UFRPE – Universidade Federal Rural de Pernambuco
Equipe: Bruno Henrique Pereira Marques;
Danielly de Moura Borba Queiroz.
Disciplina: Processamento de Imagem
Recife, 17 de fevereiro de 2017

Detecção de Placas de Carros

1. Definição do projeto

Este projeto tem como objetivo localizar placas de carros em uma determinada imagem através de algoritmos de reconhecimentos de objetos e para isso utilizaremos uma base de 382 imagens de veículos com diversas cores, ângulos e ambiente. Para a implementação, foi utilizado a linguagem Java em conjunto com a biblioteca OpenCV 3.1.0 para a realização de diversas operações com imagens.

A primeira etapa desse projeto é realizar um pré-processamento na imagem de entrada. Este pré-processamento é responsável por fazer correções de distorções e eliminação de ruídos. A segunda etapa consiste em destacar os objetos presentes nas imagens com operações morfológicas. A terceira etapa é a segmentação com objetivo de localizar a placa na imagem através do padrão citado abaixo (1.a). O diagrama que representa as fases do projeto está apresentado no item 2.

a) Padrão da placa:

O código da placa seguirá um padrão de combinações de três letras (26 símbolos, de A-Z) e quatro números (10 símbolos, de 0-9), nas cores cinza - para o fundo - e preto para os caracteres. Seu formato está apresentado na imagem (Figura 1) apresentada a seguir.

FIGURA 1: Padrão de placa para automóveis nacionais.

2. Etapas do projeto

O diagrama abaixo (Figura 2) apresenta, em alto nível, todas as etapas realizadas pelo algoritmo desenvolvido no projeto. Cada etapa tem sua descrição referente e enumerada a seguir.

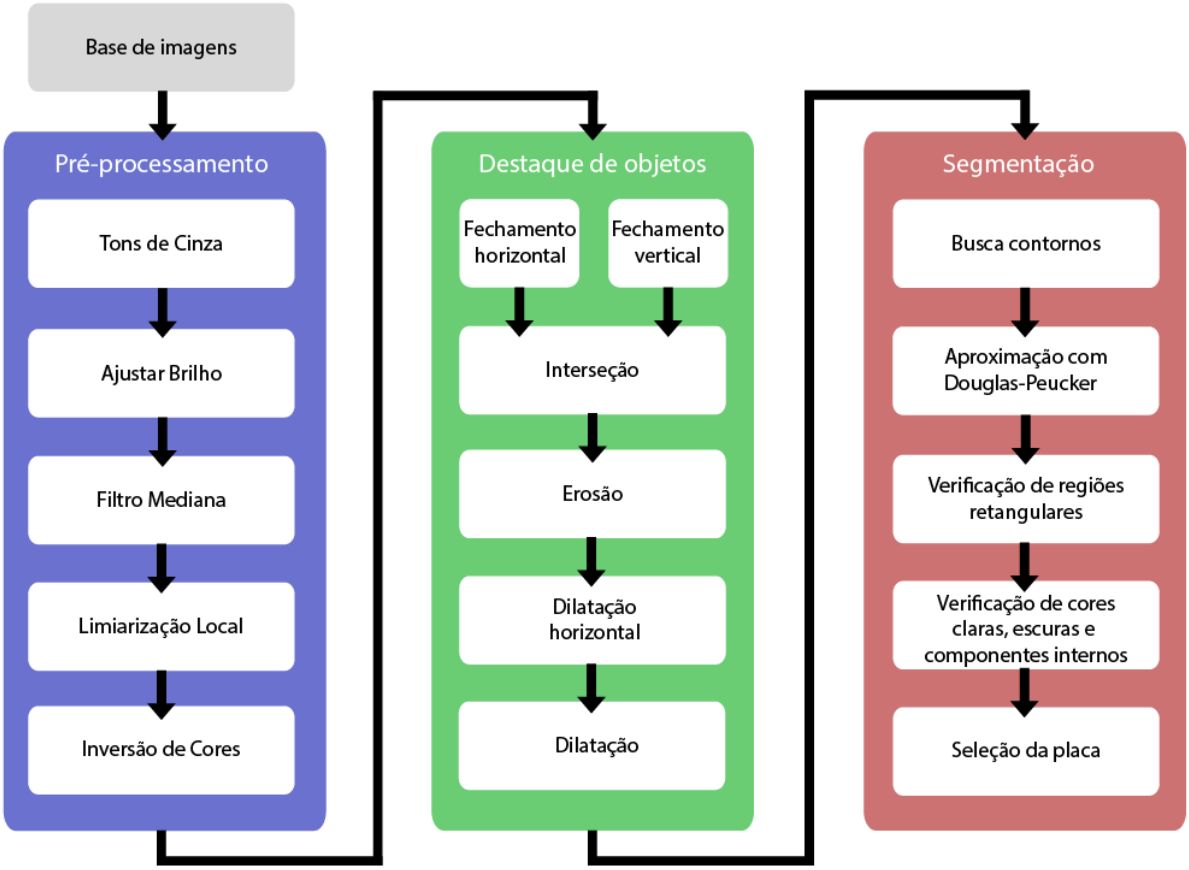


FIGURA 2: Diagrama com todas as etapas do algoritmo desenvolvido no projeto.

a) Pré-processamento:

A entrada deste projeto são imagens de automóveis tiradas de câmeras fotográficas ou de celulares, assim deve ser feito um pré-processamento para corrigir possíveis erros de captura, como também ajudará melhorando a eficiência das próximas etapas. A figura 3 é uma imagem da base de dados que será utilizada como exemplo. Os itens abaixo têm missão de diminuir ruídos e corrigir geometrias distorcidas.

FIGURA 3: Imagem de exemplo retirada da base de dados do projeto.

I. Tons de cinza:

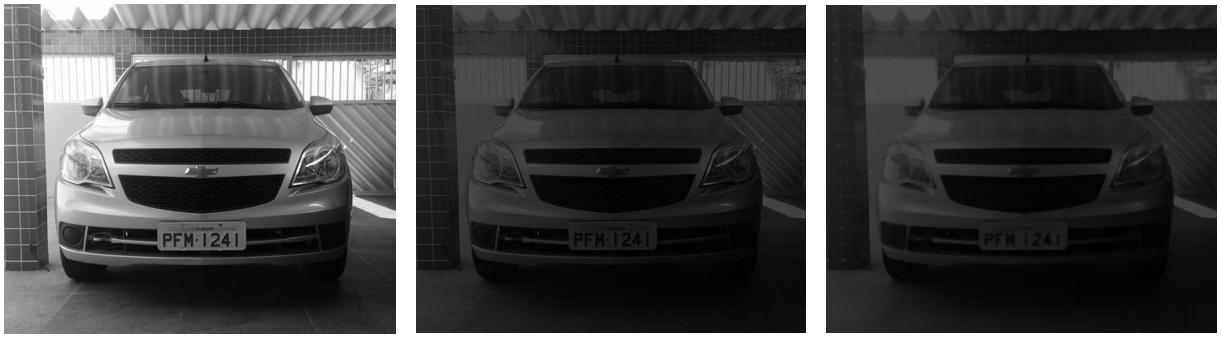
A imagem de entrada foi transformada de RGB para escala de cinza.

II. Ajustar brilho:

Nesta etapa um ajuste de brilho é realizado afim de preparar para limiarização com finalidade de transformar os pixels de tons médios em tons mais escuros e os pixels claros em tons próximos dos médios.

III. Filtro mediana:

Após o ajuste de brilho foi aplicado o filtro mediana com uma máscara 5x5 para remover ruídos sem prejudicar os detalhes da imagem. Vale ressaltar o teste com máscaras nos tamanhos 3x3, 7x7 e 9x9, sendo a máscara 5x5 a que apresentou melhor resultado.



I. Tons de cinza II. Ajustar brilho III. Filtro mediana

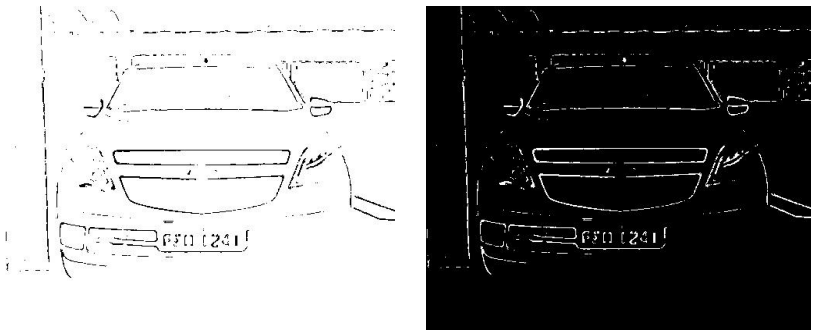
FIGURA 4: Imagem com os resultados das respectivas etapas.

IV. Limiarização local:

Nesta etapa a imagem é convertida para preto e branco com o Threshold Adaptativo Gaussiano do OpenCV. O valor do limiar é a soma ponderada dos valores da vizinhança onde os pesos são uma janela gaussiana.

V. Inversão de cores:

Como o resultado da etapa anterior deixa as regiões de interesse na cor preta, é necessária a inversão para que estas fiquem brancas. Além disso, prepara a imagem para as operações das próximas etapas.



IV. Limiarização local V. Inversão de cores

FIGURA 5: Imagem com os resultados das respectivas etapas.

b) Destaque de objetos:

As operações aplicadas anteriormente apresentaram bons resultados na eliminação de ruídos. A partir desse resultado, foram aplicadas as operações morfológicas de **fechamento**, **dilatação** e **erosão**. As etapas e os resultados estão explicitados a seguir.

I. Fechamento horizontal:

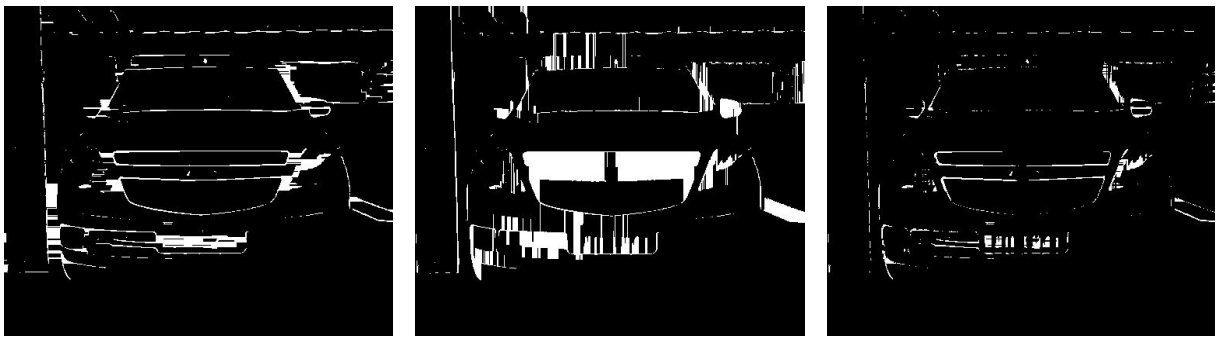
Para executar a operação morfológica de fechamento foi feito a aplicação de uma dilatação seguida de uma erosão. Com a finalidade de juntar objetos com pequenas separações, preencher pequenas brechas e contorno. E para o fechamento ser horizontal foi utilizado um elemento estruturante de 1 pixel de altura e o tamanho da imagem original dividido por 30 (melhor fator encontrado nos testes) de largura, afim de destacar elementos com orientação horizontal:

II. Fechamento vertical:

Tem funcionamento análogo ao fechamento horizontal, a diferença é o destaque de elementos orientação vertical com um elemento estruturante de 1 pixel de largura e o tamanho da imagem original dividido por 15 (melhor fator encontrado nos testes) de altura.

III. Interseção:

Nessa etapa é realizado uma junção (através da operação **AND**) das imagens resultantes do Fechamento Vertical e Horizontal, com finalidade de que apenas os objetos em comum permaneçam na imagem de resultante.



I. Fechamento Horizontal II. Fechamento Vertical III. Interseção

FIGURA 6: Imagem com os resultados das respectivas etapas.

IV. Erosão:

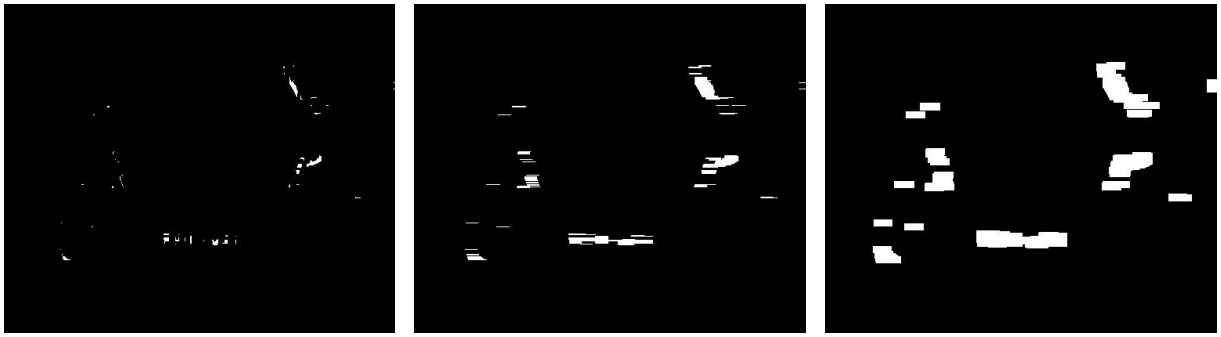
A erosão é um operador morfológico que tem como objetivo diminuir as áreas em branco da imagem. Essa diminuição é realizada através de um elemento estruturante que é construído pela função `getStructuringElement` do openCV 3.1.0. Foi utilizado a erosão com um elemento estruturante de ordem 3 com a finalidade de eliminar pequenos detalhes adquiridos pela etapa anterior, evitando com que esses objetos se unam com a placa. Vale ressaltar que outras ordens (5, 7 e 9) foram testadas.

V. Dilatação horizontal:

Seu funcionamento é análogo ao fechamento horizontal, porém tem objetivo de aumentar as áreas em branco na imagem. Também foi utilizado um elemento estruturante de 1 pixel de altura com o tamanho da imagem original de largura dividido por 30 (melhor fator encontrado nos testes).

VI. Dilatação:

Foi utilizada a dilatação simples para corrigir pequenas falhas que não foram corrigidas pela etapa anterior com um elemento estruturante de ordem 9. Vale ressaltar que outras ordens (3, 5 e 7) foram testadas.



V. Erosão VI. Dilatação horizontal VII. Dilatação

FIGURA 7: Imagem com os resultados das respectivas etapas.

c) Segmentação:

Nesta etapa, a imagem resultante do pré-processamento será entrada para o módulo do sistema que detectará a posição da placa do carro. Analisaremos a imagem com objetivo de separar as partes de interesse na imagem de entrada de acordo com o padrão da placa (1.a). Isso será feito através de uma busca pelos padrões que caracterizam uma placa de automóvel.

I. Busca contornos:

A biblioteca OpenCV 3.1.0 fornece uma função (*findContours*) que retorna um conjunto de contornos na imagem, assim foi aplicada na imagem resultante da detecção de bordas (item C.VII), dando o primeiro passo para a detecção da posição da placa na imagem original.

II. Aproximação com Douglas-Peucker:

Com o conjunto de bordas, foi executado o algoritmo de busca (*approxPolyDP*), também presente na biblioteca OpenCV 3.1.0, que retorna um conjunto de polígonos fechados ou abertos (neste projeto, só interessa polígonos fechados obviamente pela característica da placa). A partir deste conjunto de polígonos, foi feito uma operação afim de eliminar falsos positivos. A quantidade de vértices do polígono candidato deve ser compatível com a quantidade de vértices encontrada no polígono que determina a área da placa.

III. Verificação de regiões retangulares:

Nesta etapa, é realizada o cálculo do aspecto do polígono e sua proporção com objetivo, também, de eliminar falsos positivos. Os polígonos resultantes desta operação são retângulos que possuem proporção equivalente à da placa de um veículo – De acordo com Jorge Fernando Silva Pereira Filho [4] a proporção deve estar entre 3:1 e 6:1.

IV. Verificação de cores claras, escuras e componentes internos:

Esta etapa consiste em calcular a quantidade de regiões claras, escuras e a quantidade de componentes internos delimitado pelo polígono candidato validado na etapa anterior. O objetivo fornecer dados para a seleção de uma região candidata na próxima etapa. Para as cores claras, é armazenada a quantidade de cores que se aproximam do branco, enquanto para as cores escuras, é armazenada a quantidade de cores que se aproximam do preto. Para calcular a quantidade de componentes internos, foi realizado um pequeno pós-processamento nas regiões candidatas seguindo a ordem abaixo. Após isso foi aplicado a busca de contornos e a busca de componentes conexos do OpenCV igualmente usado na etapa de segmentação mas com finalidade de contar quantos componentes foram encontrados na região candidata.

1. Equalização do histograma;
2. Filtro mediana de máscara 3 aplicada 2 vezes;
3. Limiarização global;
4. Detecção de bordas com Auto Canny;

FIGURA 8: Resultados de cada etapa do pós-processamento.

V. Seleção da placa:

Depois de realizar várias filtragens afim de diminuir a probabilidade de haver falsos positivos, esta etapa tem objetivo de encontrar a placa verdadeira dentre várias regiões candidatas. A operação é em relação a quantidade de cores claras e escuras explicitadas na etapa anterior. O algoritmo escolherá a região candidata que atende aos critérios abaixo:

- ✓ A região candidata com a maior quantidade de cores claras e escuras;
- ✓ A região candidata deve conter quantidade de cores claras superior à quantidade de cores escuras;
- ✓ A região candidata deve ter maior quantidade de componentes internos com valores de 5 a 7.

Após a execução de todas as etapas, a imagem resultante da segmentação está apresentada na imagem abaixo (figura 9). A borda de cor lilás é o polígono candidato encontrado pelo algoritmo.

FIGURA 9: Imagem da placa segmentada.

3. Resultados

A implementação do projeto foi testada no banco de imagens citado anteriormente. O resultado está apresentado na tabela abaixo. Além disso foi comparado o resultado com mais 4 propostas encontradas nos respectivos artigos científicos.

Algoritmo	Não encontradas	Segmentados	Acertos	Erros
Proposto	17	365	232	133
Ref[1]	31	351	130	221
Ref[2]	287	95	14	81
Ref[3]	0	382	100	281

Não encontradas	Caso onde nenhuma região candidata que atende às regras é encontrada. Valor em relação ao total de imagens da base.
Segmentados	Caso onde pelo menos uma região candidata é encontrada. Valor em relação ao total de imagens da base.
Acertos	Caso onde a placa é segmentada corretamente. Valor em relação a quantidade de segmentos.
Erros	Caso onde o algoritmo escolhe a região candidata errada. Valor em relação a quantidade de segmentos.

4. Referências

[1] Reconhecimento de placas de veículos através de análise de imagem:
http://www.prp.rei.unicamp.br/pibic/congressos/xviicongresso/paineis/059834.pdf

[2] Um Algoritmo para localização de placas de veículos:
http://www.lbd.dcc.ufmg.br/colecoes/wvc/2006/0065.pdf

[3] Reconhecimento Automático de Placas de Veículos:
http://www.lbd.dcc.ufmg.br/colecoes/wvc/2010/0047.pdf

[4] Um sistema de reconhecimento automático de placas de automóveis:
http://fei.edu.br/~rbianchi/publications/ENIA99.pdf

[5] Reconhecimento automático de placas de veículos utilizando processamento digital de imagens e inteligência artificial:
http://uniseb.com.br/presencial/revistacientifica/arquivos/9.pdf

[6] Localização de Placas de Licenciamento Veicular em tempo real utilizando OpenCV com CUDA:
http://wiki.ifba.edu.br/ads/tiki-download_file.php?fileId=827

[7] Documentação OpenCV 3.1.0:
http://docs.opencv.org/3.1.0/