ALGORITMO DE RECONHECIMENTO AUTOMÁTICO DE PLACAS DE VEÍCULOS BASEADO EM MATLAB E TESSERACT OCR

Roberto Espinheira da Costa Bomfim, Rebeca Tourinho Lima e Roberto Luiz Souza Monteiro

Faculdade de Tecnologia SENAI CIMATEC

Programa de Pós-Graduação em Modelagem Computacional e Tecnologia Industrial

E-mails: roberto.bomfim@fieb.org.br, rebeca.lima@fieb.org.br, roberto.monteiro@fieb.org.br

RESUMO

Este trabalho apresenta o desenvolvimento de um algoritmo destinado ao reconhecimento de placas de veículos utilizando o *toolbox* de processamento de imagens do MATLAB e a biblioteca de reconhecimento óptico de caracteres Tesseract OCR. Os resultados encontrados demonstram a viabilidade técnica da abordagem proposta, cuja precisão e consequente taxa de acertos pode ser aprimorada através do ajuste dos parâmetros do Tesseract OCR.

1. INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Nos últimos anos, um desenvolvimento notável vem ocorrendo em dois campos tecnológicos intimamente ligados às áreas de Processamento de Imagem, Visão Computacional e disciplinas correlatas:

Sistema de Aquisição de Imagens - Surgimento de dispositivos de alta definição e baixo custo (e.g. câmeras fotográficas, câmeras de vídeo);

Sistemas Computacionais - Evolução dos dispositivos através do aumento de seu poder de processamento e de sua quantidade de memória, aliada à redução de seus custos e de seu tamanho;

A evolução desses sistemas criou o cenário ideal para o desenvolvimento/aperfeiçoamento de diversas aplicações, como na área biomédica e no controle de qualidade e automação de processos industriais.

Uma aplicação bastante difundida na literatura e cada vez mais robusta é a de reconhecimento automático de placas de veículos, comumente utilizada na automação de sistemas de controle de acesso e sistemas de fiscalização eletrônica nas vias urbanas e rodovias.

Essas aplicações permitem a identificação de veículos que não estejam autorizados a trafegar e/ou acessar uma determinada localidade ou que possuam algum tipo de pendência junto ao órgão fiscalizador (e.g. multas registradas, IPVA vencido).

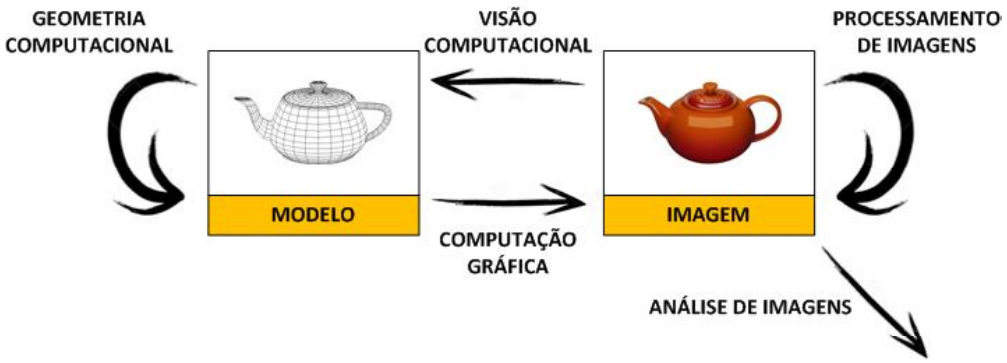
Neste contexto, *esse trabalho propõe o desenvolvimento de um algoritmo computacional de reconhecimento de placas de veículos utilizando um banco de imagens reais.*

Conforme será detalhado no tópico 2.2, o algoritmo carrega as imagens do banco de imagens reais utilizado e extrai o código de identificação contido nas placas de veículos automotivos brasileiros (sequência de letras e números).

1.2 FUNDAMENTAÇÃO TEÓRICA

Existe uma série de disciplinas intimamente relacionadas ou até mesmo em parte coincidentes com a área de *processamento de imagens*, como a *visão computacional*, a *computação gráfica* ou ainda a *análise de imagens* [1]. Não há um consenso entre os especialistas a respeito das fronteiras entre essas disciplinas, mas o quadro esquemático da **Figura 3** representa de maneira razoável a relação entre elas:

Figura 3: Relação existente entre a área de processamento de imagens e as demais disciplinas correlatas.



Fonte: Autor. Adaptado de [1].

Dentro de um contexto mais amplo, as técnicas de processamento de imagem possuem diversas funções, podendo ser normalmente enquadradas na lista abaixo:

- Aprimoramento de Imagens;
- Combinação de Imagens;
- Síntese de Imagens;
- Compressão de Dados;
- Reconhecimento de Padrões;
- Transformação de dados em informações mais facilmente manipuláveis ou interpretáveis;

Ao falar de processamento de imagens, normalmente está implícito tratar-se de *imagens digitais* (apesar de o processamento de imagens analógicas existir, pouquíssimas aplicações são desenvolvidas na prática, devido à complexidade do *hardware* envolvido, dos custos elevados e da falta de flexibilidade e modularidade dos equipamentos).

Uma imagem digital nada mais é do que uma matriz de números, onde cada elemento (i, j) da matriz é conhecido como *pixel* (do inglês *picture element*). Cada *pixel*, por sua vez, é representado por uma *n*-upla de números inteiros (normalmente entre 0 e 255), que varia de acordo com o espaço de cores utilizado. Na representação de uma imagem no espaço de cores RGB, por exemplo, cada *pixel* é representado por 3 números inteiros. Por outro lado, para representar esta mesma imagem no espaço de cores de Escala de Cinza é necessário apenas 1 número inteiro.

Ao representar uma imagem por uma matriz de *pixels*, pode-se efetuar uma miríade de operações matemáticas sobre esses dados para implementar a função que se deseja. Do ponto de vista prático, as operações matemáticas utilizadas correspondem a derivadas, convoluções, operações estatísticas (médias simples e ponderadas, medianas) transformadas (de Fourier, de Hough), dentre outras. Uma peculiaridade de se trabalhar com imagens, que pode se tornar uma das principais dificuldades de implementação, está no fato que uma imagem quase nunca é uma função "bem comportada" (i.e. contínua e derivável ao longo do intervalo de interesse). Deste modo, tornam-se necessárias algumas aproximações e adaptações nos cálculos.

Em aplicações de processamento de imagens, normalmente a 1ª etapa corresponde ao Pré-Processamento, onde a imagem é preparada de maneira a ressaltar algumas características importantes para as etapas posteriores. Em seguida, já na etapa de Processamento, é necessário segmentar a imagem. De acordo com [2], a segmentação de uma imagem consiste em subdividi-la nas regiões ou objetos que a constituem. Obviamente, o nível de subdivisão depende da aplicação em questão e deve possibilitar o isolamento do objeto de interesse.

O Reconhecimento Óptico de Caracteres (do inglês, OCR – *Optical Character Recognition*) é um campo de pesquisa na área de reconhecimento de padrões, inteligência artificial e visão computacional, que consiste na conversão mecânica ou eletrônica de imagens digitalizadas ou manuscritas, digitadas ou impressas em texto codificado [3]. Algumas abordagens OCR são discutidas a seguir [4]:

Correlação de Matrizes: Converte-se cada caractere em uma matriz e compara-se com padrões conhecidos. É uma técnica indicada para textos que utilizem a mesma fonte e estejam escritos em uma única coluna por página.

Lógica Fuzzy: Lógica de múltiplos valores, onde se permite valores intermediários em detrimento da tradicional lógica binária. Trata-se de uma tentativa de tornar o pensamento lógico computacional mais humano, ao acrescentar um grau de incerteza ao estudo.

Extração de Características: Esse método identifica cada caractere pela presença ou ausência de determinadas características-chaves tais como altura, largura, densidade, loops, linhas etc. É uma abordagem interessante para amostras de imagem de revistas, impressões a laser e imagens de alta qualidade.

Análise Estrutural: Identifica os caracteres através da comparação de suas projeções horizontais e/ou verticais. Os padrões de projeção de cada caractere são catalogados previamente e o critério de igualdade é estabelecido através de um limiar [5]. Apenas a partir da projeção vertical já é possível identificar alguns caracteres alfanuméricos, no entanto, caracteres como "N", "H" e "U", por exemplo, possuem

projeções verticais semelhantes, tornando necessária a comparação de suas respectivas projeções horizontais para eliminar ambiguidades [5]. É adequada para textos de baixa qualidade e jornais.

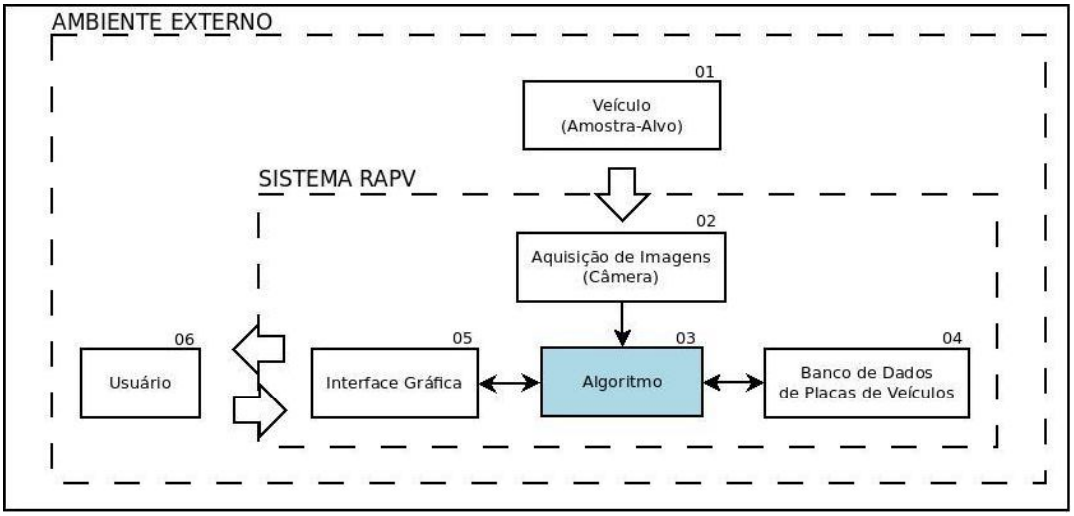
Redes Neurais: Simula o sistema neuronal humano e, neste caso, efetua uma amostragem de pixels de cada imagem para correlacioná-los com padrões de pixels previamente conhecidos. É uma abordagem interessante para documentos danificados e mensagens de fax, por exemplo.

2. METODOLOGIA

2.1 ESCOPO DO TRABALHO

Um sistema completo de Reconhecimento Automático de Placas de Veículos (RAPV) compreende um subsistema de aquisição de imagens através de uma câmera, um algoritmo de processamento, um banco de dados de placas de veículos e uma interface gráfica com o usuário, conforme **Figura 4**.

Figura 4: Sistema completo de RAPV.



Fonte: Autor.

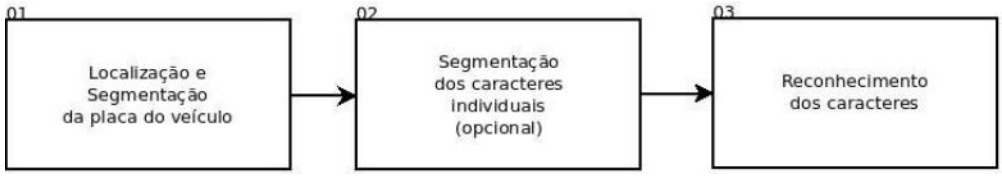
Levando-se em consideração que o escopo deste trabalho corresponde apenas ao desenvolvimento do algoritmo (destacado em azul na **Figura 4**), optou-se por simplificar o sistema completo através das seguintes modificações:

Etapas 01 e 02: Utilização do banco de imagens do Laboratório de Processamento Digital de Sinais e Imagens [6].

Etapas 04 e 05: Não implementadas (interface atual via MATLAB).

Conforme indicado na **Figura 5**, um algoritmo de RAPV normalmente consiste de 3 etapas principais (o escopo deste trabalho restringiu-se à execução das etapas 01 e 03).

Figura 5: Fluxograma das principais etapas de um algoritmo de RAPV.



Fonte: Autor.

2.2 CONSIDERAÇÕES SOBRE AS FERRAMENTAS UTILIZADAS

O projeto foi desenvolvido em ambiente híbrido, utilizando:

Windows Vista SP2: No qual foi instalado o MATLAB R2013b contendo o *toolbox* de Processamento de Imagens.

Linux, distribuição Mint Petra 16: No qual foram instalados os pacotes Tesseract OCR 3.02.01, Leptonica 1.69 e demais dependências.

2.2.1 Toolbox de Processamento de Imagens do MATLAB

Neste trabalho, utilizou-se o *toolbox* de Processamento de Imagens do MATLAB para segmentar a imagem original do veículo, isolando a região de interesse (i.e. a placa do veículo) do restante da imagem e, em seguida, preparando a região da imagem da placa para a etapa de reconhecimento óptico de caracteres (estes processos serão detalhados na seção 2.3).

2.2.2 Tesseract OCR

O Tesseract OCR é uma *engine* de reconhecimento óptico de caracteres desenvolvida nos laboratórios da Hewlett Packard (HP) entre 1985 e 1995 e hoje mantida e aprimorada pela Google [7]. É dita uma das mais precisas ferramentas de OCR de código aberto disponíveis no mercado. Em conjunto com as bibliotecas Leptonica de processamento de imagens, é capaz de ler uma infinidade de formatos de imagem e convertê-las em texto para cerca de mais de 60 línguas.

Sob licença Apache 2.0, pode ser utilizada diretamente como programa ou, no caso de programadores, pode ser utilizada via API (*Application Program Interface*). Embora não possua uma interface gráfica (GUI), há inúmeros *softwares* de terceiros que cumprem este papel.

O desempenho do Tesseract OCR está diretamente ligado à alguns cuidados com a imagem de entrada e à correta configuração de seus parâmetros. Um recurso bastante importante, por exemplo, é o de dicionários, *whitelist* e *blacklist* (para inclusão e exclusão de termos, respectivamente).

2.3 DESENVOLVIMENTO DO ALGORITMO

2.3.1 Algoritmo de Localização e Segmentação de Placas de Veículos

O fluxograma do Algoritmo de Localização e Segmentação de Placas de Veículos, desenvolvido em MATLAB com o auxílio do *toolbox* de Processamento de Imagens, pode ser visto na **Figura 6**.

Figura 6: Fluxograma do algoritmo de localização e segmentação de placas de veículos.

Fonte: Autor.

De modo a facilitar a compreensão do algoritmo, abaixo é fornecida uma breve descrição das principais funções utilizadas em sua implementação (em ordem de utilização):

imread - Carrega uma imagem a partir de um arquivo e retorna uma matriz contendo os dados da imagem.

rgb2gray - Converte a imagem do espaço de cores RGB para Escala de Cinza.

histeq - Equaliza o histograma de uma imagem. Em outras palavras, redistribui o espectro de frequências da imagem de modo a aproveitar toda a faixa de frequências e, consequentemente, aumentar o contraste. Alguns testes do algoritmo utilizando esta função foram realizados e, como os resultados obtidos foram insatisfatórios, decidiu-se por eliminar esta etapa do algoritmo.

edge - Converte a imagem de Escala de Cinza para Binária calculando o limiar através do Método de Otsu [8] e, em seguida, realiza a detecção das bordas existentes na imagem. Neste trabalho, realizou-se a detecção apenas das bordas verticais, para o qual utilizou-se o método de Sobel. A ideia central é utilizar a informação conhecida *a priori* de que uma placa de veículo costuma ter bordas verticais proeminentes em relação ao plano de fundo da imagem para dar início ao processo de delimitação da região de interesse.

bwareaopen - Remove todos os objetos da imagem menores que um tamanho especificado em pixels.

imclose - Realiza a operação morfólogica de FECHAMENTO nos objetos da imagem com base em um elemento estruturante previamente definido (neste caso uma linha horizontal).

imopen - Realiza a operação morfólogica de ABERTURA nos objetos da imagem com base em um elemento estruturante previamente definido (neste caso uma linha horizontal de tamanho 5 vezes inferior ao elemento utilizado em *imclose*.

imdilate - Realiza a operação morfólogica de DILATAÇÃO nos objetos da imagem com base em um elemento estruturante previamente definido (neste caso um quadrado de lado igual ao elemento utilizado em *imclose*).

find - Localiza, em uma imagem binária, os pixels de cor branca e retorna suas respectivas posições (criando uma máscara que será aplicada à imagem original).

A partir deste ponto, a região de interesse do problema (i.e. a placa do veículo) pode ser devidamente delimitada na imagem a partir da aplicação da máscara construída à imagem original.

im2bw - Converte a imagem de Escala de Cinza para Binária através de um limiar calculado pelo Método de Otsu [8].

imresize - Redimensiona a imagem através da aplicação de um fator de escala (neste caso utilizou-se um fator de escala igual a 4 e o método de interpolação cúbico).

~ - Inverte as cores de uma imagem binária (resultando, neste caso, em uma imagem de plano de fundo preto com objetos brancos). Esta operação é necessária à etapa posterior, visto que o MATLAB, por convenção, trabalha sempre desta forma.

bwareaopen - Remove todos os objetos da imagem menores que um tamanho especificado em pixels.

2.3.2 Algoritmo de Reconhecimento dos Caracteres

Nesta etapa do algoritmo, utilizou-se o Tesseract na forma de programa de linha de comando através do terminal do Linux. A sintaxe geral é conforme segue:

tesseract imagename outputname [-l lang] [-psm pagesegmode] [configfile...]

Onde o segundo termo corresponde ao nome do arquivo de entrada com sua respectiva extensão; o seguinte, o nome do arquivo de saída, que é gerado em formato **.txt*; a linguagem a ser utilizada e que, quando não especificada, por padrão é estabelecida a língua inglesa; um parâmetro de configuração (*page mode segmentation*) dentre 11 opções e, por fim, o nome de um arquivo de configuração no qual pode-se alterar variáveis do sistema e criar listas.

Para este trabalho, o modo de segmentação mais adequado foi o 7 (corresponde ao tratamento de uma única linha de texto).

De modo a comparar as diferentes configurações do Tesseract, foram realizados testes com 3 configurações diferentes. Primeiramente, fez-se testes com a configuração básica:

Configuração Básica:

tesseract entrada saída -psm 7

Considerando que o escopo não é a busca de palavras, um segundo conjunto de testes consistiu em manter a configuração básica adicionando um arquivo de configuração para desativar o dicionário, o que é feito atribuindo "false" a duas variáveis do sistema:

Configuração Sem Dicionário:

load_system_dawg F

load_freq_dawg F

Por fim, no último conjunto de testes, adicionou-se as seguintes listas de caracteres permitidos e proibidos, respectivamente:

Configuração com Listas:

tessedit_char_whitelist A..Z 0..9

tessedit_char_blacklist (Símbolos)

3. RESULTADOS E DISCUSSÃO

Nessa seção são apresentados os resultados obtidos através da aplicação do algoritmo desenvolvido em 4 imagens diferentes. As imagens utilizadas foram retiradas do banco de imagens do Laboratório de Processamento Digital de Sinais e Imagens [6] (foram escolhidas, de maneira arbitrária, as 4 primeiras imagens do banco de imagens).

3.1 Resultados do Algoritmo de Localização e Segmentação de Placas de Veículos

A **Figura 7** mostra alguns exemplos dos resultados obtidos através da aplicação do Algoritmo de Localização e Segmentação de Placas de Veículos às amostras do banco de imagens utilizado. Em cada uma das figuras, a imagem superior corresponde à imagem de entrada e a imagem inferior à imagem de saída.

Figura 7: Exemplos dos resultados obtidos.

Fonte: Autor.

Deste modo, conclui-se que, para os testes realizados, o algoritmo é capaz de localizar e delimitar com precisão a região de interesse do trabalho. Entretanto, pode-se observar que, em alguns casos, a imagem de saída possui resquícios de objetos que não representam caracteres, podendo ocasionar erros na etapa de OCR.

Deste modo, conclui-se que, para os testes realizados, o algoritmo é capaz de localizar e delimitar com precisão a região de interesse do trabalho. Entretanto, pode-se observar que, em alguns casos, a imagem de saída possui resquícios de objetos que não representam caracteres, podendo ocasionar erros na etapa de OCR.

3.2 Resultados do Algoritmo de Reconhecimento dos Caracteres

A **Tabela 1** sintetiza os resultados obtidos através da utilização do Algoritmo de Reconhecimento dos Caracteres (Tesseract OCR) em cada uma das imagens de saída obtidas na etapa anterior e para diferentes configurações dos parâmetros da ferramenta.

Tabela 1: Resultados obtidos através da utilização do algoritmo de reconhecimento de caracteres.

AMOSTRA	TEXTO ORIGINAL	CONFIGURAÇÃO	TEXTO DETECTADO	QNT DE ERROS
01	LOH 8516	01 (Básica)	LOH 8516	0
		02 (s/ Dicionário)	LOH 8516	0
		03 (c/ Listas)	LOH 8516	0
02	LID 5816	01 (Básica)	_.\'Cl '58 -	5
		02 (s/ Dicionário)	_.\'Cl '58 -	5
		03 (c/ Listas)	LIT3 58	3
03	LNZ 7675	01 (Básica)	LNZ 7675	0
		02 (s/ Dicionário)	LNZ 7675	0
		03 (c/ Listas)	LNZ 7675	0
04	LOF 2025	01 (Básica)	CS	7
		02 (s/ Dicionário)	CS	7
		03 (c/ Listas)	CS	7

Fonte: Autor.

Os resultados obtidos nos três conjuntos de testes mostram que, para as amostras 01 e 03, a configuração básica já foi satisfatória com 100 % de acerto. Já nas amostras 02 e 04, vê-se que nenhuma das configurações conseguiu esse índice, mesmo com os ajustes incrementais. A dificuldade de reconhecimento pode ser atribuída a fatores mencionados anteriormente como, por exemplo, a orientação da imagem, condição evidente na amostra 02.

4. CONCLUSÃO

Com base nos resultados encontrados, pode-se concluir que os objetivos propostos foram atingidos, visto que o algoritmo desenvolvido é capaz de localizar e delimitar com precisão a região de interesse da imagem (i.e. a placa do veículo) e, em seguida, realizar a identificação dos caracteres.

No caso das amostras utilizadas, observou-se que, em alguns casos, a imagem da região de interesse possui resquícios de objetos que não representam caracteres, o que pode dificultar a etapa de OCR. De modo semelhante, a taxa de acertos da detecção de caracteres por meio do Tesseract OCR foi bastante inconstante, variando de 0 a 100 % para os casos analisados. Sendo assim, para avaliar o algoritmo de maneira mais precisa e sistemática, dando mais confiabilidade aos resultados obtidos, é necessário realizar testes com uma quantidade maior de amostras do banco de dados. Entretanto, considerando que esta etapa consome bastante tempo de trabalho, sugere-se efetuar as alterações de configuração do algoritmo de detecção de caracteres sugeridas no parágrafo posterior antes de aumentar o número de amostras utilizadas.

Para aprimorar o algoritmo desenvolvido, pode-se pensar em uma série de propostas de trabalhos futuros. Primeiramente, um ponto de melhoria de performance no uso do Tesseract seria treinar o algoritmo de classificação com o banco de imagens próprio em substituição ao treinamento padrão da *engine*, que é voltado à detecção de documentos de texto. Outra melhoria seria fixar o formato do dado a ser lido, de forma que o algoritmo de classificação tenha o conhecimento prévio dos tipos de caracteres a serem identificados a cada posição do código da placa (i.e. 3 letras e 4 algarismos separados por um espaço). Por último, com o algoritmo devidamente testado e validado, o mesmo pode compor o *firmware* de um sistema embarcado, mas, para isso, é necessário transpô-lo para uma plataforma única (e.g. C++, JAVA ou Python).

REFERÊNCIAS

[1] Maintz, T.; Digital and Medical Image Processing. *Notas de Aula*, Utrecht University, 2005.

[2] Gonzalez, R. C.; Woods, R. E.; *Digital Image Processing*, 2ª ed., Prentice Hall, 2002.

[3] Banerjee, S.; A Study on Tesseract Open Source OCR Engine. *Tese de Doutorado*, Jadavpur University Kolkata, 2012.

[4] Singh, S.; Optical Character Recognition Techniques: A Survey. *Journal of Emerging Trends in Computing and Information Sciences*, **2013**, 4, 545-550.

[5] Belvisi, R. et al; Um Sistema de Reconhecimento Automático de Placas de Automóveis. *Anais do XIX Congresso Nacional da Sociedade Brasileira de Computaçao - Encontro Nacional de Inteligência Artificial*, **1999**, 4, 537-539.

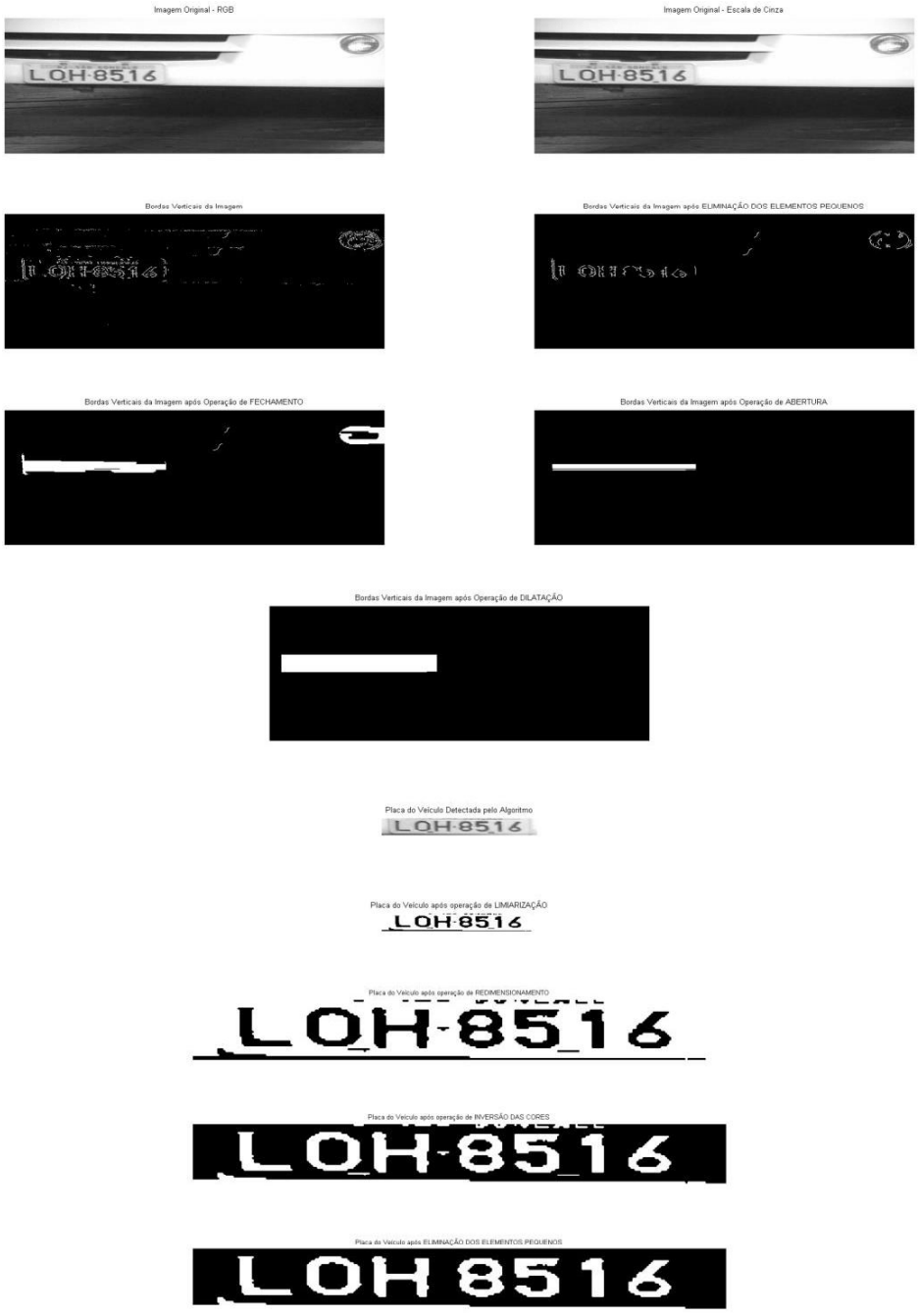
[6] LPDSI. Base de Imagens de Veículos Brasileiros. Disponível em: <http://www.cbpf.br/cat/pdsi/lpr/>. Acesso em: 01 maio 2014.

[7] Google. tesseract-ocr: An ocr engine that was developed at hp labs between 1985 and 1995... and now at google. Disponível em: < https://code.google.com/p/tesseract-ocr/>. Acesso em: 10 maio 2014.

[8] Otsu, N.; A Threshold Selection Method from Gray-Level Histograms. *Automatica*, **1975**, 11, 23-27.

APÊNDICE

Figura 8: Evolução da imagem da Amostra 01 ao longo das etapas do Algoritmo de Localização e Segmentação de Placas de Veículos.



ra 9: Evolução da imagem da **Amostra 01** ao longo das etapas do Algoritmo de Localização e ıentação de Placas de Veículos.

Fonte: Autor.