UFRPE – Universidade Federal Rural de Pernambuco
Equipe: Bruno Henrique Pereira Marques;
Danielly de Moura Borba Queiroz.
Disciplina: Processamento de Imagem
Recife, 17 de outubro de 2016

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 386 imagens de veículos em diversos ângulos e ambientes. 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, eliminação de ruídos e em seguida detectar bordas. Na segunda etapa faremos o reconhecimento da posição da placa na imagem através do padrão citado abaixo (1.a) e por fim o reconhecimento de caracteres contidos na placa. 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.

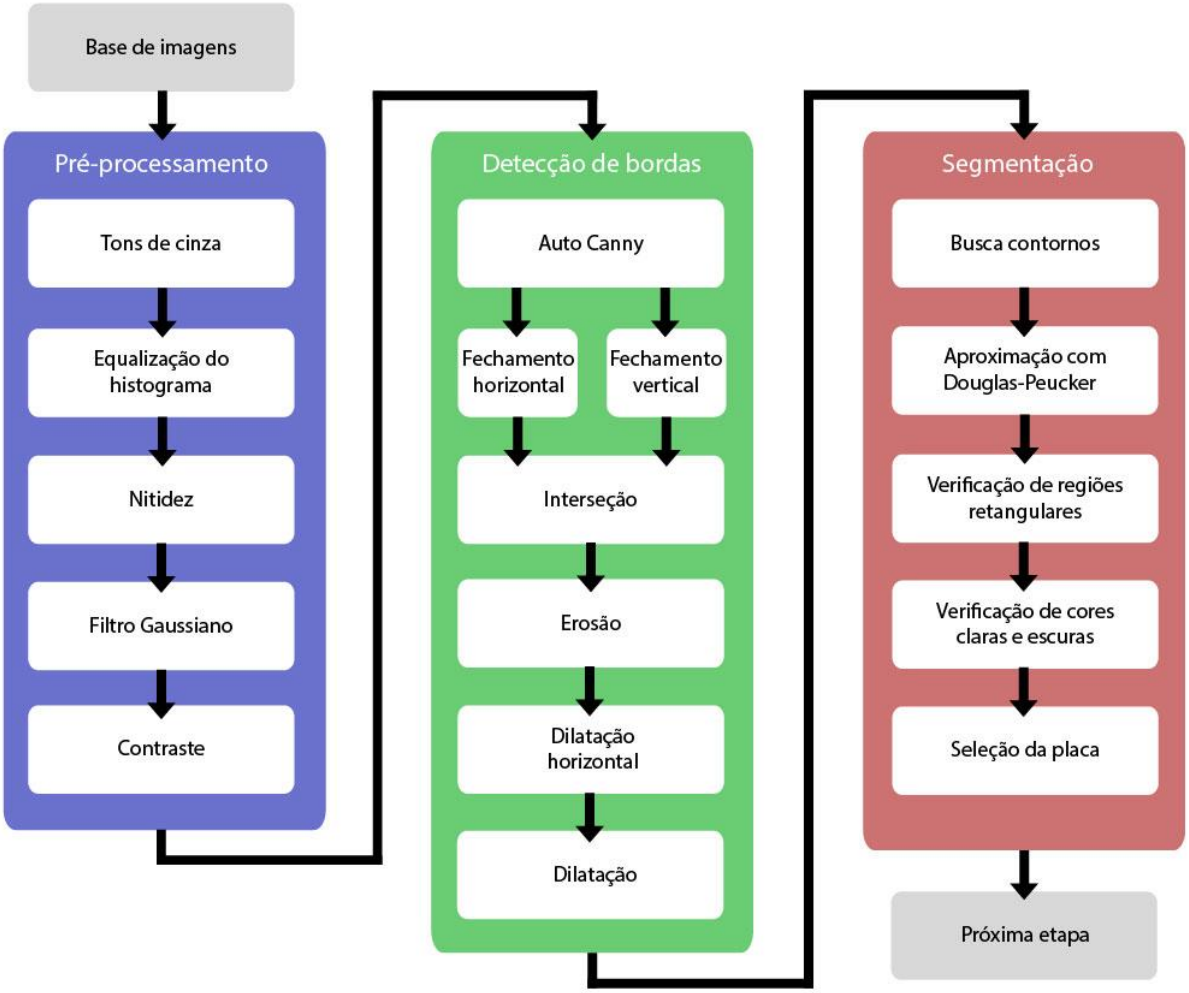


FIGURA 2: Diagrama com todas as etapas do algoritmo desenvolvido no projeto.

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. Equalização do Histograma:
O histograma da imagem foi equalizado para melhorar a distribuição dos tons de cinza. Para aplicar a equalização do histograma foi utilizado o método `equalizeHist` do OpenCV 3.1.0.

III. Nitidez:
Após a equalização do histograma foi necessário um aumento de nitidez para eliminar leves borramentos e destacar a placa do veículo. Foram testados manualmente vários valores para os parâmetros sgima, alfa, beta e gama, da função `addWeighted` e o melhor resultado obtido, para grande parte das imagens da base, foi com os valores (7, 0.75, -0.5, 0) respectivamente para os parâmetros.

I. Tons de cinza II. Equalização do Histograma III. Nitidez

FIGURA 4: Imagem com os resultados das respectivas etapas.

IV. Filtro Gaussiano:
Com o objetivo de suavizar a imagem e eliminar distorções e ruídos foi aplicado o filtro Gaussiano com uma máscara 3x3. Vale ressaltar o teste com máscaras nos tamanhos 5x5, 7x7 e 9x9, sendo a máscara 3x3 a que apresentou melhor resultado.

V. Contraste:
Foi aplicado a técnica de realce de contraste com o objetivo de melhorar a qualidade da imagem e realçar a diferença de brilho entre as áreas claras e escuras da imagem, deixando o histograma com regiões mais definidas.

IV. Filtro Gaussiano V. Contraste

FIGURA 5: Imagem com os resultados das respectivas etapas.

b) Detecção de bordas:

As operações aplicadas anteriormente apresentaram bons resultados no destaque dos objetos de cores e formas parecidos com placas. A partir desse resultado, foi aplicado o filtro de detecção de bordas **Auto Canny** e as operações morfológicas de **fechamento**, **dilatação** e **erosão**. Cada etapa está explicitada a seguir.

I. Auto Canny:
Foi aplicado o operador ótimo de **Canny** para detecção de bordas. Foi percebido, apesar do pré-processamento, a presença de bordas de pequenos ruídos. O Canny possui dois *thresholds* (mínimo e máximo) como parâmetro, mas, como a base possui imagens variadas, foi implementado um algoritmo para extrair os dois parâmetros dinamicamente. O algoritmo consiste em calcular a média dos pixels da imagem e em seguida calcular o mínimo e máximo utilizando um peso (sigma, valores a partir de 0,33) atendendo às equações abaixo e a imagem resultante está apresentada na figura 6:

$$\text{thresh1(x)} = \max(0, (1 - sigma) * mean)$$

Cálculo do threshold mínimo

$$\text{thresh2(x)} = \min(255, (1 + sigma) * mean)$$

Cálculo do threshold máximo

I. Auto Canny

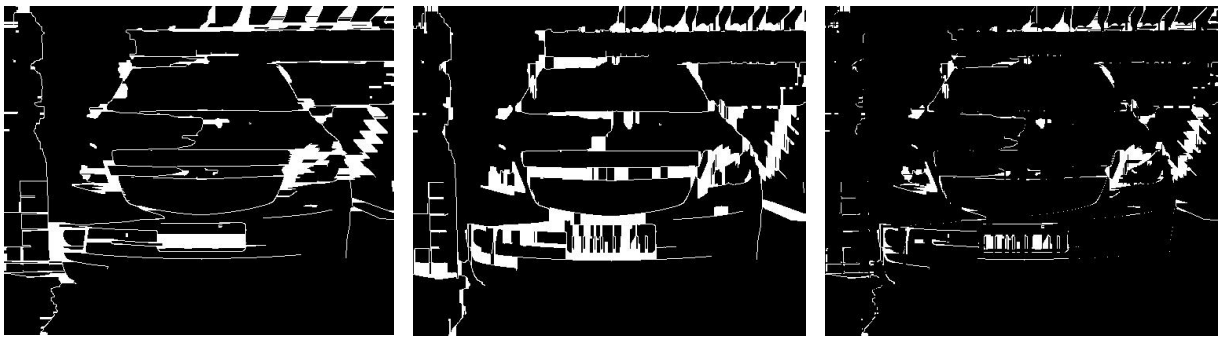
FIGURA 6: Imagem com a operação Auto Canny.

II. 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.

III. 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 30 (melhor fator encontrado nos testes) de altura.

IV. 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.



II. Fechamento Horizontal III. Fechamento Vertical IV. Interseção

FIGURA 7: Imagem com os resultados das respectivas etapas.

V. 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.

VI. 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).

VII. 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 8: 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 e escuras:
Esta etapa consiste em calcular a quantidade de regiões claras e escuras delimitada 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.

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;

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.

6. Referências

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

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

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

[4] 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