CENTRO UNIVERSITÁRIO FEEVALE

IDEMAR FELAU

ESTUDO PARA DESENVOLVIMENTO DE UM SISTEMA DE EXTRAÇÃO DE CARACTERÍSTICAS DE PLACAS DE VEÍCULOS AUTOMOTORES

Novo Hamburgo, junho de 2008.

IDEMAR FELAU

ESTUDO PARA DESENVOLVIMENTO DE UM SISTEMA DE EXTRAÇÃO DE CARACTERÍSTICAS DE PLACAS DE VEÍCULOS AUTOMOTORES

Centro Universitário Feevale
Instituto de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação
Trabalho de Conclusão de Curso

Professora Orientadora: Marta Rosecler Bez

Novo Hamburgo, junho de 2008.

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial:

A Professora Marta que teve muita paciência na orientação de meu trabalho, mostrando a melhor maneira de colocação do texto, bem como idéias para o trabalho e alguns materiais de apoio.

A Denise Kreisig, que contribuiu com idéias para o trabalho, auxiliando na leitura e correção.

Ao Fabiano Bender por me auxiliar na leitura e correção.

A minha esposa e pais, que me incentivaram durante meus estudos. Aos meus familiares, amigos e às pessoas que convivem comigo.

RESUMO

Este trabalho abordará a utilização de técnicas de processamento de imagens na identificação das letras e números das placas de automóveis. Sendo assim, serão estudadas algumas técnicas de processamento de imagens para melhorar a qualidade e encontrar a melhor maneira de localizar e segmentar os caracteres da placa. Alguns testes de processamento de imagens serão apresentados, demonstrando técnicas de operações e segmentações sobre as mesmas, servindo de base para o desenvolvimento de um protótipo. Desta forma, o objetivo é desenvolvê-lo para localizar e segmentar as placas de veículos automotores a partir de imagens capturadas, para a futura implementação de um software que, além de localizar e segmentar, seja capaz de reconhecer os caracteres contidos nas placas.

Palavras-chave: Processamento de imagens. Localização. Segmentação. Placas de veículos automotores.

ABSTRACT

This work will address the use of image processing techniques in the identification of letters and numbers of plates of cars. So, will be studied some of image processing techniques to improve quality and find the best way to find and target the characters of the plate. Some tests of image processing will be presented, demonstrating techniques of operations and targets on them, providing the basis for the development of a prototype. Thus, the goal is to develop it to track and target the plates of motor vehicles from images taken in for further implementation of software that, in addition to locate and target, is capable of recognizing the characters contained in the cards.

Key words: Processing of images. Location. Targeting. Plates of motor vehicles.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RGB	Formato da imagem que utiliza as cores red, green, blue.
CMY	Formato da imagem que utiliza as cores cyan, magenta, yellow.
YIQ	Formato da imagem onde, Y corresponde luminância, I e Q cromáticos.
HSI	Formato da imagem que identifica matiz, saturação, intensidade.
HSV	Formato da imagem que identifica matiz, saturação, valor.
TV	Televisão.
UFRJ	Universidade Federal do Rio de Janeiro.
UFRGS	Universidade Federal do Rio Grande do Sul.
RS	Estado do Rio Grande do Sul.

SUMÁRIO

1 INTRODUÇÃO

Esta monografia apresenta o estudo de algumas técnicas para elaboração de um protótipo para localização e segmentação de placas de veículos, de forma automática, a partir de um banco de imagens capturadas. Neste capítulo será apresentada a motivação, objetivo e técnicas a serem estudadas.

1.1 Motivação

A crescente movimentação de veículos em empresas em geral, em especial nas do ramo automotivo, tem gerado a necessidade de aprimorar o controle dessas movimentações, buscando, com isso, maior segurança e melhores controles para essas organizações. Um exemplo é as concessionárias de veículos que apresentam diariamente um grande fluxo de automóveis em suas dependências, tanto na área de vendas quanto na de manutenção.

Um grupo empresarial atuante no estado do RS, do qual fazem parte 14 concessionárias de veículos, tendo como interesse adquirir um software para reconhecimento de placas de veículos, realizou pesquisas procurando por esse produto. Poucas empresas foram encontradas que disponibilizassem este tipo de software e, as existentes, o oferecem com preços elevados.

A movimentação nestas empresas, ultrapassando 12000 carros ao mês, cria uma necessidade de controlar o fluxo de automóveis próprio e de seus clientes.

Para isto, é necessário um sistema automático eficiente de reconhecimento de placas de veículos, que, depende basicamente de três características fundamentais: da qualidade das imagens dos veículos a ser adquiridas; da obtenção de boas características associadas às imagens que se pretende reconhecer e, finalmente, do desempenho do modelo computacional usado para o reconhecimento (RODRIGUES, 2003).

Considerando esses fatores, percebe-se um mercado promissor para desenvolvimento de um sistema de reconhecimento de placas, tendo oportunidade de explorar este mercado ainda carente, fornecendo soluções para as etapas complexas de localização e segmentação de caracteres em imagens de placas de veículos automotores.

1.2 Objetivo

O objetivo principal desta monografia é desenvolver um protótipo que localiza a placa a partir da imagem de um veículo automotor de forma automática e segmenta os caracteres da placa, preparando as informações para o desenvolvimento futuro de um software.

O processo de localização e segmentação será aplicado sobre um banco de imagens capturadas, havendo a necessidade de processar o reconhecimento das placas a partir da imagem e extrair as letras e números para ser aplicado o processo de reconhecimento para cada letra e número.

Alguns sistemas brasileiros serão apresentados, destacando algumas características como o percentual de acertos no processo de reconhecimento das placas, a plataforma em que foram desenvolvidos e o tempo necessário para o processamento das imagens. Estes dados serão importantes para efeito de comparação com o protótipo a ser desenvolvido, procurando alcançar as características apresentadas e, se for possível, melhorar alguns aspectos.

1.3 Técnicas Estudadas

Para demonstração de casos práticos apresentados neste trabalho, será utilizada a ferramenta Matlab, e para cada etapa estudada, algumas funções serão demonstradas na prática. Algumas técnicas serão citadas a seguir.

Para as imagens coloridas capturadas no formato RGB, devem ser estudadas técnicas de conversão de imagens coloridas para tonalidades de cinza.

Na melhoria das imagens, algumas técnicas de realce e extração de ruídos serão estudadas, procurando realçar características que serão importantes para segmentação e eliminar interferências para não prejudicar a localização e segmentação.

Para segmentação algumas funcionalidades como binarização e detecção de contornos serão estudados, apresentando alguns casos práticos para ilustrar o resultado aplicado sobre uma imagem.

2 PROCESSAMENTO BÁSICO DE IMAGENS DIGITAIS

O processamento de imagens digitais consiste em extrair as informações de uma imagem a partir de sucessivas transformações de modo que o resultado final seja adequado para uma aplicação específica (REIS; ALBUQUERQUE; CASTRO, 2001).

O processamento de imagens digitais é geralmente expresso de forma algorítmica e a maioria das funções pode ser implementada em software. A maior necessidade do hardware é quanto a velocidade de algumas aplicações que precisam vencer as limitações fundamentais da computação (GONZALEZ E WOODS, 2000).

Para entender melhor o processamento digital, será apresentado, no próximo tópico, a constituição de uma imagem e algumas operações que podem ser aplicadas sobre a mesma.

2.1 Constituição da Imagem

Conforme Gonzales e Woods (2000) uma imagem digital pode ser considerada uma matriz cujos índices de linhas e colunas identificam um ponto na imagem. Os elementos dessa matriz são chamados de elementos da imagem e elementos da figura, "*pixels*", sendo este o valor da intensidade luminosa de cada ponto armazenado na respectiva coordenada da matriz.

Uma imagem é dada por uma função de intensidade de luz bidimensional *f(x, y)* composta por *pixels* (pontos da imagem), resolução espacial de coordenadas espaciais para *x* e *y,* e resolução dos níveis de cinza (brilho) para o valor de *f* (DIAS, 2005).

Os objetos que os olhos capturam no dia a dia indicam luz refletida sobre as mesmas. Estas características podem ser de duas naturezas básicas: a) luminância; que é a quantidade de luz incidente sobre a cena, indicado por *i(x, y)*, e b) reflectância; que é a quantidade de luz

que se reflete sobre os objetos dados pela função r*(x, y)*. O resultado se define da seguinte forma: $f(x, y) = i(x, y) \cdot r(x, y)$.

2.2 Operações sobre a imagem

As operações sobre uma imagem são, em geral, aplicadas *pixel* a *pixel* ou entre *pixels* de imagens distintas, modificando as características da imagem (BARBOZA, 2004).

Algumas operações para melhora das imagens serão apresentadas nos tópicos a seguir. Podem ser citados como exemplo: conversão em tonalidades de cinza, binarização, extração de ruídos e realce em imagens.

2.2.1 Conversão para tonalidades de cinza

Segundo Gonzalez e Woods (2000, p.160), "*um modelo de cor é uma especificação de um sistema de coordenadas tridimensionais e um subespaço dentro deste sistema, onde cada cor é representada por um único ponto*".

Os modelos de cores mais usuais são: RGB (*red*, *green*, *blue*) usado em monitores e câmeras de vídeo, CMY (*cyan*, *magenta*, *yellow*) usado em impressoras coloridas, YIQ (*Y* corresponde a luminância, *I* e *Q* a componentes cromáticos) usado em transmissão de TV colorida. Modelos usados em manipulação de imagens coloridas são: HSI (matiz, saturação, intensidade) e o HSV (matiz, saturação, valor) (AZEVEDO; CONCI, 2003).

O modelo de cores aqui apresentado se restringe apenas para o processamento de imagens em pseudo-cores, que aborda a atribuição de cores para imagens monocromáticas baseado em seus conteúdos de níveis de cinza.

De acordo com Gonzalez e Woods (2000) duas transformações se destacam: fatiamento por intensidade e níveis de cinza para transformações de cores.

Fatiamento por intensidade, como o nome sugere, nada mais é do que fatiar a imagem em níveis de cinza. Cada plano fatia a função na área de intersecção.

Qualquer *pixel* que esteja acima ou igual ao nível do plano será codificado com uma cor e o que está abaixo será codificado com outra. O resultado é uma imagem com duas cores,

que pode ser controlado por um plano de fatiamento para cima e para baixo do eixo de níveis de cinza.

Níveis de cinza são capazes de alcançar resultados mais amplos em relação ao fatiamento. As transformações fundamentais de um modelo de cor *RGB* são basicamente três resultados separados, nas cores vermelho, verde e azul, realizando três transformações independentes sobre os valores de níveis de cinza de qualquer *pixel* de entrada de uma imagem.

$$f(x,y) = \begin{cases} \Rightarrow \text{Transformação no vermelho} \rightarrow I_R\,(x,y) \\ \Rightarrow \text{Transformação no verde} \rightarrow I_G\,(x,y) \\ \Rightarrow \text{Transformação no azul} \rightarrow I_B\,(x,y) \end{cases}$$

No exemplo anterior cada *pixel* de entrada de sua respectiva cor (vermelho, verde e azul) será transformado na tonalidade de nível de cinza.

A figura 2.1 é apresentada uma imagem no padrão RGB, contendo a intensidade de cor de cada camada “R”, “G” e “B” executado no Matlab.

Figura 2.1 – Imagem no padrão RGB.

Na figura 2.3 é ilustrado um exemplo prático usando o Matlab na aplicação do filtro *rgb2gray* para conversão da imagem colorida em tonalidades de cinza. A figura 2.2 é uma imagem original em que foi aplicado o filtro de conversão.

Figura 2.2 – Imagem original.

Figura 2.3 – Imagem aplicada a função *rgb2gray*.

2.2.2 Binarização

O processo de binarização é a transformação de uma imagem em duas cores, onde os objetos geralmente são nas cores brancas e pretas.

O processo de binarização normalmente é utilizado para segmentar uma imagem colocando uma cor em evidência, definindo um limiar do nível de cinza de cada *pixel*, como será abordado no capítulo três.

Uma imagem colorida, dependendo de sua configuração, pode conter até 16 milhões de cores, perfazendo um total de 24 bits de cores, enquanto que, em uma imagem binária, necessita apenas um *bit* por *pixel,* totalizando duas cores (MARQUES FILHO E VIEIRA NETO, apud GAZZONI, et al., 2006). A binarização facilita as aplicações no processamento da imagem e extração de informações.

O algoritmo de binarização consiste em passar para branco os *pixels* que estão acima de um limiar de cores com valor estipulado e para preto os *pixels* que estão abaixo deste valor. (RODRIGUES, 2003).

Se a[m,n] >= "limite" *a[m,n] = branco*

Senão *a[m,n] = preto*

A binarização separa os objetos da imagem em dois grupos, denominados de fundo e objeto. As classificações de binarização são citadas a seguir:

Conforme Conci, Azevedo e Leta (2008) formalmente, a limiarização converte uma imagem de entrada *f(x, y)* de *N* níveis de cinza em uma imagem *g(x, y)*. A definição de *T* é a função de variáveis na forma: *T = T [x, y, p(x, y), f(x, y)]* onde *p(x, y)* é uma propriedade local de *x, y*.

Binarização global manual: depende só de *f(x, y)*, onde é definido o valor limiar de separação com base na amostragem de toda a imagem a ser utilizada (CONCI; AZEVEDO; LETA, 2008).

Binarização global automática: onde estima o valor limiar de separação com base na densidade de probabilidade da luminância, uma para região clara e outra escura. A escolha do valor *T* é determinada por métodos estatísticos dependendo somente dos *pixels* da imagem, o que dificulta o processo de binarização quando possui vários objetos com vários limiares de separação (DIAS, 2005).

Binarização local adaptativa: a separação se dá a cada *pixel,* definindo um limiar baseado na sua vizinhança, que utiliza a média e o desvio padrão da vizinhança do *pixel* para determinar seu limiar dinâmico baseado no nível de cinza (DIAS, 2005). Sua desvantagem é a sensibilidade aos ruídos da imagem. O limiar é chamado local se *T* depender de *f(x, y)*, de *p(x, y)* e das coordenadas espaciais (*x, y*) (CONCI; AZEVEDO; LETA, 2008).

Binarização iterativa: consiste em mudar o valor do limiar de modo iterativo até binarizar corretamente o componente conexo dos objetos de uma imagem (DIAS, 2005).

Para converter uma imagem em binário, a função *im2bw* do Matlab pode ser usada: *g = im2bw(f, T)*, onde *g* é a imagem binária produzida da intensidade da imagem *f*, pelo *thresholding T*, que define a escala de níveis de cinza dos valores específicos entre *0* e *1*.

Na aplicação do limiar para imagens de tonalidades de cinza mais claro se usa o parâmetro *T* com valores próximos de '*0*' e, para imagens de tonalidades de cinza mais escuras, se aplica o parâmetro *T* com valores próximos de *1*. Caso não informado o parâmetro na função, o valor padrão do limiar *T* é 0.5 (GONZALEZ, WOODS E EDDINS, 2004).

Nas figuras a seguir serão apresentados alguns casos práticos aplicados no Matlab utilizando a função *im2bw* sobre uma figura de tonalidade de cinza, com exemplos de mais de um limiar.

A figura 2.4 é uma imagem em tonalidades de cinza utilizada na aplicação da função *im2bw* conforme mostrado na figura 2.5 com limiar de 0.5. Na figura 2.6 foi aplicada a função com limiar de 0.4. Notam-se algumas características que se perderam na imagem da figura 2.6 em relação à figura 2.5.

Figura 2.4 – Imagem original em tonalidade de cinza.

Figura 2.5 – Imagem aplicada a função *im2bw* com limiar 0.5.

Figura 2.6 – Imagem aplicada a função *im2bw* com limiar 0.4.

2.2.3 Extração de Ruídos

A remoção de ruídos deve ser aplicada quando utilizado o algoritmo de corte ou após a binarização da imagem. Para a eliminação de ruídos de uma imagem, pode-se utilizar um filtro de suavização pela mediana modificada (RODRIGUES, 2003).

Na redução de ruídos através do filtro por mediana, o nível de cinza de cada *pixel* é substituído pela mediana dos níveis de cinza na vizinhança daquele *pixel*. Este filtro preserva

a agudeza das bordas, sendo classificado como filtro mediano não linear (GONZALEZ E WOODS, 2000).

Para calcular a filtragem por mediana em uma vizinhança de um *pixel* são selecionados os valores dos *pixels* e de seus vizinhos, se determina a mediana e atribui este valor ao *pixel*. No exemplo de uma matriz de 3x3, a mediana é o 5° maior valor, para uma matriz 5x5 é o 13°, e assim dependendo do tamanho da matriz para se identificar o maior valor da mediana. No caso de haver vários valores iguais, esses devem ser agrupados. Esse filtro não faz convolução, mas sim ordena a intensidade dos pixels dentro da área da máscara em ordem crescente (CONCI; AZEVEDO; LETA, 2008).

A função *medfilt2,* da ferramenta Matlab, é uma implementação especializada em filtros medianos: *g = medfilt2(f, [m n], padopt)* onde a tupla *[m n]* define o tamanho da matriz na qual a mediana será computada, e a especificação *padopt* é uma de três possibilidades de opções de tamanho de bordas: '*zeros*' é o padrão, '*symmetric*' no qual *f* (figura) é estendido simetricamente e '*indexed*' para o processo de uma imagem indexada (GONZALEZ, WOODS E EDDINS, 2004).

A seguir será apresentado um caso prático usando o Matlab aplicando um filtro *imnoise* do tipo *gaussian* para gerar ruídos em uma imagem de tonalidades de cinza (figura 2.7). Este exemplo será usado para mostrar o resultado dos filtros de suavização por mediana das matrizes *3x3* (figura 2.8) e *5x5* (figura 2.9) utilizando a função *medfilt2*.

Figura 2.7 – Imagem aplicada ruído *gaussiano*.

Figura 2.8 – Imagem com filtro *medfilt2 – 3x3*.

Figura 2.9 – Imagem com filtro *medfilt2 - 5x5*.

No exemplo a seguir é aplicado o filtro *medfilt2* pelas matrizes de *2x2* (figura 2.11), *3x3* (figura 2.12) e *5x5* (figura 2.13) na figura binarizada pela função *im2bw* com limiar de *0.5* (figura 2.10).

Figura 2.10 – Imagem binarizada.

Figura 2.11 – Imagem com filtro *medfilt2 - 2x2*.

Notam-se poucas variações das características da figura 2.11 na aplicação do filtro *medfilt2* na figura 2.10. As características se destacam mais nas figuras 2.12 e 2.13 em relação à figura 2.11. Havendo necessidade de aumentar o tamanho da matriz para se observar as diferenças, por motivo de existir pouco ruído na imagem.

Figura 2.12 – Imagem com filtro *medfilt2 - 3x3*.

Figura 2.13 – Imagem com filtro *medfilt2 - 5x5*.

2.2.4 Realce em Imagens

O realce das imagens tem por finalidade destacar determinadas características da imagem, tendo por objetivo evidenciar detalhes da imagem para uma aplicação específica (GONZALES E WOODS, 2000).

Para realçar o contraste de objetos da imagem, uma das técnicas utilizadas na computação gráfica é o histograma, que contabiliza a quantidade de *pixels* para cada nível de cinza de um espectro (RODRIGUES, 2003).

A equalização de uma imagem se dá pela alteração do valor de cada *pixel* para uniformizar a distribuição de níveis de cinza, melhorando a iluminação dos objetos da imagem (SANTOS, 2006).

Para melhorar o contraste da imagem se deve realçar as partes mais escuras, tornando-as mais claras, e escurecendo as partes mais claras.

Conforme Souza e Correia (2008), para que as tonalidades de cinza sejam distribuídas estatisticamente em uma imagem *P*, deve-se considerar o nível de cinza *l* que ocorre n_l vezes em uma imagem na quantidade de *n pixels*.

$$P(l) = n_l/n$$

No exemplo a seguir é apresentado um modelo de histograma aplicado sobre uma imagem com tonalidades de cinza (figura 2.4), resultando na figura 2.14 após ser aplicado a função *imhist* da ferramenta Matlab, utilizada para demonstrar um histograma.

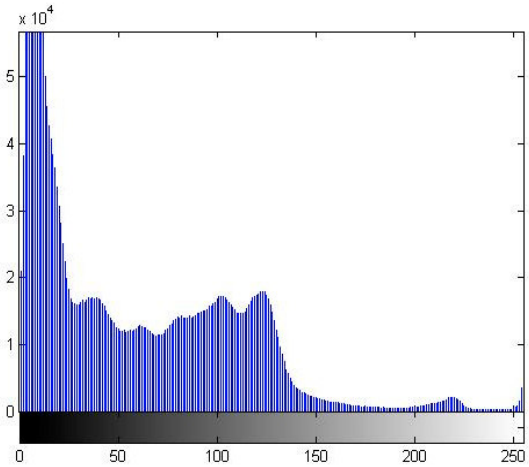

Figura 2.14 – Demonstração de um histograma pela função *imhist*.

A equalização pelo histograma é implementada na ferramenta Matlab pela função *histeq*, na seguinte sintaxe: *g = histeq(f, nlev),* onde *f* é a entrada da imagem e *nlev* é o nível de intensidade para saída da imagem. Se *nlev* é igual ao total do número de possibilidades dos níveis de entrada, se mantém o valor de intensidade na saída processada. Se o *nlev* for inferior ao total do número de entrada, então *histeq* tenta distribuir os níveis de intensidade, para que eles se aproximem do plano do histograma. O valor padrão para *nlev* é 64, caso o parâmetro não seja informado (GONZALEZ, WOODS E EDDINS, 2004).

Outra função, extraída do mesmo autor, apresentada no exemplo a seguir, é *imadjust*, usado para ajustar a intensidade da imagem. Possui a seguinte sintaxe: *g = imadjust(f, [low_in high_in], [low_out hight_out], gamma)*, onde *low_in* e *high_in* e *low_out* e *high_out* são valores específicos entre 0 e 1, os valores para *low* identificam o contraste para tonalidade escura e *high* para tonalidades claras e *gamma* especifica a intensidade dos valores de *f* para criar *g*.

No exemplo a seguir é apresentado um caso prático utilizando o Matlab. Na figura 2.16 é aplicada a função *histeq* com limiar de *128* sobre uma imagem de tonalidade de cinza. Na figura 2.17 é aplicada a função *imadjust* com limiar entre *0.3* e *0.7* e na figura 2.18 com limiar alterado entre *0.3* e *0.49*.

Figura 2.15 – Imagem original em tonalidades de cinza.

Figura 2.16 – Imagem aplicada a equalização da função *histeq*.

Figura 2.17 – Imagem aplicada a equalização da função *imadjust* com limiar 0.3 e 0.7.

Figura 2.18 – Imagem aplicada a equalização da função *imadjust* com limiar de 0.3 e 0.49.

No próximo capítulo serão demonstradas algumas técnicas de segmentação de características em imagens, que auxiliarão nos testes e desenvolvimento do protótipo do TCC II. Definições e exemplos de cada processo serão demonstrados.

3 TÉCNICAS DE SEGMENTAÇÃO E EXTRAÇÃO DE CARACTERÍSTICAS EM IMAGENS DIGITAIS

A segmentação consiste em definir na imagem, recortes dos objetos de interesse, evidenciar as cores do objeto, separar os objetos do fundo e encontrar diferenças entre dois objetos ou mais, dentre outras opções.

Algumas técnicas de segmentação serão apresentadas a seguir, como detecção de bordas, limiarização, segmentação orientada a regiões e relação entre *pixels*.

3.1 Detecção de bordas

A detecção de bordas é importante para identificação de objetos em uma imagem e, para isso, pode ser utilizado um operador derivativo local para detectar as variações de intensidade.

Para entender melhor a detecção de bordas será apresentada a detecção de descontinuidades que apresenta três tipos básicos: pontos, linhas e bordas. A detecção basicamente procura por descontinuidades através da varredura da imagem orientada a vizinhança, utilizando o conceito de convolução com máscaras.

Conforme Neves e Pelaes (2001), a cada posição relativa da máscara sobre a imagem, o pixel central da subimagem em questão será substituído, em uma matriz denominada imagem destino, pela soma dos produtos dos coeficientes com os níveis de cinza contidos na sub-região envolvida pela máscara.

A figura 3.1 é um exemplo de máscara de detecção de pontos isolados, que quando aplicado a uma imagem, destacará *pixels* brilhantes circundados por *pixels* mais escuros.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figura 3.1 – Modelo de máscara para detecção de pontos.

Na detecção de linhas a segmentação ocorre da mesma forma que na detecção do ponto, mediante uma máscara de peso em uma linha, desde que sejam adequadas as máscaras de acordo com a figura 3.2 para realçar as partes de uma reta (GONZALEZ E WOODS, 2000).

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

$+45^{O}$.

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

2	-1	-1
-1	2	-1
-1	-1	2

-45^{O}.

Figura 3.2 – Máscaras para detecção de linhas.

De acordo com Neves e Pelaes (2001), para detecção de bordas tem-se dois tipos de filtros, os gradientes baseado na luminosidade da imagem e o Laplaciano.

O filtro laplaciano é uma função derivada de segunda ordem, e como definição, o *pixel* central é positivo e os *pixels* externos são negativos e as somas dos coeficientes devem ser nulas. A figura 3.3 mostra um exemplo da máscara 3x3 que ilustra o valor dos coeficientes, que detecta a passagem por zero da segunda derivada. É muito sensível ao ruído e pouco usado para detecção de bordas (CONCI; AZEVEDO; LETA, 2008).

0	-1	0
-1	4	-1
0	-1	0

Figura 3.3 – Modelo de máscara usada na função *Laplaciana*.

Os módulos de gradientes podem ser aproximados por operadores 3x3 ou máscaras de convolução. Alguns filtros de detecção de bordas podem ser citados, como *Sobel, Roberts, Log, Prewitt*, entre outros.

A figura 3.4 apresenta um exemplo da máscara do operador *Sobel* que detecta o declive máximo da curva de níveis de cinza.

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figura 3.4 – Modelo de máscara do operador de Sobel.

O gradiente detector de contorno de Canny (CANNY, 1983) utiliza dois limiares, um para detectar contornos “fortes” e outro para os “fracos”. Os contornos “fracos” somente serão utilizados se forem conexos aos “fortes”. Desta forma, é possível detectar partes dos contornos com baixo gradiente (DIAS, 2005).

Nas figuras a seguir é apresentada a função *edge*, do Matlab, aplicando os operadores de *Sobel* (figura 3.7), *Canny* (figura 3.6), *Log* (figura 3.9), *Prewitt* (figura 3.8), *Roberts* (figura 3.13), *vertical* (figura 3.11) *e horizontal* (figura 3.12) e o filtro *Laplaciano* (figura 3.10), sobre uma imagem em tonalidades de cinza.

Figura 3.5 – Imagem original.

Figura 3.6 – Imagem aplicado operador de Canny.

Figura 3.7 – Imagem aplicado operador de Sobel.

Figura 3.8 – Imagem aplicado o operador de Prewitt.

Figura 3.9 – Imagem aplicado operador Log.

Figura 3.10 – Imagem aplicado o filtro Laplaciano.

Figura 3.11 – Imagem aplicado à função edge na vertical.

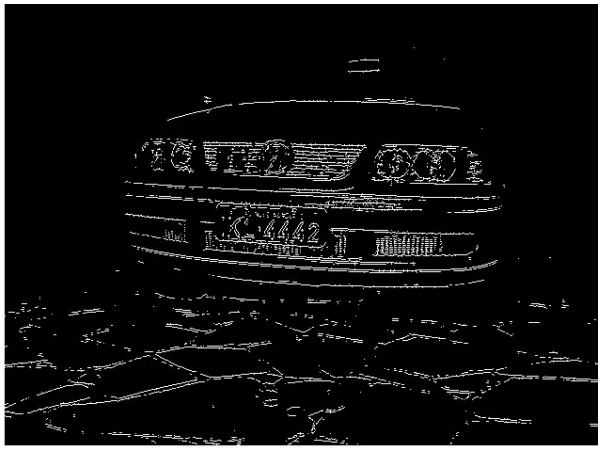

Figura 3.12 – Imagem aplicado à função edge na horizontal.

Figura 3.13 – Imagem aplicado operador Roberts.

3.2 Limiarização

A limiarização é uma binarização da imagem de tonalidades de cinza colocando em evidência um objeto pela divisão dos agrupamentos de cinza através da seleção de um limiar. Esta técnica foi apresentada na seção 2.2.2 que demonstra a aplicação do limiar em uma imagem em tonalidades de cinza.

Para se alcançar o sucesso de uma limiarização, necessita-se encontrar o melhor valor para o limiar que é a separação dos *pixels* entre os agrupamentos de níveis de cinza. Através do histograma devem ser analisados as escalas de cinza e encontrar a região que separa os agrupamentos de níveis de cinza (GONZALEZ, WOODS E EDDINS, 2004).

Em uma limiarização simples, o histograma é varrido *pixel* a *pixel* para encontrar o particionamento entre o objeto e o fundo, definido por um limiar *T* que depende do nível de cinza que seja maior ou menor que o limiar.

Conforme Gonzalez e Woods (2000), a iluminação da imagem apresenta um efeito na segmentação da mesma. Conforme demonstrado na figura 3.14, quando existirem mais de um limiar, ou seja, variações grandes nas tonalidades de cinza, dificulta o particionamento da imagem por mais de um limiar, sendo que, em uma imagem que possui apenas um limiar, pode-se particionar facilmente.

Na identificação do limiar em um histograma se considera a quantidade de *pixels* que são crescentes até o ponto máximo encontrado, denominado o pico da variação de níveis de cinza. A partir deste ponto a quantidade de *pixels* se torna decrescente até a quantidade

mínima de *pixels*. Esta variação entre os valores crescentes e decrescentes após o pico se encontra o início e fim do limiar de uma imagem.

No caso de uma imagem de iluminação pobre pode ser bastante difícil a segmentação, pois, neste caso, a variação de níveis de cinza é mínima, dificultando a identificação do limiar.

A figura 3.14 ilustra o histograma com dois limiares aplicado sobre a imagem de tonalidades de cinza da figura 2.4 pela função *imhist*.

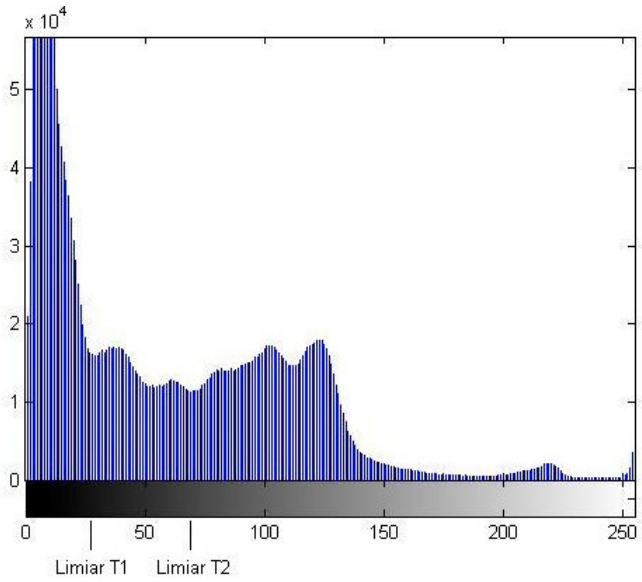

Figura 3.14 – Ilustração do histograma particionado por dois limiares pela função *imhist*.

A figura 3.15 apresenta uma imagem em tonalidades de cinza onde foi aplicada a limiarização pelo valor de $T = 64$.

Figura 3.15 – Imagem aplicada limiarização de $T = 64$.

3.3 Segmentação orientada a regiões

A segmentação orientada a regiões é o particionamento das regiões da imagem. As segmentações, conforme Conci, Azevedo e Leta (2008) se dividem em duas etapas, que são: crescimento de regiões por agregação de *pixels* e divisão e fusão de regiões.

3.3.1 Crescimento de regiões

Crescimento de regiões é um procedimento que agrupa *pixels* ou sub-regiões em regiões maiores, que comecem com um conjunto de pontos e, a partir deles, vai anexando, *pixel* a *pixel*, os de propriedades similares, tais como nível de cinza, textura e cor e, a partir deles, cresce as regiões (GONZALEZ E WOODS, 2000).

Um dos problemas do crescimento de regiões é a seleção de pontos da região de interesse, onde a seleção de um ou mais pontos iniciais pode gerar um problema de crescimento. Outro problema no crescimento de regiões é estabelecer uma condição de parada, onde o sistema deverá identificar quando nenhum outro *pixel* satisfizer os critérios de inclusão da região (CONCI; AZEVEDO; LETA, 2008).

A seguir será ilustrado o crescimento de regiões de uma imagem pela função *regiongrown* do Matlab. A sintaxe é: *[g, NR, SI, TI] = regiongrown(f, S, T)*, onde *f* é a imagem a ser segmentada e *S* pode ser uma matriz do tamanho de *f*, com 1 para as coordenadas de cada ponto a iniciar o crescimento e os demais pontos com 0s. *S* pode ser também um intervalo que define o valor da intensidade de todos os pontos de *f*. O *T* é similar ao *S*, pode ser uma matriz *T* do tamanho de *f*, contendo um valor limite para cada *pixel* em *f*, ou *T* pode ser um intervalo, neste caso, torna-se o "threshold" global.

A saída *g* é o resultado do crescimento de regiões, com cada região marcada por um inteiro diferente. O valor *NR* é o número de regiões retornado pela função. O *SI* é a última imagem do ponto de crescimento utilizada pelo algoritmo, e *TI* é uma imagem contendo os *pixels* passados de "threshold", onde eles são testados antes do processamento por conectividade (GONZALEZ, WOODS E EDDINS, 2004).

A figura a seguir ilustra o resultado da aplicação de *regiongrown* em uma imagem em tonalidades de cinza com parâmetros em escala de 45 para *S* e com valor de 44 para *T*.

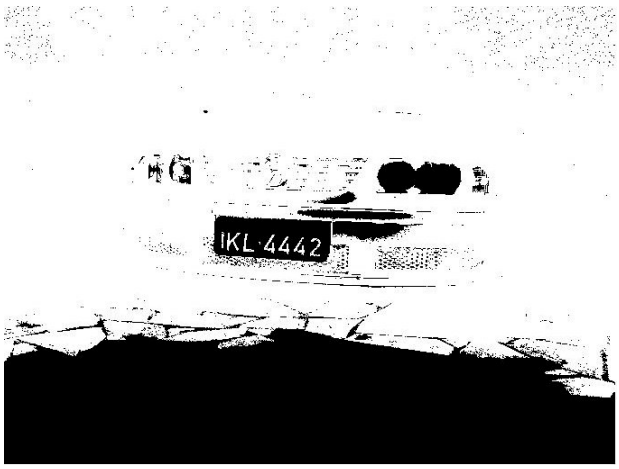

Figura 3.16 – Crescimento de região aplicado sobre uma imagem em tonalidades de cinza.

3.3.2 Divisão e fusão de regiões

O procedimento discutido no tópico anterior apresentou o crescimento de regiões a partir de um conjunto de pontos. Uma alternativa é subdividir a imagem em um conjunto de regiões arbitrárias e disjuntas, e então realizar a divisão e/ou fusão das regiões (GONZALEZ E WOODS, 2000).

Na divisão e fusão de regiões o processo de segmentação pode ser visto como um processo que particiona R em n sub-regiões R_1, R_2, R_3,....,R_N. No exemplo apresentado a seguir, P é o predicado e R a região da imagem. $P(R_i)$ é um predicado lógico sobre os pontos no conjunto R_i e deve ser uma região conectada $i = 1,2,3,...,n$. As regiões devem ser distintas para todo i e j, $i \neq j$. As propriedades devem ser satisfeitas pelos *pixels* numa região segmentada, enquanto $P(R_i) = Verdadeiro$ e todos os *pixels* em R_i tiverem a mesma intensidade não subdivide, ou seja, se o predicado de R_i for verdade, mantém a região completa. Se for $P(R_i) = Falso$, divide a imagem em quadrantes. Esta forma de representação é chamada de *quadtree* (NEVES; PELAES, 2001).

A ilustração do *quadtree* será apresentada a seguir, onde a tabela apresenta quatro regiões que sofreram segmentação e apenas o R_4 foi subdividido em quatro novos quadrantes (figura 3.17). A representação da árvore de nós do *quadtree* é apresentada na figura 3.18.

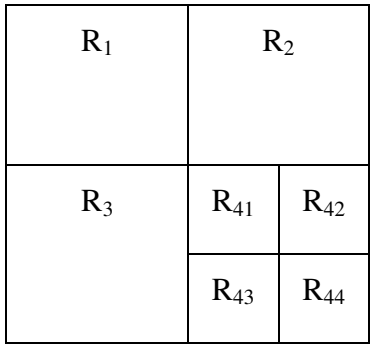

Figura 3.17 – Imagem particionada.

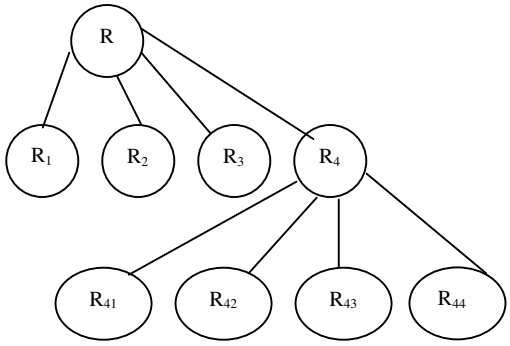

Figura 3.18 – Árvore de nós da imagem particionada.

Na partição final, após as divisões, poderia existir regiões com propriedades idênticas. Neste caso, as partes iguais devem ser fundidas no processo contrário a divisão. No caso, duas regiões adjacentes R_i e R_j são fundidas apenas se $P(R_i \cup R_j) = Verdadeiro$.

Para demonstrar a divisão e fusão será aplicada a função *splitmerge* no Matlab. A sintaxe da função é: *g = splitmerge(f, mindim, fun)*, onde *f* é a imagem de entrada da função e em *mindim* é especificado o mínimo da dimensão de *quadtree* da subimagem da região. E *fun* é o valor de *predicate, flag = predicate(region)* uma função do Matlab que retorna verdadeiro se os *pixels* na região satisfizerem o predicado definido pelo código na função, e para os outros valores, retorna falso.

A informação de *flag* é verdadeira se a intensidade do *pixel* na *region* possuir desvio padrão excedido de 10 e a intensidade de *flag* for entre 0 e 125, senão a informação de *flag* é falsa.

Na imagem a seguir foi aplicada a função *splitmerge* com valor de 2 para *mindim* (figura 3.19). Na figura 3.20 foi aumentado o valor do parâmetro *mindim* para 4 para se observar a variação das características em relação à figura anterior. O mesmo foi aplicado na figura 3.21 com aumento do valor de *mindim* para 16.

Figura 3.19 – Imagem aplicado divisão e fusão de regiões, valor 2 para *mindim*.

Figura 3.20 – Imagem aplicado divisão e fusão de regiões, valor 4 para *mindim*.

Figura 3.21 – Imagem aplicado divisão e fusão de regiões, valor 16 para *mindim*.

3.4 Relação entre *pixels*

A relação entre *pixels* nada mais é do que analisar a vizinhança de cada *pixel* para se tomar uma decisão. Conforme Dias (2005) as vizinhanças mais utilizadas são *N4* e *N8*.

Na vizinhança *N4* de um *pixel p* nas coordenadas *(x, y)* são avaliados seus quatro vizinhos, o vizinho acima do *pixel p*, abaixo do *pixel p*, o vizinho na esquerda e na direita do *pixel p*. Sendo assim, dois horizontais e dois verticais. A equação se resume em $(x + 1, y)$, $(x - 1, y)$, $(x, y + 1)$, $(x, y - 1)$. A figura a seguir ilustra a vizinhança *N4* do *pixel p* (CONCI; AZEVEDO; LETA, 2008).

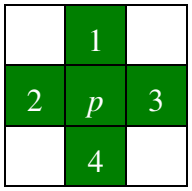

Figura 3.22 – Vizinho *N4* do *pixel p*.

Na vizinhança *ND* de um *pixel p* nas coordenadas (*x, y*), o vizinho superior e inferior diagonal do *pixel p* tanto do lado direito quanto esquerdo são considerados vizinhos. A equação se resume em (*x - 1, y - 1*), (*x - 1 , y + 1*), (*x + 1, y - 1*) e (*x + 1, y + 1*). A figura a seguir ilustra a vizinhança de *D* diagonal do *pixel p* (DIAS, 2005).

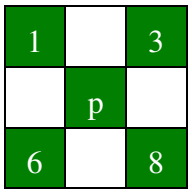

Figura 3.23 – Vizinho diagonal de *p*.

Na vizinhança *N8* de um *pixel p* nas coordenadas (*x, y*) a avaliação é feita da mesma forma que na vizinhança N4, considerando mais os vizinhos das diagonais. A equação se resume em (*x + 1, y*), (*x - 1, y*), (*x, y + 1*), (*x, y - 1*), (*x + 1, y + 1*), (*x + 1, y - 1*), (*x - 1, y + 1*), (*x - 1, y - 1*). A figura a seguir ilustra a vizinhança de *N8* do *pixel p* (CONCI; AZEVEDO; LETA, 2008).

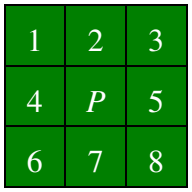

Figura 3.24 – Vizinho *N8* do *pixel p*.

Depois de entender como é analisada a vizinhança dos *pixels* é necessário conhecer a conectividade que se aplica especialmente na detecção de bordas ou regiões homogêneas em uma imagem. Conforme Barboza (2004), são quatro tipos de conectividade entre *pixels*.

A conectividade de quatro considerando que os *pixels p* e *q* são ditos conectados atendem a *S* (similaridade de cinza entre os valores, texturas ou cores), seus atributos são iguais e *q* está no conjunto $N_4(p)$ (CONCI; AZEVEDO; LETA, 2008).

A conectividade diagonal é análoga à conectividade de quatro, onde *q* deve pertencer a $N_D(p)$. A conectividade de oito, semelhante à conectividade diagonal, trocando-se $N_D(p)$ por

$N_8(p)$ (BARBOZA, 2004). Conforme UDESC (2008), a conectividade de *m*, nesse caso o *q*, deve atender ao critério de cor e pertencer à vizinhança de quatro de *p* ou atender ao critério de cor e não existir nenhum elemento comum entre os conjuntos da vizinhança de quatro de *q*, vizinhança de quatro de *p*. Dois *pixels p* e *q* com valores em *S* é: $q \supset N4(p)$ ou $q \supset ND(p)$ e $N4(p) \cap N4(q)$.

O capítulo seguinte apresentará alguns sistemas de identificação de placas existentes no mercado brasileiro, para demonstrar sua utilidade e suas características em comum. Este estudo servirá como modelo de comparação e aproximação para os testes e desenvolvimento do protótipo a ser desenvolvido no TCC II.

4 SISTEMAS EXISTENTES

Alguns sistemas de reconhecimento de placa de veículos encontrados no Brasil serão apresentados neste capítulo. Poucas características são citadas pelos autores sobre seus sistemas, obtendo poucas informações para realizar comparações mais detalhadas.

4.1 Sistema KAPTA

O sistema KAPTA foi desenvolvido no Laboratório de Inteligência Computacional da UFRJ. O sistema localiza, extrai e reconhece automaticamente as placas das imagens de veículos. A metodologia empregada no reconhecimento de placas é a de redes neurais. Segundo a universidade, a taxa de acerto aproximado é de 95% (KAPTA, 2008). O sistema leva em torno de 250 milisegundos para identificar o veículo para cada três capturas de imagem (GUINGO; STIELBER; PONTES; THOMÉ, 2008).

Figura 4.1 – Interface gráfica do sistema Kapta.
Fonte: (GUINGO; STIELBER; PONTES; THOMÉ, 2008)

4.2 Sistema SIVEM

O sistema SIVEM foi desenvolvido pela empresa Compuletra com a cooperação da UFRGS. O sistema faz o reconhecimento em tempo real dos veículos que trafegam nas ruas (COMPULETRA, 2008). A metodologia utilizada no reconhecimento de placas é a de redes neurais. Segundo a empresa, a taxa de acerto aproximado é de 95%. A ferramenta de desenvolvimento utilizado foi C e Java (INFO, 2008). Segundo a Info (2008), o policial recebe a informação do veículo cerca de 200 metros antes deste chegar ao posto rodoviário.

4.3 Sistema SIAV 2.0

O sistema foi desenvolvido pela empresa Automatisa Ltda e a Universidade Federal do Rio Grande do Sul. O sistema pode ser aplicado em controle de estacionamentos, verificação de veículos roubados e segurança. A metodologia utilizada, conforme a empresa, é a de redes neurais. A taxa de acerto aproximado é de 82,7%. O sistema leva em torno de 3,2 segundos para identificar a placa do veículo processado sobre uma plataforma de um K6 II – 400 MHz. A resolução das imagens utilizadas foi de 320x240 *pixels* e 256 tons de cinza. A ferramenta de desenvolvimento utilizado foi o C++ (UFRGS, 2008).

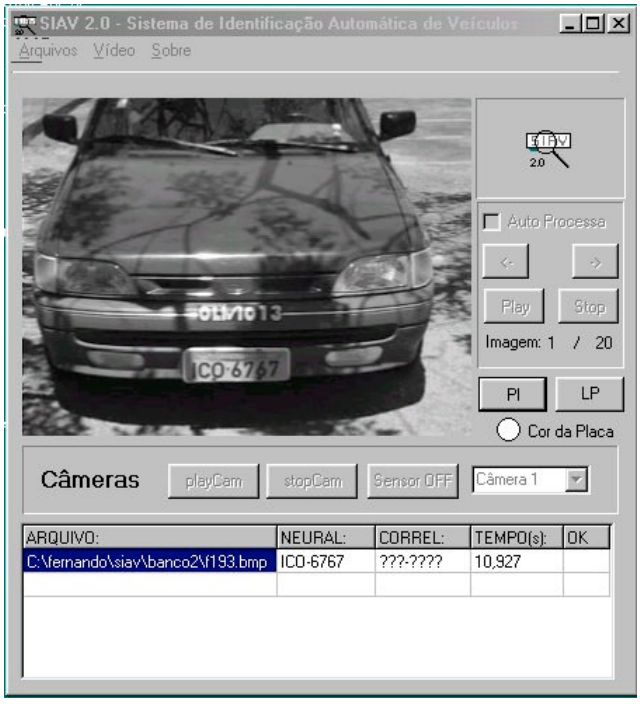

Figura 4.2 – Interface gráfica do sistema SIAV 2.0.
Fonte: (UFRGS, 2008).

4.4 Sistema PONFAC

Desenvolvido pela empresa Ponfac S/A para reconhecimento de placas de veículos, este produto está em comercialização. A metodologia adotada para reconhecimento das placas é a de redes neurais. A taxa de acerto, segundo a empresa, varia de 85% à 97%. O sistema leva em torno de 2 segundos para identificar a placa do veículo. A ferramenta de desenvolvimento da aplicação foi C++ (PONFAC, 2008).

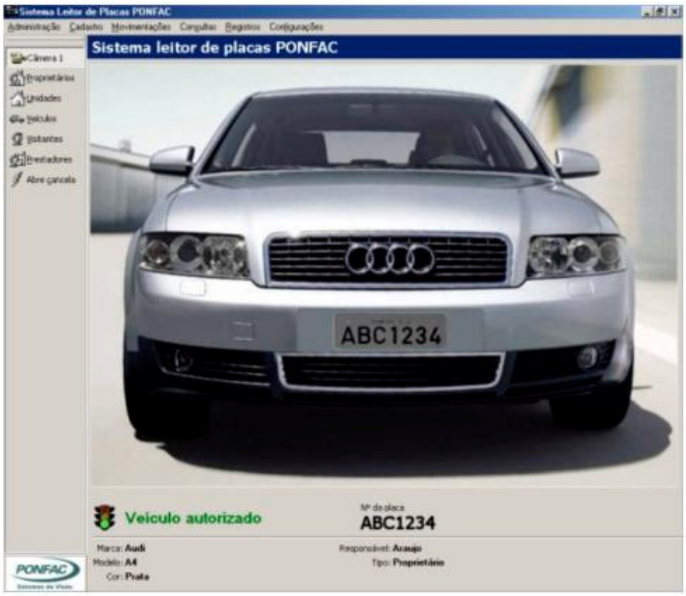

Figura 4.3 – Interface gráfica do sistema Ponfac.
Fonte: (PONFAC, 2008).

4.5 Resumo comparativo entre os sistemas citados neste trabalho

É apresentado, a seguir, uma tabela comparativa dos sistemas abordados neste capítulo.

Tabela 4.1 – Tabela resumo dos sistemas de reconhecimento de placas no Brasil.

Sistemas	Metodologia Empregada	Tempo	Taxa de Acerto
KAPTA	Redes Neurais	250 milisegundos	95%
SIVEM	Redes Neurais	Não divulgado	95%
SIAV 2.0	Redes Neurais	3,2 segundos	82,7 %
PONFAC	Redes Neurais	2 segundos	85% a 97%

Com base nos estudos realizados nos capítulos dois e três é possível definir algumas características do sistema a ser desenvolvido no TCC II.

No próximo capítulo serão abordadas todas as fases do sistema, destacando algumas abordagens que apresentaram bom desempenho nos testes.

5 PROPOSTA DO SISTEMA

Nos capítulos anteriores foram apresentados vários métodos de operações para melhorias em imagens e segmentação. Baseado nesta análise, será apresentada uma combinação de métodos para alcançar a localização e segmentação, a partir das imagens já capturadas. Este processo deve ocorrer de forma automática.

O desenvolvimento deste projeto, bem como os testes do mesmo, está sendo realizados no software Matlab versão 7.4.0.287.

O equipamento utilizado é um micro com processador AMD Athlon 2.5 GHz, 1Mb de cachê e 1.5 Gb de memória.

Para o desenvolvimento do protótipo é necessário um arquivo de parâmetros contendo todas as configurações do tamanho aproximado da placa, altura das letras e números, distância entre as mesmas, que deverá ser alimentado após realizar testes com as imagens encontrando o valor mínimo e máximo para cada parâmetro que será citado no arquivo de inicialização.

As informações que compõe a lista dos possíveis parâmetros são: tamanho mínimo da placa, tamanho máximo da placa, altura máxima da letra, altura mínima da letra, largura mínima da letra, largura máxima da letra, distância máxima entre as letras e números, distância mínima entre as letras e números e a distância entre a letra e número da placa.

Estes parâmetros podem sofrer variações caso a distância do veículo for maior do que os valores citados no arquivo de parâmetros, havendo necessidade de ajustes manuais de acordo a nova distância.

5.1 Obtenção das imagens

As imagens serão obtidas por uma máquina fotográfica da Sony D35 de 7.0 Megapixels, com resolução de 640x480 *pixels*, com a distância de três passos do veículo e meio metro de altura. Desta forma, é possível obter uma profundidade do veículo na imagem com poucas variações.

Serão obtidas 100 imagens para testar as possíveis variações como: luminosidade da imagem, ângulos, cores dos veículos, características de fundo do veículo, entre outras que poderão influenciar na localização e segmentação.

Estas imagens serão obtidas nas ruas e também em ambientes fechados, considerando as variações de luminosidades do dia, bem como da noite.

5.2 Conversão para tonalidades de cinza

A imagem adquirida pela máquina fotográfica é do padrão RGB, necessitando da conversão da imagem colorida para tonalidades de cinza, conforme apresentado na seção 2.2.1.

Uma das funções do Matlab a ser aplicada na conversão das imagens é a *rgb2gray* que será utilizada nos testes e desenvolvimento do protótipo.

A figura 5.2 apresenta a aplicação da função *rgb2gray* sobre uma imagem colorida da figura 5.1, resultando em uma imagem em tonalidades de cinza.

Figura 5.1 – Imagem original.

Figura 5.2 – Imagem aplicada a função *rgb2gray*.

5.3 Realce em imagens

Nesta etapa é importante aplicar uma equalização na imagem para melhorar a qualidade e salientar características no processo de binarização da imagem.

Uma das funções que será aplicada na equalização da imagem é a *histeq* do Matlab, apresentado em detalhes na seção 2.2.4. O limiar a ser aplicado na função *histeq* será de acordo com a maior quantidade de *pixels* de variação de tonalidades de cinza, para isto será aplicada a função *imhist* que retorna a quantidade de *pixels* para cada variação de tons da imagem.

O cálculo do limiar será com base na média dos níveis de cinza da maior quantidade de *pixels* retornados pela função *imhist*. Se a maior quantidade de *pixels* for escuro, o limiar tenderá a ficar mais próximo de 256, ou se for mais claro, o limiar tenderá a ficar mais próximo de 0.

A figura 5.4 apresenta a aplicação da função *histeq* com limiar de 128 sobre a figura 5.3 de tonalidades de cinza, destacando detalhes não observados na imagem original.

Figura 5.3 – Imagem original em tonalidades de cinza.

Figura 5.4 – Imagem aplicada a equalização da função *histeq*, com limiar de 128.

5.4 Extração de ruídos

A extração de ruídos auxiliará na diminuição da interferência dos ruídos e permitirá que se tenha uma imagem binarizada com melhor qualidade.

Uma das funções do Matlab que será aplicada no teste é o *medfilt2* que tem o objetivo de reduzir o ruído e preservar as bordas para não comprometer os próximos

processos a ser executados sobre a imagem. Para alcançar uma boa eficiência, será aplicada uma matriz de 3x3, que não distorce as arestas da imagem. Maiores detalhes podem ser vistos na seção 2.2.3.

A figura 5.6 apresenta a aplicação da função *medfilt2* sobre a figura 5.5 com uma matriz 3x3 para não interferir no processo de binarização.

Figura 5.5 – Imagem aplicada a equalização da função *histeq*.

Figura 5.6 – Imagem aplicada a função *medfilt2* - 3x3.

5.5 Binarização

A binarização é um procedimento importante, pois, é o momento de separar o fundo dos demais objetos da imagem. Para que a binarização tenha sucesso, deve ser encontrada, através do histograma, a variação das tonalidades de cinza na busca do valor correto do limiar, ou se ocorrer mais de um limiar, aplicar a binarização da média dos valores de cada limiar encontrado através do resultado do histograma.

Para iniciar o processo de binarização será aplicada a função *im2bw*, que necessita de um limiar para separar o fundo dos demais objetos. Para isto será necessário obter os valores da função *imhist* do Matlab que retorna a amostragem dos *pixels* e suas variações na tonalidade de cinza da imagem, gerando uma matriz com seus respectivos valores. A seguir será feita a busca por possíveis limiares em base da quantidade de *pixels* e variações de cinza registrada na matriz. Nas seções 2.2.2 e 3.2, pode ser acompanhada a teoria referente a binarização.

A figura 5.7 apresenta o histograma da figura 5.6, mostrando as variações nas tonalidades de cinza da imagem.

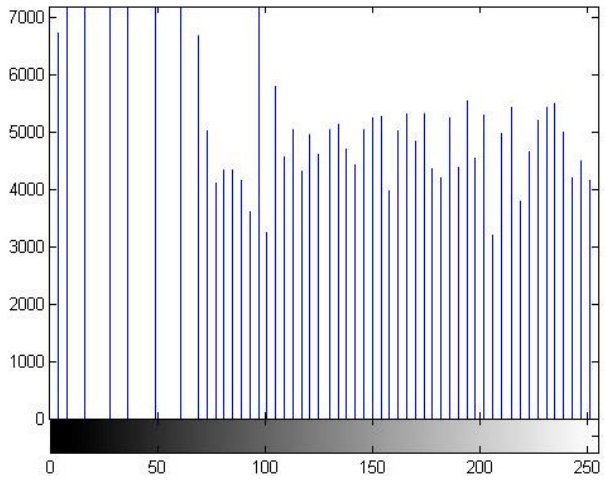

Figura 5.7 – Histograma da figura 5.6.

O cálculo do limiar é feito com base na maior amostragem de *pixels* encontrados na matriz. Se encontrado mais de um limiar, será aplicada uma média entre o intervalo dos limiares encontrados. A maior amostragem de *pixels* indica a cor em evidência na imagem.

Na figura 5.8 foi aplicada a função *im2bw* com limiar de 0.9 sobre a figura 5.6, resultando em uma imagem binarizada.

Figura 5.8 – Imagem binarizada.

5.6 Remoção do fundo

Completada a binarização da imagem, é necessário identificar cada objeto da mesma. Este procedimento deve avaliar se o objeto é de forma retangular e se o tamanho do objeto está próximo ao tamanho registrado no arquivo de parâmetros.

O arquivo de parâmetros deverá ser ajustado pelo usuário de acordo o ambiente a ser instalado, ou melhor, onde serão adquiridas as imagens. Será levado em consideração à distância da imagem capturada permitindo um desvio padrão para as possíveis variações encontradas.

Neste procedimento, é necessário que seja aplicada a metodologia da vizinhança de *pixels*, agrupando os *pixels* de acordo a regra N8, como explicado na seção 3.4. Este agrupamento de *pixels* identificará um objeto que será avaliado na sua forma e tamanho.

A figura 5.9 apresenta a imagem binarizada mostrando possíveis grupos de *pixels*, identificando alguns objetos da imagem em cor vermelha.

Figura 5.9 – Imagem binarizada indicando alguns objetos.

5.7 Filtragem dos grupos de objetos

Alcançada a localização da placa, as letras e números da região devem ser encontrados e segmentados.

Para que isto seja possível, é necessária a aplicação da metodologia da vizinhança dos *pixels* para agrupar os mesmos, conforme a regra N8, como explicada na seção 3.4, e encontrar os objetos da placa localizada.

Será aplicado, nesta avaliação dos objetos, o cálculo de altura, largura, distância das letras, números e alinhamento dos mesmos. A conferência verificará se totalizou sete objetos. Se não encontrou, reprocessará a imagem em busca de nova localização, desconsiderando esta localização mal sucedida. As informações de altura, largura, distância e o desvio padrão deverão constar no arquivo de parâmetros.

A figura 5.10 apresenta um exemplo de uma placa localizada a partir da figura 5.9. A figura 5.11 ilustra os objetos que identificam as letras e números a ser segmentada a partir da figura 5.10.

Figura 5.10 – Imagem da placa localizada.

Figura 5.11 – Imagem da placa indicando os possíveis objetos.

5.8 Testes do Sistema

O protótipo a ser desenvolvido será comparado com as características de alguns sistemas existentes no Brasil, citados no capítulo quatro, visando alcançar o tempo de processamento de cada imagem capturada com valores próximos aos sistemas propostos.

A tabela 4.1 serve de modelo para os testes a ser realizadas com o sistema proposto no capítulo cinco, servindo de base para alcançar a taxa de acertos e o menor tempo para localização e segmentação das letras e números da placa.

Outras características como resolução das imagens utilizadas nos sistemas, citadas no capítulo quatro, serão analisadas. O processamento das imagens com mais ou menos resolução, pode oferecer ganho em tempo de processamento das mesmas. Devem ser levadas em consideração as características da imagem, que poderão ser perdidas, caso a diminuição da resolução seja muito baixa, dificultando o processo de localização e segmentação.

Outro aspecto importante é o processamento de mais de uma captura de imagem, se é obtido algum ganho significante comparado ao tempo de processamento gasto para isto. Pois, conforme apresentado no sistema Kapta (2008) o autor cita no processo de reconhecimento três capturas, considerando que sua taxa de acerto é bem elevado.

CONCLUSÃO

O objetivo deste trabalho é alcançar a localização e segmentação de um banco de imagens capturadas, por um processo de reconhecimento de placas e seus objetos, como letras e números, que deverá acontecer de forma automática.

Para este estudo, foi necessário buscar informações a respeito de como operar e processar as imagens, através de técnicas de processamento digital de imagens.

Algumas técnicas foram apresentadas e estudadas, como: converter as imagens em tonalidades de cinza, realçar as características, extrair ruídos para amenizar as interferências e binarizar mediante uma média aplicada sobre a maior quantidade de *pixels* que identificará um intervalo de cores, resultando no limiar a ser aplicado, ou seja, converter a imagem em duas tonalidades de cor, preto e branco, colocando em evidência o objeto de interesse.

Após entender o funcionamento destas técnicas, apresentam-se os resultados obtidos das operações propostas. Resumindo em uma imagem com objetos evidenciados pela limiarização. Neste contexto houve a necessidade de identificar os objetos. Para isto, foi necessário buscar informações de como identificá-las. Nesta etapa informações bibliográficas foram pesquisadas de como relacionar os *pixels* com suas vizinhanças.

Encontrado as várias maneiras de se relacionar os *pixels*, foi proposto no capítulo cinco agrupar os *pixels*, formando os objetos para que sejam avaliados sua forma e tamanho. Assim, facilitará a análise dos objetos da imagem no processo de localização da placa, conseguindo, dessa forma, encontrar o objeto que se encaixa no perfil de uma placa de automóvel.

Da mesma forma para segmentar as letras e números da placa, serão avaliados os objetos agrupados, pela sua altura, largura, distância e alinhamento dos mesmos. Deve ser

levado em consideração o arquivo de parâmetros que delimitarão a variação do tamanho dos objetos da imagem a ser processada.

Testes de cada uma das etapas de processamento foram apresentados, tendo como base sempre a mesma imagem, para garantir a confiança em todo o processo e comparação entre as funções relacionadas.

Com base no resultado dos testes de cada fase do sistema, foi proposto uma seqüência para desenvolvimento do protótipo. Faz-se necessário realizar testes com outras imagens para ajustar os parâmetros do mesmo.

Este arquivo conterá os seguintes parâmetros: tamanho mínimo da placa, tamanho máximo da placa, altura máxima da letra, altura mínima da letra, largura mínima da letra, largura máxima da letra, distância máxima entre as letras e números, distância mínima entre as letras e números e a distância entre a letra e número da placa.

Conforme citado na tabela 4.1, espera-se alcançar os percentuais de acerto no reconhecimento das placas de veículos, bem como, o tempo gasto para processar cada imagem adquirida, podendo alcançar bons resultados do protótipo, que deverá ser testado com pelo menos 100 imagens a ser obtidas em vários ambientes, considerando a variação de luminosidade do dia, bem, como da noite.

REFERÊNCIAS BIBLIOGRÁFICAS

AZEVEDO, Eduardo; CONCI, Aura. Computação Gráfica – Teoria e Prática. Rio de Janeiro: Elsevier, 2003.

BARBOSA, Jorge Leonardo Lima. Processamento de Imagens Aplicado à Monitoração de Processos. (Dissertação de Mestrado). Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2004.

CANNY, J. F. Finding Edges and Lines. Tech Report 720. Massachusetts Institute of Technology, Cambridge, MA, USA, 1983.

COMPULETRA, Sistema de Identificação de Veículos em Movimento – SIVEM. Disponível em: <http://www.compuletra.com.br>. Acesso em 02 de maio de 2008.

CONCI, Aura; AZEVEDO, Eduardo; LETA, Fabiana R. Computação Gráfica: Teoria e Prática. v.2, Rio de Janeiro: Elsevier, 2008.

DIAS, Fábio Gaiotto. Melhorias para Sistemas de Reconhecimento da Placa de Licenciamento Veicular. (Dissertação de Mestrado da Faculdade de Engenharia Elétrica e de Computação). UNICAMP, Campinas, 2005.

GAZZONI, Jean Carlos, et al.. Limiarização e binarização na análise de objetos em imagens digitais. Synergismus scyentifica UTFPR, Pato Branco / PR, 2006. Disponível em: <http://pessoal.pb.cefetpr.br/eventocientifico/revista/artigos/0607008.pdf>. Acesso em 10 de março de 2008.

GONZALEZ, Rafael C.; WOODS, Richard E.. Processamento de Imagens Digitais. Editora Edgard Blucher, Brazil, 2000. 509 p.

GONZALEZ, Rafael. C.; WOODS Richard E.; EDDINS Steven L. Digital Image processing using MATLAB. Upper Saddle River, N.J: Prentice Hall, 2002. 609 p.

GUINGO, Bruno Clemente; STIELBER, Guilherme Matosinho; PONTES, Thiago da Silva; THOMÉ, Antonio Carlos Gay. KAPTA - Um sistema de reconhecimento automático de placas de veículos e suas aplicações potenciais. Disponível em: <http://www.labic.nce.ufrj.br/downloads/cpet_2004.pdf>. Acesso em 02 de maio de 2008.

INFO. No Brasil, seis pontos de fronteira já têm um sistema de reconhecimento de placas. Disponível em: <http://info.abril.com.br/edicoes/229/arquivos/4988_1.shl>. Acesso em 02 de maio de 2008.

KAPTA. Kapta – Sistema de Reconhecimento Automático de Placas de Veículos Automotores. Disponível em: <http://www.labic.nce.ufrj.br/projetos.htm>. Acesso em 02 de maio de 2008.

NEVES, S. C. M.; PELAES, E. G. Estudo e implementação de técnicas de segmentação de imagens, 26/10/2001. Edição N° 02 - outubro / 2001. Disponível em: <http://www.cultura.ufpa.br/rcientifica/ed_anteriores/pdf/ed_02_scmn.pdf>. Acesso em 02 de maio de 2008.

PONFAC. Leitor de Placas PONFAC. Disponível em: <http://www.ponfac.com.br>. Acesso em 02 de maio de 2008.

REIS, Caimi F.; ALBUQUERQUE, Marcelo Portes; CASTRO, Sérgio B. Introdução ao Reconhecimento de Padrões utilizando Redes Neurais. Rio de Janeiro: CBPF, publicação CBPF-NT-002/01, 2001. Disponível em: <ftp://ftp2.biblioteca.cbpf.br/pub/apub/2001/nt/nt_zip/nt00201.pdf>. Acesso em 10 de março de 2008.

RODRIGUES, Roberto José. Segmentação e Extração de Características para Reconhecimento Automático de Caracteres – Estudo e Propostas. (Dissertação de Mestrado). Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2003.

SANTOS, Vitor M. F. Elementos de Imagem e Visão por Computador. Aula Teórica Robótica Industrial, 2006. Disponível em: <http://www2.mec.ua.pt/activities/disciplinas/RoboticaIndustrial/2006-2007/AulasTeoricas/Slides/ImagemVisaoPorComputadorEusoDaCor.pdf>. Acesso em 10 de março de 2008.

SOUZA, Taciana; CORREIA, Suzete. Estudo de técnicas de realce de imagens digitais e suas aplicações. Disponível em: <http://www.redenet.edu.br/publicacoes/arquivos/20080127_131848_INFO-022.pdf>. Acesso em 07 de março de 2008.

UDESC. O Sistema de Visão Artificial - Vizinhança. Disponível em: <http://www.joinville.udesc.br/processamentodeimagens/vizinhanca.html>, acesso em 16 de maio de 2008.

UFRGS, Projeto SIAV - Sistema de Identificação Automática de Veículos (versão 2.0). Universidade Federal do Rio Grande do Sul - UFRGS/ Escola de Engenharia Departamento de Engenharia Elétrica – DELET / Laboratório de Processamento de Sinais e Imagens – LaPSI. Disponível em: <http://www.lapsi.eletro.ufrgs.br/projetos/siav/siav2.htm#Plataforma>. Acesso em 02 de maio de 2008.