a ser resolvido;
- ✓ Realizar estudos sobre as técnicas existentes para identificação de placas veiculares (formas retangulares específicas) e caracteres próprios contidos em imagens;
- ✓ Desenvolver protótipo físico eficaz para embarque dos componentes de *hardware*;
- ✓ Desenvolver algoritmos de reconhecimento das placas, identificação dos caracteres, interface para controle das funcionalidades do sistema e gerenciamento e estruturação dos bancos de dados;

- ✓ Executar testes visando obter a porcentagem de acertos dos algoritmos de reconhecimento e realizar prováveis correções, a fim de conceder maior precisão e exatidão aos resultados (EM PARTES) (∗);
- ✓ Desenvolver software integrado que caracterize o sistema, relacionando os algoritmos de identificação e gerenciamento de bancos de dados à interface gráfica a ser disponibilizada para o usuário.

(∗) Necessário ainda obter uma quantidade maior de imagens de placas no padrão considerado ótimo para realização de testes. Além disso, é visto como necessário gerar correções no algoritmo de reconhecimento de caracteres, devido a necessidade de realização de estudos ainda mais aprofundados sobre o processo completo de possíveis condições na qual a placa possa estar (condições extremas de luminosidade, borrões, sujeira ou ausência de tinta em partes da placa, intempérie como chuva constante, etc.).

5.2 Propostas de continuidade

As principais melhorias a serem levadas em consideração, definidas com base na experiência adquirida durante o decorrer do projeto e também de acordo com as pendências vistas como mais importantes após a conclusão deste trabalho, são:

1. Englobar a identificação de placas e caracteres de outros tipos de veículos, como motos, ônibus, táxis e caminhões (formato e cores são diferentes das convencionais);
2. Implementar tratativa de luminosidade incidente na placa (ambiente extremamente claro ou escuro, como, por exemplo, o farol do veículo aceso/apagado à noite, e até mesmo a influencia da chuva);
3. Aprimorar *layout* da interface gráfica utilizada no sistema, visando atingir um padrão mais moderno e intuitivo;
4. Incrementar novos periféricos ao *hardware*, como sensor de presença e luminosidade, a fim de abstrair informações sobre o cenário ao qual a placa está envolvida no momento da captura e possibilitar diferentes tratativas da imagem automaticamente, otimizando o processo de identificação;
5. Acoplar um novo algoritmo de identificação, visando reconhecer também o novo modelo de placas veiculares brasileiras (padrão Mercosul), que hoje é obrigatório em todo o território nacional.

Apêndice A

Códigos

A.1 Disponível em

`https://github.com/BernardoSlailati/codigostcc`

Referências

BARROW, H. *et al*. *Computer vision systems*. SRI International, 1978.

BEARDSLEY, P.; TORR, P.; ZISSERMAN, A. 3D model acquisition from extended image sequences. In: EUROPEAN CONFERENCE ON COMPUTER VISION. *Anais...* 1996. p.683–695.

BESL, P. J.; JAIN, R. C. Three-dimensional object recognition. *ACM Computing Surveys (CSUR)*, v.17, n.1, p.75–145, 1985.

BJÖRKLUND, T. *et al*. Robust License Plate Recognition using Neural Networks Trained on Synthetic Images. *Pattern Recognition*, 2019.

BMVA. *BMVA - The British Machine Vision Association and Society for Pattern Recognition*. Disponível em `http://www.bmva.org/visionoverview`. Acessado em: 25 abr. 2019.

BODEN, M. A. *Mind as machine*: a history of cognitive science. Oxford University Press, 2008. 781p.

BOVIK, A. C. *The essential guide to image processing*. Academic Press, 2009.

BRADSKI, G.; KAEHLER, A. *Learning OpenCV*: computer vision with the opencv library. "O'Reilly Media, Inc.", 2008.

BURT, P.; ADELSON, E. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on communications*, v.31, n.4, p.532–540, 1983.

DAVIS, L. S. A survey of edge detection techniques. *Computer graphics and image processing*, v.4, n.3, p.248–270, 1975.

DEBEVEC, P. E.; TAYLOR, C. J.; MALIK, J. *Modeling and rendering architecture from photographs*. University of California, Berkeley, 1996.

DIGITAL SECURITY. *SVA Tech é eleita uma das três melhores startups de computer vision*. Disponível em `https://revistadigitalsecurity.com.br/sva-tech-e-eleita-uma-das-tres-melhores-startups-de-computer-vision/`. Acessado em: 28 mar. 2019.

FAPESP. *FAPESP e Microsoft oferecem até R$ 4,5 milhões para apoio à pesquisa sobre visão computacional aliada à Inteligência Artificial*. Disponível em `http://www.fapesp.br/11261`. Acessado em: 24 mar. 2019.

FAUGERAS, O.; HEBERT, M. The representation, recognition, and positioning of 3-D shapes from range data. In: *Machine Intelligence and Pattern Recognition*. Elsevier, 1986. v.3, p.13–51.

FENG, Y.; LI, S.; PANG, T. Research and System Design of Intelligent License Plate Recognition Algorithm. *2018 37th Chinese Control Conference (CCC)*, p.9209–9213, 2018.

FISCHLER, M. A.; ELSCHLAGER, R. A. The representation and matching of pictorial structures. *IEEE Transactions on computers*, n.1, p.67–92, 1973.

FUNG, J.; MANN, S. Using graphics devices in reverse: gpu-based image processing and computer vision. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO, 2008. *Anais...* 2008. p.9–12.

G1. *Roubos e furtos a condomínios crescem 56% no estado de SP em 2018*. Disponível em `https://g1.globo.com/sp/sao-paulo/noticia/roubos-e-furtos-a-condominios-crescem-56-no-estado-de-sp-em-2018.ghtml`. Acessado em: 22 mar. 2019.

GALLUP. *Gallup Global Law and Order Report*. Disponível em `https://www.insightcrime.org/wp-content/uploads/2018/06/Gallup_Global_Law_And_Order_Report_2018.pdf`. Acessado em: 20 mar. 2019.

GEOVISION. *Leitura de Placas Veiculares - GV-LPR*. Disponível em `http://www.geovisiondobrasil.com.br/products.php?product=GV%252dLPR`. Acessado em: 28 mar. 2019.

GONZALEZ C. RAFAEL, W. C. R. *Processamento de Imagens Digitais*. Blucher, 2000.

JIANG, S. Mechanical Parts Defect Detection Approach Based on Computer Vision Technology. In: INTERNATIONAL CONFERENCE ON SMART CITY AND SYSTEMS ENGINEERING (ICSCSE), 2016. *Anais...* 2016. p.582–586.

JOHN, C. A Computational Approach to Edge Detection. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTEKKUGENCE*, v.8, n.6, 1986.

KIL, I. *et al*. Development of computer vision algorithm towards assessment of suturing skill. In: IEEE EMBS INTERNATIONAL CONFERENCE ON BIOMEDICAL & HEALTH INFORMATICS (BHI), 2017. *Anais...* 2017. p.29–32.

KUMAR, V. *et al*. Computer vision based object grasping 6DoF robotic arm using picamera. In: INTERNATIONAL CONFERENCE ON CONTROL, AUTOMATION AND ROBOTICS (ICCAR), 2018. *Anais...* 2018. p.111–115.

LIN, C.-H.; LIN, Y.-S.; LIU, W.-C. An efficient license plate recognition system using convolution neural networks. In: IEEE INTERNATIONAL CONFERENCE ON APPLIED SYSTEM INVENTION (ICASI), 2018. *Anais...* 2018. p.224–227.

MARQUES FILHO, O.; NETO, H. V. *Processamento digital de imagens*. Brasport, 1999.

NALWA, V. S.; BINFORD, T. O. On Detecting Edges. *IEEE transactions on pattern analysis and machine intelligence*, n.6, p.699–714, 1986.

NISHANI, E.; ÇIÇO, B. Computer vision approaches based on deep learning and neural networks: deep neural networks for video analysis of human pose estimation. In: MEDITERRANEAN CONFERENCE ON EMBEDDED COMPUTING (MECO), 2017. *Anais...* 2017. p.1–4.

OPENCV. *OpenCV*. Disponível em `https://opencv.org/about/`. Acessado em: 26 abr. 2019.

ROBERTS, L. Machine perception of three-dimensional solids In: tippet et al., eds. *Optical and Electro-optical Information Processing*, p.159–197, 1965.

ROSENFELD, A. Quadtrees and Pyramids for Pattern Recognition and Image Processing. In: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 5. *Proceedings...* 1980. p.802–809.

ROSENFELD, A. *Multiresolution Image Processing and Analysis*. Springer Science & Business Media, 1984. v.12.

SANTHI, P. V. *et al*. Sensor and vision based autonomous AGRIBOT for sowing seeds. In: INTERNATIONAL CONFERENCE ON ENERGY, COMMUNICATION, DATA ANALYTICS AND SOFT COMPUTING (ICECDS), 2017. *Anais...* 2017. p.242–245.

SENTHILKUMARAN, N.; RAJESH, R. Edge detection techniques for image segmentation-a survey of soft computing approaches. *International journal of recent trends in engineering*, v.1, n.2, p.250, 2009.

SQLITE. *SQLite. Small. Fast. Reliable. Choose any three.* Disponível em `https://www.sqlite.org/index.html`. Acessado em: 23 mar. 2021.

SZELISKI, R. *Computer Vision*: algorithms and applications. Springer, 2010.

TSAI, Y.-M. *et al*. An intelligent vision-based vehicle detection and tracking system for automotive applications. In: IEEE INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS (ICCE), 2011. *Anais...* 2011. p.113–114.

VAQUERO, D. A. *Piramides de Imagens*. Instituto de Matemática e Estatística - Universidade de São Paulo, 2004.