Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

EMMANUEL AIRES URQUIZA DE CARVALHO

IDENTIFICAÇÃO ÓPTICA DE CARACTERES DE PLACAS DE TRÂNSITO

Campina Grande, Paraíba
Maio de 2015

EMMANUEL AIRES URQUIZA DE CARVALHO

IDENTIFICAÇÃO ÓPTICA DE CARACTERES DE PLACAS DE TRÂNSITO

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento de informação.

Orientador:
Marcos Ricardo Alcântara Morais, D. Sc.

Campina Grande, Paraíba
Maio de 2015

EMMANUEL AIRES URQUIZA DE CARVALHO

IDENTIFICAÇÃO ÓPTICA DE CARACTERES DE PLACAS DE TRÂNSITO

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento de informação.

Aprovado em / /

Jaidilson Jó da Silva
Universidade Federal de Campina Grande
Avaliador

Marcos Ricardo Alcântara Morais, M. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Campina Grande, Paraíba
Maio de 2015

Dedico este trabalho aos meus pais. À minha mãe por todos os esforços que fez para que eu pudesse concluir meus estudos. E ao meu pai, que me incutiu o amor pela ciência e pelo conhecimento, mas que não viveu para ver minha graduação, de modo que, infelizmente, esta dedicatória terá que ser *in memoriam.*

AGRADECIMENTOS

A obtenção do meu título de engenheiro e, talvez de qualquer título, de qualquer pessoa, foi praticamente um trabalho coletivo, tal a quantidade de ajuda que recebi de familiares, amigos, professores e funcionários da UFCG ao longo da minha graduação. Devo, pois, utilizar este espaço para agradecer a todos os que me permitiram chegar aqui.

Aos colegas, professores e funcionários que me ajudaram, gostaria que me perdoassem por não citá-los nominalmente neste espaço. Faço isso porque não confio na minha própria memória e tenho medo de ser ingrato e esquecer algum amigo, já que, felizmente, tenho muitos. Portanto, me dirijo a todos os inestimáveis colegas, funcionários e professores que me ajudaram: foi uma honra.

A minha família certamente merece um agradecimento mais profundo do que o que eu sou capaz de pensar ou escrever neste modesto espaço. Certamente não sou capaz de expressar como gostaria a minha eterna gratidão pelo extraordinário e permanente apoio. O que posso fazer é usar todo o resto da minha vida para honrá-los e transmitir os seus (nossos) valores.

"Aqueles que buscam o sucesso precisam, primeiro, fazer as perguntas certas."

Aristóteles.

RESUMO

O presente estudo tem como objetivo o desenvolvimento de um sistema que permita a identificação de uma placa de automóvel recebendo como entrada uma imagem contendo o veículo. O sistema deve detectar uma placa na imagem e reconhecer os caracteres. Para isto fez-se uma pesquisa bibliográfica em visão computacional, processamento de imagens digitais e reconhecimento de padrões. As ferramentas computacionais utilizadas foram o ambiente de programação Visual Studio, a biblioteca OpenCV e o *software* Tesseract. Para a detecção da placa na imagem utilizou-se um treinamento de Haar. Em seguida a imagem da placa foi separada do todo e processada. Após o processamento a imagem foi submetida ao reconhecedor de caracteres. O processo de detecção da placa na imagem atingiu uma precisão de 50%. Um dos algoritmos testados separou a imagem da placa e seus caracteres adequadamente. Entretanto o identificador de caracteres produziu resultados aquém do esperado. A causa provável é que o Tesseract precisa ser treinado para a fonte das placas após o processamento da imagem.

Palavras-chave: Visão Computacional, OCR, LPR, Tesseract , OpenCV.

ABSTRACT

This study aims to develop a system to identify an automobile license plate receiving as an input an image containing a picture of the vehicle. The system should detect the plate in the image and recognize the characters. In order to do this, a bibliographic research was carried out in computer vision, digital image processing and pattern recognition. The Visual Studio programming environment, the library OpenCV and the software Tesseract were the computational tools used to accomplish the objective. For the detection of the plate in the image, a Haar training was applied. After detection, the image of the plate was separated from the rest and processed and then submited to the character recognition phase. The process of detection reached a precision of 50% and one of the algorithms tested separated the image from the plate and its characters adequately, however the character identifier produced results which were below the expected threshold. The likely cause for that is that Tesseract needs to be trained in the specific font used in the plates after image processing has been carried out.

Keywords: OpenCV, OCR, License Recognition, Tesseract, Computer Vision

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O objetivo da pesquisa é o desenvolvimento de um sistema que seja capaz de identificar a placa de um carro em uma imagem e reconhecer os caracteres da placa. Por dificuldades técnicas em obter imagens de alta resolução não identificaremos o nome da cidade e do estado, presentes na placa.

A tecnologia de reconhecimento automático de placas de veículos, também conhecida pela sigla em inglês "LPR" (*license plate recognition*), pode ser usada para diversas aplicações. A tecnologia é limitada pelas condições ambientais como iluminação, velocidade do veículo, posição, dano mecânico na placa (Kwasnicka, 2002) e fundo da imagem estático (Gilly, 2013).

Essencialmente um sistema de LPR deve transformar os *pixels* de uma placa em caracteres correspondentes aos caracteres escritos na placa (Sundari, 2014). Os sistemas de LPR, normalmente operam em três fases: capturam a imagem do carro, separam a imagem dos caracteres da placa e por fim, identificam os caracteres (Sundari, 2014).

Os sistemas de reconhecimento de caracteres de placas de automóveis têm sido utilizados para diversos fins desde aplicações policiais até segurança comercial, como, por exemplo, controle de um estacionamento (Gilly, 2013). Um sistema automático de leituras de caracteres também pode beneficiar prefeituras que promovem rodízio de carros que podem trafegar.

Segundo o relatório do departamento de polícia do estado canadense de British Columbia (Cohen, 2007), um sistema de LPR foi instalado nas ruas e rodovias e a polícia descobriu que o roubo de carro associa-se a um número considerável de outros crimes como assalto a mão armada, roubo residencial e crimes relacionados a tráfico de drogas Um assaltante tende a roubar um veículo para transportar o produto do roubo e facilitar a fuga, antes de cometer outros assaltos. A mesma coisa ocorre para infrações de trânsitos mais leves. Por exemplo, foi descoberto que um terço das pessoas que utilizam vagas de estacionamento de deficientes tinham registro criminal, quase metade (49 por cento) estão envolvidas em outras infrações de trânsito, 21 por cento eram de interesse imediato da polícia e 18 por cento eram suspeitas de outros crimes.

Ainda segundo o relatório da polícia de British Columbia, uma outra vantagem percebida de um sistema de LPR associado a um bom banco de dados das placas dos veículos roubados ou de fugitivos da justiça é o aumento da objetividade da análise da placa. Pelas limitações do contingente policial de quase qualquer cidade não é possível que os agentes sejam capazes de averiguar se a placa de cada veículo que passe pela estrada está na lista das placas de veículos roubados, então, é necessário que os policiais tomem decisões subjetivas baseados na experiência e no comportamento suspeito do motorista. Não é uma suposição absurda que deixem de perceber placas de criminosos

Para maior eficiência dos sistemas, o tempo de duração do reconhecimento deve ser o mais rápido possível (Kwasnicka, 2002), embora algumas aplicações sejam mais dependentes do tempo do que outras. Um sistema instalado em cada vaga de um estacionamento em que o tempo médio de parada seja de quinze minutos depende menos do tempo do que um sistema de reconhecimento instalado em uma rodovia movimentada ou mesmo em um estacionamento em que o sistema fique instalado na entrada e não na vaga do carro. Se a captura da imagem ocorrer com o veículo em movimento o problema torna-se mais complexo, sendo necessário o uso de técnicas para diminuir o borrado das imagens (Sundari, 2014). As principais questões técnicas dos sistemas de LPR são: a eficiência do software utilizado e a qualidade da imagem obtida pela câmera e iluminação (Sundari, 2014).

A etapa de reconhecimento da placa utiliza diversos elementos de inteligência artificial ou reconhecimento de padrões tais como Redes Neurais Artificiais (Gilly, 2013), algoritmos genéticos e evolutivos (Sundari, 2014). Tais técnicas são necessárias devido à diversidade de imagens produzidas. As placas não são homogêneas em relação a ferrugem e sujeira, podem estar posicionadas de forma diferente em relação à câmera e a iluminação pode estar diferente.

O reconhecimento ótico de caracteres (em inglês, *optical character regonition*, ou OCR), utilizado na etapa posterior dos sistemas de LPR, após isolarmos a placa da imagem, é por si só um campo de estudos envolvendo inteligência artificial e reconhecimento de padrões e existem diversos sistemas tanto de código aberto como fechado disponíveis. É importante escolher o melhor sistema de OCR de acordo com a aplicação.

Diversos sistemas utilizam dicionários internos e seu algoritmo tende a priorizar palavras deste dicionário. Para uma aplicação em reconhecimento de placas de veículo este tipo de prioridade tende a produzir resultados negativos, pois placas não formam

palavras conhecidas de dicionários, sendo necessário desativar esta propriedade no algoritmo e limitá-lo ao reconhecimento das letras e números, sem viés em relação à formação ou não de uma palavra em algum idioma conhecido.

A metodologia deste trabalho consistiu em pesquisa bibliográfica e testes estatísticos.

2 DESENVOLVIMENTO

O presente trabalho pode ser dividido em três tópicos principais, são eles:

- Identificação da placa na imagem
- Pré-processamento da imagem
- Detecção dos caracteres

No decorrer deste capítulo, cada um destes tópicos será explicado detalhadamente.

2.1 IDENTIFICAÇÃO DA PLACA NA IMAGEM

Para que o sistema seja efetivo em sua função de monitoramento é necessário, preliminarmente, que o sistema seja capaz de identificar a presença de uma placa de automóvel em uma imagem. Caso haja uma placa na imagem o sistema deve atuar para processar a imagem da placa. Quando o sistema identifica uma placa em uma imagem em que há uma placa diz-se que aconteceu um "positivo verdadeiro" ou um *hit*. Caso o sistema identifique uma placa em uma imagem em que não há nenhuma placa, trata-se de um "falso positivo".

Existem várias formas de se identificar uma placa em uma imagem e o método utilizado dependerá do conhecimento prévio a respeito da posição da placa na imagem. No melhor caso possível, seria conhecida a posição exata da placa e só seria necessário isolar esta posição na imagem. No pior caso, não seria conhecida nenhuma informação à cerca da posição da placa na imagem.

A intenção do trabalho é a criação de um sistema mais geral, que requeira menos conhecimento prévio sobre a imagem. Entretanto, o *data set*, isto é, o conjunto de imagens utilizado, possui imagens relativamente similares entre si, e, para este conjunto específico, com as características das imagens que foram testadas, o ideal é uma abordagem simples, determinando a área média em que as placas se situam na imagem. Mas o conjunto de testes serve apenas para testar as hipóteses de forma mais simples, já que um treinamento de Haar utilizando um conjunto de treinamento maior teria tempo duração maior. Caso as técnicas utilizadas neste *data set* mostrem-se eficazes, pode-se fazer inferências à respeito das aplicações destas mesmas técnicas em um conjunto de imagens maior e com maior variedade em seus elementos.

Para identificar imagens com maior variedade é necessário usar classificadores mais complexos. O classificador escolhido é denominado “Cascata de Haar” e seu algoritmo, que já está implementado na biblioteca OpenCV.

2.1.1 CASCATA DE HAAR.

O algoritmo deve efetuar um treinamento e produzir uma árvore de dados que descreve o padrão a ser reconhecido. No caso do executável utilizado pela biblioteca OpenCV em questão a árvore de dados está expressa em um arquivo .xml, que pode ser interpretado por funções da mesma biblioteca para reconhecer o padrão em uma imagem. O algoritmo precisa, como dados de entrada, imagens positivas, que contém o objeto a ser detectado e imagens negativas, que não contém o objeto (Viola e Jones, 2001). .

Inicialmente consideram-se as imagens mostradas na Figura 1. Tais imagens são chamadas de modelos de Haar.

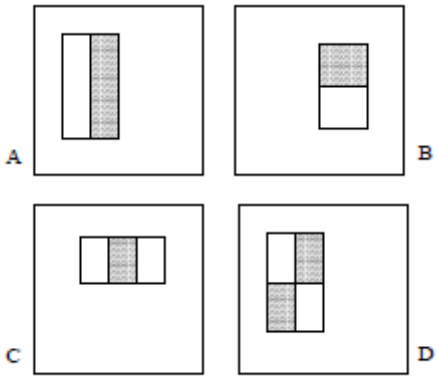

Figura 1: Modelos de Haar.

Uma característica de Haar é definida como um desses modelos sobreposto à imagem do objeto a ser identificado. Associado a este modelo um valor calculado como a soma das intensidades dos *pixels* da região da imagem original que está sobreposta pela área preta menos a soma das intensidades dos *pixels* da região da imagem original que está sobreposta pela área branca. A Figura 2 exemplifica isto.

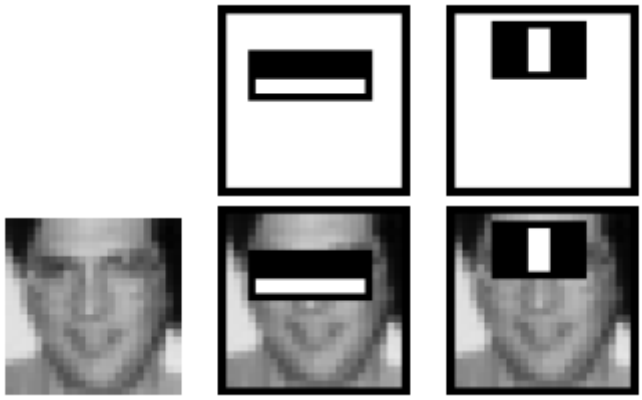

Figura 2: Exemplo de modelos de Haar aplicados a uma imagem.

É necessário fornecer a região de interesse do objeto nas amostras positivas. A menor região será usada como parâmetro para o cálculo da quantidade de características existentes.

As características de Haar podem ter tamanho de dois *pixels* até serem do tamanho da imagem que estão se sobrepondo. Também podem estar em qualquer posição dentro da imagem. Em uma imagem de tamanho 24 por 24 a quantidade de características será maior do que 160.000, segundo o site oficial da biblioteca OpenCV. Imagens maiores possuirão ainda mais características.

Com a finalidade de minimizar o esforço computacional, Viola e Jones utilizaram-se do conceito de imagem integral.

Uma imagem integral é uma função de uma outra imagem, neste caso a imagem do objeto a ser detectado. A relação entre a imagem integral e a imagem de entrada é que, na imagem integral, cada *pixel*, em uma posição definida por um par ordenado x e y, terá valor de intensidade igual à soma de todos os *pixels* da imagem original que estão à esquerda de x e acima de y (a imagem original está no mesmo sistema de coordenadas). Para exemplificar a Figura 3 apresenta as imagens de entrada e saída, representadas por matrizes cujos valores dos elementos correspondem a intensidade do *pixel* na posição referente ao elemento.

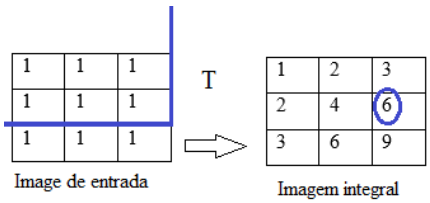

Figura 3: Transformação em imagem integral.

O elemento circulado na imagem integral tem o valor da soma de todos os componentes que estejam acima ou na mesma coordenada vertical e que estejam a esquerda ou na mesma coordenada horizontal.

Definida a imagem integral é possível utilizá-la para calcular a soma das intensidades dos *pixels* em quaisquer regiões retangulares da imagem de entrada realizando-se apenas duas somas e uma subtração. A Figura 4 será utilizada para mostrar como isso é feito.

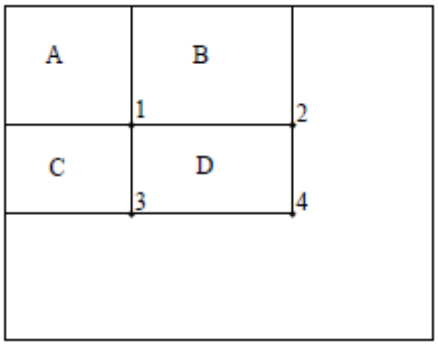

Figura 4: Exemplo do uso da imagem integral.

Para calcular o valor da soma das intensidades dos *pixels* na região D, correspondente à imagem de entrada sem o uso da imagem integral seria necessário fazer uma varredura na região e somar as intensidades.

Na imagem integral o *pixel* correspondente à posição 1 já possui o valor do somatório das intensidades em A, o *pixel* correspondente à posição 2 possui os valores de A + B, o correspondente a 3 possui os valores de A+C e o *pixel* 4 possui a informação de A+B+C+D. O valor da soma das intensidades da região D está na Equação 1.

$$D = (P4 + P1) - (P3 + P2) \tag{1}$$

.

Onde P1, P2, P3 e P4 são os valores de intensidades associados aos *pixels* 1, 2, 3 e 4 na imagem integral.

Este método é usado para calcular as características de Haar, considerando cada característica como um conjunto dos seus sub-retângulos pretos ou brancos e calculando a soma das intensidades correspondente a cada um desses sub-retângulos.

Definidas todas as características, é necessário escolher as que são relevantes e as que não são.

Cada imagem positiva e negativa tem suas características extraídas, com seus valores determinados. Para cada característica estabelece-se o valor de limiar que separa a maior quantidade de imagens positivas das negativas. Caso este limiar não consiga separar em mais de 50% as imagens positivas das negativas, a característica é descartada.

É recomendado que se utilizem as 200 características que separaram mais imagens positivas das negativas (Viola e Jones, 2001).

Escolhidas as características utilizadas define-se uma função como a da Equação 2.

$$h_j(x) = 1, \text{ se } p_j c_j < p_j T_j$$
$$h_j(x) = 0, \text{ caso contrário} \tag{2}$$

Onde $h_j(x)$ é uma função que tem como entrada uma imagem x, caso o valor da característica c_j esteja acima ou abaixo de um limiar T_j o valor da função será 1. A

variável de paridade p_j indica se o valor da caraterística deve ser maior ou menor que o limiar.

Essas funções são chamadas de classificadores fracos. Define-se uma função $F(x)$ como a combinação linear de todos os classificadores fracos. A função $F(x)$ é descrita pela Equação 3.

$$F(x) = h_1 a_1 + h_2 a_2 + ... + h_{final} a_{final} \quad (3)$$

.

Um tipo de classificador com taxa de acerto superior ao de cada classificador fraco pode ser implementado utilizando-se a Equação 3((Viola e Jones, 2001).). Este classificador é denominado classificador forte e sua função está na equação 4((Viola e Jones, 2001).).

$$H(x) = 1 \text{ , se}$$
$$F(x) >= \frac{1}{2} \sum_{j=1}^{final} a_j \quad (4)$$
$$\text{Caso contrário}$$
$$H(x) = 0$$

Para determinar os coeficientes a_j da combinação linear que otimizam a taxa de acertos é utilizado um algoritmo denominado *AdaBoos*t. O algoritmo é formado pelos seguintes procedimentos:

- Dados os exemplos de n imagens $(x_1, y_1) \dots (x_n, y_n)$, onde x_i representa a imagem em si e y_i indica se imagem é uma amostra positiva ou negativa, sendo igual a 1 caso a imagem seja positiva e zero caso seja negativa.
- Inicializar as variáveis correspondente aos pesos $w_{i,1} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0,1$, respectivamente, onde m e l são os números de amostras negativas e positivas, respectivamente.
- De t=1 até T(número de iterações) fazer:

a) Normalizar os pesos, isto é, $w_{t,i} \leftarrow w_{t,i} / \sum_{j=1}^{n} w_{t,j}$. Assim w_t torna-se uma distribuição de probabilidade.

b) Para cada característica j calcular o erro do classificador fraco correspondente h_j desconsiderando-se os outros classificadores. Este processo denomina-se treinamento do classificador fraco. O cálculo do erro é dado por $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.

c) Escolhe-se o classificador h_t com o menor erro ϵ_t.

d) Define-se a variável $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

e) Atualiza-se os valores dos pesos:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

Onde e_i =0 se o exemplo x_i foi classificado corretamente e e_i=1 , caso contrário.

- Os coeficientes do classificador forte são definidos pela fórmula:

$$a_t = \log \frac{1}{\beta_t}$$

- Finalmente tem-se a função do classificador forte H(x).

$$H(x) = 1 \text{ se } \sum_{t=1}^{T} a_t h_t \geq \frac{1}{2} \sum_{t=1}^{T} a_t$$

$$H(x) = 0, \text{ caso contrário.}$$

Após definida a função do classificador forte o algoritmo deve varrer a imagem com sub-retângulos do tamanho da menor região de interesse nas amostras positivas. Para minimizar o esforço computacional de, em cada sub-retângulo, ocorrer o teste do classificador forte foi desenvolvida uma técnica específica de detecção que utiliza o que foi chamado de cascata de Haar(Viola e Jones, 2001).. Ao invés de utilizar diretamente o classificador forte completo, utilizam-se vários classificadores fortes incompletos, cada qual utilizando apenas algumas características de Haar. A imagem seria primeiro submetida pelos classificadores mais genéricos, pois as taxas de falso negativo são normalmente muito próximas de zero mesmo para classificadores com poucas características de Haar.Caso a imagem seja sinalizada como negativa neste classificador, não será submetida ao classificador seguinte. Entretanto se for positiva deverá ser submetida a um classificador seguinte cuja taxa de falsos positivos é menor. O processo está apresentado na Figura 5.

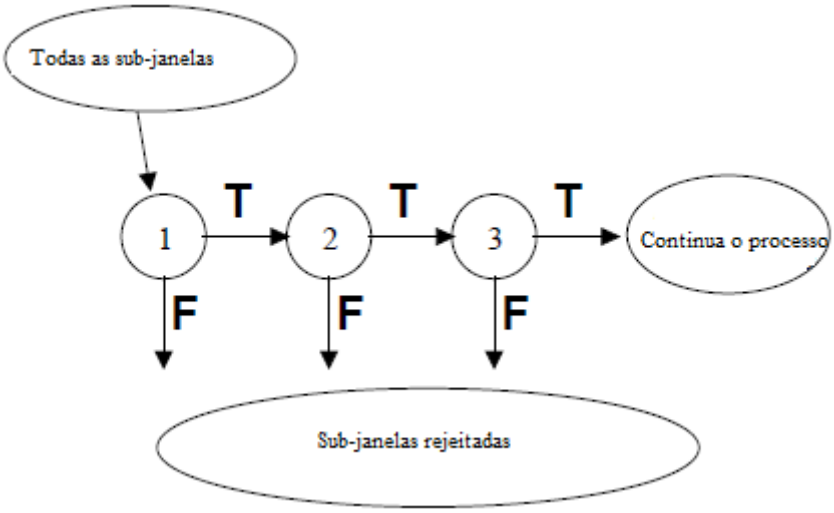

Figura 5: Cascata de Haar(adaptado de Viola e Jones, 2001)..

Cada classificador forte da cascata é chamado de estágio. Para projetar uma cascata será necessário:

- Determinar o número de estágios.
- Determinar o número de características de Haar detectada por cada estágio.
- Determinar o limiar de cada estágio, isto é, o termo $\frac{1}{2}\sum_{t=1}^{T} a_t$, da definição de classificador forte.

O algoritmo proposto, por Viola e Jones, para a determinação destes parâmetros requer que se definam alguns critérios preliminares. São eles:

- A taxa máxima aceitável f de falsos positivos em qualquer estágio.
- Taxa mínima d de acerto em qualquer estágio.
- A taxa global Ft de falsos positivos.

O algoritmo descrito em pseudocódigo fica da seguinte maneira:

P=conjunto de imagens positivas;

N=conjunto de imagens negativas;

F=1;(variável em ponto flutuante);

D=1;(variável em ponto flutuante);

h[] =0 (vetor de objetos que correspondem a classificadores fortes e são saídas do algoritmo *Adaboost*);

i=0 ;(inteiro que corresponde ao número do estágio)

Enquanto (F >=Ft) {

f=0;

d=0;

n=0

F(i-1)=F;

D(i-1)=D;

i=i+1;

Enquanto(F>=f*F(i-1) e D<=d*D(i-1)){

n=n+1;

(Insira n características no *Adaboost* e o treine com os conjuntos P e N)//no executável da biblioteca OpenCV utilizam-se até 200 características.

h(i)=Saída do Adaboost

F=Cálculo da taxa global de falsos positivos;

D=Cálculo da taxa global de acertos ;

Decremento do limiar do i-ésimo classificador até que D>= d*D(i-1) (isso também afeta F)

}

N=Ø;

SE(F>Ft)

N=conjunto de imagens correspondentes aos falsos positivos na cascata "i"

}

Retorno h[estágios]; onde cada elemento representa o classificador forte de um estágio da cascata.

Uma estimativa da taxa global de falsos positivos da cascata resultante desse algoritmo está descrita na Equação 5.

$$F = f^{e}. \quad (5)$$

Onde a variável “e” representa o número de estágios máximos e a variável “f” corresponde à taxa máxima de falsos positivos por estágio. O cálculo de uma estimativa da taxa de acertos D está descrito na Equação 6.

$$D = d^{e}. \quad (6)$$

Onde a variável “e” representa o número de estágios máximos e a variável “d” corresponde à taxa mínima de acertos por estágio.

Estes valores referem-se ao acerto e aos falsos positivos desde que se use o conjunto de dados do treinamento. Para avaliar o algoritmo com relação a outro conjunto de imagens somente efetuando-se os testes com este conjunto.

A taxa de falsos positivos tenderá a diminuir no final da cascata, o que é vantajoso para o classificador. Entretanto a taxa de acertos também cairá, o que é uma desvantagem. Para minimizar essa desvantagem deve-se adotar como taxa mínima de acerto valores muito próximos de 1.

A cascata de Haar implementada neste trabalho foi feita utilizando-se 50 amostras positivas e 100 amostras negativas, porque este foi o maior conjunto disponível gratuitamente e com placas brasileiras.

A taxa mínima de acertos por estágio escolhidas foi de 0,99. Inicialmente foi tentada 0,999 pois é o valor padrão do executável utilizado na biblioteca OpenCV, mas o tempo de execução do algoritmo ultrapassou três dias e não concluiu a cascata, tendo sido necessário a interrupção do processo e a diminuição do valor do parâmetro.

A taxa máxima de falso positivos por estágio escolhida foi de 0,5. Este valor foi escolhido por ser o padrão do executável utilizado na biblioteca OpenCV.

Nas Figuras 6 e 7 estão mostradas imagens correspondentes às amostras positivas (que contém placas de automóveis) e negativas (que não contém placas de automóveis), respectivamente.

Figura 6: Exemplo de amostra positiva utilizada no treinamento de Haar.

Figura 7: Exemplo de amostra negativa utilizada no treinamento de Haar.

2.1.2 CÓDIGO DE DETECÇÃO DE PLACAS.

Inicialmente foi feito um código único que deveria reconhecer a placa na imagem e efetuar o recorte, ou seja, a seleção da região de interesse.

O código possui uma função que indexa um arquivo com os nomes das imagens em uma pasta chamada “entrada”, que está no mesmo diretório do executável do programa. O algoritmo deve submeter cada amostra da pasta de entrada, que conste na lista de arquivos, a análise e caso identifique uma placa, deve exibir o nome do arquivo

de entrada seguido de um “1”, caso não identifique, o nome deve ser seguido de um “0”. Além disso, em caso de detecção, o algoritmo deve separar a região da placa e salvá-la como imagem na pasta de nome “saída”. Como a região definida com base na cascata de Haar nem sempre abrange a região real de interesse, existe uma etapa adicional antes do recorte, que consiste em aumentar a região detectada em três vezes.

O código foi feito utilizando funções específicas do Visual Studio, que separam uma região retangular definida de uma imagem. Como a região de saída da função cascata de Haar são retângulos (um dos sub-retângulos, que varrem a imagem), então basta utilizar as funções prontas. Exemplos de imagens de entrada e saída estão nas Figuras 8 e 9, respectivamente.

Figura 8:Exemplo de imagem de entrada.

.

Figura 9: Exemplo de imagem de saída.

2.1.3 RESULTADOS DA DETECÇÃO.

Para o teste de detecção foram utilizadas 50 imagens em tons de cinza. Esta quantidade de imagens de teste foi escolhida porque o *data set* de imagens mais completo e gratuito

encontrado possuía 100 imagens positivas e decidiu-se que metade seria usada para treinamento e metade para os testes.

A taxa de acertos do algoritmo foi de 56%, aproximadamente. Este valor foi considerado satisfatório devido ao grande número de concessões que foram feitas para que o arquivo da cascata de Haar fosse criado rapidamente. Estas concessões incluem tanto a alteração dos critérios de convergência quanto o baixo número de amostras positivas e negativas utilizadas no treinamento.

O mesmo procedimento foi repetido com imagens que não continham placas. De cinquenta imagens, somente uma produziu um falso positivo. Ou seja, a taxa de falsos positivos foi de 2%.

2.2 TESSERACT OCR

Antes de se iniciar a implementação do algoritmo de pré-processamento da imagem foi necessário fazer um estudo preliminar capaz de determinar a sensibilidade da ferramenta de OCR. As perguntas que este tópico se propõe responder são:

- Algum pré-processamento é necessário?
- Qual é a posição e a cor dos caracteres na imagem que otimizam a identificação? O pré-processamento deverá buscar converter a imagem para estas condições.
- Qual o grau de acertos esperado na melhor das condições?
- O sistema identifica mais facilmente os caracteres dentro de palavras ou isolados? Dependendo da resposta pode-se desenvolver um algoritmo para isolar os caracteres.

Mas antes de ser possível responder estas perguntas foi necessário adaptar o Tesseract para que seja capaz de detectar caracteres escritos nas fontes das placas brasileiras.

2.2.1 TREINAMENTO E ALGORITMO.

O Tesseract contém arquivos correspondentes a fontes e dicionários da linguagem que se pretende reconhecer. No arquivo denominado eng, por exemplo, existem informações sobre diversas possíveis fontes e um dicionário de palavras da língua inglesa. Se houvesse, neste arquivo, informações sobre a fonte dos caracteres das placas brasileiras, só seria necessário desativar as penalidades atribuídas a palavras fora do dicionário. Porém o Tesseract não contém informações acerca da fonte das placas, sendo necessária a criação de um arquivo que contenha esta informação. O processo de criação deste arquivo é chamado de treinamento do Tesseract. Embora seja possível criar o arquivo utilizando apenas o Tesseract, existem diversas ferramentas utilizadas para simplificar o processo. Neste trabalho utilizou-se o *software* Serak Tesseract Trainer. O treinamento foi feito de forma que o arquivo não impusesse penalidades para palavras fora do dicionário de qualquer idioma específico.

Segundo a resolução 231 do Conselho Nacional de Trânsito (Cotran), as "tipologias dos caracteres das placas devem seguir o modelo especificado na fonte Mandatory". A fonte é a mesma das placas da Grã-Bretanha. O arquivo da fonte foi copiado do site http://www.dafont.com/mandatory.font, acessado em dezembro de 2014. Exemplos de caracteres escritos nesta fonte estão mostrados na Figura 10.

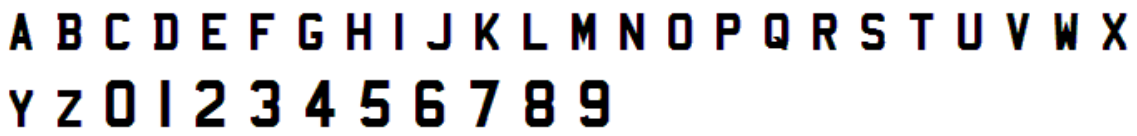

Figura 10: Caracteres das placas brasileiras

Após o treinamento pode-se agora tentar responder as perguntas que abrem este tópico.

2.2.2 RESULTADOS DOS TESTES DE DETECÇÃO DE CARACTERES

A primeira pergunta, das que abrem o tópico 2.2, foi respondida com um código contendo apenas uma função que recebe como entrada as imagens das placas detectadas

após o recorte tal como mostradas no tópico anterior e as submetendo ao Tesseract. Nenhum caractere foi reconhecido.

A segunda pergunta foi respondida consultando-se o *website* da desenvolvedora. A melhor situação possível é de uma imagem como um texto, ou seja, letras pretas em um fundo branco e sem ruído, como manchas ou descontinuidades nas letras Também é preferível que as letras sejam grandes e ocupem a maior área da imagem.

Para testar-se os índices de acerto no melhor caso possível, foram feitas imagens correspondentes às placas, entretanto na forma de texto. A Figura 11 mostra um exemplo de imagem utilizada.

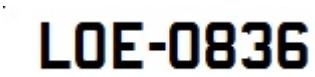

Figura 11: Exemplo de imagem ideal para a leitura de caracteres

Os resultados colhidos estão na Tabela 1.

Tabela 1: Teste Tesseract.

Placa	Resultado	Caracteres corretos
LNP-7221	LNP722	6
LAK-8614	LKB	2
LNI-2993	LN23	4
LOE-0836	LOEB36	5
KPC-7603	KPC763	6
KQH-7731	KQH773	7
AAY-5127	Y527	4
LOA-7993	LO7	3
LNR-0491	LNR4	4
KNP-9837	KNPB37	5
KMG-0982	KMGB2	4
LAV-5233	LV5233	6
LBO-1230	D23	2
KOD-7347	KOD7347	7
LCN-7724	LCN7724	7
ABY-9198	BYSSB	2
GZA-2477	G2447	3
LAR-7207	R727	4
CTB-1569	CTB56	5
Nada	Nada	0
LOF-3405	LOF345	6
LIK-3988	LK3BB	3

Nada	Nada	0
LNZ-4292	LN422	5
JLW-9464	JLW464	6

Como pode-se perceber o Tessaract utilizado sobre a imagem de uma palavra inteira só pôde detectar todos os caracteres de duas placas em um total de 23, mesmo em condições muito próximas das ideais.

O procedimento foi repetido, desta vez considerando-se caracteres isolados. Um exemplo de imagem usada está na Figura 12.

Figura 12: Imagem de caractere isolado

Fez-se também testes utilizando imagens cujas fontes não fossem as do treinamento e sim Times New Roman, para comparar a sensibilidade do Tesseract a mudanças na fonte. . Os resultados se encontram na Tabela 2

Tabela 2: Teste de identificação por caractere

Caractere	Resultado	Caracteres corretos(Fonte Mandatory)	Caracteres Corretos(Fonte Times New Roman)
A	A	1	0
B	B	1	0
C	C	1	1
D	D	1	0
E	E	1	1
F	F	1	1
G	G	1	1
H	H	1	0
I	I	1	0
J	J	1	0
K	K	1	1
L	L	1	1
M	M	1	0
N	N	1	0
O	O	1	0
P	P	1	0
Q	Q	1	0
R	R	1	0
S	S	1	0

T	T	1	1
U	U	1	1
V	V	1	1
W	W	1	0
X	X	1	0
Y	Y	1	0
Z	Z	1	0
0	O	0	1
1	I	0	0
2	2	1	1
3	3	1	1
4	4	1	1
5	S	0	0
7	7	1	0
8	B	0	0
9	9	1	1

Pode ser verificado que o índice de acertos é total para as letras e com alguns erros nos caracteres numéricos, quando a fonte da placa está correta. Este erro deve-se ao fato de que as fontes para números são iguais ou muito próximas.

Na fonte das placas de carro utilizadas, e mostradas na Figura 10, a letra “i” possui o mesmo caractere que o número 1, assim como a letra “o” e o número “0”. Outros erros foram o número “8” identificado pela letra “b” e o número “5” identificado como a letra “s”.

Como ocorreram apenas erros de identificação entre números e letras a forma avaliada como mais simples de se resolver isto foi levar em consideração a posição do símbolo na imagem da placa. Como os três primeiros caracteres sempre serão letras e os demais números, basta, quando for desenvolvido o programa, incluir um tratamento de exceções que substitua os caracteres pelos mais adequados.

Verificou-se também que uma mudança na fonte da placa acarreta uma queda de 30,76% em relação às letras detectadas e de 50% em relação aos números.

2.3 PRÉ-PROCESSAMENTO DA IMAGEM

Processar uma imagem significa realizar operações nela de modo que se torne mais adequada para uma aplicação específica. Dependendo do objetivo que se tem

diferentes operações serão necessárias durante o processamento da imagem. Como o objetivo do trabalho é a leitura de caracteres, um pré-processamento eficiente seria o que tornassem os caracteres da placa o mais simples possível de serem reconhecidos pelo *software* de reconhecimento óptico.

Como foi visto no tópico anterior, a imagem mais fácil de se reconhecer é aquela que mais parece um texto comum, sem manchas, apenas letras pretas sobre um fundo branco, como na Figura 11. E, se possível, deve-se isolar cada caractere antes de tentar a identificação.

Sendo assim um pré-processamento eficiente deve deixar a imagem o mais próxima possível desse formato. Diversos testes foram feitos com a finalidade de desenvolver um algoritmo apenas para esta dissertação. Outro algoritmo testado foi um que foi publicado em 2013 por engenheiros da Universidade de Goa, da Índia.

2.3.1 ALGORITMO DESENVOLVIDO

2.3.1.1 PRINCIPAIS OPERAÇÕES UTILIZADAS

Para aumentar o contraste da imagem e tornar os caracteres mais nítidos foi testada a técnica de equalização do histograma.

O histograma de uma imagem é a função discreta dentro do intervalo [0, L-1]. A função pode ser definida pela Equação 7. A Equação 7, assim como todas as mostradas daqui pra frente foram extraídas do livro *Digital image processing* (Woods, Gonzales, 2002)

$$h(r_k) = n_k . \tag{7}$$

Onde r_k é o k-ésimo tom de cinza e n_k a quantidade de *pixels* que apresentam o tom r_k. Se a função for normalizada dividindo-se n_k pela quantidade n total de *pixels* tem-se uma função de densidade de probabilidade da ocorrência dos respectivos tons de cinza.

Para analisar-se a relação entre contraste da imagem e seu histograma tomar-se-á os seguintes exemplos de imagens e seus histogramas, na Figura 13.

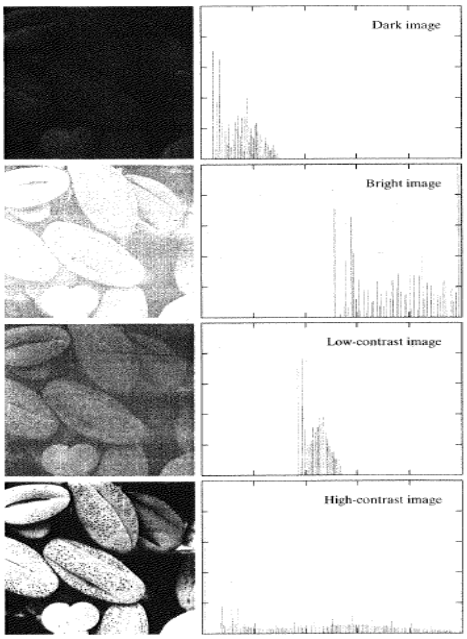

Figura 13: Exemplos de imagens e seus histogramas

No histograma da imagem mais escura, os valores de intensidade menores são os que possuem maior frequência. No histograma da imagem mais clara, os valores de intensidade maiores são os mais frequentes. Na imagem cujo histograma apresenta maior frequência de *pixels* nas intensidades intermediárias, vê-se uma imagem de tom de cinza não tão escura quanto a imagem de maior frequência nas intensidades baixas nem tão clara quanto a de maior frequência nas intensidades altas.

Finalmente na imagem em que o histograma é mais homogêneo é onde o contraste da imagem é maior e, consequentemente, mais detalhes podem ser percebidos. Isto deve-se ao fato de que uma imagem com mais tons conterá, na maioria das vezes, mais detalhes do objeto, do que uma imagem com menos tons, ou com tons de intensidade cujos valores são muito próximos entre si.

Uma maneira, portanto, de aumentar o contraste de uma imagem consiste em distribuir uniformemente as frequências das intensidades dos tons de cinza na imagem. Tal operação é denominada equalização do histograma.

Para demonstrar a operação, considera-se, inicialmente, uma função contínua, em que a variável r representa a intensidade dos tons de cinza da imagem a ser equalizada. Inicialmente, consideremos que r foi normalizada, e que r=0, representa

preto e r=1, representa branco. Sendo s uma função que representa os valores dos tons de cinza de saída, definida na Equação 8.

$$s = T(r). \tag{8}$$

Onde:

- $T(r)$ é bijetora no intervalo $0 \leq r \leq 1$;
- $T(r)$ é monotônica crescente no intervalo $0 \leq r \leq 1$;
- $0 \leq T(r) \leq 1$ no intervalo $0 \leq r \leq 1$;

A primeira condição garante que a função $T(r)$ seja invertível. É importante que a função seja montônica porque o objetivo é que haja contraste sem que a imagem tenha partes negativadas, ou seja, pode ser que uma área escura da imagem original fique mais clara em relação a si mesma mas ela não irá se tornar mais clara que uma área correspondente a uma região clara da imagem original. A terceira condição implica que a função dos tons de cinza de saída existe no mesmo intervalo da função de entrada.

Sendo $T(r)$ bijetora há uma inversa $T(s)^{-1}$, representada na Equação 9.

$$T(s)^{-1} = r. \tag{9}$$

Os tons de cinza na imagem, podem ser descritos como variáveis aleatórias no intervalo [0, 1]. Uma forma de descrever variáveis aleatórias é definindo-as por uma função de densidade de probabilidade. Sendo p_s a função densidade de probabilidade de s e p_r a função densidade de probabilidade de r. Não se pretende demonstrar neste trabalho o resultado que será utilizado a seguir, entretanto sabe-se que, se p_r e $T(r)$ são conhecidos e $T(s)^{-1}$ é monotônica crescente então a FDP de $s$está expressa na Equação 10.

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right| \tag{10}$$

Logo a função densidade de probabilidade da variável (FDP) s dependerá da FDP de r e da função de transformação escolhida.

Supondo que a função de transformação seja conforme a Equação 11.

$$s = T(r) = \int_0^1 p_r(w)dw. \tag{11}$$

Onde w é uma variável de integração qualquer. Esta é uma função de distribuição cumulativa, e como uma FDP é sempre positiva então s será monotônica crescente e bijetora. E, como é a integral de uma FDP, $s$estará no intervalo [0,1]. Sendo assim, a função de distribuição cumulativa cumpre todas as propriedades especificadas inicialmente. Esta função tem, além disso, uma propriedade importantíssima para estes propósitos. Sua FDP corresponde à distribuição uniforme de probabilidade, um resultado muito conhecido em análise de processos estocásticos. A demonstração começa na Equação 12.

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = \frac{d}{dr}\left[\int_0^1 p_r(w)dw\right] = p_r \tag{12}$$

.

Substituindo (10) em (12), e levando em consideração que as funções são sempre crescentes, tem-se a Equação 13.

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = p_r(r)\frac{1}{p_r(r)} = 1 \tag{13}$$

.

No intervalo $0 \leq s \leq 1$. Considerando que uma função de densidade de probabilidade uniforme possui a forma expressa na Equação 14:

$$f(x) = \frac{1}{a-b}. \tag{14}$$

No intervalo $a \leq s \leq b$. Como $a = 0$ e $b = 1$ conclui-se que (13) satisfaz (14). Um histograma cuja distribuição de tons de cinza seja uniforme é ideal para os

propósitos deste trabalho. Logo, utilizando a equação (11), pode-se produzir uma imagem com distribuição de tons uniforme a partir de uma imagem de entrada qualquer.

Será necessário aplicar o mesmo raciocínio para funções discretas. Neste caso a probabilidade de ocorrência dos tons de cinza não será dada pela FDP e sim pela Equação 15.

$$p_r(r_k) = \frac{n_k}{n}. \tag{15}$$

- Onde k=0,1,2,3...L-1;
- Onde n corresponde ao número total de *pixel;*
- Onde n_k corresponde à quantidade de *pixels* que possuem o valor de intensidade r_k .
- Onde L corresponde ao total de tons de cinza possíveis.

Sendo assim, para variáveis discretas, a Equação 11 pode ser substituída pela Equação 16.

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n} \tag{16}$$

.

- Onde k=0,1,2,3...L-1;

Esta transformação é chamada de equalização de histograma. Ao contrário do caso contínuo, no caso discreto não se pode garantir que todos os tons terão a mesma frequência na imagem. Entretanto irá ocorrer uma tendência à maior uniformidade na distribuição dos tons. O resultado desta operação nas imagens da Figura 13 estão representados na Figura 14.

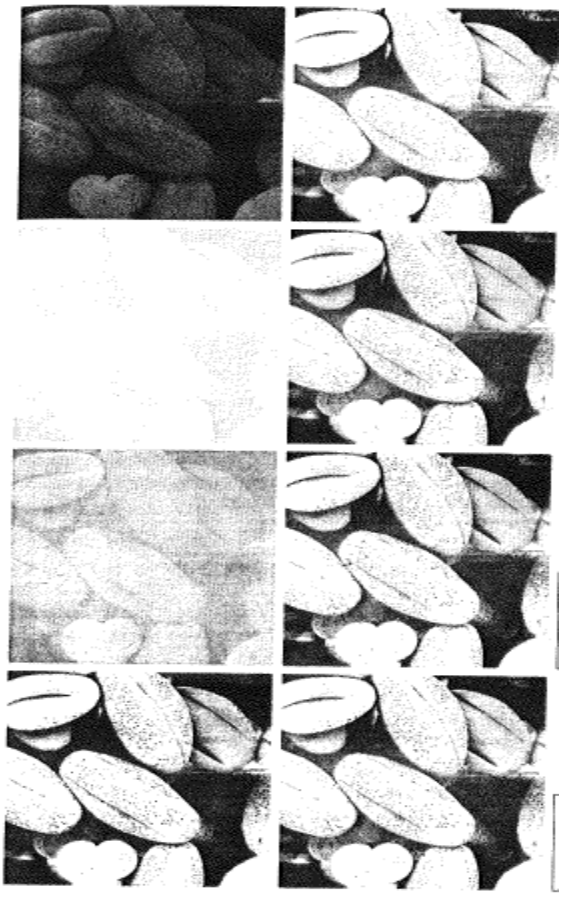

Figura 14: Imagens com suas respectivas equalizações de histograma. Extraído de GONZALES,Rafael C.; WOODS,Richard E.**Digital Image Processing.**Prentice Hall. Segunda Edição.2002

Independente do grau de contraste da imagem original, os resultados, após a equalização do histograma, ficam bastante similares entre si. Na imagem com melhor contraste, o efeito da equalização foi desprezível.

Após a equalização do histograma, ou seja, após o aumento da nitidez da imagem efetuou-se um *treshold.*

A operação de *treshold,* também chamado de "limiarização", consiste em uma operação que torna a imagem binária, ou seja, com apenas as cores preta e branca. Qualquer *pixel* cujo valor de intensidade se situe abaixo de um determinado limiar torna-se preto, caso este valor esteja acima ou igual, torna-se branco. No caso do trabalho em questão, o valor de limiar foi definido como 140, por ter permitido melhor visibilidade da região da placa nos conjuntos de imagens utilizados. Os resultados destas duas operações estão mostrados na Figura 15, na Figura 16 e na Figura 17 .

Figura 15: Imagem antes da equalização e do *treshold.*

Figura 16: Imagem após a equalização.

Figura 17: Imagem após a equalização e o *treshold*

Com apenas estas operações efetuadas, tentou-se identificar os caracteres das placas, mas nenhum resultado foi satisfatório. Mesmo considerando o acerto de caracteres, quase nenhum foi identificado.

Concluiu-se, portanto, que seriam necessárias mais operações na imagem. Seriam necessárias operações que retirassem as imperfeições da imagem, preenchesse os espaços vazios pequenos, dando uma consistência mais homogênea à imagem.

Duas operações foram consideradas capazes de produzir o efeito desejado, se efetuadas em conjunto: dilatação e erosão. A explicação das duas operações seguirá a seguir.

Seja B uma imagem, expressa na Figura 18, consistindo em um quadrado cujo *pixel* central seja preto, chamado de *kernel,* e os demais sejam brancos (uma imagem binária, portanto). Na figura cada quadrado menor corresponde a um *pixel*. Chama-se B de elemento estruturante da dilatação.

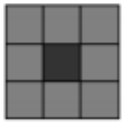

Figura 18: Elemento estruturante.

Supondo outra imagem binária A, expressa na Figura 19, que será o objeto a ser dilatado. B será transladado por A, iniciando-se na origem (ponto superior esquerdo), até o final da imagem (ponto inferior direito). Em termos de teoria dos conjuntos, A e B são conjuntos de Z^2 e a dilatação de A equivale ao conjunto correspondente a todos os deslocamentos z tal que B e A se sobreponham em pelo menos um elemento não-nulo. A expressão morfológica correspondente a dilatação de A e B, representada pela expressão $"A \oplus B"$ está definida na Equação 17.

$$A \oplus B = \left\langle z \in Z^2 \mid \left[(B^s)_z \cap A \right] \neq \varnothing \right\rangle \tag{17}$$

.

Onde B^s representa o conjunto correspondente à translação de B por todo o plano Z^2 .

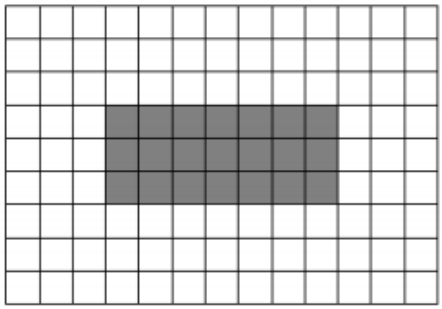

Figura 19: Imagem a ser dilatada.

A Figura 20 representa o resultado da dilatação da imagem da Figura 19 pelo elemento estruturante da Figura18.

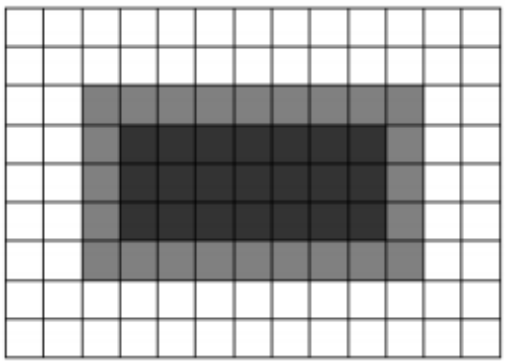

Figura 20: Imagem dilatada.

Deve-se ressaltar que na definição da operação utilizou-se o termo elementos não-nulos. Para fins de programação é necessário saber se, no *software* utilizado, estes elementos são os *pixels* ou os brancos. No caso da ferramenta que está sendo utilizada, os *pixels* brancos são os "não-nulos" da expressão da dilatação (embora nas figuras 18, 19 e 20 estejam sendo usados *pixels* escuros para facilitar a identificação).

Esta operação foi utilizada para homogeneizar as partes brancas da imagem. Se houvesse apenas uma mancha preta dentro de uma grande área branca, esta mancha seria preenchida.

A operação de erosão é similar entretanto verifica-se os termos nulos entre a área varrida pelo conjunto B em Z^2 e B. A operação pode ser expressa como "$A-B$" e está expressa na Equação 18.

$$A - B = \left\langle z \in Z^2 \mid \left[(B^s)_z \cap A\right] = \varnothing \right\rangle \tag{18}$$

.

Esta operação foi utilizada para homogeneizar as partes pretas da imagem.

A operação de dilatação possui este nome pois nela há uma aparente dilatação do fundo da imagem, um aumento das partes brancas. Na erosão há uma diminuição aparente do fundo da imagem, ou seja, uma erosão do fundo da imagem. Estas duas operações combinadas podem ser utilizadas para minimização do ruído em uma imagem. Deve ser ressaltado que, embora as operações sejam contrárias, estas não são inversas entre si. Caso utilize-se uma dilatação seguida de uma erosão não será obtida a imagem original. Esta afirmação é coerente com o fato de que, tanto na dilatação quanto na erosão há uma perda de informação da imagem original.

Embora tenha-se aqui utilizado como exemplo um quadrado como elemento estruturante, este pode ser qualquer figura geométrica. Assim como a localização do *kernel* (*pixel* não-nulo do elemento estruturante), considerado no exemplo como centro do elemento, pode, na verdade estar localizado em qualquer ponto deste. As Equações 16 e 17 não mudam com isso.

As operações de dilatação e erosão são cumulativas, associativas e distributivas. Além disso, estas duas operações são consideradas operações básicas no estudo de análise de imagens digitais. Utilizando a imagem da Figura 21, pode-se exemplificar os efeitos da dilatação e da erosão, que estão, respectivamente nas Figuras 22 e 22.

Figura 21: Imagem a ser testada. Fonte: http://docs.opencv.org/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html

Figura 22: Figura 22 após dilatação.

Figura 23:Figura 22 após erosão.

Em termos de programação, estas operações podem ser implementadas no Visual Studio, a partir de rotinas que recebem como entrada as especificações do elemento estruturante e a imagem que pretende-se operar. Na biblioteca OpenCV, já existem elementos prontos, como elipses, retângulos e cruzes.

Foram feitos testes utilizando como entradas as imagens como as da Figura 17 (imagens da placa após a equalização e o *treshold*), assim como imagens sem a equalização, apenas com *treshold,* exemplificadas na Figura 24.

Figura 24: Imagem da Figura 11 após treshold

Estes testes foram efetuados porque em diversas imagens observou-se que equalização muitas vezes adicionava ruído à imagem durante o processo de realce. Como em muitas placas os dígitos da placa são perfeitamente nítidos é preferível, nesses casos, não efetuar a equalização do histograma.

Foram efetuadas uma dilatação e uma erosão nas imagens de entrada. O elemento estruturante utilizado foi um retângulo 3x3 com seu *kernel* no centro. Nas Figuras 25 e 26 tem-se o resultado das operações efetuadas sobre a Figura 24 (sem equalização do histograma) e nas Figuras 27 e 28 tem-se o resultado das operações sobre a Figura 17 (com equalização do histograma).

Esta operação chama-se “fechamento” ou “fecho” de uma imagem. Esta operação tende a suavizar a silhueta dos objetos, entretanto, há uma tendência a fundir descontinuidades muito estreitas, preenchendo, assim, fendas muito afuniladas e buracos muito pequenos. Em morfologia matemática esta operação pode ser expressa pela Equação 19.

$$A \bullet B = (A \oplus B) - B \ . \tag{19}$$

Onde A é função que representa a imagem e B é a função que representa o elemento estruturante utilizado na dilatação e na erosão.

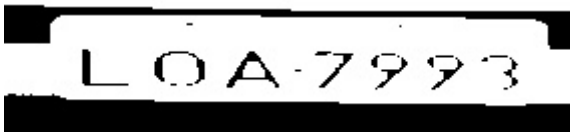

Figura 25: Figura 24 após dilatação.

Figura 26: Figura 24 após dilatação e erosão.

Figura 27: Figura 17 após dilatação

Figura 28: Figura 17 após dilatação e erosão

Também foi testada a operação contrária ao fechamento, denominada abertura.

Chama-se abertura a operação morfológica constituída por uma erosão seguida de uma dilatação. Esta operação tende a suavizar a silhueta de um objeto, quebra istmos muito estreitos e elimina pequenas protuberâncias. Em termos de morfologia matemática a abertura pode ser expressa pela Equação 20.

$$A \circ B = (A - B) \oplus B \ . \tag{20}$$

Onde A é função que representa a imagem B é a função que representa o elemento estruturante utilizado na erosão e na dilatação .

2.3.1.2 TESTES E TABELAS.

Em seguida testou-se qual dos dois métodos de fechamento, com ou sem equalização, produz resultados mais fáceis de serem identificados pelo OCR utilizado. Considerando-se as 25 amostras de placas detectadas no tópico 2.1.2, será verificado quais são identificadas. Os resultados estão nas Tabelas 3 e 4.

Tabela 3: Teste sem equalização.

Placa	Resultado	Caracteres corretos
LNP-7221	Nada	0
LAK-8614	nada	0
LNI-2993	Nada	0
LOE-0836	LOF84	3
KPC-7603	Nada	0
KQH-7731	Nada	0
AAY-5127	Nada	0

LOA-7993	L7PP3	3
LNR-0491	Nada	0
KNP-9837	U	0
KMG-0982	Nada	0
LAV-5233	Nada	0
LBO-1230	J	0
KOD-7347	Nada	0
LCN-7724	Nada	0
ABY-9198	B	1
GZA-2477	nada	0
LAR-7207	nada	0
CTB-1569	nada	0
Nada	nada	0
LOF-3405	nada	0
LIK-3988	nada	0
Nada	nada	0
LNZ-4292	H	0
JLW-9464	nada	0

Tabela 4: Teste com equalização

Placa	Resultado	Caracteres corretos
LNP-7221	Nada	0
LAK-8614	Nada	0
LNI-2993	Nada	0
LOE-0836	Nada	0
KPC-7603	Nada	0
KQH-7731	Nada	0
AAY-5127	Nada	0
LOA-7993	Nada	0
LNR-0491	Nada	0
KNP-9837	Nada	0
KMG-0982	Nada	0
LAV-5233	Nada	0
LBO-1230	Nada	0
KOD-7347	Nada	0
LCN-7724	RPC	1
ABY-9198	Nada	0
GZA-2477	Nada	0
LAR-7207	Nada	0
CTB-1569	Nada	0
Nada	Nada	0
LOF-3405	Nada	0
LIK-3988	S	0
Não há placa	Nada	0

LNZ-4292	Nada	0
JLW-9464	Nada	0

Foi feito também o teste de identificação de caracteres sem equalização, erosão e dilatação, apenas com o *treshold,* o resultado se encontra na Tabela 5.

Tabela 5: Teste de identificação de caracteres da imagem submetida apenas ao treshold.

Placa	Resultado	Caracteres corretos
LNP-7221	Nada	0
LAK-8614	LBS	1
LNI-2993	Nada	0
LOE-0836	LEOSS	1
KPC-7603	Nada	0
KQH-7731	Nada	0
AAY-5127	Nada	0
LOA-7993	Nada	0
LNR-0491	LCN724	1
KNP-9837	Nada	0
KMG-0982	Nada	0
LAV-5233	Nada	0
LBO-1230	Nada	0
KOD-7347	Nada	0
LCN-7724	RPC	1
ABY-9198	Nada	0
GZA-2477	Nada	0
LAR-7207	Nada	0
CTB-1569	Nada	0
Não há placa	Nada	0
LOF-3405	Nada	0
LIK-3988	Nada	0
Nada	2366	0
LNZ-4292	nada	0
JLW-9464	4	0

Os teste de abertura foi feito com a imagem sem equalização, pois os resultados da equalização sugerem que este processo dificulta a identificação, quando utilizado em uma imagem que já possui um nível de contraste excelente, como as utilizadas. Os resultados estão na Tabela 6.

Tabela 6: Teste da abertura

Placa	Resultado	Caracteres corretos
LNP-7221	Nada	0
LAK-8614	Nada	0
LNI-2993	Nada	0
LOE-0836	LOE	3
KPC-7603	Nada	0
KQH-7731	1	1
AAY-5127	Nada	0
LOA-7993	Nada	0
LNR-0491	TJRCD	1
KNP-9837	Nada	0
KMG-0982	Nada	0
LAV-5233	Nada	0
LBO-1230	Nada	0
KOD-7347	724	2
LCN-7724	RPC	1
ABY-9198	BYSTSB	2
GZA-2477	Nada	0
LAR-7207	Nada	0
CTB-1569	Nada	0
Não há placa	Nada	0
LOF-3405	OF	2
LIK-3988	Nada	0
Nada	Nada	0
LNZ-4292	S	0
JLW-9464	4	1

A aplicação da abertura na imagem produziu um índice de 13 caracteres corretamente identificados em um total de 161. O fechamento sem equalização identificou 7 caracteres corretamente em um total de 161. O fechamento com equalização, por sua vez, identificou corretamente somente 1 caractere. O *treshold* puro identificou corretamente 4 caracteres.

Foi avaliado que a técnica sem equalização é ligeiramente superior à técnica com equalização e a técnica que envolve a abertura da imagem é superior às demais. Uma conclusão que deve ter uma explicação mais detalhada é a inferioridade do fechamento com equalização em relação às demais. Apesar da equalização de histograma ser uma técnica muito usada na literatura para aprimoramento de imagens,

seu resultado neste trabalho foi o pior de todos, tendo identificado menos caracteres do que o simples *treshold.*

A razão para isto, como já foi dito, é que as imagens do *data set* escolhido são legíveis e perfeitamente discerníveis em sua maioria. Isso implica que a equalização do histograma seria mais uma fonte de ruído do que um processamento efetivo para a imagem. Entretanto a opção por testar a equalização no pré-processamento deve-se à decisão de criar-se um método geral e não um que funcionasse apenas para um conjunto específico de imagens. No caso de uma situação geral o algoritmo mediria o contraste da imagem e, caso estivesse abaixo ou acima de um determinado limiar. Entretanto, este algoritmo não chegou a ser implementado as razões disto serão discutidas no tópico 2.3.1.3.

Pela mesma razão a técnica apenas com o *treshold* mostrou-se apenas um pouco inferior às técnicas de fechamento e abertura. O efeito da dilatação e da erosão na diminuição do ruído é limitado devido ao fato de que as imagens escolhidas já estão quase livres de ruído

. Foi feita a suposição de que a razão dos resultados seja o fato de que os caracteres não estão suficientemente isolados na imagem. As imagens, então foram editadas de modo a serem mostrados apenas os caracteres. As Figuras 29 e 30 mostram a edição que foi feita nas imagens. A Figura 31 mostra a imagem submetida ao *treshold.*

Figura 29: Imagem antes da edição.

Figura 30: Imagem após a edição.

Figura 31: Imagem após o *treshold*

Os resultados da identificação com as imagens editadas se encontram na Tabela 7.

Tabela 7: Resultado da identificação com as imagens editadas.

Placa	Resultado	Caracteres corretos
LNP-7221	TCT	0
LAK-8614	LKS4	3
LNI-2993	LN273	4
LOE-0836	L0E0S	4
KPC-7603	Nada	0
KQH-7731	SC77	2
AAY-5127	Y27	3
LOA-7993	L7PP3	2
LNR-0491	ROLT	1
KNP-9837	KPP637	4
KMG-0982	Q	0
LAV-5233	V	0
LBO-1230	LBOT23	5
KOD-7347	37347	4
LCN-7724	LCN724	6
ABY-9198	YST	1
GZA-2477	G2477	3
LAR-7207	L727	3
CTB-1569	CT	2
Não há placa	Nada	0
LOF-3405	LOFO	3
LIK-3988	LKPB	2
Nada	Nada	0
LNZ-4292	LP	1
JLW-9464	2LW464	5

Ou seja 58 caracteres encontrados em um total de 161 (36%). Considerando que, com as imagens editadas, existe menos cenário para se misturar com as letras em caso de dilatação ou erosão verificou-se, com as imagens editadas, os resultados da aplicação da erosão e dilatação.

Os resultados da aplicação apenas da erosão nas imagens editadas estão na Tabela 8.

Tabela 8: Teste de identificação dos caracteres das imagens editadas após erosão.

Placa	Resultado	Caracteres corretos
LNP-7221	Nada	0
LAK-8614	LKB4	3

LNI-2993	LNP	2
LOE-0836	L0E0BB	4
KPC-7603	C	1
KQH-7731	C277	2
AAY-5127	BT27	2
LOA-7993	L7PP3	3
LNR-0491	TR4	2
KNP-9837	KNPP837	7
KMG-0982	Nada	0
LAV-5233	VBB	1
LBO-1230	LQTQ	1
KOD-7347	3	1
LCN-7724	LCN7724	7
ABY-9198	BB8	1
GZA-2477	G2477	3
LAR-7207	LR707	5
CTB-1569	CTB	3
Não há placa	Nada	0
LOF-3405	LOF34	5
LIK-3988	LKBPBB	2
Nada	Nada	0
LNZ-4292	LN4P	3
JLW-9464	JL	2

O total de caracteres encontrados foi 60 em um total de 161 (37%). Testou-se, também, a detecção de caracteres com as imagens editadas submetidas apenas a dilatação. Os resultados encontram-se na Tabela 9.

Tabela 9: Teste de identificação dos caracteres das imagens editadas após dilatação.

Placa	Resultado	Caracteres corretos
LNP-7221	E	0
LAK-8614	LQJ84	3
LNI-2993	LT	1
LOE-0836	1CF	0
KPC-7603	Nada	0
KQH-7731	Nada	0
AAY-5127	Y727	3
LOA-7993	F	0
LNR-0491	IROT	1
KNP-9837	Nada	0
KMG-0982	BTT	0
LAV-5233	233	2
LBO-1230	JSJC	0
KOD-7347	KCO737	3

LCN-7724	LCTJ77J	4
ABY-9198	C	0
GZA-2477	G247	4
LAR-7207	J57	1
CTB-1569	Nada	0
Não há placa	Nada	0
LOF-3405	IIFYT	0
LIK-3988	LTPH	1
Não há placa	Nada	0
LNZ-4292	FJJ	0
JLW-9464	LH	2

O total de caracteres identificados corretamente, efetuando-se apenas a dilatação, é de 25 caracteres em um total de 161.

Na Tabela 10 será verificado os resultados da erosão seguida de dilatação, sobre as imagens editadas.

Tabela 10: Teste de identificação dos caracteres das imagens editadas após *treshold* e erosão seguida de dilatação.

Placa	Resultado	Caracteres corretos
LNP-7221	X	0
LAK-8614	LKBS4	3
LNI-2993	LP273	3
LOE-0836	LOEO	4
KPC-7603	Nada	0
KQH-7731	277	2
AAY-5127	Y27	3
LOA-7993	LO7PP3	4
LNR-0491	T4	1
KNP-9837	KNP67	4
KMG-0982	TF	0
LAV-5233	V	1
LBO-1230	23	2
KOD-7347	KD7347	6
LCN-7724	LCN724	6
ABY-9198	8YT8	2
GZA-2477	G2477	5
LAR-7207	LR727	5
CTB-1569	CT5	3
Não há placa	Nada	0
LOF-3405	LOF345	6
LIK-3988	LK3P88	5
Não há placa	Nada	0
LNZ-4292	PGP	0

JLW-9464	LW454	4

O processo de erosão seguido de dilatação produziu um total de 69 caracteres identificados em um total de 161. O índice de detecção será testado, também com as operações contrárias, ou seja, dilatação seguido de erosão. Os resultados estão na Tabela 11.

Tabela 11:Teste de identificação dos caracteres das imagens editadas após treshold e dilatação seguida de erosão.

Placa	Resultado	Caracteres corretos
LNP-7221	B	0
LAK-8614	LKBE	2
LNI-2993	LN273	3
LOE-0836	LOFJJK	2
KPC-7603	Nada	0
KQH-7731	Nada	0
AAY-5127	Y27	3
LOA-7993	L7P3	3
LNR-0491	R0	2
KNP-9837	Nada	0
KMG-0982	T	0
LAV-5233	V233	4
LBO-1230	FJ	0
KOD-7347	KOD7347	7
LCN-7724	LCPJ7	3
ABY-9198	EVOP	0
GZA-2477	G247	4
LAR-7207	27	2
CTB-1569	Nada	0
Não há placa	Nada	0
LOF-3405	LOF1	3
LIK-3988	PBF	0
Não há placa	Nada	0
LNZ-4292	H	0
JLW-9464	LW4	3

O número de caracteres identificados corretamente após o processo de dilatação seguida de erosão foi de 41 caracteres. Outro procedimento foi testado: *treshold,* erosão, dilatação, erosão e dilatação. Esta operação denomina-se “abertura e fechamento” e é utilizada para a diminuição do ruído. Os resultados estão na Tabela 12.

Tabela 12:Teste de identificação dos caracteres após "abertura e fechamento"

Placa	Resultado	Caracteres corretos
LNP-7221	X	0
LAK-8614	LKBS4	3
LNI-2993	LP273	3
LOE-0836	LOE0	3
KPC-7603	Nada	0
KQH-7731	277	2
AAY-5127	Y27	3
LOA-7993	LO7PP3	4
LNR-0491	T4	0
KNP-9837	KNP67	3
KMG-0982	TF	0
LAV-5233	23	2
LBO-1230	FJ	0
KOD-7347	KD7347	6
LCN-7724	LCN724	6
ABY-9198	8YT8	2
GZA-2477	G2477	4
LAR-7207	LR727	5
CTB-1569	CT5	2
Não há placa	Nada	0
LOF-3405	LOF345	6
LIK-3988	LK3P88	5
Não há placa	Nada	0
LNZ-4292	PGP	0
JLW-9464	LW454	4

O total de caracteres identificados corretamente após a operação de “abertura e fechamento” foi de 63 caracteres em um total de 161 (39%).

O melhor resultado obtido, utilizando-se placas editadas, foi o referente à operação de erosão seguida de dilatação. Nas Figuras 32 e 33, tem-se exemplos de imagens de entrada e saída após estas operações.

Figura 32: Imagem de entrada

Figura 33: Imagem de saída

Foi testado, também, o resultado obtido isolando-se cada caractere. As imagens dos caracteres receberam a erosão e a dilatação e foram submetidas à ferramenta de

reconhecimento de caracteres. Em seguida o resultado foi agrupado e apresentado na Tabela 13.

Tabela 13: Teste de identificação de caracteres ao separar cada caractere da placa

Placa	Resultado	Caracteres corretos
LNP-7221	Nada	0
LAK-8614	LK4	3
LNI-2993	LN73	3
LOE-0836	LBE3	3
KPC-7603	Nada	0
KQH-7731	377	2
AAY-5127	Y627	3
LOA-7993	L73	3
LNR-0491	T4T	0
KNP-9837	KN37	4
KMG-0982	253F	0
LAV-5233	LV33	4
LBO-1230	LI33	2
KOD-7347	K7347	5
LCN-7724	LCN7724	7
ABY-9198	YT	1
GZA-2477	C2477	4
LAR-7207	L77	3
CTB-1569	C	1
Não há placa	Nada	0
LOF-3405	LF346	4
LIK-3988	LK	2
Não há placa	Nada	0
LNZ-4292	N422	4
JLW-9464	LW44	4

Com um total de 62 caracteres identificados corretamente, o método de avaliar cada caractere separadamente mostrou-se quase igual ao método com processamento de imagens equivalente porém para a imagem da placa inteira.

Este resultado contraria o que foi mostrado no tópico anterior em que avaliou-se que a ferramenta de reconhecimento de caracteres é mais eficaz ao analisar um caractere por vez. Uma possível explicação é que foi adicionado erro durante o processo de edição. Outra explicação é que, como o caractere é deformado pelo processamento, o Tesseract não produz resultados satisfatórios. Mais detalhes serão abordadas no tópico de conclusão.

2.3.1.3 ALGORITMO DESENVOLVIDO QUE NÃO FOI IMPLEMENTADO.

Neste tópico está uma parte do algoritmo que foi concebida e estudada, mas não implementada. O objetivo do algoritmo é a medição do contraste da imagem. Caso o contraste ultrapasse determinado limiar, não será feita a equalização do histograma, caso contrário, será feita a equalização.

A medição do contraste de uma imagem utilizando algum algoritmo simples o suficiente para ser implementado neste trabalho resultou em algumas conclusões. Como contraste é uma medição do grau de uniformidade do histograma de uma imagem, qualquer algoritmo utilizado para medi-lo deveria receber o vetor contendo o histograma.

Inicialmente considerou-se a medição da dispersão estatística do histograma como forma de avaliar o grau uniformidade deste. Existem várias formas de medir-se a dispersão de uma distribuição, mas a escolhida foi a medição da variância. Se esta hipótese estiver correta, uma pequena variância significará que os elementos do histograma estão mais próximos entre si, caso a variância seja maior, significará que estão mais dispersos.

Em um conjunto X de amostras n , $[x_1, x_2, \dots, x_n]$, o valor médio μ será dado pela Equação 21.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \ . \tag{21}$$

A variância de X é dada pela Equação 22.

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2 \ . \tag{22}$$

Outro método utilizado para a medição do contraste foi resultado do estudo de imagens digitais no domínio da frequência e requer uma explicação mais detalhada.

Uma imagem, representada como uma função discreta de x e y(coordenadas da área da imagem), pode, como qualquer outra função, ser submetida à Transformada de Fourier. Dada uma função discreta $f(x,y)$ de tamanho $M \times N$ e sua transformada de Fourier $F(u,v)$. A relação entre ambas é dada pela Equação 23.

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)} \tag{23}$$

Algumas considerações podem ser feitas a respeito do domínio da frequência, dentre elas a mais importante para os propósitos desse estudo é a relação entre os componentes de alta frequência e o contraste da imagem. A relação é positiva, quanto mais componentes de alta frequência maior a quantidade de detalhes. Foge aos objetivos deste trabalho a demonstração rigorosa deste resultado, bastando aqui deixar claro que se frequência está ligado a taxa de mudança. Pode-se notar, também que para $u = v = 0$ então $F(u,v)$ corresponde ao valor médio de $f(x,y)$.

Para ilustrar estas conclusões, temos a imagem da Figura 34, que mostra um circuito integrado ampliado por um microscópio eletrônico de varredura.

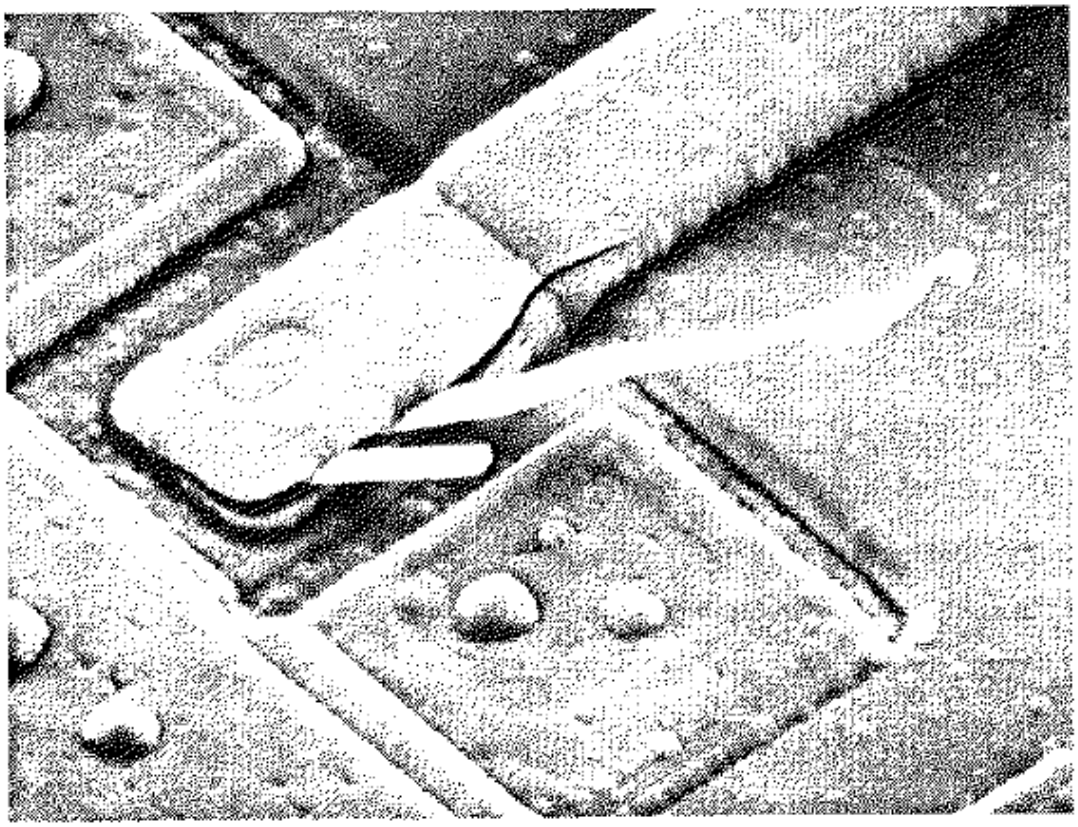

Figura 34: Imagem de circuito integrado tirada por um microscópio eletrônico.

A Figura 35 representa esta imagem após passar por uma transformada de Fourier ter seus componentes de baixa frequência retirados.

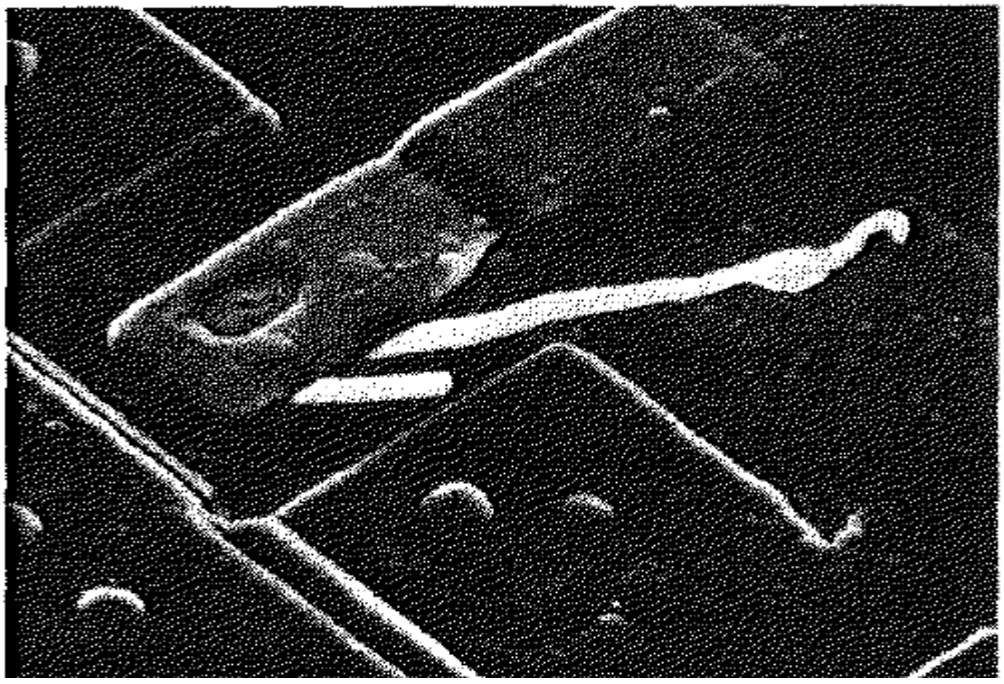

Figura 35: figura 34 após filtragem em passa-alta

Na filtragem passa-alta os contornos da imagem são acentuados e que o que não é contorno torna-se escuro. Na Figura 36, tem-se a imagem da Figura 34 após ser uma filtragem passa-baixa.

Figura 36:Figura 34 após filtragem passa-baixa

Na filtragem passa-baixa a imagem perde os detalhes de contorno.

Nem o algoritmo que utiliza a variância nem o que utiliza a transformada de Fourier foram implementados.

Verificou-se que seriam inócuos para a resolução do problema que este trabalho se destina caso os dois algoritmos não fossem devidamente testados e comparados em um trabalho que ocuparia ainda mais tempo. Como os resultados dos testes tanto com equalização como sem equalização se mostravam praticamente ilegíveis pela ferramenta de OCR, não haveriam critérios para comparar ambos os algoritmos. A comparação teria que ser feita de maneira qualitativa por um humano, o que em um trabalho como este não teria muito significado e despenderia muito tempo.

A determinação do valor limiar para a equalização também seria problemática. A única forma pensada foi a criação de uma rede neural, para determinar o valor ótimo de contraste. Isso envolveria um novo conjunto de imagens em várias condições de nitidez. Isso afastaria o trabalho do seu propósito original.

2.3.2 ALGORITMO MIRASHI.

O algoritmo que será explicado neste tópico foi publicado no *International Journal of Science and Research,* por quatro pesquisadores da Universidade de Goa, na Índia. Os nomes dos autores são Vinay Mirashi, Jairam Parab, Ramesh Kudaskar e Samarth Borkar.

Para comparar os resultados obtido aqui com os obtidos pelos pesquisadores, utilizaremos inicialmente a mesma imagem utilizada como exemplo no artigo dos autores originais. A imagem está na Figura 37.

Figura 37: Imagem de teste do algoritmo

Inicialmente, diz o artigo, deve-se converter a imagem para tons de cinza "suprimindo a informação de matiz (*hue*) e saturação (*saturation*) mas mantendo a luminosidade (*lightness*) constante". Este trecho já indica que a imagem não está sendo representada pelo espaço de cores RGB e sim pelo espaço de cores HLS.

Até então quando se disse "intensidade" de um *pixel* em tom de cinza, tinha-se em mente o espaço de cores RGB, baseado nos tipos de células capazes de captar cores no olho humano, chamadas "cones". Assim como existem três tipos de cones no olho humano normal, capazes de captar luz vermelha, verde e azul, no sistema RGB cada *pixel* é associado a três números (R, G e B) correspondentes ao valor da predominância de cada cor. Caso os valores de R, G e B sejam iguais, o valor do *pixel* será cinza. Converter uma imagem colorida para uma imagem em tons de cinza, utilizando o espaço de cores RGB, significa substituir os valores dos parâmetros RGB, de cada *pixel* da imagem, pela média destes valores.

Para se perceber as particularidades do espaço de cores HLS convém expressá-lo por um sistema de coordenadas cilíndricos, como mostrado na Figura 38.

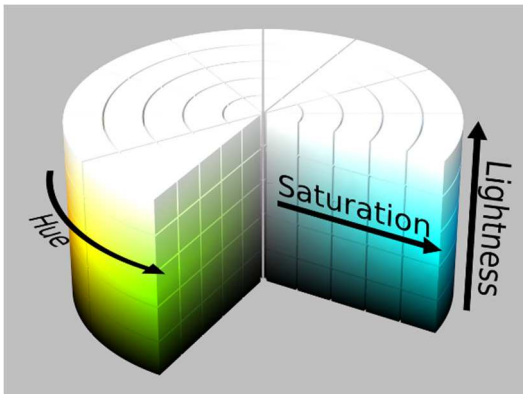

Figura 38: Espaço de cores HLS. Referências:"HSL color solid cylinder alpha lowgamma" by HSL_color_solid_cylinder.png: SharkDderivative work: SharkD Talk - HSL_color_solid_cylinder.png. Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wi

A matiz (*hue),* corresponde ao comprimento de onda dominante e forma a base do cilindro. É mais simples perceber isto, se isolar-se a base do cilindro, como pode ser visto na Figura 39.

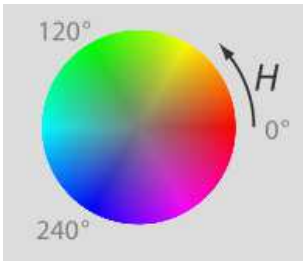

Figura 39: "Matiz" H, representada como base do cilindro de cores. Referências:"Hsl-hsv models" by Jacob Rus - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Hsl-hsv_models.svg#/media/File:Hsl-hsv_models.svg

A saturação (*saturation*) é uma medida do grau de pureza da cor, ou seja, do quão misturada a matiz está com a cor branca. Quanto mais saturada a cor menos “desbotada” ela parece, quanto menos saturada, mais desbotada. A palavra “desbotada” foi escolhida propositadamente por ser de compreensão intuitiva, já que a facilidade de ser compreendido por humanos é uma das razões pela qual o sistema HSL é utilizado.

Na Figura 39, a saturação varia do centro do círculo para a sua circunferência, crescendo no sentido do raio do círculo.

Por fim, a luminosidade (*lightness*), que corresponde à intensidade do brilho e cresce no sentido da altura do cilindro de cores. Quanto maior a luminosidade mais clara a imagem, quanto menor a luminosidade mais escura. Pode-se perceber, então, porque suprimir as informações de matiz e saturação corresponde a passar a imagem para tons de cinza.

A imagem em tons de cinza mostrada no artigo está na Figura 40.

Figura 40: Imagem em tons de cinza mostrada no artigo.

A imagem obtida neste trabalho está na Figura 41. Considerou-se que a diferença entre ambas pode ser atribuída ao fato de que os autores utilizaram uma imagem de entrada com resolução diferente(640x480) e maior à da que foi usada neste trabalho(187x142), que foi extraída do artigo com baixa resolução.

Figura 41: Imagem em tons de cinza obtida.

Em seguida deve-se equalizar o histograma da imagem para aumentar o contraste. O resultado apresentado no artigo está na Figura 42.

Figura 42: Imagem do artigo após a equalização do histograma.

Este procedimento foi reproduzido neste trabalho e obteve-se a imagem da Figura 43.

Figura 43: Imagem obtida após a equalização

Para demonstrar que o histograma equalizado utilizou-se um *software* de edição de imagens, chamado Gimp, para verificar o histograma das duas figuras. O histograma da Figura 41 está na Figura 44 e o histograma da Figura 43 está na Figura 45.

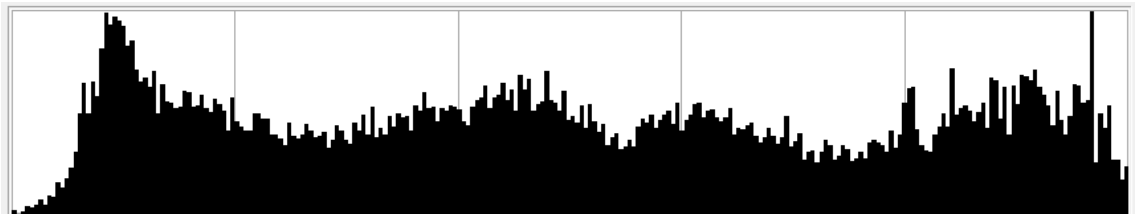

Figura 44: Histograma da imagem obtida em tons de cinza.

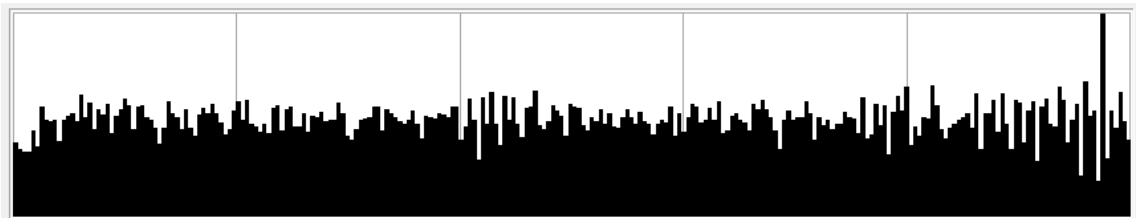

Figura 45: Histograma após a equalização

Mesmo assim, existem diferenças em relação a imagem obtida e a imagem mostrada no artigo. Isto pode ser causado pela fonte de erros já mencionada, a diferença de resolução entre as imagens de entrada, e também por um algoritmo diferente de equalização.

A seguir tornou-se a imagem binária, ou seja, aplicou-se um *treshold* na imagem. O resultado do artigo está na Figura 46.

Figura 46: Imagem do artigo após o *treshold*.

O artigo não informava o valor de limiar escolhido para a binarização da imagem. O que valor escolhido foi 140, por garantir uma boa visibilidade na região da placa. O resultado obtido está na Figura 47.

Figura 47: Imagem obtida após o *treshold*.

Em seguida efetuou-se uma operação de abertura na imagem. Entretanto, pela imagem mostrada pelos autores do artigo, e mostrada na Figura 48, está claro que antes da abertura os autores devem ter, necessariamente, invertido a imagem, tornando as regiões pretas em brancas e vice-versa. E a imagem mostrada é resultante da inversão. As razões ficarão mais claras a seguir.

Figura 48: Imagem do artigo após a inversão.

A imagem obtida após a inversão está na Figura 49.

Figura 49: Imagem obtida após inversão.

Após a inversão foi efetuada a abertura, cuja imagem obtida está na Figura 50. Como já foi dito, no artigo a figura identificada como a imagem após a abertura é a que está representada na Figura 48. Será demonstrado porque isso não pode ser verdade e que a Figura 49, na verdade, representa a inversão.

Figura 50: Imagem obtida após a abertura.

A primeira vista parece haver um erro, pois toda a informação do interior da placa foi perdida, entretanto o objetivo destas operações, como ficará claro, é isolar apenas a região da placa. Segundo o artigo a operação seguinte foi o *filling,* também chamada de preenchimento, sobre o resultado da abertura.

Esta operação, na forma como foi executada pelos autores, consiste em tornar brancos os *pixels* pretos que não estão conectados a algum *pixel* preto da borda da imagem através de outros *pixels* pretos. O resultado desta operação apresentado no artigo está na Figura 51. Na Figura 52 está o resultado obtido neste trabalho.

Figura 51: Imagem do artigo invertida após o *filling*.

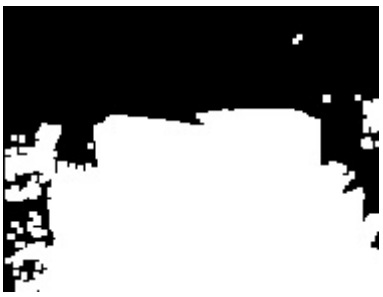

Figura 52: Imagem obtida após o *filling*.

Em seguida efetua-se a operação de fechamento da imagem.O artigo não mostra os resultados da operação de fechamento.

Após o fechamento os autores realizaram uma multiplicação entre imagem da Figura 46 (logo após o *treshold*) e o resultado do fechamento, para obter as áreas onde

pode estar a placa. A imagem mostrada no artigo está na Figura 53 e obtida neste trabalho está na Figura 54.

Figura 53: Regiões onde a placa pode estar, na imagem do artigo.

Figura 54: Regiões onde a placa pode estar, obtidas neste trabalho

Em seguida as regiões brancas onde é menos provável de encontrarem-se as placas foram eliminadas, o critério utilizado foi o tamanho da região. A Figura 55 mostra a região da placa, mostrada no artigo.

Figura 55: Região de interesse da placa.

Para efetuar este passo, decidiu-se utilizar a técnica conhecida como “rotulagem dos componentes conectados”.

Esta análise consiste em saber quantos componentes existem na imagem. Para determinar um componente, deve-se executar um algoritmo que será explicado a seguir. Como está-se lidando com imagens binárias sabe-se que a diferença de intensidade entre dois *pixels* quaisquer será sempre ou zero ou 255. Existem formas gerais de se fazer este algoritmo, levando em consideração até mesmo cores, de fato, o método pertence a teoria dos grafos e, a rigor, pode ser usado a diversas estruturas que nem mesmo pertençam ao campo de processamento de imagens digitais. Entretanto como aqui está-

se a considerar imagens binárias, a técnica será aplicada levando-se isto em consideração.

Definindo-se, inicialmente, duas imagens A e B. A imagem A é a imagem que contém a figura cujos componentes devem ser determinados e que, neste caso, é binária. A imagem B é uma imagem, do mesmo tamanho de A, inicialmente com todos os seus *pixels* iguais. O algoritmo deve iniciar a varredura da imagem A ao partir de algum *pixel,* chamado de *seed* (semente, em inglês). Sendo a *seed,* o primeiro *pixel* da imagem, no canto superior esquerdo. O algoritmo analisa cada um dos oito *pixels* vizinhos da semente e coloca todos os vizinhos que possuem o mesmo valor de intensidade da semente em uma estrutura de dados do tipo fila e na imagem B todos os *pixels* destas posições são rotulados. Ou seja são atribuídos valores aos *pixels* pertencentes à fila. O mesmo procedimento é feito com cada *pixel* da fila. Quando a fila estiver vazia pela primeira vez, terá sido obtido o primeiro componente. Todos os *pixels* correspondentes a este componente estarão rotulados na imagem B. Após a fila ficar vazia, faz-se uma nova varredura na imagem e escolhe-se uma nova *seed* dentre os *pixels* que ainda não foram rotulados. Quando todos os *pixels* de B possuírem um rótulo o processo está concluído.

Após todos os componentes conectados terem sido rotulados fez-se uma triagem daqueles que poderiam ser a placa. Esta triagem foi baseada no tamanho da área do componente. Qualquer elemento com menos de 200 *pixels* de área está descartado. A imagem obtida após esta operação está na Figura 56.

Figura 56: Imagem da região de interesse obtida.

Após isto os autores multiplicaram a imagem original binarizada pela imagem da região de interesse, depois efetuaram o recorte da imagem. O resultado deles está na Figura 57 e o resultado obtido neste trabalho está na Figura 58.

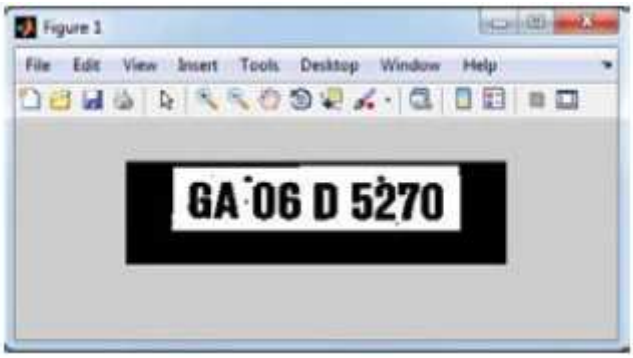

Figura 57: Imagem mostrada no artigo.

Figura 58: Imagem obtida.

Após o recorte, a imagem fica como na Figura 59.

Figura 59: Imagem obtida após recorte.

A imagem obtida após o recorte foi diferente da mostrada no artigo porque o algoritmo utilizado foi diferente. O algoritmo utilizado no trabalho consistiu em descobrir as coordenadas do *pixel* inicial (superior esquerdo) do retângulo branco da Figura 59, e também a altura e a largura deste. Sabendo a região retangular, é possível fazer o recorte da mesma maneira que foi feito no algoritmo de detecção de placas, ou seja, utilizando-se uma função específica do OpenCV. O recorte foi feito entre o retângulo encontrado e a imagem da Figura 56 .

Para determinar o retângulo foram feitos dois laços de "for" para varrer cada *pixel* da imagem da Figura 56. Quatro variáveis foram inicializadas da seguinte maneira:

- "Xmaior" inicializada com o valor zero;
- "Xmenor" inicializada com o número total de linhas da imagem;

- “Ymaior" inicializada com o valor zero;
- “Ymenor” inicializada com o número total de colunas da imagem;
- “Linha” inicializada com o valor zero (variável do laço “for” correspondente às linhas);
- “Col” inicializada com o valor zero (variável do laço “for” correspondente às colunas);

A imagem é testada *pixel* a *pixel* onde as variáveis “linha” e “coluna” correspondem ao valor das coordenadas do *pixel* que está sendo testado. A varredura assume a seguinte forma:

```
Enquanto(Linha=0;Linha<=(total de linhas); Linha=Linha+1)
  Enquanto(Col=0;Col<=(total de colunas); Col=Col+1)
      SE(pixel_atual(Linha,Coluna)=branco) ENTÃO {
      SE(Linha>Xmaior) ENTÃO Xmaior=linha;
      SE(Linha<Xmenor) ENTÃO Xmenor=Linha;
      SE(Coluna>Ymaior) ENTÃO Ymaior=Coluna;
      SE(Coluna<Ymenor) ENTÃO Ymenor=Coluna; }
```

Por simples geometria cartesiana sabe-se que a largura do retângulo será a diferença entre Xmaior e Xmenor e a altura a diferença entre Ymaior e Ymenor.

Após o recorte os autores do artigo isolaram os caracteres em *bounding boxes*, que vêm a ser os menores quadrados possíveis em que cabem cada caractere. A imagem mostrada no artigo está na Figura 60.

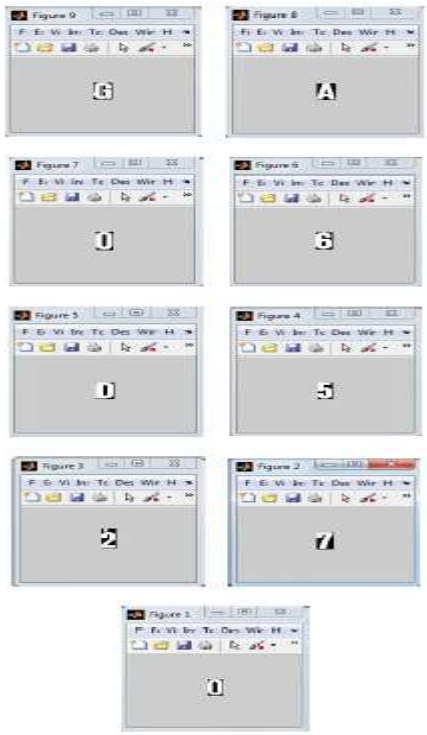

Figura 60: Caracteres isolados em *bounding boxes* no artigo.

O algoritmo utilizado no artigo para isolar os caracteres não é explicado em detalhes. Tendo sido necessário o desenvolver um algoritmo específico. A única coisa que foi dita é que foram utilizadas *bounding boxes.* Para este trabalho partiu-se do princípio de que foi possível extrair os caracteres em um fundo branco (como mostra a Figura 59). O algoritmo apenas faz uma varredura coluna a coluna. Quando acha um *pixel* preto determina o início do caractere quando acha a próxima coluna toda composta de *pixels* brancos sabe que a coluna anterior é o fim do caractere. Não utilizou-se, neste trabalho *bounding boxes* porque a altura do retângulo é a altura máximo da imagem. Esta escolha foi feita por considerar que ela produziria menos erros no tratamento dos caracteres. O mesmo foi feito com a largura do retângula pois, embora com o algoritmo seja possível saber a largura exata do caractere, foi dado espaços em branco nas laterais com o intuito de facilitar a identificação.

O resultado obtido foi praticamente igual ao obtido pelos autores do artigo, com a única diferença de que dois caracteres foram considerados apenas um. Mas isso pode ser atribuído à diferença de resolução de ambas as imagens. O resultado obtido está na Figura 61.

Figura 61: Segmentação de caracteres obtida.

Devido à baixa resolução da imagem que estava sendo utilizada, para que fosse feito o teste de segmentação foi utilizada a imagem da Figura 57 não da Figura 59.

Embora o método tenha sido reproduzido com resultados consideravelmente similares aos obtidos pelos autores do artigo quando aplicados ao conjunto de imagens de teste o método se provou dependente demais das características particulares de cada imagem. Para cada imagem havia praticamente uma combinação específica de *treshold* e de tamanho limite do componente conectado que forma a placa. Os valores dos componentes não podem ser muito baixos pois o método não deveria capturar mais de um componente que forme a placa. Os autores do artigo não detalham como resolveram estes problemas. Uma forma suposta mas não implementada é tornar o processo iterativo para cada imagem. Ajustando-se os valores de *treshold* e tamanho de componentes até a região de interesse ser apenas uma mancha retangular.

Entretanto não foi possível obter uma imagem como a da Figura 59 na maioria dos casos, com apenas os caracteres pretos sobre um fundo branco. Na verdade uma imagem de placa obtida está na Figura 62 (com *treshold 140* e tamanho de componente 120).

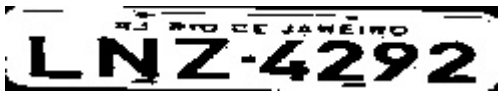

Figura 62:Imagem da placa isolada

Como pode-se notar não seria possível extrair os caracteres desta placa utilizando o algoritmo até então mostrado, devido a partes pretas contínuas nas bordas da imagem.

Desenvolveu-se outro algoritmo de extração de caracteres, que foi considerado mais próximo do que é mostrado no artigo. Neste algoritmo utilizou-se mais uma vez a análise de componentes conectados. Diferente do primeiro uso que foi feito desta técnica, aqui a imagem foi varrida em busca dos oito maiores componentes pretos. Após esta varredura isolou-se estes elementos fazendo-se recorte na imagem de entrada que, neste caso, é a imagem da Figura 62. O resultado obtido está na Figura 63.

Figura 63: Placa segmentada obtida.

A vantagem deste método é que permite a obtenção dos caracteres separadamente, o que facilita a identificação das placas. O índice de segmentação obtido com as placas oblíquas até 65 graus com a câmera é de 92% e o índice de extração é de 98%. Com a placa na condição ótima de noventa graus em relação à câmera o índice de extração permanece em 98% porém o índice de segmentação sobe para 97%(Mirashi, 2013). A partir de 30 graus de inclinação o algoritmo não extrai nenhuma placa e, logicamente, não segmenta nada, também.

3 CONCLUSÃO

A primeira conclusão é de que quanto mais o sistema for capaz de isolar a placa, mais controladas devem ser as condições de funcionamento, em termos de iluminação e ângulo de inclinação ou maior deve ser o poder computacional disponível para fazer um treinamento em reconhecimento de padrões que seja robusto o suficiente.

Foi possível detectar as placas com um grau considerável de acerto, considerando-se o tamanho pequeno do banco de dados para o treinamento, apenas 50 imagens de amostras positivas e 100 de amostras negativa, quando os valores recomendados é de pelo menos 1000 positivas e 2000 negativas.

A principal limitação na implementação do sistema foi a própria ferramenta de identificação Tesseract.

Como foi mostrado anteriormente, mesmo nas imagens segmentadas das placas após o pré-processamento, o índice de acertos é inferior ao obtido na literatura.

A causa possível deste resultado é que embora a ferramenta de OCR tenha sido treinada para identificar a fonte dos caracteres das placas brasileiras, após o pré-processamento a fonte não é mais a mesma. Ligeiras alterações nos caracteres dificultam a identificação sobremaneira.

Como pôde-se perceber da Tabela 2, uma troca de fonte produziu uma variação negativa de 30,76% no acertos em relação às letras.

A solução proposta para este problema é a execução de um novo treinamento para o Tesseract desta vez utilizando várias imagens obtidas do pré-processamento para abarcar a maioria das possibilidades de fontes possíveis.

BIBLIOGRAFIA

KWASNICKA, Halina; WAWRZYNIAK, Bartosz.**License plate localization and recognition in camera pictures.**Novembro de 2002

GILLY, Divya; RAIMOND, Kumudha. **A survey of license plate recognition systems.** Janeiro de 2013

SUNDARI, Guna. **;**VIDHYA, N. **A survey on localization and recognition of license plate number.**Setembro de 2014.

COHEN, Irwin M.; PLECAS, Darryl; MCCORMICK, Amanda V. **A report on the utility of the automated license plate recognition system in British Columbia.**2007

JONES, Michael ;VIOLA, Paul.**Rapid object detection using a boosted cascade of simple features.** 2001

JONES, Michael ;VIOLA, Paul.**Robust Real-time object detection.** Julho de 2001.

http://www.denatran.gov.br/download/resolucoes/resolucao_231.pdf, acessado pela última vez no dia 13/05/2015

GONZALES,Rafael C.; WOODS,Richard E.**Digital Image Processing.**Prentice Hall. Segunda Edição.2002.

MIRASHI, Vinay; SHIRVOIKAR, Manisha; KUDASKAR, Ramesh;BORKAR Samarth. **License Plate Detection and Reconigtion for Goan Vehicles.**Fevereiro de 2013.

http://biblioteca.asav.org.br/vinculos/tede/EduardoDeOliveiraComputacao.pdf , acessado pela última vez no dia 13/05/2015

http://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework, acessado pela última vez no dia 13/05/2015