PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RECONHECIMENTO DE PLACAS DE CARRO EM UM SISTEMA EMBARCADO

DANIEL ANTONIAZZI AMARANTE E
MATTHIAS OLIVEIRA NUNES

Trabalho de Conclusão I apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Roland Teodorowitsch

Porto Alegre
2016

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

O controle e identificação de veículos é usado nas mais diversas áreas, indo desde serviços de pagamentos automatizados, como pedágios, até aplicações mais críticas, como segurança de fronteiras, sistemas de vigilância de tráfego [2] e sistema de busca por carros roubados. Uma solução para identificar é, inclusive, parte do plano de governo do atual prefeito eleito de Porto Alegre Nelson Marchezan. Ele pretende utilizar os sistemas de controle de velocidade da cidade para também monitorar as placas de carros com o objetivo de identificar carros roubados [8].

Com o crescimento constante da frota de carros no Brasil, aplicações para auxiliar neste trabalho tornam-se cada vez mais nescessárias. Com isso em mente, propõe-se neste trabalho uma solução de aplicação embarcada de reconhecimento de placas, visando a crescente nescessidade de controle na área e as peculiaridades das placas automotivas brasileiras, que impossibilitam a utilização de ferramentas configuradas para placas estrangeiras, nescessitando pesquisas locais neste tema.

O reconhecimento automático de placas de carros (*Automatic License-Plate Recognition*, ALPR) é a extração das informações das placas de veículos a partir de uma imagem ou de uma sequência de imagens. A sua utilização na vida real precisa de um processamento rápido e bem sucedido de placas sob diferentes condições ambientais. Deve-se considerar as diferenças entre as placas de diferentes nações, que terão cores, fontes, símbolos, padrões e línguas diferentes. Também é preciso superar casos onde as placas possam estar parcialmente cobertas com sujeira, luzes e acessórios dos carros, e também a iluminação do ambiente e qualidade da imagem adquirida. [9]

O objetivo deste trabalho é desenvolver um software embarcado em um Raspberry Pi (um computador de tamanho muito pequeno) equipado com módulo de câmera que seja capaz de reconhecer veículos e identificá-los baseando-se em sua placa. Todo o processamento deve acontecer em tempo real com base nas imagens da câmera captadas no momento. A informação coletada e processada deve ser enviada para um servidor que possa utilizar estes dados em tempo real.

A solução proposta tem como diferencial o fato de permitir uma análise e processamento das imagens em tempo real. Isso será realizado utilizando um software embarcado, que estará coletando as imagens ao mesmo tempo que as analisa, enviando o dado já processado para um servidor, ocupando menos largura de banda na transmissão. Com essa abordagem, será possível que esses dados sejam úteis para uma tomada de decisão imediata das "autoridades"de controle de tráfego ou policiamento.

Um exemplo de uma aplicação, onde a velocidade e a disponibilidade imediata das informações é crucial para a viabilidade do produto, seria em um *software* de identificação de carros roubados. O sistema analisaria as placas dos carros que trafegam em uma rodo-

via e identificaria quais daqueles carros têm placas que correspondem a placas de carros roubados. Para que o *software* seja eficiente é nescessário que o processamento seja feito em tempo real, qualquer atraso na identificação do veículo permite que ele se distancie muito, impedindo que as autoridades reajam a tempo.

Uma abordagem alternativa seria fazer a gravação das imagens em um momento, e o processamento em outro. Entretanto, desta maneira, a utilidade do *software* seria limitada, pois na maioria das aplicações reais estes dados precisam estar disponíveis imediatamente.

Com o resultado da implementação proposta será feita uma prova de conceito utilizando o estacionamento da PUCRS, se for possível adquirir permissão. Posicionando o computador com a câmera em um ponto estratégico do estacionamento, serão analisados os carros que passarem. O programa então coletará informações referentes ao número de carros que passaram por aquele ponto e suas respectivas placas.

O fluxo esperado do software é o seguinte. O computador Raspberry pi ficará postado em uma rota de fluxo frequente de carros. Ele ficará posicionado de maneira que consiga capturar imagens das placas em boa qualidade com sua câmera. Será nescessário o módulo de câmera e de bateria, ou haver uma fonte de energia por perto. Ao capturar as imagens, o computador irá processá-las localmente, seguindo os passos do reconhecimento de placas e contagem de carros. Após obtida, a informação será enviada para um servidor simples que irá armazená-la. Para o envio das informações, o computador também precisará estar equipado com algum módulo de Internet.

Este trabalho está organizado da seguinte maneira: no capítulo 2, serão apresentados trabalhos relacionados aos temas de reconhecimento de placas e contagem de carros. No capítulo 3, será apresentado o modelo proposto. No capítulo 4, será especificado os passos a serem seguidos no desenvolvimento da aplicação, classificando os algoritmos a serem utilizados em cada etapa. No capítulo 5, será demonstrada como deve ser feita a configuração e instalação das ferramentas nescessárias para o desenvolvimento do trabalho. E nos capítulos 6 e 7, serão mostrados o cronograma planejado e os recursos nescessários para a conclusão deste trabalho respectivamente.

2. TRABALHOS RELACIONADOS

Uma solução para detectar e reconhecer placas de licenciamento brasileiras foi proposta por Serro [11] na PUCRS. Neste trabalho foram utilizadas técnicas de segmentação de imagens, histograma, cisalhamento de imagens e reconhecimento ótico de caracteres. A metodologia utilizada consistiu das seguintes etapas, calibração do sistema para definir a região de interesse e o ângulo de cisalhamento, detecção da placa, segmentação dos caracteres e aplicação do reconhecedor ótico de caracteres.

Com a solução proposta por Serro [11] foi obtida uma taxa de acerto de aproximadamente 54%, com tempo médio de execução de 0,062 segundos por imagem. A baixa taxa de acerto pode ter sido por problemas de foco e nitidez, o tamanho da placa em *pixels* nas imagens e a existência de outros objetos em cena. Todos esses problemas foram citados no desenvolvimento do projeto.

Em Ahmad et al. [2] foi feito um estudo comparativo dos sistemas de reconhecimento de placas automotivas automáticos. Segundo estes autores, o processo de ler o conteúdo de uma placa passa por três estágios. O primeiro é a localização ou extração da placa, que consiste no processo de localizar a placa do carro na imagem. O segundo estágio é a separação dos caracteres, onde cada caractere individual é separado dos outros para reconhecimento. E o terceiro e último estágio é o reconhecimento do caractere em si, onde os caracteres extraídos da imagem são identificados.

Neste trabalho foram implementados três diferentes métodos de localização de placa e dois diferentes métodos de reconhecimento de caracteres, resultando em 6 diferentes abordagens para o reconhecimento de placas. Todas essas combinações foram então testadas contra diferentes conjuntos de dados.

Os resultados obtidos por Ahmad et al. [2] na experimentação não foram muito animadores, variando entre 20 e 40 por cento. Um dos motivos para os maus resultados foi a variedade de parâmetros nas imagens do conjunto de dados de teste. Tais parâmetros incluem variações na distância, ângulo, iluminação e ambiente. Acreditamos que estes erros poderiam ser mitigados em sistemas reais com uma câmera com resolução fixa e de boa qualidade. A variação do tamanho da placa afetou o desempenho de alguns algoritmos, mas em uma câmera fixa é possível obter uma consistência e conseguir resultados mais aceitáveis.

Outros motivos para a baixa taxa de acerto foram a falta de pré-processamento das imagens, que a análise não considerou, e a utilização de mais dados de aprendizado para o reconhecimento ótico de caracteres.

Em Abtahi et al. [1] foram feitas novas abordagens para a segmentação de caracteres em imagens. De acordo com eles, o método padrão de segmentação baseado em projeção sofre com variações consideráveis na região da placa ao redor dos caracte-

res, portanto estes autores propuseram duas abordagens. A primeira é feita adaptando um método de aprendizado por reforço, criando um agente que consiga achar os melhores caminhos para a segmentação. A segunda abordagem usa um método híbrido que utiliza a simplicidade e velocidade do método de projeção, mas com o poder do aprendizado por reforço.

De acordo com Wafy e Madbouly [12], o reconhecimento de uma placa consiste em dois mecanismos principais: detecção de uma placa e em seguida a sua identificação. O algoritmo proposto nesse artigo, faz os dois passos e se baseia na distribuição semi-simétrica dos pontos de canto nas imagens de carros e placas, e nas características morfológicas da região da placa. Essa solução teve uma taxa de acerto de 97,5% no processo de detecção e 92,8% na identificação, com o maior tempo de execução para um dos processos sendo de 0,3s. Com estes resultados seria possível utilizar este método em aplicações de tempo real.

Com relação ao uso de sistema embarcado para executar o reconhecimento, Arth et al. [3] trabalharam no desenvolvimento de um sistema de reconhecimento de placas de carro em um processador de sinal digital (*Digital Signal Processor*, DSP). *DSP* são microprocessadores especializados em processar sinais digitais como áudio ou vídeo em tempo real. [13] O processador utilizado neste trabalho específico foi um *Texas Instruments C64* com *1MB* de *cache RAM* e um outro bloco de memória mais lento, *SDRAM* de *16MB*. O processador não possui camera integrada mas permite a conexão de uma fonte de vídeo analógica ou digital. Na solução implementada foi utilizada uma camera com resolução de *352x288 pixels*.

Com sua implementação, Arth et al. [3] foram capazes de conseguir localizar a placa em *7.30 ms*, levando mais aproximadamente *1 ms* para identificar cada caractere. Não é informado, no artigo, a taxa de sucesso de cada reconhecimento. Os autores ainda concluem que por o tempo de detecção da placa ser superior ao tempo de reconhecimento dos caracteres, e que este algoritmo deve ser melhorado.

Analisando os trabalhos feitos na área, nota-se que por mais que os algoritmos variem, a base da detecção de placas permanece parecida. Eles costumam ser divididos em pelo menos duas partes, a localização da placa e a detecção dos caracteres. Inclusive, Ahmad et al [2] misturou diferentes algoritmos, utilizando a localização de um e a detecção de outro, demonstrando que os dois passos são bem independentes.

Pode-se notar que o reconhecimento de placas de carros é uma área bastante estudada, com diversas abordagens diferentes e grande variação de resultados. Entretanto, um problema que ainda existe, é a grande diferença nas placas de diferentes países, no estilo, fonte, caracteres utilizados e padrão do texto. A solução deste problema é a criação de uma solução local de reconhecimento de placas.

O *software open source Openalpr*[1] criou uma solução de reconhecimento de placas que permite que a comunidade contribua treinando o reconhecedor de caracteres a reconhecer as placas de seus países para contornar esse problema.

[1] https://github.com/openalpr/openalpr

3. COMPONENTES

Para o desenvolvimento do trabalho serão utilizados alguns componentes externos, tanto de *hardware* quanto de *software*. Será utilizado o computador *Raspberry Pi* e seu módulo de camera. Será utilizada a linguagem de programação *Python*. Será utilizada a biblioteca de visão computacional *OpenCV* juntamente com outras bibliotecas auxiliares que permitam integrar o *OpenCV* com a linguagem de programação *Python* e o módulo de camera do *Raspberry Pi*. E também será utilizada a ferramenta de reconhecimento ótico de caracteres *Tesseract*.

3.1 Raspberry Pi

Figura 3.1 – O Raspberry Pi 3 model B

Raspberry Pi é um computador construído em uma placa de circuito do tamanho de um cartão de crédito desenvolvido pela Raspberry Pi Foundation[1]. Existem diversos modelos diferentes de Raspberry Pi no mercado, o que será utilizado no trabalho é um dos mais recentes, o Raspberry Pi 3 model B. Será utilizado o sistema operacional *Raspbian*, que é o sistema operacional oficial suportado pela *Raspberry Pi Foundation*.

O computador ainda tem suas limitações, com um processador quad-core ARMv8 de 1.2GHz e apenas 1GB de memória RAM. Pela limitação do *hardware* é possível que algumas aplicações fiquem mais lentas do que ficariam em um computador mais potente, apesar disso, o Raspberry Pi é um computador bem completo e capaz de exercer todas as funções de um computador normal.

O modelo utilizado de Raspberry Pi contém módulo de WiFi embutido, sem a necessidade de periférico, que será utilizado no trabalho para enviar as informações proces-

[1] https://www.raspberrypi.org/

sadas pela Internet. Também será utilizado um módulo externo de câmera para coletar as imagens em tempo real.

3.2 OpenCV

OpenCV (*Open Source Computer Vision Library*) é uma biblioteca *open source* de visão computacional e aprendizado de máquina. Contém mais de 2500 algoritmos otimizados nessas áreas, incluindo algoritmos clássicos e recentes. A biblioteca é escrita nativamente em C++, e dispõe de interfaces para C, C++, Python, Java e MATLAB, suportando os sistemas operacionais Windows, Linux, Android e Mac OS.[2]

No desenvolvimento deste trabalho será utilizada a linguagem de programação *Python*, e algumas bibliotecas são utilizadas para trabalhar com *OpenCV*. A biblioteca *numpy*[3], é uma biblioteca de computação científica em *Python*, que inclui funções de processamento numérico e vetores que são utilizados pelo *OpenCV* para representar as imagens.

Também será utilizada o pacote *picamera*[4], que possui uma interface em *Python* para se comunicar com o módulo de camera do *Raspberry Pi*. É possível utilizar as funções próprias de captura de imagem da camera do *OpenCV*, como por exemplo o `cv2.VideoCapture(0)`, mas será optado por utilizar o *picamera*, por ser uma interface mais específica para a camera utilizada.

Um exemplo de código de captura de imagem utilizando a camera do *Raspberry Pi* em *Python*, utilizando as bibliotecas citadas, pode ser visto abaixo. No exemplo, os quadros são capturados pela camera do *Raspberry Pi*, convertidos para escalas de tons de cinza (*grayscale*), e mostrados na tela, dado que haja um monitor conectado ao computador *Raspberry Pi*. Converter a imagem para *grayscale* é um dos primeiros passos de pré-processamento para iniciar o reconhecimento da placa.

[2] http://opencv.org/
[3] http://www.numpy.org/
[4] https://picamera.readthedocs.io/en/release-1.12/

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import cv2

camera = PiCamera()
rawCapture = PiRGBArray(camera, size=(640, 480))

for frame in camera.capture_continuous(rawCapture, format="bgr",
    use_video_port=True):
   image = frame.array
   gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
   cv2.imshow("Frame", gray_image)
   rawCapture.truncate(0)
```

3.3 OCR

Reconhecimento Ótico de Caracteres (*Optical Character Recognition*, OCR) consiste da conversão de textos em formato de imagem para o formato reconhecido por máquina. É o método mais eficiente para fazer o processamento de imagem para texto de acordo com Mohit et al. [7].

Uma ferramenta conhecida de OCR é o Tesseract[5]. É uma ferramenta *open source* de reconhecimento ótico de caracteres que suporta múltiplas línguas. É essencialmente um algoritmo de comparação de *templates*, e as amostras de caracteres podem ser autotreinadas. [5] Esta ferramenta será utilizada para fazer o reconhecimento dos caracteres neste trabalho.

Um exemplo do uso do *Tesseract* na linguagem de programação *Python* pode ser visto abaixo. É nescessário utilizar uma biblioteca chamada *pytesseract*, que possui uma interface em *Python* para a utilização da ferramenta. O exemplo abaixo lê uma imagem do disco e imprime no console quais os caracteres que foram reconhecidos.

[5]https://github.com/tesseract-ocr/tesseract

```
from PIL import Image
import pytesseract

print(pytesseract.image_to_string(Image.open('license_plate.jpg')))
```

O *Tesseract* pode ser treinado para reconhecer diferentes fontes e línguas. O treinamento deve ocorrer previamente, e não enquanto executa o reconhecimento das placas. Deverá ser coletado um significativo número de exemplos de caracteres como eles serão segmentados pelo algoritmo, e com esse conjunto de treinamento, deve se executar a ferramenta de treinamento do *Tesseract*. Um manual de como utilizar o *software* de treinamento pode ser encontrado no repositório do *Tesseract*[6], onde o *software open source* está disponível.

3.4 Placa de Transito Brasileira

Segundo o código de transito brasileiro [4], todos os veículos são identificados por meio de placas, dianteira e traseira. Elas são identificadas por uma tarja na parte superior contendo a sigla do estado e o nome do município, e pelo código de identificação único, composto por três letras, seguidas por quatro digitos, separados por um hífen.

Veículos particulares, de aluguel, oficial, de experiência, de aprendizagem e de fabricante têm suas dimensões de 130mmx400mm e altura dos caracteres de 63mm(figura 3.2). Caso a placa não caiba no receptáculo ela pode ser reduzida em até 15%. As placas de motocicleta, motoneta, ciclomotor e triciclos autorizados tem dimensões de 136mmx187mm e altura de caracteres de 42mm(figura 3.3).

Figura 3.2 – Placa de um carro

[6] https://github.com/tesseract-ocr/tesseract

Figura 3.3 – Placa de uma moto

A tipologia dos caracteres das placas utiliza a fonte Mandatory 3.4, e as placas de categorias diferentes de veículos são diferenciadas pelas suas cores na tabela 3.1.

Figura 3.4 – Tipografia das placas

Tabela 3.1 – Cores das placas automotivas

Categoria do Veículo	**Cor de Fundo**	**Cor de Caracteres**
Particular	Cinza	Preto
Aluguel	Vermelho	Branco
Experiência/Fabricante	Verde	Branco
Aprendizagem	Branco	Vermelho
Coleção	Preto	Cinza
Oficial	Branco	Preto
Missão Diplomática	Azul	Branco
Corpo Consular	Azul	Branco
Organismo Internacional	Azul	Branco
Corpo Diplomático	Azul	Branco
Organismo Consular/ Internacional	Azul	Branco
Acordo Cooperação Internacional	Azul	Branco
Representação	Preto	Dourado

4. IMPLEMENTAÇÃO

O trabalho será dividido nos quatro passos propostos por S Du [9]: aquisição da imagem, extração da placa, segmentação dos caracteres e reconhecimento dos caracteres. Para a localização da placa e separação dos caracteres serão utilizadas a biblioteca OpenCV e a linguagem de programação Python. O motivo da escolha dessa linguagem se dá por ser uma linguagem muito usada para OpenCV, contendo muita documentação. Para o reconhecimento dos caracteres será utilizado o software Tesseract, havendo a possibilidade de precisar ser treinado para reconhecer a fonte da placa de transito brasileira. As escolhas a serem feitas tem como objetivo maximizar os resultados ao final do trabalho, tentando criar um balanço entre facilidade de implementação e qualidade do reconhecimento.

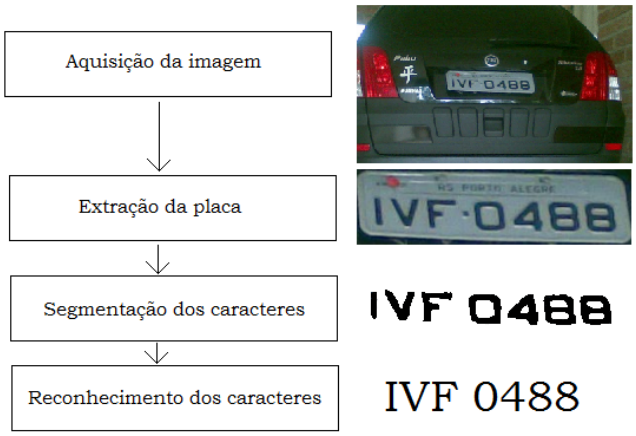


Figura 4.1 – Quatro estágios do reconhecimento de placa

4.1 Aquisição das imagens

O primeiro passo para o reconhecimento de uma placa é a extração das imagens. Diversos fatores externos podem afetar o reconhecimento da placa, como a iluminação, a distância e o ângulo da imagem. A influência desses fatores pode ser minimizada utilizando-se de imagens de qualidade.

A aquisição das imagens será feita utilizando o módulo de câmera do *Raspberry Pi*. É uma câmera de 5 *megapixels* capaz de gravar videos em 1080p. Uma das vantagens de utilizar essa combinação do *Raspberry Pi* e seu módulo de câmera é a sua portabilidade. Por serem pequenos e leves, caso a câmera tenha dificuldade em capturar imagens de qualidade para o reconhecimento das placas, é possível movê-los e colocá-los em posições privilegiadas para otimizar o processo.

4.2 Extração da placa

A fase de extração é a fase mais importante em um sistema de reconhecimento de placas, porque todas as outras fases dependem da exata extração da área da placa. Essa extração é difícil pois influencia na precisão do sistema como um todo [6]. Muitas dificuldades podem ocorrer durante a fase de extração pelos seguintes motivos:

- A eficiência da extração é afetada pela complexidade da cena.
- Diferentes veículos possuem suas placas em diferentes posições.
- Pode ocorrer ruído durante a captura da câmera.
- Condições do tempo pode influenciar no ruído.
- Hora do dia afeta na luminosidade e resulta em erros de contraste.
- Outros caracteres, quadros e parafusos podem introduzir confusão.
- Mal posicionamento da câmera, ou placa, pode resultar em distorção que afeta na eficiência.
- Luminosidade baixa, ou desigual, imagem desfocada, baixa resolução, reflexão, sombra afetam a eficiência da extração.

Kaur [6] propõe um método eficiênte de extração que segue o seguinte fluxograma:

1. Conversão de RGB para escala de tons de cinza
2. Remoção de ruído por filtro bilateral
3. Aumento de contraste usando equalização de histograma adaptativo
4. Operações Morfológicas de Abertura e Subtração de Imagem
5. Binarização da Imagem
6. Detecção de borda pelo operador Sobel
7. Detecção de Área de Placa Candidata por Operações Morfológicas de Abertura e Fechamento
8. Extração da área da placa real
9. Aprimoramento da região extraída

4.2.1 Pré-processamento

O objetivo básico do pré-processamento é melhorar o contraste da imagem de entrada, para reduzir ruído e, em consequência, a velocidade de processamento. O melhoramento do contraste é feito pela equalização do histograma, alongamento de contraste etc. Varios filtros são usados para a remoção de ruído.

4.2.2 Conversão de RGB para escala de tons de cinza

A imagem capturada está no formato RGB. O primeiro passo do pré-processamento é converter essa imagem para a escala de tons de cinza. O objetivo dessa conversão é reduzir o número de cores. Os componentes R, G e B são separados do valor de cor de 24 bits de cada pixel (i,j), e um valor de 8 bits em cinza é calculado.

Figura 4.2 – Imagem em escala de tons de cinza
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

4.2.3 Remoção de ruído por filtro bilateral

O filtro bilateral é um filtro não linear, que preserva as arestas e reduz ruído. Ele funciona substituindo o valor de intensidade de cada *pixel* em uma imagem por uma média ponderada dos valores de intensidade dos *pixels* próximos. O objetivo básico da filtragem é remover ruído e distorção da imagem. O ruído pode ocorrer durante a captura pela câmera ou pelas condições do tempo. No método que Kaur [6] propõe, filtro bilateral iterativo é utilizado para remover ruído. Ele é não linear, provê mecanismo para remoção de ruído enquanto preserva bordas mais efetifamente que o filtro mediano. O filtro mediano funciona substituindo o valor de cada *pixel* pela mediana dos *pixels* vizinhos.

Figura 4.3 – Aplicação de filtro bilateral em uma imagem em escala de tons de cinza
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

Aumento de contraste usando equalização de histograma adaptativo

Contraste é definido como a diferença entre o nível mais baixo e alto de intensidade. Equalização de histograma é o método para distribuir de forma mais efetiva o histograma de *pixels*. Equalização de histograma adaptativo mostra melhor contraste em relação a equalização de histograma.

Figura 4.4 – Aumento de contraste usando equalização de histograma adaptativo
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

4.2.4 Operações Morfológicas de Abertura e Subtração de Imagem

A operação de abertura morfológica é realizada na imagem de escala de cinza aumentada de contraste usando um elemento de estrutura em forma de disco. Na subtração de imagem, a imagem morfológica aberta é subtraída da imagem de escala de cinza aumentada de contraste.

Figura 4.5 – Efeito de abertura
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

Figura 4.6 – Imagem subtraída
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

4.2.5 Binarização da Imagem

Nesta operação a imagem de escala de cinza subtraída é convertida em imagem binária. Em primeiro lugar, o nível de limiar é calculado pelo método de Otsu. O algoritmo do método de Otsu assume que a imagem contém duas classes de pixels seguindo um histograma bi-modal. Ele então calcula o limiar ótimo separando as duas classes para que o seu espalhamento combinado seja mínimo. Tendo o limiar calculado, a imagem de escala de cinza subtraída é convertida em imagem em preto e branco.

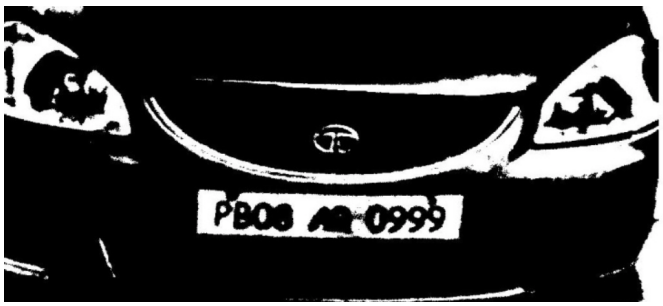


Figura 4.7 – Imagem Binarizada
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

4.2.6 Detecção de borda pelo operador Sobel

A borda vertical é detectada pelo operador sobel e o resultado do operador sobel ser aplicado à imagem binarizada é mostrado na figura 4.8

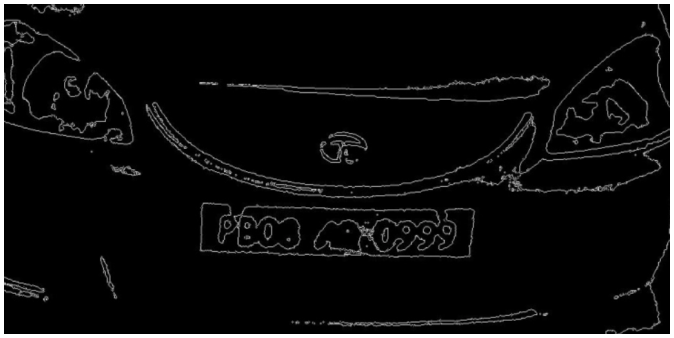


Figura 4.8 – Detecção de borda pelo operador Sobel
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

4.2.7 Detecção de Área de Placa Candidata por Operações Morfológicas de Abertura e Fechamento

Com as operações morfológicas, os objetos indesejados na imagem são removidos. Para a detecção da área da placa, uma operação de dilatação é aplicada na imagem e depois os buracos são fechados. Em seguida, operações de abertura morfológica e erosão são usadas para a detecção exata da área da placa.

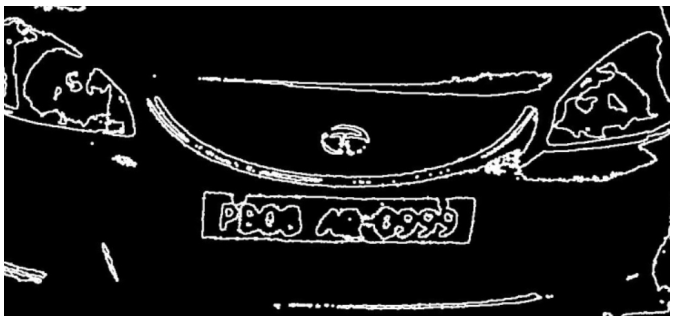


Figura 4.9 – Dilatação morfológica
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

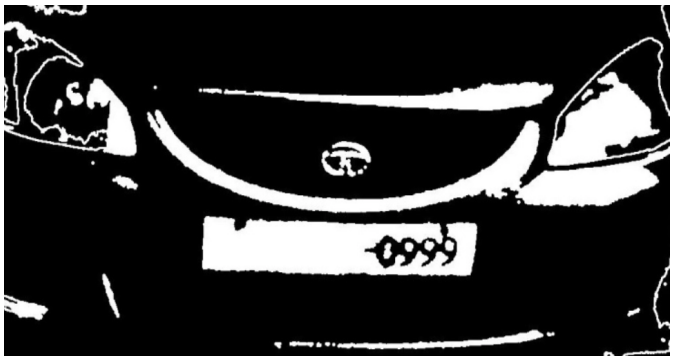


Figura 4.10 – Após preenchimento dos buracos
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

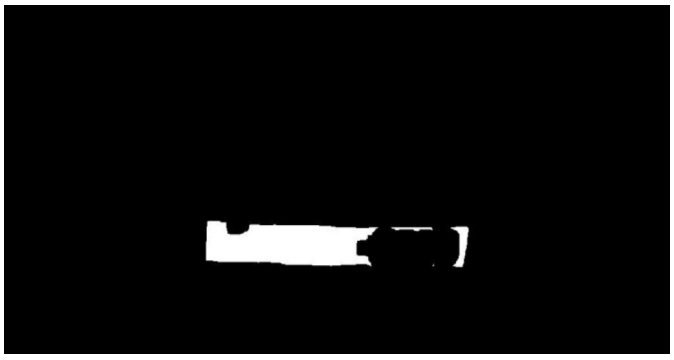


Figura 4.11 – Detecção da área da placa
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

4.2.8 Extração da área da placa real

Após a detecção da área da placa, essa área é extraída da imagem. Em primeiro lugar, os índices de linha e coluna da área da placa são encontrados por análise de componentes conectados.

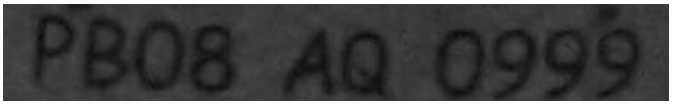


Figura 4.12 – Placa extraída
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

4.2.9 Aprimoramento da região extraída

A placa de número extraída pode consistir em vários ruídos ou furos indesejados.

Figura 4.13 – Placa aprimorada
Fonte: An efficient approach for number plate extraction from vehicles image under image processing [6]

4.3 Segmentação dos caracteres

O terceiro passo para o reconhecimento da placa é a segmentação dos caracteres. A segmentação dos caracteres consiste na extração dos caracteres utilizando estratégias como projetar as suas informações de cores, rotulá-los ou comparar suas posições com modelos. A placa extraída no passo anterior pode conter problemas de inclinação ou iluminação, mas o algoritmo de segmentação deve superar todos esses problemas com pré-processamento. [9]

Badawy [9] faz uma análise dos algoritmos de segmentação mais utilizados com seus prós e contras. Os principais algoritmos utilizados são: segmentação utilizando conectividade de *pixels*, segmentação utilizando perfis de projeção, segmentação utilizando conhecimento anterior dos caracteres, segmentação utilizando contorno dos caracteres e segmentação utilizando características combinadas.

Analisando os resultados foi definido que a segmentação dos caracteres utilizando perfis de projeção foi o mais eficiente, e que mais se encaixa no problema proposto. Sanyuan [10] utiliza essa técnica de segmentação de caracteres, e, utilizando-a juntamente de remoção de ruídos e análise de sequência de caracteres, obteve uma taxa de acerto de 99.2% e uma velocidade de processamento de 10 a 20 milisegundos, o que é bem animador. As vantagens deste método são que a segmentação independe das posições dos caracteres, e consegue lidar bem com rotações. Suas desvantagens são que é afetada ruído na imagem e requer o conhecimento do número de caracteres na placa, o que não será problema devido ao fato de as placas brasileiras terem um número constante de caracteres.

Este método utiliza-se da diferença entre a cor dos caracteres da placa e a cor do fundo da placa, por terem valores diferentes eles têm valores binários opostos em uma imagem binária. Portanto, o método de segmentação consiste em projetar a placa extraída verticalmente para determinar o início e final dos caracteres e depois projetar os caracteres extraídos horizontalmente para extrair cada caracter independente.

Badawy [9] ainda afirma que é evidente que este método de segmentação é o mais comum e o mais simples presente na literatura.

4.4 Reconhecimento dos caracteres

O último passo para reconhecimento da placa é o reconhecimento ótico dos caracteres extraídos. Um dos maiores desafios da extração de caracteres é o fato de os caracteres extraídos não terem o mesmo tamanho e grossura, devido ao *zoom* da câmera. Outro problema, ao criar um reconhecedor de propósito geral, é as diferentes fontes das diferentes placas ao redor do mundo. [9] Este segundo não será um problema para este trabalho pois só há interesse em reconhecer placas brasileiras. Outro desafio é o reconhecimento de caracteres semelhantes, como o D e o 0 [5]. Esse problema será mitigado com o conhecimento prévio das placas brasileiras, sabendo onde é possível haver letras e onde é possível haver números.

Para o reconhecimento de caracteres foi escolhido utilizar o *software Tesseract*. A ferramenta possui execução veloz e eficiente, permite o treinamento para reconhecer novas fontes e suporte para diversas línguas.

5. CONFIGURAÇÃO

O desenvolvimento deste trabalho requer o uso de *hardware* e *software* existentes que nescessitam de um certo nivel de configuração para serem utilizados concorrentemente.

5.1 Raspberry Pi e módulo de câmera

No *Raspberry Pi* será instalado o sistema operacional *Raspbian*. Sua imagem pode ser baixada no site oficial[1]. O módulo de câmera é facilmente acoplado no *Raspberry Pi* e para funcionar só é nescessário habilitar a câmera na ferramenta de configuração do sistema que pode ser acessada utilizando o comando `sudo raspi-config`. Para garantir que a câmera está funcionando pode-se usar o comando `raspistill` para capturar uma imagem.

5.2 OpenCV

A instalação do *OpenCV* no *Raspberry Pi* pode ser feita de diversas maneiras e customizações. Para este trabalho optamos por baixar o código fonte do *github*[2] e instalar utilizando o *software CMake*. Para utilizar o *OpenCV* com *Python*, também é nescessário instalar a biblioteca *numpy*, utilizando o gerenciador de pacotes para programas em *Python pip*.

Uma vez instalado, pode-se acessar a biblioteca *OpenCV* do código *Python* importando o pacote *cv2*. Para integrar a camera do *Raspberry Pi* com o *OpenCV* também é recomendado instalar o pacote *picamera*, mas não nescessário.

5.3 Tesseract

A ferramenta *Tesseract* pode ser instalada no *Raspbian* utilizando o gerenciador de pacotes padrão do sistema, utilizando o comando `sudo apt-get install tesseract-ocr`. Com isso é possível utilizar o *Tesseract* na linha de comando. Para utilizar com *Python*, pode-se instalar o pacote *pytesseract* com o gerenciador de pacotes *pip*, que é um geren-

[1] https://www.raspberrypi.org/downloads/raspbian/
[2] https://github.com/opencv/opencv

ciador de pacotes para programas em *Python*. Para instalar o *pytesseract* executa-se o comando `sudo -H pip install pytesseract`.

6. CRONOGRAMA

O desenvolvimento deste trabalho prevê as seguintes atividades:

I Pesquisar e estudar trabalhos relacionados atuais.

II Estudar as ferramentas e bibliotecas OpenCV, Tesseract e Raspberry Pi.

III Definir modelo e iniciar implementação.

IV Integrar o OpenCV e o Raspberry Pi e implementar uma prova de conceito que utilize os dois.

V Implementar uma aplicação que localize a placa de um carro em uma imagem utilizando OpenCV.

VI Escrever o volume final de TCI.

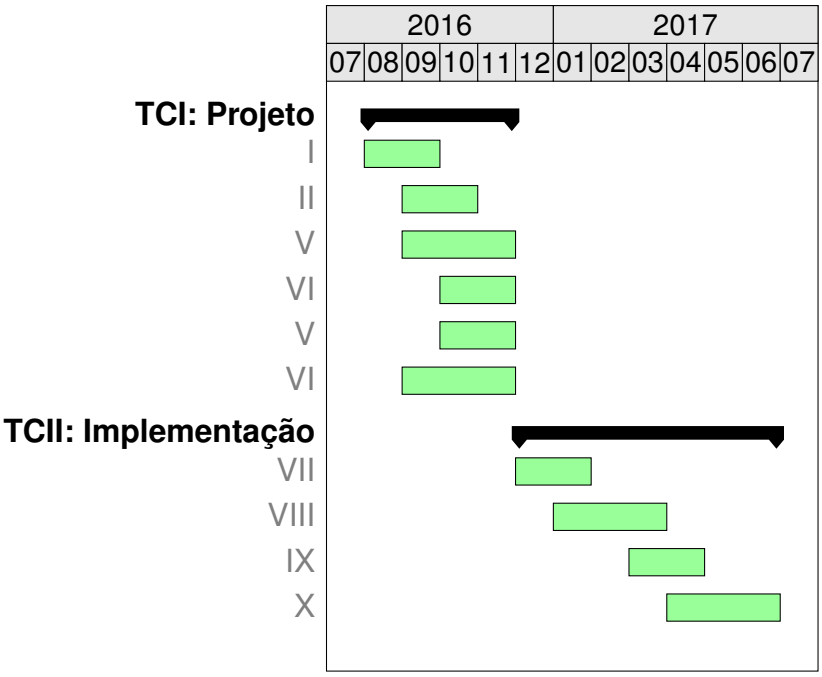
VII Implementar uma aplicação de contagem de carros.

VIII Implementar a aplicação de reconhecimento de placas.

IX Colocar a aplicação a prova em um ambiente real e refinar com base nos resultados.

X Escrever o volume final de TCII.

Estas atividades tem como objetivo definir o modelo que será implementado e iniciar a implementação.



7. RECURSOS NECESSÁRIOS

Estão aqui discriminadas as peças físicas de *hardware* que integrarão o produto final desenvolvido, as bibliotecas e ambientes de programação nescessários no desenvolvimento do projeto, as aplicações que servirão de auxílio, tanto no desenvolvimento do *software* quanto no desenvolvimento do artigo, e os computadores pessoais utilizados no desenvolvimento do trabalho. Para o desenvolvimento do trabalho serão nescessários os seguintes recursos:

- Raspberry Pi.
- Módulo de câmera para Raspberry Pi.
- Ambiente de programação com as linguagens C/C++ e Python.
- Biblioteca OpenCV de visão computacional.
- Sistema LaTeXpara documentação.
- Software de versionamento git.
- Ferramenta para reconhecimento ótico de caracteres Tesseract.
- Editor de texto vi
- IDE Clion e Pycharm.
- Dois computadores pessoais para propósito geral.

8. COMENTÁRIOS FINAIS

Com base na revisão dos trabalhos relacionados, descritos no capítulo 2, e no modelo de implementação definido no capítulo 4, o modelo proposto do presente trabalho tentará criar uma solução eficiente de reconhecimento de placas de trânsito de veículos brasileiros utilizando *software* embarcado. O objetivo do produto final será possuir um *software* que tenha a capacidade de detectar as placas com baixa taxa de erro e alto desempenho, possibilitando que seja utilizado em aplicações reais.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Abtahi, F.; Zhu, Z.; Burry, A. M. “A deep reinforcement learning approach to character segmentation of license plate images”. In: Machine Vision Applications (MVA), 2015 14th IAPR International Conference on, 2015, pp. 539–542.

[2] Ahmad, I. S.; Boufama, B.; Habashi, P.; Anderson, W.; Elamsy, T. “Automatic license plate recognition: A comparative study”. In: 2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), 2015, pp. 635–640.

[3] Arth, C.; Limberger, F.; Bischof, H. “Real-time license plate recognition on an embedded dsp-platform”. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.

[4] BRASIL. “Lei nº 9.503”, 23 de setembro de 1997.

[5] Ho, C.-Y.; Lin, H.-Y.; Wu, L.-T. “Intelligent speed bump system with dynamic license plate recognition”. In: 2016 IEEE International Conference on Industrial Technology (ICIT), 2016, pp. 1669–1674.

[6] Kaur, S.; Kaur, S. “An efficient approach for number plate extraction from vehicles image under image processing”, *International Journal of Computer Science and Information Technologies*, vol. 5–3, 2014, pp. 2954–2959.

[7] MOHIT BANSAL, BINAY BINOD KUMAR, P. V. “Designing of licensed number plate recognition system using hybrid technique from neural network & template matching”, *IEEE*, 2015, pp. 1–6.

[8] PSDB. “Marchezan destaca propostas para a segurança pública em porto alegre”. Acesso em 19/11/2016.

[9] S Du, B. “Automatic license plate recognition (alpr): A state of the art review”, *Circuits and Systems for Video Technology, IEEE Transactions on*, –99, 2013, pp. 311–325.

[10] Sanyuan, Z.; Mingli, Z.; Xiuzi, Y. “Car plate character extraction under complicated environment”. In: Systems, Man and Cybernetics, 2004 IEEE International Conference on, 2004, pp. 4722–4726.

[11] Serro, R. M. “Detecção e reconhecimento de placas de carros em imagens digitais”, 2012.

[12] Wafy, M.; Madbouly, A. M. “Efficient method for vehicle license plate identification based on learning a morphological feature”, *IET Intelligent Transport Systems*, 2016.

[13] Yovits, M. C. “Advances in computers”. Indianapolis, Indiana: Academic Press, 1993, vol. 37.