CARLOS EDUARDO NOVELLETTO RICARDO

DESENVOLVIMENTO DE UM SISTEMA DE RECONHECIMENTO DE PLACAS VEICULARES EM PLATAFORMA EMBARCADA

Florianópolis
2012

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE PÓS-GRADUAÇÃO EM DESENVOLVIMENTO DE PRODUTOS ELETRÔNICOS

CARLOS EDUARDO NOVELLETTO RICARDO

Desenvolvimento de um Sistema de Reconhecimento de Placas Veiculares em Plataforma Embarcada

Trabalho de conclusão de curso submetido à banca examinadora do curso de Pós-Graduação em Desenvolvimento de Produtos Eletrônicos do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, como requisito parcial à obtenção do título de Especialista em Desenvolvimento de Produtos Eletrônicos.

Professor Orientador: Fernando S. Pacheco, Dr. Eng.

Florianópolis, 2012

CDD 621.3815
R488d

Ricardo, Carlos Eduardo Novelletto
Desenvolvimento de um sistema de reconhecimento de placas veiculares em plataforma embarcada [monografia] / Carlos Eduardo Novelletto Ricardo; orientação de Fernando S. Pacheco. – Florianópolis, 2013.

1 v. : il.

Monografia de especialização (Desenvolvimento de Produtos Eletrônicos) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina. Curso de Pós-graduação em Desenvolvimento de Produtos Eletrônicos.

Inclui referências.

1. OCR. 2. Processamento de imagens. 3. Placas veiculares. 4. Sistema embarcado. I. Pacheco, Fernando S. II. Título.

Sistema de Bibliotecas Integradas do IFSC
Biblioteca Dr. Hercílio Luz – Campus Florianópolis
Catalogado por: Edinei Antonio Moreno CRB 14/1065
Rose Mari Lobo Goulart CRB 14/277

Desenvolvimento de um Sistema de Reconhecimento de Placas Veiculares em Plataforma Embarcada

CARLOS EDUARDO NOVELLETTO RICARDO

Este trabalho foi julgado adequado para obtenção do Título de Especialista em Desenvolvimento de Produtos Eletrônicos e aprovado na sua forma final pela banca examinadora do Curso de Pós-Graduação em Desenvolvimento de Produtos Eletrônicos do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 21 de dezembro de 2012.

Banca Examinadora:

_____
Fernando S. Pacheco, Dr. Eng.
Orientador

_____
André Luís Dalcastagnê, Dr. Eng.

_____
Carlos Gontarski Speranza, M. Eng.

Acima de tudo a Deus, nosso senhor e criador de todas as coisas.
À minha mãe, Ana.
Ao meu pai, Antônio.
A toda minha família.

AGRADECIMENTOS

À Deus, por mais esta oportunidade de crescimento e desenvolvimento nos meus estudos.

Aos meus pais, que sempre me apoiaram e acreditaram nos meus esforços.

A minha família, sempre disponível a me distrair e encorajar.

Ao professor Fernando S. Pacheco, pela orientação e incentivo.

RESUMO

Com a frota brasileira superando 64 milhões de veículos automotivos, é de suma importância para órgãos governamentais, escolas, *shoppings centers*, estacionamentos privados etc. a identificação de veículos para atender às demandas de diversas aplicações como, por exemplo, controle de acesso, suporte à polícia, identificação de padrões estranhos de comportamento. Este trabalho apresenta o desenvolvimento de um sistema de reconhecimento de placas veiculares em um sistema embarcado. Este sistema é dividido em quatro etapas: localização, verificação, extração e *OCR* (do inglês, *optical character recognition*). O processo de detecção é baseado na localização dos caracteres, onde são apontadas regiões na imagem candidatas a possuírem caracteres. Após a localização, é necessária a verificação das regiões candidatas geradas na etapa anterior, com o objetivo de reduzir a ocorrência de falso-positivos. Com as regiões da imagem que são candidatas a possuírem texto apontadas e verificadas, é iniciado o processo de extração. Este processo transforma a imagem colorida original em uma imagem binária, para assim ser utilizada em um sistema *OCR* de terceiros. O sistema *OCR* é responsável por produzir um arquivo texto possuindo os caracteres da imagem. Neste trabalho emprega-se o software Tesseract, de código aberto. As quatro etapas são executadas na plataforma embarcada *Beagleboard*. As etapas de localização, verificação e extração foram totalmente desenvolvidas em linguagem C, apresentando uma taxa de acerto de 92% em um banco de testes composto por 30 fotos. O resultado da etapa de OCR com o software *Tesseract* foi de 90% de acerto.

Palavras-Chave: *OCR, processamento de imagens, reconhecimento de placas veiculares, sistema embarcado.*

ABSTRACT

With the Brazilian fleet exceeding 64 million of vehicles, it is essential for government agencies, schools, malls, private parkings etc., the automatic identification of license plates to meet the demands of various applications such as access control, police support, identifying strange patterns of behavior, etc. This paper presents the development of a license plate recognition system for an embedded platform. This system is divided into four stages: localization, verification, extraction, and OCR (optical character recognition). The goal of the detection process is to find (locate) regions with characters in the image. Since this first stage may generate false positives, a second step is necessary. In the verification, some parameters are evaluated, trying to point out only regions with characters. The third stage is called extraction. This process transforms the original image into a binary image, for further processing in a third-party OCR system. An open source software called Tesseract is used in this work. The OCR system is responsible for producing a text file with the characters of the image. The four stages are performed on the BeagleBoard embedded platform. The steps of localization, verification and extraction were developed entirely in the C programing language, with an accuracy of 92% on a test database composed of 30 photos. OCR processing resulted in 90% of accuracy.

Key-Words: OCR, image processing, license plate recognition, embedded system.

LISTA DE FIGURAS

LISTA DE TABELAS

Lista de Equações

SUMÁRIO

1 INTRODUÇÃO

Atualmente existe uma enorme e crescente frota de veículos em todo o Brasil. Segundo dados do Denatran, em 2010 a frota basileira passava de 64 milhões de veículos automotivos. Tal frota exige, tanto de poder público quanto da iniciativa privada, aplicações cada vez mais complexas. Uma delas é o monitoramento em tempo real das placas dos veículos circulantes. Este monitoramento das placas veiculares é importante nas rodovias, como também em ambiente fechados.

Com o avanço da eletrônica, a tarefa de monitoramento pode ser efetuada de forma automática. Para que isso seja possível, é necessário um sistema composto por *hardware* e *software*. O *hardware* engloba a aquisição da imagem e execução dos algoritmos de forma automática. O *software*, por sua vez envolve algoritmos de processamento de visão computacional em uma linguagem que o *harware* entenda e possa executá-lo.

Uma das principais atividades de um sistema de visão computacional dentro da tarefa de detecção de placas é selecionar as áreas de interesse. Esses itens de interesse são os caracteres da imagem sem outros objetos, denominados ruídos. No entanto, esta tarefa não é algo trivial, pois requer inúmeros passos e métodos complexos.

Existem algoritmos baseados na busca da placa veicular em imagens dinâmicas, como é o caso de ARTH, LIMBERGER E BISCHOF (2007). Naquele trabalho os autores propõem um sistema dividido em duas etapas: localização da placa veicular e extração dos caracteres. A localização da placa é baseada nos algoritmos *AdaBoost* (FREUND; SCHAPIRE,1995) e *Haar-like* (Wikipédia, 2012). A extração dos caracteres é baseada na segmentação dos caracteres na imagem, através de filtros que os diferenciam do fundo, de ruído e de elementos espúrios.

Uma dificuldade comum no reconhecimento de placas veiculares é a variação tanto das condições climáticas quanto da posição da placa na imagem. CONCI, CARVALHO E RAUBER (2009) desenvolveram uma solução com três módulos para assim detectar placas de veículos em condições adversas. O sistema é dividido em três etapas: localização da placa, a segmentação de seus elementos e o reconhecimento dos

caracteres. A localização da placa é baseada, dentre outras, na técnica de *top-hat* por fechamento (CAIRVALHO, 2006), onde a imagem passa por diversos processos até se detectar o objeto de interesse. Através da contagem de componentes conectados e um processo de limiarização multinível de OTSU (1979), a imagem da placa é segmentada. Após a segmentação, é proposto um sistema de OCR (do inglês, *optical character recognition*) que reconhece os caracteres da imagem e os transcreve para um arquivo texto.

Além de propostas no âmbito acadêmico, há também sistemas comerciais, como o *License Plate Recognizer* (*INTELLI-VISION*, 2012), desenvolvido pela *IntelliVision*. Este sistema suporta vários tipos de câmeras para aquisição de imagens, porém com limitações no tamanho relativo da placa em relação à imagem e no ângulo da placa em relação à câmera. Pode ser visualizado nas Figuras 1 e 2 exemplos de detecção de placas veiculares com esse software comercial.

O algoritmo utilizado neste trabalho é baseado em uma estrutura de três etapas: localização dos caracteres, verificação e extração. Esta proposta foi aplicada em um sistema de extração textual, desenvolvido por TAHIM (2009). No presente trabalho, esta técnica é aplicada para o caso particular de reconhecimento de placas veiculares.

Para a execução dos algoritmos, é necessário uma plataforma computacional. Existem várias opções, desde um computador pessoal até um cluster de alto desempenho, para processamento de um grande número de imagens. Neste trabalho, o foco é a utilização de um sistema portátil e autônomo, fazendo assim uso de uma plataforma computacional embarcada. A plataforma utilizada é a *Beagleboard* (*BEAGLEBOARD*, 2011), um *hardware* embarcado de alto desempenho com performance semelhante à de um *notebook* de pequeno porte. É projeto da Texas Instruments (TI) em parceria com a DigiKey.

FIGURA 1 - Detecção de placa veicular através do software License Plate Recognizer.
Fonte: *INTELLI-VISION*, 2012.

FIGURA 2 - Detecção de placa veicular através do *software License Plate Recognizer*. Detalhe de numeração da placa veicular.
Fonte: *INTELLI-VISION*, 2012**.**

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver um sistema de reconhecimento de placas veiculares embarcado de baixo custo, para ambientes com boa iluminação e velocidade do veículo em análise zero (veículo parado).

1.1.2 Objetivos Específicos

- O sistema deve processar imagens no formato *jpeg*.
- Desenvolver o *software* do sistema embarcado utilizando ferramentas gratuitas e de código aberto.
- Avaliar algoritmos já existentes para os processos de localização, verificação e extração dos caracteres em uma imagem.
- Avaliar o desempenho do sistema em relação à taxa de acerto e ao tempo de processamento.
- Como resultado, o sistema deve apresentar uma imagem com os caracteres extraídos e o resultado do OCR em um arquivo texto.

1.2 Escopo e Delimitação do Sistema Proposto

Há vários sistemas de localização de placas veiculares existentes no mercado, com diferentes características: detecção de placas de carros em movimento ou parados, em condições climáticas adversas, etc.

O sistema aqui desenvolvido processa imagens provenientes de uma câmera digital direcionada para a placa veicular. Na Figura 3 pode ser visto uma possível aplicação do sistema onde uma câmera é direcionada a placa veicular e realiza o processo de extração da mesma. Pode também ser utilizado em outras aplicações em que o veículo observado está com velocidade zero. Tal limitação, no estágio atual do trabalho,

deve-se ao fato de não ser verificado se a imagem encontra-se fora de foco ou borrada, o que é provável em velocidade maior.

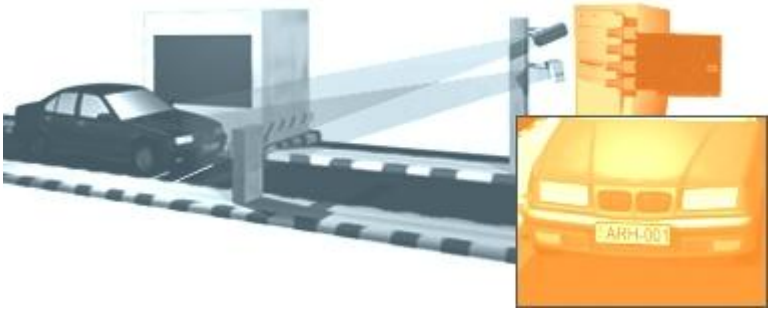


FIGURA 3 - Sistema de localização de placas veiculares para cancelas.
Fonte: <http://www.platerecognition.info>. Acessado em 10/10/2012.

Além disso, com um sistema embarcado para o processamento de vídeo, os recursos de processamento são reduzidos em comparação com um computador de maior porte. Assim, não se pode garantir a emissão do resultado do sistema de forma instantânea de tempo (tempo de processamento menor que 1 s).

1.3 Justificativa

Com a frota brasileira superando 64 milhões de veículos automotivos, é de suma importância para órgãos governamentais, escolas, *shoppings centers*, estacionamentos privados etc. a identificação de placa veicular para atender as demandas de diversas aplicações.

A tecnologia de reconhecimento de placas veiculares pode ser aplicada em controle de acesso, auxiliar órgãos privados e governamentais em operações antifurto e de rastreabilidade e, de modo geral, auxiliar na segurança em prol da população.

Visto a grande demanda para esse tipo de aplicação nos mais diversos nichos de mercado, há uma notória oportunidade para tecnologias que atendam este segmento em suas mais diversas variações de investimentos.

O sistema proposto, diferentemente de outros sistemas comerciais, é de baixo custo, pois faz uso de tecnologias de *hardware* de larga escala e bibliotecas de *software* gratuitas que facilitam e aumentam a velocidade de implementação do sistema.

Além das vantagens de custo de tamanho reduzido também é um sistema que pode ser um produto que futuramente pode ser empregado em rastreamento de placas veiculares para rastrear veículos roubados ou até de sequestros, emissão de ticket de multas instantâneo, entre outras.

1.4 Diretrizes Metodológicas

As diretrizes aplicadas neste trabalho o dividem em duas áreas. A primeira é o estudo do *hardware* e do ambiente de desenvolvimento. A segunda envolve o estudo e a implementação dos algoritmos de visão computacional.

Em uma 1ª etapa, o estudo do ambiente de desenvolvimento e do *hardware* utilizado compõe-se por:

- Ambientação na plataforma de desenvolvimento *Beagleboard,* com pesquisa de referências que possam servir como base para a utilização do *DSP* integrado ao *OpenCV*;

- Testes com aplicações exemplos, para certificar-se de que a integração dos módulos necessários para utilizar o *DSP* embarcado está funcionando;

- Desenvolvimento de aplicações com a biblioteca O*penCV*, integradas ao *Codec Engine* e ao *DSP*, para ambientação com o sistema.

Em um segundo momento, faz-se a implementação dos algoritmos de visão computacional, com as seguintes etapas:

- Estudar algoritmos de visão computacional de Tahim (2009) e estudá-los na ferramenta matemática *Matlab*;

- Comparar os resultados obtidos com a execução dos algoritmos de localização, verificação e extração na *Beagleboard* com o *Matlab*;

- Validação da implementação com, no mínimo, 25 fotos.

2 REVISÃO BIBLIOGRÁFICA

Para a localização de placas veiculares, o sistema deve executar uma série de passos distintos, aqui divididos em quatro etapas: localização, verificação, extração e, por fim, um sistema *OCR*. Isso compõe um sistema de extração da informação textual (EIT), como mostra a Figura 4.

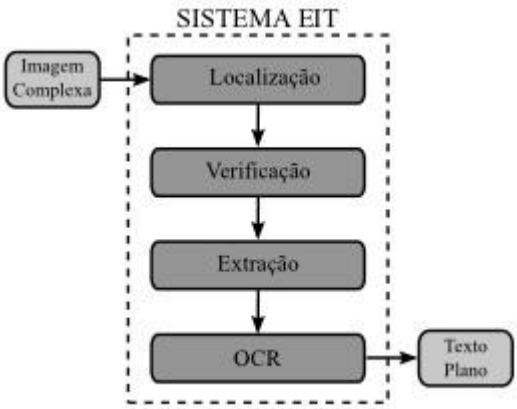


FIGURA 4 – Visão geral das etapas de um sistema de extração da informação textual – EIT.
Fonte: TAHIM, 2009.

A etapa de localização tem como objetivo apontar onde estão os caracteres em uma imagem complexa. Essa identificação gera como resultados retângulos denominados *Bound Boxes* (*BBs*), que delimitam uma região candidata a possuir caracteres. Nesse processo, há a possibilidade de geração de vários falsos positivos, isto é, regiões sem caracteres, mas apontadas como proprietárias de caracteres.

Desta maneira, é necessária uma etapa de verificação que tem como objetivo descartar regiões que não possuem caracteres e selecionar apenas aquelas que são de fato caracteres.

Após essa “filtragem” dos caracteres, é necessário gerar uma imagem binária que contemple apenas os caracteres, processo denominado de extração, para posterior uso de um *OCR.* Essa última etapa gera um arquivo texto com os caracteres

da imagem, ou seja, obtendo-se, ao final do processo, um arquivo ASCII.

Neste trabalho, as quatro etapas são executadas na *Beagleboard*. As três primeiras foram implementadas em linguagem C. Por uma questão de tempo e complexibilidade, optou-se por usar um *OCR* desenvolvido por terceiros na última etapa. Os algoritmos e técnicas de cada uma das etapas são discutidos a seguir.

2.1. Etapa de Localização

A etapa de localização de caracteres tem como objetivo percorrer a imagem e apontar regiões candidatas a possuírem caracteres. Existem várias formas de executar esta tarefa: métodos baseados em textura ou em região, como classificado na Figura 5, por TAHIM (2009).

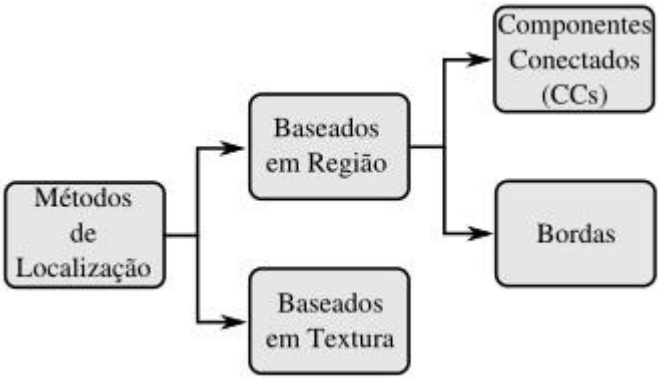


FIGURA 5 - Métodos de localização de caracteres.
Fonte: TAHIM, 2009.

2.1.1 Métodos Baseados em Região

Os métodos baseados em região buscam encontrar bordas onde há grande variação de contraste na imagem, de modo a delimitar regiões com grande probabilidade de possuir caracteres.

Na Figura 6 FIGURA 6é possível notar que nos limites das letras da palavra “PARE” com o fundo, sempre existe um

contraste muito forte, fazendo com que este parâmetro seja muito conveniente para delimitar regiões candidatas a possuírem caracteres em uma imagem.

FIGURA 6 – Imagem para exemplo em que se observa a variação de contraste entre caractere e plano de fundo.

2.1.1.1 Métodos Baseados em Componentes conectados - CCs

Representa uma classe de métodos que varrem a imagem à procura de componentes conectados (CCs), ou seja, *BBs* que possuem características similares e que estão dispostos sobre um eixo.

De modo geral, esse método apresenta quatro etapas distintas:

- Pré-processamento, como clusterização de cor e redução de ruído.
- Geração dos CCs.
- Filtragem dos CCs não textuais.
- Agrupamento dos CCs.

A Figura 7 mostra um exemplo de aplicação das etapas deste algoritmo sobre uma imagem exemplo.

FIGURA 7 - Geração de CCs - Métodos baseados em CCs. (a) Imagem original com caracteres de baixa densidade e artefatos incluídos durante o processo de compressão. (b) Extração dos CCs da imagem original, em que cada CC está representado por uma cor e delimitado por um BB em preto. (c) Geração correta do CCs da imagem. (d) Agrupamento dos CCs.
Fonte: TAHIM, 2009.

2.1.1.2 Métodos Baseados em Bordas

Diferentemente do método de componentes conectados, que explora similaridade entre regiões, este é baseado apenas em delimintar as bordas que possuem alto contraste entre a região de fundo e texto para assim gerar os *BBs*. As bordas normalmente são geradas através da aplicação de algoritmos como *Canny, Sobel,* entre outros, sobre a imagem. Também são empregados filtros para a detecção de bordas textuais.

2.1.2 Métodos Baseados em Textura

Observando que os caracteres possuem certa similaridade em tamanhos, cores e distância, essas características podem ser usadas para localizar caracteres a partir de métodos baseados em textura.

2.1.3 Método de localização utilizado neste trabalho

O método utilizado é uma customização dos métodos baseados em bordas, através da extração do gradiente da imagem. Desse modo, onde há mudanças abruptas de contraste, são geradas bordas nas regiões candidatas a possuírem caracteres.

Nas seções seguintes são apresentados os passos para extração dessas bordas.

2.1.3.1 Cálculo do Gradiente de intensidade

O cálculo do gradiente de intensidade é efetuado em alguns passos.

Primeiramente, a imagem em tons de cinza é filtrada por dois filtros derivativos, horizontal H_h e vertical H_v de BLANCHET e CHARBIT (2006):

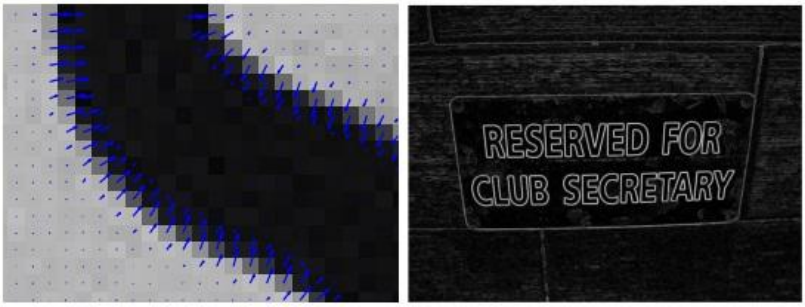
$$H_h = \frac{1}{3} \times \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad H_v = \frac{1}{3} \times \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \tag{1}$$

A filtragem é executada através da convolução da imagem utilizando os *kernels* da equação 1, que, por sua vez, geram duas imagens G_x *e* G_y. Com isso, é possível calcular a magnitude e o vetor do gradiente para cada pixel:

$$G_{mag}(i,j) = \sqrt{G_x(i,j)^2 + G_y(i,j)^2} \tag{2}$$

$$G_\theta(i,j) = arctan\left(\frac{G_y(i,j)}{G_x(i,j)}\right) \tag{3}$$

O resultado da equação (2) e da equação (3) pode ser visto na Figura 8, onde existem vetores em azul indicando a magnitude do gradiente e sua direção.



(a) (b)

FIGURA 8 - Gradiente da imagem-exemplo. (a) Apresentação dos vetores gradiente de cada pixel (setas azuis) em uma região da imagem-exemplo. (b) Imagem representando a magnitude do gradiente G_{mag}(quanto maior a magnitude, mais próximo da cor branca os pixels se apresentam).
Fonte: TAHIM, 2009.

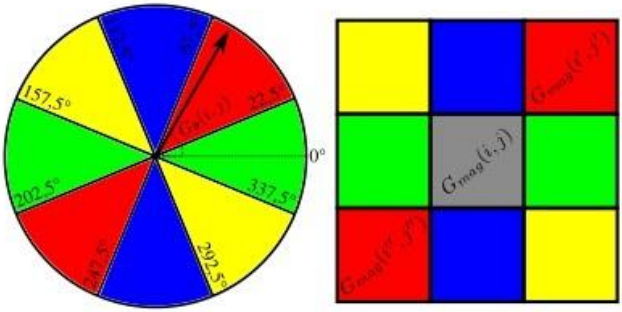
2.1.3.2 Extração dos Contornos Candidatos a Texto

As regiões candidatas a contornos de texto são aquelas em que geralmente o contraste é elevado entre os limites da área das letras e a cor de fundo. Existe um valor para verificar quais bordas são candidatas a ser caracteres, denominado liminar L_m, que é calculado a partir de vários fatores, descritos a seguir.

Uma borda é candidata a caracteres quando:

1. Conforme Tahim (2009), a magnitude do gradiente do pixel (i,j), deve ser maior do que a magnitude de ambos os pixels da vizinhança-8(i',j') e (i'',j'') relacionados à direção do gradiente do pixel (i,j). Assim,

$$G_{mag}(i,j) > G_{mag}(i',j') \; e \; G_{mag}(i,j) \; > G_{mag}(i'',j'') \qquad (4)$$



(a)

FIGURA 9 - Exemplos da correspondência entre o ângulo do vetor gradiente do pixel (i,j) sob avaliação e os pixels da vizinhança-8 (i′,j′) e (i″,j″).
Fonte: TAHIM, 2009.

Note que no gráfico em pizza da Figura 9 existe $G_\theta(i,j)$, que é o ponto do centro do gráfico, e o vetor de magnitude está sobre uma área vermelha que corresponde ao pixel $G_{mag}(i',j')$. O pixel $G_{mag}(i'',j'')$ é localizado na área oposta a de $G_{mag}(i',j')$.

No entanto, numa imagem isso não infroma a localização dos pixels de interesse, para tal eles devem ser remapeados para a matriz de cores ao lado do gráfico em pizza. Desta maneira $i' = i + 1\,, j' = j - 1$ e $i'' = i - 1\,, j' = j + 1$.

2. Para satisfazer o requisito de pixel de borda candidata a caractere é necessário que o pixel de G_{mag} possua valor maior que o limiar L_m, determinado pela equação 5:

$$L_m = \frac{\sum_{(i,j)\in P1}\left(G_{mag}(i,j)\cdot|G_{Diff}(i,j)|\right)}{\sum_{(i,j)\in P1}|G_{Diff}(i,j)|} \tag{5}$$

$$G_{Diff}(i,j) = G_{mag}(i',j') - G_{mag}(i'',j'') \tag{6}$$

onde $\boldsymbol{P1}$, é a imagem em níveis de cinza; $G_{mag}(i',j')$ e $G_{mag}(i'',j'')$ são as magnitudes do gradiente dos pixels da vizinhança-8 do pixel (i,j), como ilustrado na Figura 9.

2.2 Etapa de Verificação

O método de localização descrito anteriormente gera bordas que não estão relacionadas ao caracteres de interesse. Qualquer borda que possua um grau de contraste elevado muito provavelmente será dada como candidata a caractere.

Para minimizar estas limitações do algoritmo, é necessário realizar uma análise sobre as regiões candidatas para verificar se são de fato caracteres, pois um sistema *OCR* não será capaz de reconhecer as letras quando houver muita "poluição" na imagem.

Neste trabalho, apenas são considerados os parâmetros estruturais a seguir, a partir do trabalho desenvolvido por TAHIM (2009).

- A média da magnitude do gradiente;
- A variância da magnitude do gradiente;
- O *skewness* da magnitude do gradiente;
- O *kurtosis* da magnitude do gradiente;
- A média da direção do gradiente;
- A variância da direção do gradiente;
- O *skewness* da direção do gradiente;
- O *kurtosis* da direção do gradiente;
- A máxima variação da direção do gradiente;
- A razão entre o número de pixels de contorno e a dimensão do *BB*;
- A razão entre o número de pixels de contorno e a área do *BB*.

Esses parâmetros são detalhados a seguir.

2.2.1 Média da magnitude do gradiente

De acordo com LIU, GOTO e IKENAGA (2006) a média da magnitude do gradiente dos pixels de contorno de caracteres tende a ser maior do que outros objetos na imagem. Dessa

maneira, faz com que seja um ótimo parâmetro para verificar se a região de análise é um caractere ou não.

LIU, GOTO e IKENAGA (2006) propuseram o atributo de média da magnitude do gradiente dos pixels de contorno, cuja equação é dada por:

$$M_{avg} = \frac{\sum_{(i,j)\in CC} M(i,j)}{n \cdot L_m} \tag{7}$$

onde n representa o número de pixels de contorno; L_m um liminar adaptativo obtido da imagem e $M(i,j)$ é a magnitude do gradiente do pixel.

2.2.2 Variância da magnitude do gradiente

Os contornos de caracteres normalmente apresentam uma variação de magnitude do gradiente superior aos contornos não textuais. Isso pode ser observado na Figura 6, que possui regiões com variações abruptas de direção (cantos) ou segmentos de reta acoplados a curvas suaves, conforme TAHIM (2009).

2.2.3 Skewness da magnitude do gradiente

O *skewness* é uma medida do grau de assimetria de uma distribuição de probabilidade. Neste trabalho, o *skewness* é definido como:

$$S = \frac{\mu_3}{\sigma^3} \tag{8}$$

onde μ_3 é o terceiro momento central e σ, o desvio padrão.

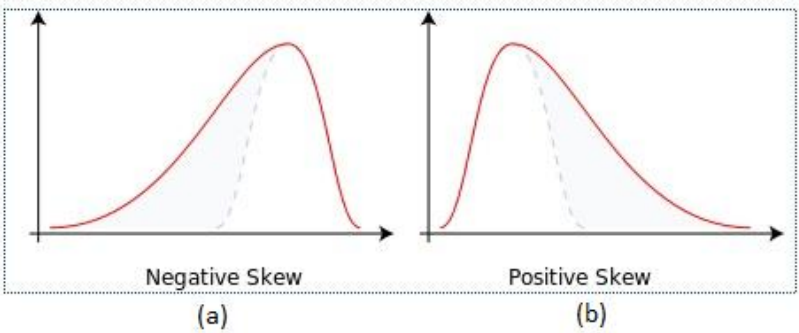


FIGURA 10 - Visualização de distribuição de probabilidade com skewness negativo (a) e positivo (b). Fonte:<http://en.wikipedia.org/wiki/Skewness>. Acessado em 18/10/2012

A Figura 10 mostra duas distribuições de probabilidade, sendo que em (a) apresenta-se um *skew* negativo (tem cauda à esquerda da distribuição) e em (b) positivo, pois a cauda está à direta.

Essa é uma característica que pode, geralmente, ser utilizada para separar blocos de imagem textuais de não-textuais. De acordo com TAHIM (2009), contornos não-textuais, por possuírem uma maior predominância de segmentos de reta, possuem uma assimetria cujo *skew* é positivo.

Desta maneira o *skewness* positivo, em uma amostra da imagem, representa comumente uma região não-textual.

2.2.4 Kurtosis da magnitude do gradiente

O *kurtosis* é uma medida do grau de “achatamento” de uma distribuição de probabilidade. Neste trabalho, o *kurtosis* é definido como:

$$S = \frac{\mu_4}{\sigma^4} \tag{9}$$

onde μ_4 é o quarto momento central e σ é o desvio padrão.

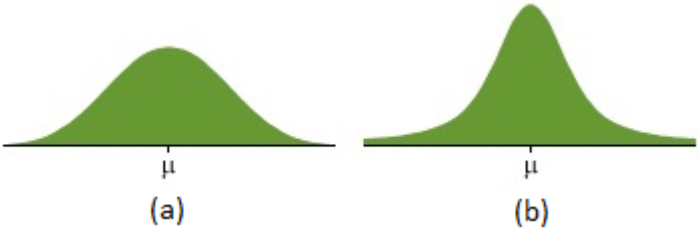


FIGURA 11 - Visualização de distribuição de probabilidade com *kurtosis* menor (a) e maior (b).
Fonte: <http://www.riskglossary.com/link/kurtosis.htm>. Acessado em 18/10/2012.

A Figura 11 mostra duas distribuições de probabilidade, sendo que a Figura 11 (a) possuiu *kurtosis* menor. É menor pois existe uma cauda mais curta que na Figura 11 (b) e possui um achatamento vertical menor.

2.2.5 Média da direção dos pixels de contorno do gradiente

A média da direção dos pixels de contorno, $G_\theta(i,j)$, deve encontrar-se na região de $0\ a\ \pi$, devido ao fato que o contorno percorre uma região fechada, conforme TAHIM (2009).

2.2.6 Variância da direção dos pixels de contorno do gradiente

A variância da direção dos pixels de contorno num caractere é normalmente maior que de regiões não-textuais devido ao fato de apresentarem contornos fechados. Esse parâmetro encontra-se com valores entre $0\ a\ 2\pi$, de acordo com TAHIM (2009).

2.2.7 Skewness da direção do gradiente

CLARK e MIRMEHDI (2000) observaram que a direção dos pixels de contorno do gradiente é a mesma para a maioria, quando a amostra for um caractere, apresentando no entanto sentido oposto.

Desta maneira, regiões com caracteres possuem pixels de contorno simétricos, fazendo com que o *skew* seja próximo a zero.

2.2.8 Kurtosis da direção do gradiente

Regiões de imagens que possuem contornos textuais apresentam similaridade na direção fazendo com que exista um *kurtosis* elevado nestas áreas, conforme TAHIM (2009).

2.2.9 Máxima variação da direção do gradiente

LIU, GOTO e IKENAGA (2006) verificaram que a máxima variação da direção do gradiente é um atributo muito conveniente para separar regiões textuais de não-textuais devido ao fato de que estas possuem regiões de variação de, no máximo, $0 \; a \; 2\pi$.

2.2.10 Razão entre o número de pixels de contorno e as dimensões dos BB

Devido ao fato de que regiões textuais apresentam contornos com perímetros fechados, LIU, GOTO e IKENAGA (2006) observaram que a razão dos pixels de contorno entre a sua maior dimensão (largura ou altura) do *BB*, deve ser, no mínimo, duas vezes maior.

2.2.11 Razão entre o número de pixels de contorno e a área do BB

A razão entre a quantidade de pixels e a área do *BB* pode também ser utilizada para filtrar as regiões de interesse, embora, conforme TAHIM (2009), não apresente uma separação nítida de regiões. Pode-se utilizar este método em combinação com outros fatores.

2.3 Etapa de Extração

Após a localização e verificação, são obtidas regiões com caracteres. Entretanto, essas regiões podem apresentar vários

problemas tais como, baixa resolução, ruídos, baixo contraste, entre outros. Mesmo em um sistema *OCR* atual, essas características dificultam a obtenção correta do texto.

Por este motivo, é necessário reduzir tais ruídos para que as imagens, posteriormente, tenham seus caracteres extraídos corretamente. Esta é a etapa de extração, que elimina os ruídos da imagem de forma geral e entrega ao *OCR* uma imagem binária com adequado contraste.

Serão apresentados, a seguir, uma série de algoritmos de extração e suas características. Existem, basicamente, três tipos principais de algoritmos: clusterização de cor, limiarização global e limiarização local.

2.3.1 Abordagem de Limiarização Global

O método de limiarização global busca um único limiar para assim gerar a cor branca ou preta para a imagem de extração. Desta maneira, os pixels da imagem, em escala de cinza, que possuírem valor maior que este limiar possuem a cor preta.

Na Figura 12 pode ser visualizada uma imagem que passou pelo processo de extração global.

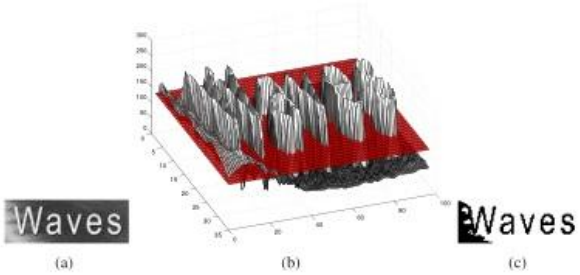


FIGURA 12 - Exemplo de limiarização global. (a) Imagem original (nível de cinza). (b) Imagem original apresentada como uma superfície de valores de intensidade (cinza) e segmentada pelo plano de limiarização global (vermelho). (c) Imagem extraída.
Fonte: TAHIM, 2009.

2.3.1.1 Método de Otsu

Classificado com um método de limiarização global, pois faz uma análise a partir da imagem inteira, assume que as imagens contêm dois grupos ou classes de pixels, extraídos através de histogramas. Com a variância entre os pixels da imagem calcula-se um liminar de extração global para a imagem que será extraída. Desta maneira encontra-se um limiar T ótimo através da seguinte equação:

$$\sigma^2_{Entre-Classes}(T) = \omega_{PF}(T)(\mu_{PF}(T) - \mu_I)^2 + \omega_O(T)(\mu_O(T) - \mu_I)^2 \quad (10)$$

onde $\omega_{PF}(T)$ e $\omega_O(T)$ representam a soma das probabilidades das intensidades dos pixels do plano de fundo e dos pixels do objeto, respectivamente. As variáveis $\mu_{PF}(T)$, $\mu_O(T)$ e μ_I representam a média da intensidade dos pixels do plano de fundo, do objeto e da imagem, respectivamente.

2.3.2 Abordagem de Limiarização Local

Os métodos de limiarização local, diferentemente dos de limiarização global, geram um limiar $T(x, y)$ para cada pixel.

Para gerar o limiar para cada pixel, existe uma janela $\boldsymbol{W}$, que é propagada no BB, extraindo da imagem várias características, tais como, média, variância, outras características já presentadas e outras diferentes. Estas características servem como base para gerar o limiar $T(x, y)$.

Com a determinação do limiar, é gerada uma superfície de extração para, por fim, gerar a imagem binária.

2.3.2.1 Método de Niblack

Este método é baseado na extração de características baseados em janela, fazendo com que seja de limiarização local.

O Método de NIBLACK (1986) calcula o limiar $T(x, y)$, para cada pixel, através da média e do desvio padrão dos pixels na janela $\boldsymbol{W}$**.** Desta maneira tem-se a equação:

$$T(x,y) = m(x,y) + K \cdot s(x,y) \quad (11)$$

onde $m(x,y)$ denota a média e $s(x,y)$, o desvio padrão da intensidade dos pixels presentes em uma região da imagem sob a janela W. A constante K determina quanto da região das bordas do objeto é considerado como parte do objeto.

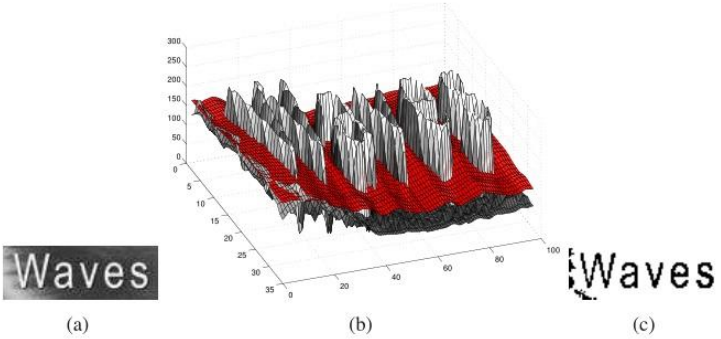


FIGURA 13 - Exemplo de limiarização local. (a) Imagem original (em níveis de cinza); (b) Imagem original apresentada como uma superfície de valores de intensidade (cinza) e segmentada pela superfície de limiarização local (vermelho). (c) Imagem extraída. Fonte: TAHIM, 2009.

2.3.2.2 Método de Chen e Yuille

O método de Niblack tem uma limitação muito forte no que diz respeito ao tamanho da janela, que é fixo. Desta maneira, CHEN e YUILLE (2004), propuseram um algoritmo que gera várias janelas até encontrar aquela que atende o requisito em questão. Para a janela ser apropriada, o desvio padrão da janela deve ser maior que um liminar T_σ.

2.3.2.3 Método de Sauvola

SAUVOLA e INEN (2000) propõe uma versão modificada do método de limiarização local de Niblack, incluindo para o cálculo do limiar uma faixa dinâmica para o desvio-padrão e um

ganho, também para o desvio-padrão, utilizando a média da janela. Segue assim a equação:

$$T(x,y) = m(x,y) \cdot \left[1 + K \cdot \left(\frac{s(x,y) - R}{R}\right)\right] \quad (12)$$

onde R é a constante de sensibilidade.

2.3.2.4 Método de Bernsen

O método proposto por BERNSEN (1986) desliza uma janela sobre a imagem e procura o menor e o maior pixel. O limiar $T(x,y)$ é aplicado ao pixel de centro da janela se a diferença entre o maior e menor pixels for maior que um liminar L. Caso contrário, o pixel central é considerado plano de fundo. Este método segue a seguinte equação:

$$T(x,y) = \begin{cases} \dfrac{Z_{max} - Z_{min}}{2}, & (Z_{max} - Z_{min}) \geq L \\ \text{Plano de fundo}, & \text{caso contrário} \end{cases} \quad (13)$$

onde $Z_{max} - Z_{min}$ representam o maior e menor pixel da janela; L o liminar da diferença entre $Z_{max} - Z_{min}$.

2.3.3 Avaliação dos algoritmos

Nas Figuras 14 e 15 são mostradas duas imagens que passaram pelo processo de extração no *Matlab*. É possível verificar que o algoritmo que teve o melhor desempenho é o de *Otsu*, no Anexo A é possível verificar que este algoritmo se mostrou eficaz também para a extração dos caracteres das placas. O resultado possui formas bem definidas nas letras, bom preenchimento e definição se comparado as outras imagens. Devido aos fatos mostrados e também pelo menor custo computacional, o algoritmo escolhido para extração é o de *Otsu*.

(a)

(b)

(c)

(d)

(e)

FIGURA 14 - Comparação entre os algoritmos de extração. (a) Imagem original. (b) Niblack. (c) Otsu. (d) Sauvola. (e) Bernsen.

(a)


(b)


(c)


(d)


(e)

FIGURA 15 - Comparação entre os algoritmos de extração. (a) Imagem original. (b) Niblack. (c) Otsu. (d) Sauvola. (e) Bernsen.

2.4 *Software OCR*

As etapas anteriores visam à extração dos caracteres de forma que a imagem possa ser utilizada num sistema *OCR*

Os sistemas *OCR* requerem certos padrões de imagens para operar de forma adequada. Normalmente requerem um

estilo de documento, sem texto em perspectiva nem texto aleatório, visto que possuem dicionários internos para comparação.

Atualmente, existe uma infinidade de softwares *OCRs*, pagos ou não, e com diferentes taxas de acerto. Neste trabalho, o escolhido é o *Tesseract*, da *Google*, pelos seguintes motivos:

- É gratuito e de código aberto;
- É portável para diversas arquiteturas de hardware e diversos sistemas operacionais;
- Pode ser utilizado por diversas interfaces gráficas de terceiros, pois é executado através de linhas de comando.

3 DESENVOLVIMENTO

Este capítulo mostra o que foi realizado para que houvesse sucesso na implementação do sistema. É apresentado o *hardware* utilizado, as principais ferramentas e *software* desenvolvido.

Nos apêndices A ao D é possível visualizar vários tópicos necessários para a configuração correta do ambiente de desenvolvimento.

3.1 *Hardware* embarcado utilizado

Atualmente existem muitas opções de *hardware* embarcado. Essas opções variam muito no requisito de memória *SDram*, *Flash*, interface de vídeo e arquitetura do processador.

Atualmente uma das arquiteturas mais usadas no mercado embarcado é a *ARM*. Essa família está presente desde microcontroladores de baixo custo até processadores de alto desempenho para aplicações multimídia. Na Figura 16, apresenta-se uma comparação simplificada entre os vários tipos de núcleo *ARM* e suas aplicações.

Para processamento de vídeo são necessário processadores de alto desempenho, representados na aba “Aplicação” da Figura 16. Desta maneira, considerando estes requisitos básicos de performance e disponibilidade, o *hardware* escolhido para ser utilizado como *host* da aplicação é a *Beagleboard*. Este *hardware* contempla um *ARM Cortex-A8* e é destinado a aplicações de multimídia embarcada.

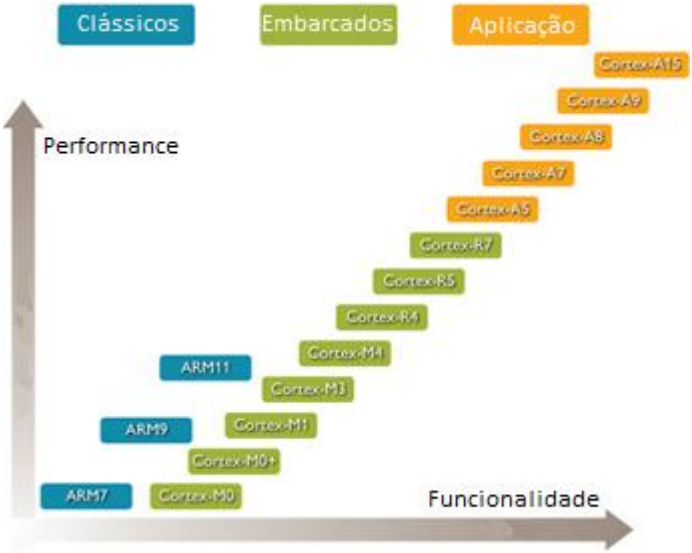


FIGURA 16 - Comparação entre processadores *ARM*.
Fonte: <http://www.arm.com/products/processors/index.php>.
Acessado em 20/10/2012.

A *Beagleboard* é uma plataforma *open hardware*, desenvolvida em conjunto pela *Texas Instruments* (TI) e *DigiKey*. Foi desenvolvida com a ideia de ser uma plataforma educacional, para levar às universidades a ideia de *open hardware* e *open software* em aplicações embarcadas. Também pode ser utilizada para aplicações comerciais quando for o caso.

É utilizada nas mais diversas aplicações, desde robótica (projeto *OpenROV*), visão computacional (biblioteca *OpenCV*), entre outras.

Existem, atualmente, três versões da *Beagleboard*. Aqui será apresentada apenas a *Beagleboard-xM* (Figura 17), que tem as seguintes especificações:

- Processador Super-Scalar ARM Cortex-A8;
- 512MB de memória RAM;
- Porta USB de 2.0 de alta velocidade;
- Saídas *DVI-D e S-Video;*
- Entrada/Saída de áudio estéreo;
- Entrada para cartão *microSD*;

- JTAG;
- Entrada para Câmera (CCD e CMOS);
- Frequência de operação de 1GHZ;
- Processador de Sinais Digital da família C64x integrado ao processador principal.

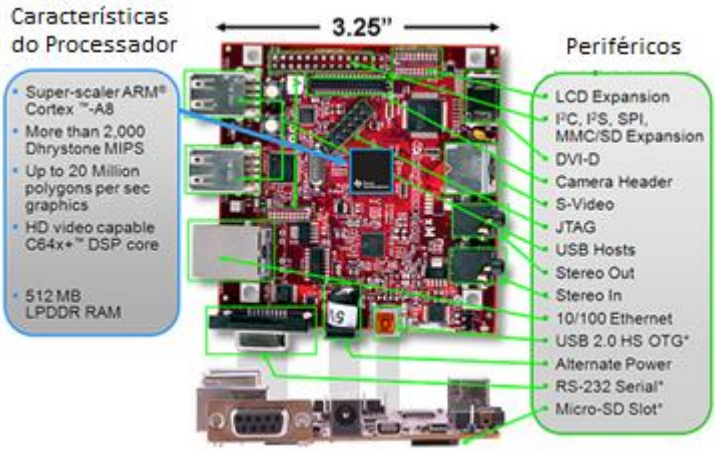


FIGURA 17 – Imagem e Características *Beagleboard*-xM.
Fonte: < http://*Beagleboard*.org/hardware-xm >. Acessado em 27/10/2012.

3.2 Visão geral do Processador

O coração da *Beagleboard* é composto por um DM3730, que é um processador multimídia altamente integrado da *TI*. Na Figura 18 apresenta-se o diagrama em blocos deste processador que é muito extenso. Dentre seus blocos principais podem ser citados: *DMA*, acelerador gráfico e periféricos.

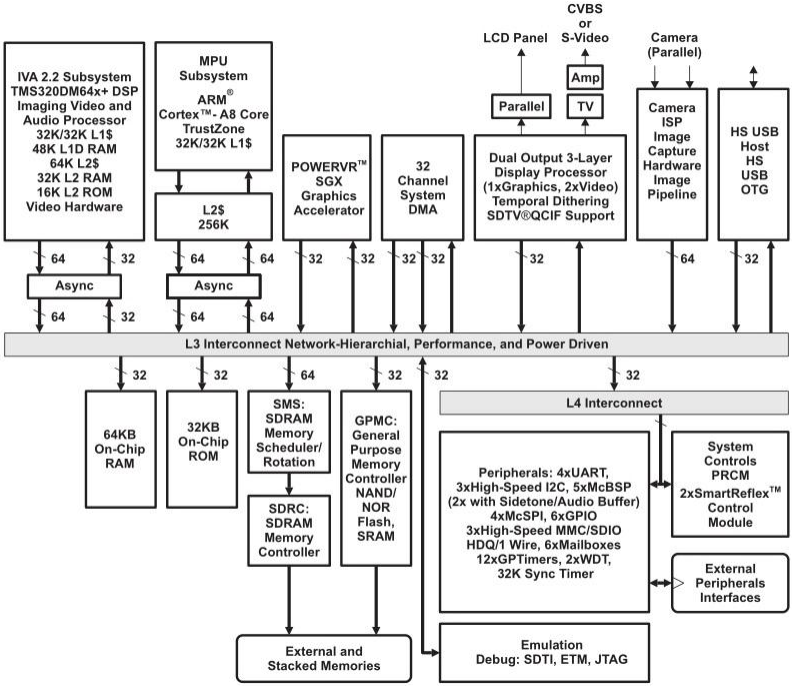


Figura 18 - Diagrama de blocos do DM3730.
Fonte: http://www.ti.com/lit/ds/symlink/dm3730.pdf>. Acessado em 27/10/2012.

3.3 Detalhes de implementação

Nesta seção são descritos os principais passos efetuados para a detecção dos caracteres.

Entre as Figuras 19 e 22 é possível verificar as principais funções desenvolvidas para executar a etapa de localização, baseada no algoritmo da seção 2.1.3.

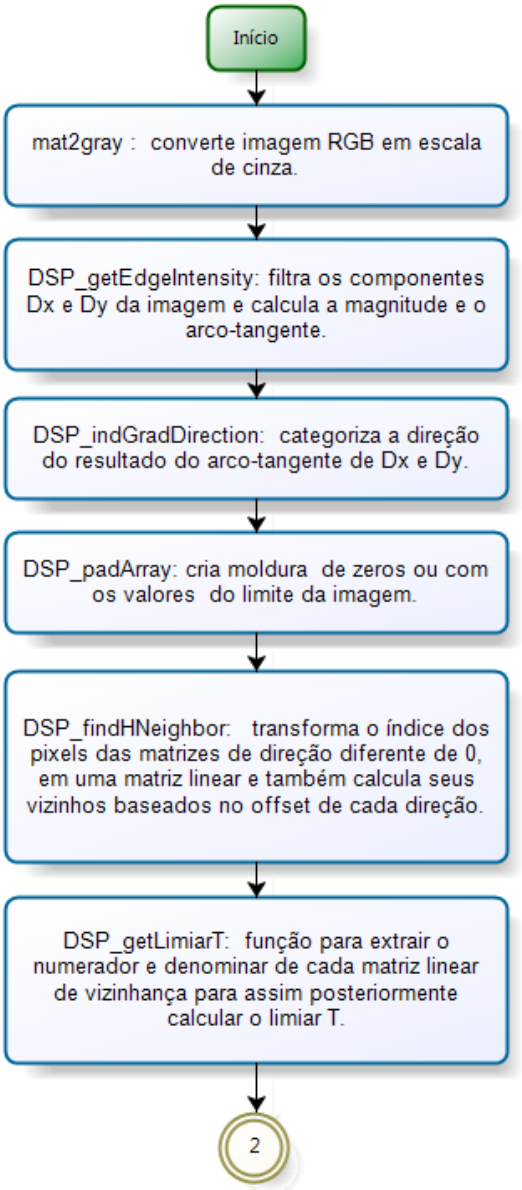


FIGURA 19 - Principais passos da execução do algoritmo na etapa de localização (Parte 1).

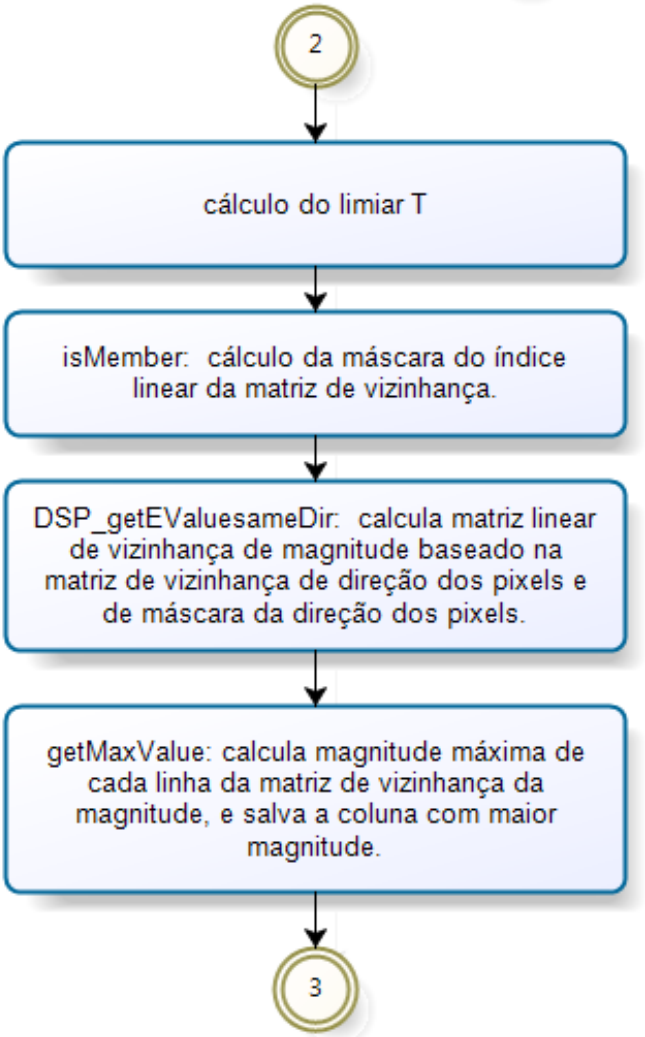


FIGURA 20 - Principais passos da execução do algoritmo na etapa de localização (Parte 2).

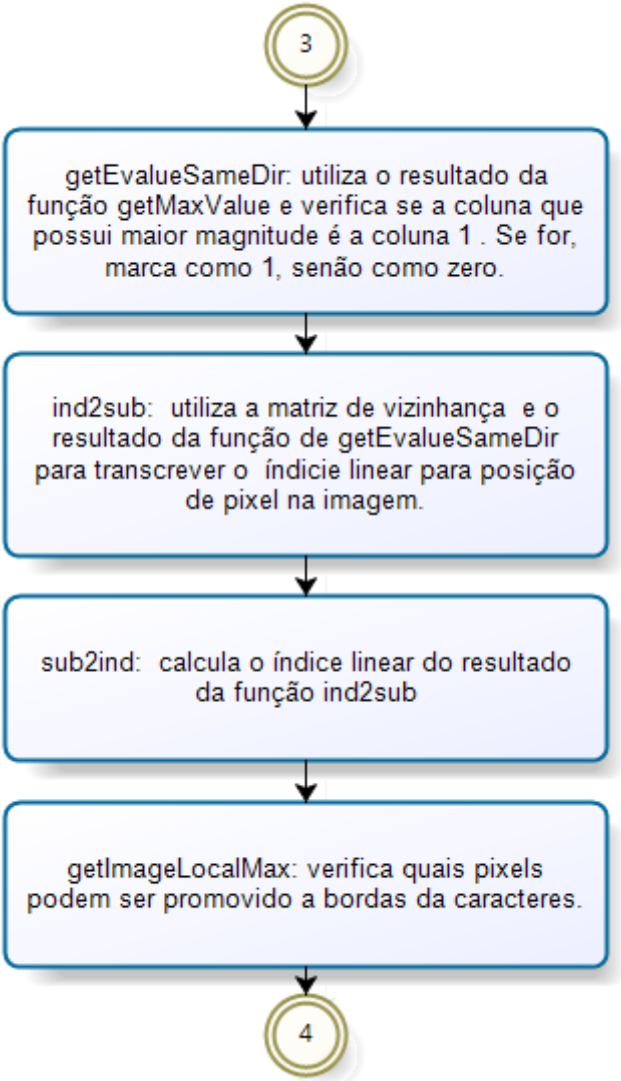


FIGURA 21 - Principais passos da execução do algoritmo na etapa de localização (Parte 3).

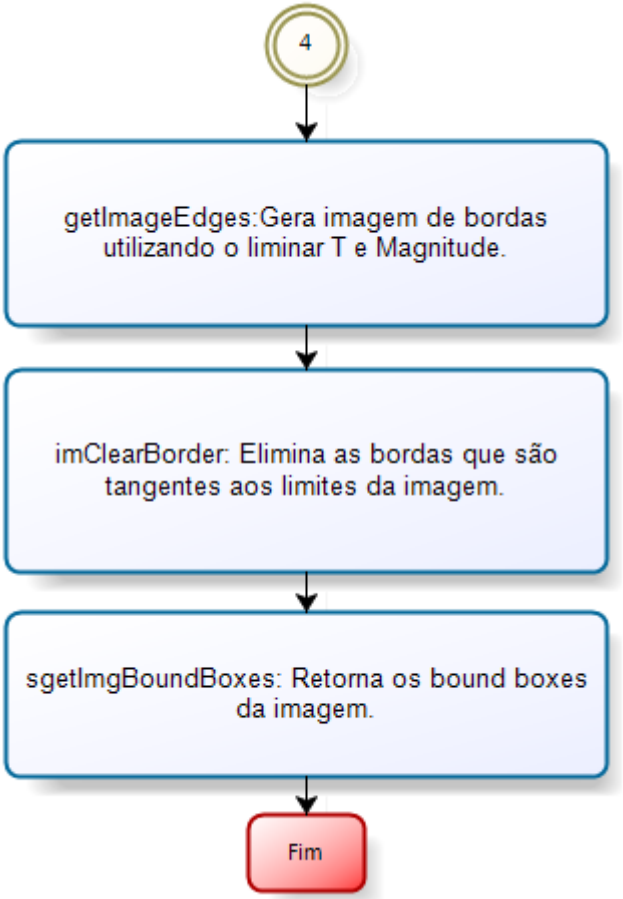


FIGURA 22 - Principais passos da execução do algoritmo para na etapa de localização (Parte 4).

Na Figura 23 existe o fluxograma da etapa de verificação apresentada na seção 2.2.

A Figura 24 mostra a etapa de extração.

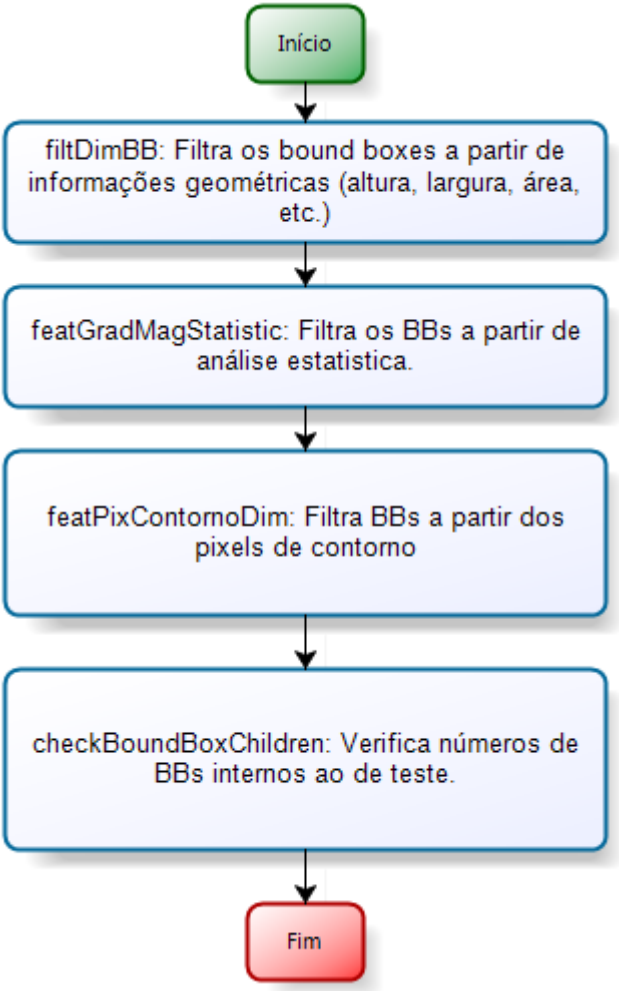


FIGURA 23 - Principais passos da execução do algoritmo na etapa de verificação.

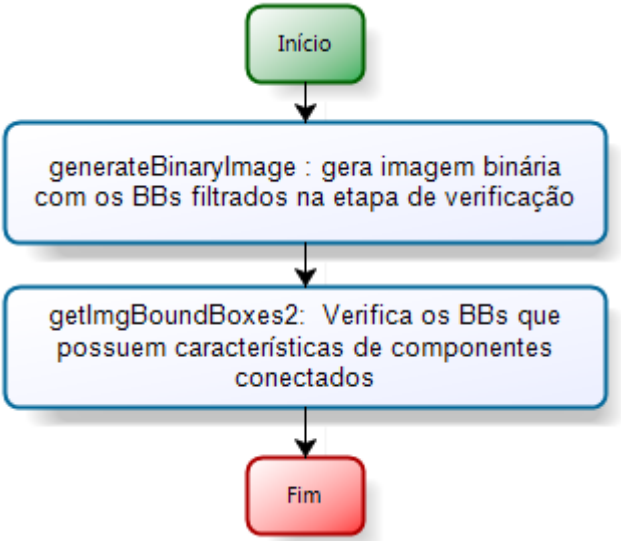


FIGURA 24 - Principais passos da execução do algoritmo na etapa de extração.

O código fonte está disponível em:
https://code.google.com/p/rec-placas-veic/

4 RESULTADOS OBTIDOS

Neste capítulo são apresentados os resultados obtidos em cada etapa do algoritmo de localização de placas veiculares.

4.1 Localização dos Caracteres

A localização dos caracteres segue o que foi descrito na seção 2.1.1.3.

A etapa de geração das imagens G_x *e* G_y, através dos filtros derivativos horizontal H_h e vertical H_v, foi efetuada através do *DSP.* utilizando funções da biblioteca de processamento de imagens. $G_{mag}(i,j)$ e $G_\theta(i,j)$ foram geradas utilizando funções do *openCV (*BRADSKI; KAEHLER, 2008*)*.

A extração dos contornos candidatos a texto foi gerada, muitas vezes reescrevendo funções do *Matlab* em linguagem C, pois não existia suporta nas bibliotecas.

Na FIGURA 25 é possível visualizar uma foto de um carro do banco de imagens de teste. Nas Figuras 26 e 27 existem imagens após a detecção de bordas candidatas a texto extraídas do *Matlab* e *OpenCV*, respectivamente.

FIGURA 25 - Imagem original.

FIGURA 26 - Detecção de bordas no *Matlab* (etapa de localização dos Caracteres).

FIGURA 27 - Detecção de bordas na *Beagleboard* (etapa de localização dos Caracteres).

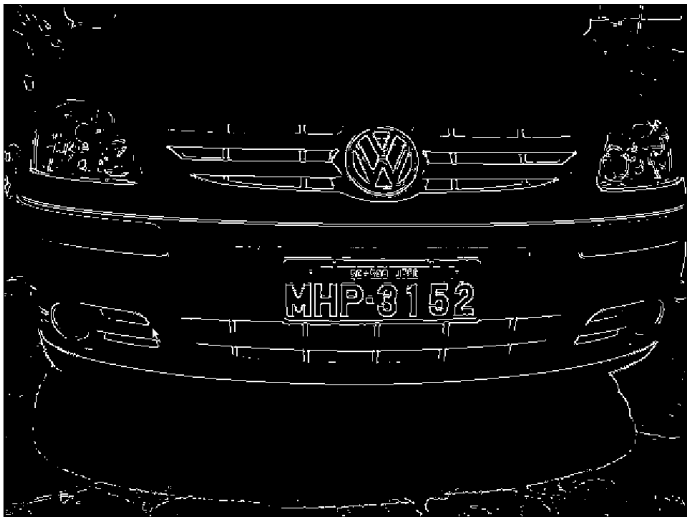


FIGURA 28 - Imagem com extração dos contornos que partem das extremidades no *Matlab* (etapa de localização dos Caracteres).

As bordas que partem das extremidades, praticamente representam sempre ruídos na imagem, desta forma sempre são descartadas como pode ser visto nas Figuras 28 e 29, extraídas do *Matlab* e *Beagleboard*, respectivamente.

Nas Figuras 30 e 31 podem ser visto os *BBs* selecionados que representam candidatos a regiões com caracteres, extraídos do *Matlab* e *Beagleboard*, respectivamente.

A geração dos BBs pôde ser gerada através de funções do próprio *OpenCV*. Embora existam outras bibliotecas que poderiam ser utilizadas em conjunto com o *OpenCV,* não houve necessidade disso.

FIGURA 29 - Imagem com extração dos contornos que partem das extremidades na Beagleboard (etapa de localização dos Caracteres).

FIGURA 30 - Imagem original com os *BBs* localizados no *Matlab* (etapa de localização dos Caracteres).

FIGURA 31 - Imagem original com os *BBs* localizados na *Beagleboard* (etapa de localização dos Caracteres*)*.

4.2 Verificação dos Caracteres

A etapa de verificação gerou um trabalho complexo, pois não existia suporte do *OpenCV* para os itens descritos na seção 2.2. Todas as verificações foram codificadas em linguagem C.

No entanto foram todas geradas e o resultado, após o processo de verificação, pode ser visto na Figura 32, proveniente do *Matlab*, e Figura 33, gerada na *Beagleboard*.

Note que mesmo assim ainda existem *BBs* que marcam regiões que não são de texto.

FIGURA 32 - Verificação dos caracteres: imagem original com os BBs após o processo de verificação dos *BBs* no *Matlab*.

FIGURA 33 - Verificação dos caracteres: imagem original com os *BBs* após o processo de verificação dos BBs na *Beagleboard*.

4.3 Extração dos Caracteres

A extração dos caracteres também foi realizada codificando e testando as funções necessárias porque não existia nenhum tipo de suporte do *OpenCV* para este fim.

O resultado final do algoritmo de extração desenvido no *Matlab* pode ser visto na Figura 30. Nesta figura é possível visualizar que ainda existem ruídos que não foram eliminados no processo de verificação.

Para que não houvesse ruídos na imagem, após o processo de extração foi realizado outro de verificação no qual é baseado no processo de localização de componentes conectados descrito na seção 2.1.1.1.

A Figura 35, gerada na *Beagleboard*, mostra a imagem extraída mostrando apenas caracteres, pois são aqueles que possuem componentes semelhantes conectados.

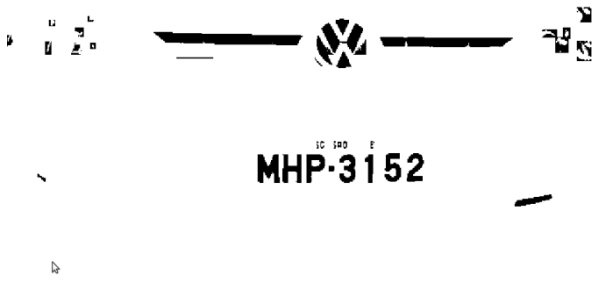


FIGURA 34 - Extração dos caracteres: após o processo de extração no *Matlab*.

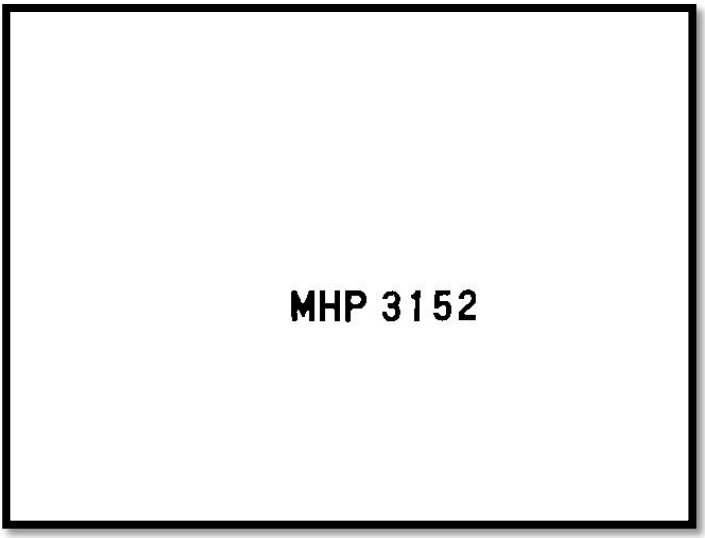


FIGURA 35 - Extração dos caracteres: após o processo de extração e validação final dos caracteres na *Beagleboard*.

4.4 Resultados do software OCR

O *software Tesseract* foi usado em linha de comando externamente à aplicação desenvolvida.

Para a imagem mostrada Figura 35, o *Tesseract* foi capaz de reconhecer as letras sem problemas. No entanto, em outras imagens não foram reconhecidas, muitas vezes sem motivo aparente. O Anexo A apresenta as imagens do banco de teste com os resultados do processo.

4.5 Tempo de execução dos algoritmos

O tempo de execução dos algoritmos a medida do tempo necessária para realizar todo o algoritmo, seguindo as etapas de localização, verificação e extração e também o *OCR.*

O tempo médio para executar todas essas aplicações na *Beagleboard* é de 15 segundos. É um tempo que pode ser alto dependendo da aplicação, mesmo para ser utilizado em uma cancela, se for necessário emitir os caracteres da placa em tempo real.

Existem alguns motivos para a aplicação tomar todo este tempo. Um dos motivos é que o algoritmo de localização foi implementado de forma a privilegiar o entendimento, e não a otimização.

4.6 Resultados do Sistema

Neste capítulo são apresentadas inúmeras imagens e será verificado o aproveitamento do sistema de forma geral.

Existe um banco de imagens (PACHECO; SANTOS, 2010) (Figura 40 até a Figura 69), que foram digitalizadas utilizando uma câmera digital com resolução de 640x400 pixels. Nestas imagens, foram aplicados os algoritmos de localização, verificação e extração apresentados anteriormente e posteriormente aplicado o *OCR*. Todo o processo foi executado na *Beagleboard*.

O resultado do processamento de vídeo pode ser visualizado no Anexo A.

Nas Tabelas 1 e 2 existe uma análise do acerto após o processo de extração. Existe uma coluna com o número da figura (Fig.) em análise, outras colunas que correspondem ao acerto das letras da placa (L. 1 – Letra 1, L. 2 – Letra 2, L. 3 – Letra 3, sendo L.1 o extremo esquerdo das letras da placa e L.3 o extremo direito), outras que correspondem aos números (N. 1 – Número 1, N. 2 – Número 2, N. 3 – Número 3, N. 4 – Número 4, sendo N. 1 o extremo esquerdo dos números da placa e N. 4 o extremo direito) e T. representa o total de acerto de cada figura. Quanto houve acerto da letra na extração o mesmo é sinalizado como “S”, caso contrário “N”.

TABELA 1 - Porcentagem dos acertos de Extração.

Fig .	L 1	L 2	L 3	N 1	N 2	N 3	N 4	T.
Fig.40	S	S	S	S	S	S	S	**100%**
Fig.41	S	S	S	S	S	S	S	**100%**
Fig.42	S	S	S	S	S	S	S	**100%**
Fig.43	S	S	S	S	S	S	S	**100%**
Fig.44	S	S	S	S	S	S	S	**100%**
Fig.45	S	S	S	S	S	S	S	**100%**
Fig.46	S	S	S	S	S	S	S	**100%**
Fig.47	S	S	S	S	S	S	N	**86%**
Fig.48	S	S	S	S	S	S	S	**100%**
Fig.49	S	S	S	S	S	S	S	**100%**
Fig.50	S	S	S	S	S	S	S	**100%**
Fig.51	S	S	S	S	S	S	S	**100%**
Fig.52	S	S	S	S	S	S	S	**100%**
Fig.53	S	S	S	S	S	S	S	**100%**
Fig.54	S	S	S	S	S	S	S	**100%**

Na Figura 36 é realizada uma análise do acerto médio geral de cara caractere da imagem. Os caracteres do meio possuem uma extração mais eficiente daqueles localizados nos extremos da placa.

TABELA 2 - Porcentagem dos acertos de Extração – Continuação.

Fig .	L 1	L 2	L 3	N 1	N 2	N 3	N 4	T.
Fig.55	S	S	S	S	S	S	S	**100%**
Fig.56	S	S	S	S	S	S	S	**100%**
Fig.57	S	S	S	S	S	S	S	**100%**
Fig.58	S	S	S	S	S	S	S	**100%**
Fig.59	N	N	N	S	S	S	S	**57%**
Fig.60	S	S	S	S	S	S	S	**100%**
Fig.61	S	S	S	S	S	S	S	**100%**
Fig.62	S	S	S	S	S	N	N	**71%**
Fig.63	N	N	S	S	S	S	S	**71%**
Fig.64	N	S	S	S	N	S	S	**71%**
Fig.65	S	S	S	S	S	S	S	**100%**
Fig.66	S	S	S	S	S	N	N	**71%**
Fig.67	N	N	S	S	S	S	S	**71%**
Fig.68	S	S	S	S	S	N	N	**71%**
Fig.69	S	S	S	S	S	S	S	**100%**
Total								**92%**

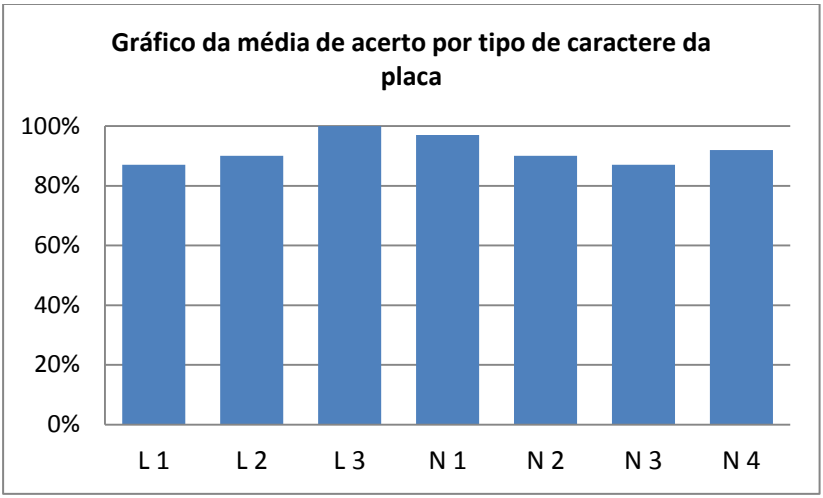


FIGURA 36 - Gráfico da média de acerto por tipo de caractere da placa.

Nas Tabelas 3 e 4 é possível visualizar o acerto do *OCR*. É comparada a numeração real da placa com a extraída do *OCR* para assim se obter a taxa de acerto. Existe uma coluna chamada de “acerto ajustado” onde a letra errada pode ser substituída sempre por outra que condiz com a real. O resultado é dividido em dois acertos, acerto médio de placas e acerto médio de caracteres.

TABELA 3 - Porcentagem de acertos do OCR.

Caracteres da Placa	Resultado OCR	Acerto Total Real	Acerto Total Ajustado	Obs.
MFV-0397	MFVU397	86%	86%	-
MHT-8913	MHT89|3	86%	100%	Substituição do | por 1
MHK-9920	MHK 9920	100%	100%	-
MEL-2608	MEL 2£O8	86%	86%	-
MBJ-2946	MBJ 22é6	71%	71%	-
MHD-2161	MHDZIBI	43%	86%	Substituição do Z por 2 e do I por 1
MGM-8239	MGM 8239	100%	100%	-
MCE-8006	MCE GOL	57%	57%	-
ATQ-1919	ATDIQIQ	29%	57%	Substituição do I por 1
MHV-7024	MHV 7024	100%	100%	-
MGB-2617	MGB 2EI7	71%	86%	Substituição do I por 1
MFN-0204	MFN |]2U4	71%	71%	-
MHP-3152	MHP 3152	100%	100%	-
HHR-6224	HHR 8224	86%	86%	-
MBI-0616	MBIUSIS	43%	57%	Substituição do I por 1

TABELA 4 - Porcentagem de acertos do OCR – Continuação.

Caracteres da Placa	Resultado OCR	Acerto Total Real	Acerto Total Ajustado	Obs.
LZM-8322	LZM 8322	100%	100%	-
MDI-0404	MDI 0404	100%	100%	-
MHF-7048	MHF 7048	100%	100%	-
CHW-2834	CHW 2834	100%	100%	-
LZY-I062	I DB2	29%	29%	-
LZG-5462	LZG 5452	100%	100%	-
MDH-5060	MDH SUEU	43%	57%	Substituição do S por 5
MGL-6188	MGLSI	43%	57%	Substituição do I por 1
IBW-3684	W 3684	71%	71%	-
MBW-3178	BH3 78	57%	57%	-
MGA-4937	MGA 4937	100%	100%	-
MGB-2014	MGB 2U	57%	57%	-
MJF-3350	F 335U	57%	57%	-
MBQ-0823	MBO G823	71%	71%	-
MEC-9301	MEC 9301	100%	100%	-
Média de acerto das placas Geral	-	**75%**	**80%**	**-**
Acerto médio das placas somente com erros de OCR	-	**-**	**90%**	**-**

Para classificar os erros do *OCR* foi realizada uma análise que pode ser vista na Figura 37. Esta análise classifica os principais motivos dos erros.

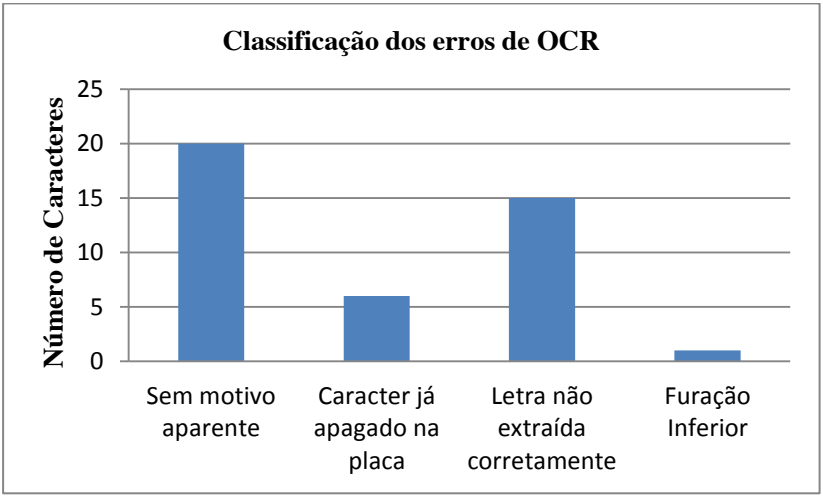


Figura 37 - Classificação dos erros de OCR

5 CONCLUSÃO

Os algoritmos de localização, verificação e extração mostraram possuir uma boa eficiência total, como pode ser visto na análise das Tabelas 1 e 2. No entanto, ainda existem pontos a serem trabalhados. Algumas placas com caracteres de boa aparência, como nas Figuras 62, 63 e 64, apresentaram extração incompleta dos caracteres da placa. Isso acontece devido ao algoritmo de seleção dos caracteres da placa do carro. Este algoritmo selecione caracteres que possuam características de componentes conectados e eventualmente não é capaz de detectar todos aqueles que fazem parte da placa.

O *OCR* apresentou uma eficiência mais baixa do que o algoritmo de extração. Na Figura 37 é possível verificar que os principais erros são classificados como “Sem motivo aparente” ou como “Letra não reconhecida”. Parra os erros classificados como “Sem motivo aparente” a remoção do efeito dos parafusos de fixação, a redução de perspectiva dos caracteres e aumentar a intensidade e a nitidez das letras pode reduzir a ineficiência para este caso. Os erros classificados como “Letra não reconhecida” são provenientes do algoritmo de seleção das letras da placa, na etapa de extração.

Embora funcional, o algoritmo, a velocidade de extração deve ser otimizada, pois atualmente são demandados quinze segundos para completar o processo. Existem gargalos de fácil otimização, como a extração das bordas que partem dos limites da imagem como na Figura 26. Só esta etapa consome até seis segundos de processamento.

A *Beagleboard* mostrou-se extremamente versátil para o desenvolvimento de aplicações embarcadas. Existe muito suporte e extenso desenvolvimento de forma geral. No entanto utilizar o *DSP* não foi uma tarefa fácil, demandou horas de pesquisa e desenvolvimento. A performance pode ser até 15 vezes mais rápida para algoritmos que possuem dados de 8 bits. No entanto, quando utilizado 32 bits não houve melhora evidente. Isso se deve ao fato de o próprio *DSP* possuir a maioria das instruções para dados de 8bits e 16bits.

REFERÊNCIAS BIBLIOGRÁFICAS

ARTH, C.; LIMBERGER, F; BISCHOF, H. Real-time license plate recognition on an embedded DSP-platform. In: IEEE CONF. ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR'07), 2007, Minneapolis. **Proceedings**... p. 1-8.

BEAGLEBOARD. *Beagleboard*. Disponível em: <http://www.Beagleboard.org>. Acesso em: 20 de jul. 2011.

BERNSEN, J. Dynamic thresholding of grey-level images. In: Proc. INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 1986, France. **Proceedings**... p. 1251–1255.

BLANCHET, G.; CHARBIT, M. **Digital Signal and Image Processing Using MATLAB** (Digital Signal and Image Processing series). 1st ed. Wiley-ISTE, 2006. 763 p.

BRADSKI, G; KAEHLER, A. **Learning OpenCV: Computer vision with the OpenCV library**. 1st ed. Sebastopol: O'Reilly, 2008. 555 p.

Carvalho , J. E. R. **Uma abordagem de Segmentação de Placas de Automóveis Baseada em Morfologia Matemática.** 2006. 101 f. Dissertação (Mestrado) - Pós Graduação em Computação, Universidade Federal Fluminense, Niterói, 2006.

CHEN, X.; YUILLE, A. Detecting and reading text in natural scenes. In: COMPUTER VISION AND PATTERN RECOGNITION, 2004**,** Washington. **Proceedings** of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2 (June-2 July 2004), II–366–II–373 Vol.2.

CLARK, P.; MIRMEHDI, M. Combining statistical measures to find image text regions. In: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION 01, 2000, Tsukuba. **Proceedings**... pp. 450–453.

CONCI, A.; CARVALHO, J. E. R.; RAUBER, T. W. A Complete System for Vehicle Plate Localization, Segmentation and

Recognition in Real Life Scene. **IEEE Latin America Transactions**, Vol. 7, No. 5, September 2009.

Freund, Y.; Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of Computer and System Sciences**, 55(1):119–139, 1995.

GONZALEZ, RAFAEL C; WOODS, RICHARD E; EDDINS, STEVEN L. **Digital Image Processing Using MATLAB**. 2nd ed. Unites States of America: Gatesmark Publishing, 2009, 827 p.

INTELLI-VISION . *License Plate Recognizer*. Disponível em: <http://www.intelli-vision.com/wp-content/uploads/2012/09/IntelliVision_LPR_12.pdf>. Acesso em: 17 de Dez. 2012.

LAGNARIÈRE, ROBERT. **OpenCV 2 Computer Vision Application Programming Cookbook**. 1st ed. Birmingham: Packt Publishing, 2011. 304 p.

LIU, Y.; GOTO, S.; IKENAGA, T. A contour-based robust algorithm for text detection in color images. **IEICE - Transactions on Information and Systems**. Japan, March 2006. E89-D, 1221–1230.

NIBLACK, W. **An Introduction to Digital Image Processing**. 1st ed. Unites States of America: PrenticeHall, 1986, 215 p.

OTSU, N. A threshold selection method from gray-level histograms. IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS 9, 1979, **Proceedings**... *62–66*

PACHECO, FERNANDO SANTANA; SANTOS, VICTOR AUGUSTO DOS. Desenvolvimento de um sistema portátil de reconhecimento de placas de veículos baseado na plataforma *Beagleboard*, 2010, Florianópolis.

SAUVOLA, J.; INEN, M. P. Adaptive document image binarization. **Pattern Recognition**, 2000. 33, 225–236.

TAHIM, A. P. N. **Localização e extração automática de textos em imagens complexas**. 2009. 179 f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de Santa Catarina, Florianópolis. 2009.

TESSERACT-OCR. *Tesseract*. Disponível em: <http://code.google.com/p/ tesseract-OCR>. Acesso em: 18 de Nov. 2011.

WIKIPÉDIA. *Haar-like Features*. Disponível em: < http://en.wikipedia.org/wiki/Haar-like_features>. Acesso em: 06 de dez. 2012.

APÊNDICE A - COMPILAÇÃO DOS MÓDULOS *DSPLINK, LOCAL POWER MANAGER E CMEM*

Os módulos *DSPLink*, *Local Power Manager* e *Cmem* são necessários para que seja possível utilizar o *DSP* em uma aplicação.

Todos foram compilados utilizando o *kernel* do Linux versão 2.6.39.

No entanto foi diagnosticado que houve uma mudança na forma de comunicação entre os módulos e o *kernel* do Linux. A chamada ***ioctl****()*, que é responsável por entradas e/ou saídas específicas de hardware como abrir o leitor de DVD por exemplo.

A mudança existiu, pois as chamadas de *ioctl*() bloqueavam o *kernel* a atender outras chamadas de *ioctl*() e com o crescente uso dessa chamada a interface foi alterada para assim ser possível atender chamadas de forma não bloqueante.

No entanto os módulos citados acima estavam com o antigo formato de *ioctl*() fazendo com que fosse impossível utilizar na versão 2.6.39. Foi cogitado mudar a versão do *kernel*, no entanto outras funcionalidades eram perdidas como a *ethernet* sobre *link USB*.

Sendo assim a interface dos módulos foi alterada manualmente para utilizar a forma nova de *ioctl*(), desta maneira não provocaram mais a grande bloqueio do Kernel (Big Kernel Lock - BKL).

A compilação destes módulos foi realizada utilizando ferramentas de software da T.I. e *GCC*.

APÊNDICE B - COMPILAÇÃO DO *KERNEL* DO LINUX E GERAÇÃO DO ESPAÇO DO USUÁRIO

O *kernel* é responsável por manipular a memória do sistema operacional e prover interface ao usuário para os diversos hardwares que existem.

É também amplamente utilizado em diversas aplicações, desde as mais simples, onde não requerem unidade de gerenciamento de memória, como é o caso do *uClinux*, e aplicações complexas, como de desktops embarcados e também sistemas operacionais como *Android*.

O *kernel* utilizado foi o do Robert Nelson (http://www.rcn-ee.com/), que um grande contribuinte para o software livre de forma geral e prove versões de *kernels* customizadas para a *Beagleboard*.

Este *kernel* customizado possuí inúmeros *patches* e customizações para que seja possível a utilização na *Beagleboard*.

Visto que o trabalho complicado de customização foi realizado é necessário apenas compilar o *kernel* utilizando o GCC.

Desta forma uma imagem é gerada e será utilizada para gerar o espaço de usuário.

O espaço do usuário, como o próprio nome diz, é uma região de uso do usuário, nesta região estão presentes aplicações do usuário, programas em geral, interface gráfica, entre outros.

Esta tarefa foi executada utilizando o *RootStock* (https://launchpad.net/project-rootstock), que é uma ferramenta para gerar o espaço do usuário de forma customiza.

É possível especificar a interface gráfica, as bibliotecas que estarão por padrão no sistema operacional, programas em geral, e assim por diante.

APÊNDICE C - OPERAÇÕES DE *BOOT*

Antes de comentar sobre as configurações *uBoot*, é de boa prática entender os estágios básicos de boot de forma sucinta. Estes estágios serão descritos a seguir.

Primeiro estágio de *Boot*

O DM3780 vem de fábrica com um código inicial que procura o ***X-Loader***. O *X-Loader* pode vir de várias fontes, porta serial, USB, cartão de memória, entre outras e é carregada na memória RAM interna do processador. O *X-Loader* é uma aplicação muito pequena faz a configuração básica de hardware para ser possível carregar o *uBoot* na SDRAM, externa ao processador.

Segundo estágio de *Boot*

Após o *X-Loader* carregar o *uBoot* inicia-se o segunda estágio de *Boot, o* qual é caracterizado pela execução do *uBoot*, que também faz configurações e hardwares e é responsável por carregar a imagem do *kernel* na memória *SDRAM* e por final executá-lo.

Mais detalhes podem ser vistos na FIGURA 38.

Configuração para alocação de memória contínua

Para que o *Cmem* possa ser utilizado é necessário que exista memória destinada apenas para o seu uso. Desta maneira o *u-Boot* deve ser configurado para que o *Kernel* não use toda a *SDRAM* disponível na placa.

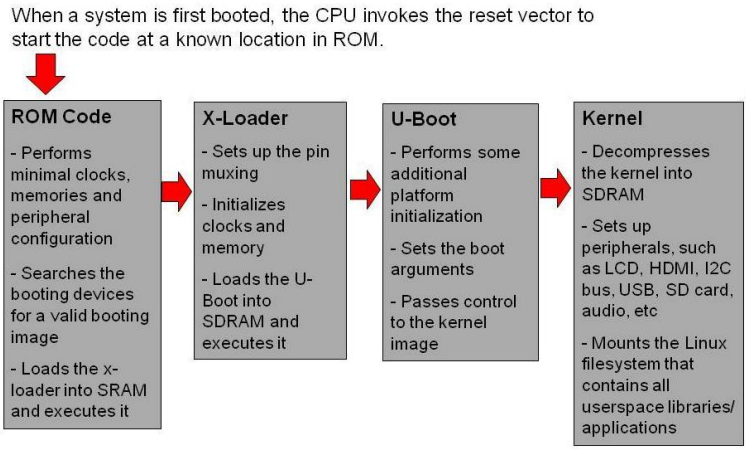


FIGURA 38 - Visão geral dos passos de *Boot*.
Fonte: <http://omappedia.org/wiki/Bootloader_Project>.
Acessado em21/10/2012.

APÊNDICE D - COMPILAÇÃO DO *OPENCV*

O *OpenCV (Open Source Computer Vision Library)* é uma ferramenta de *software* livre para a visão computacional. Possui funções para reconhecimento de facial, acompanhamento de objetos, filtros digitais, etc.

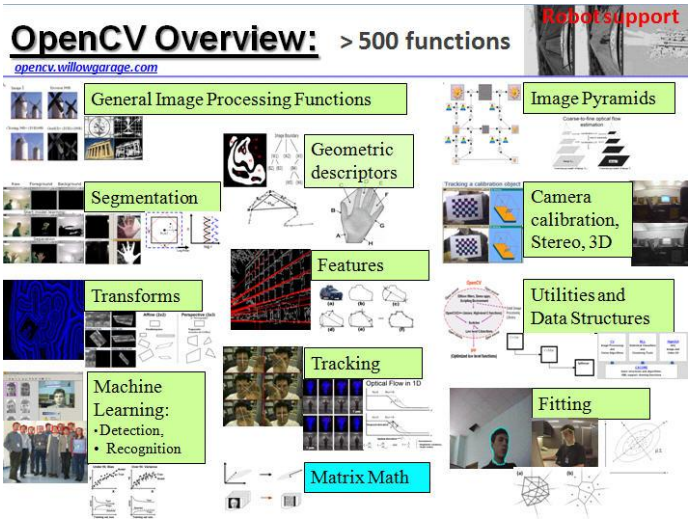


FIGURA 39 - Aplicações do OpenCV.
Fonte: <Adquirido http://opencv.willowgarage.com/wiki/ >.
Acessado em 21/10/12.

Uma grande vantagem, além de ser gratuita, é a portabilidade desta biblioteca, pois é desenvolvida para ser utilizada em multiplataforma, ou seja, em vários sistemas operacionais e também tipos de processadores (*ARM, x86*, etc.).

Para ser utilizada com na *Beagleboard*, com o *DSP*, foi necessário alterar as funções de alocação de memória. Foram alteradas para utilizar aquelas disponíveis pelo módulo *Cmem*. Desta maneira toda a alocação de memória dinâmica, aquelas que utilizam funções como *malloc*, por exemplo, são executadas pelo módulo de alocação contínua.

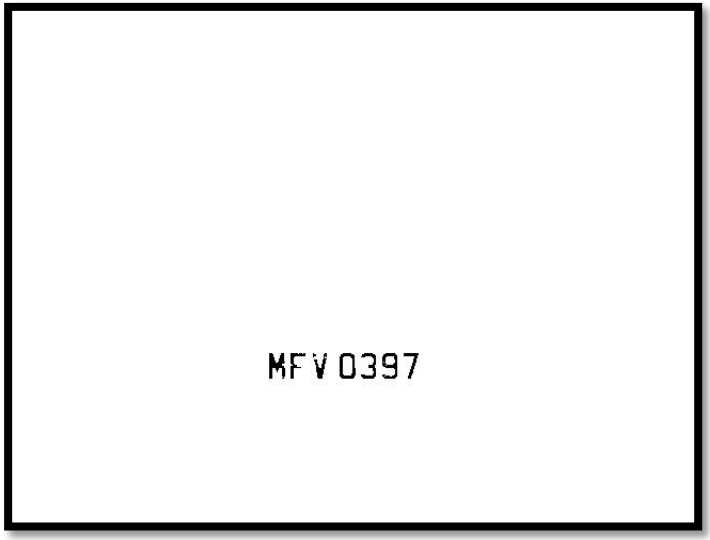
Para ser utilizado na *Beagleboard*, com o uso ou não do *DSP*, é necessário realizar o *cross-compiling* para gerar os binários compatíveis com o *ARM Cortex-A8*. No entanto o

OpenCV possuí inúmeras dependências, devido a este motivo foi executada uma compilação nativa, ou seja, foi compilado diretamente na *Beagleboard* para assim gerar as bibliotecas estáticas para serem utilizadas na aplicação. Esta tarefa foi realizada através da ferramenta ***cMake*** que um software para compilar aplicações multiplataformas.

ANEXO A – BANCO DE IMAGENS

Este anexo apresenta todas as imagens do banco de imagens (PACHECO; SANTOS, 2010), totalizando 30 fotos de veículos. É possível verificar a imagem original em (a) e o processamento de vídeo em (b), iniciando na Figura 40 e finalizando na Figura 69.

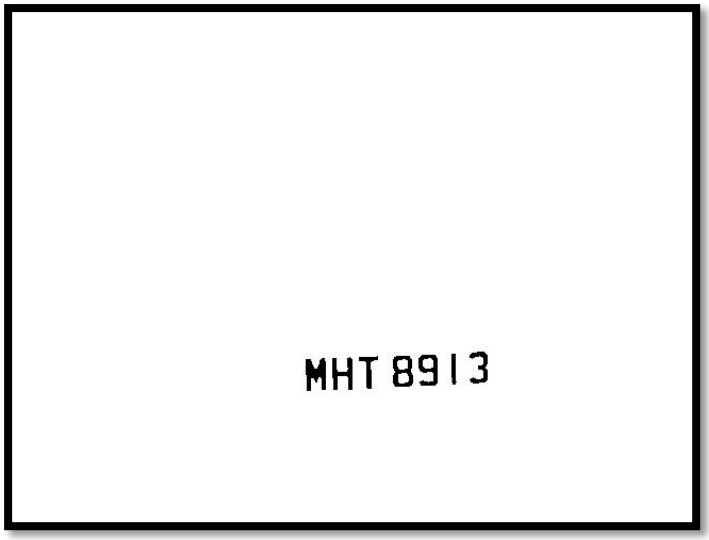
(a)



(b)

FIGURA 40 - Resultado do Sistema - Foto A. (a) Imagem original. (b) Imagem processada e extraída.

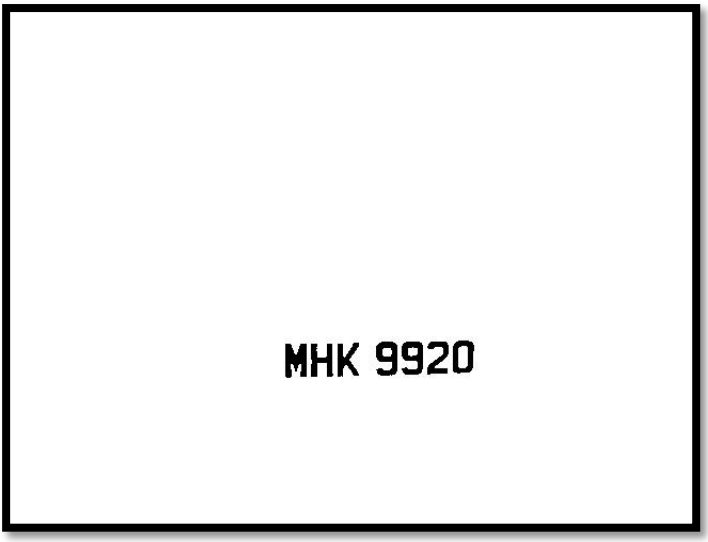
(a)



(b)

FIGURA 41 - Resultado do Sistema - Foto B. (a) Imagem original. (b) Imagem processada e extraída.

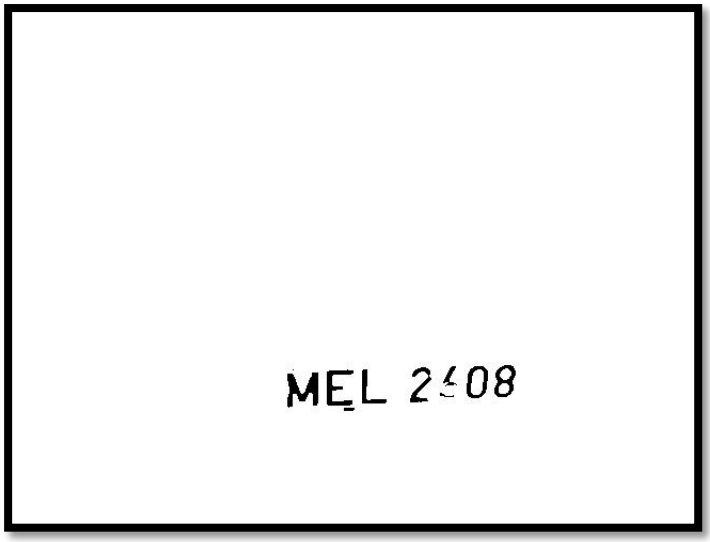
(a)



(b)

FIGURA 42 - Resultado do Sistema - Foto C. (a) Imagem original. (b) Imagem processada e extraída.

(a)



(b)

FIGURA 43 - Resultado do Sistema - Foto D. (a) Imagem original. (b) Imagem processada e extraída.

(a)

(b)

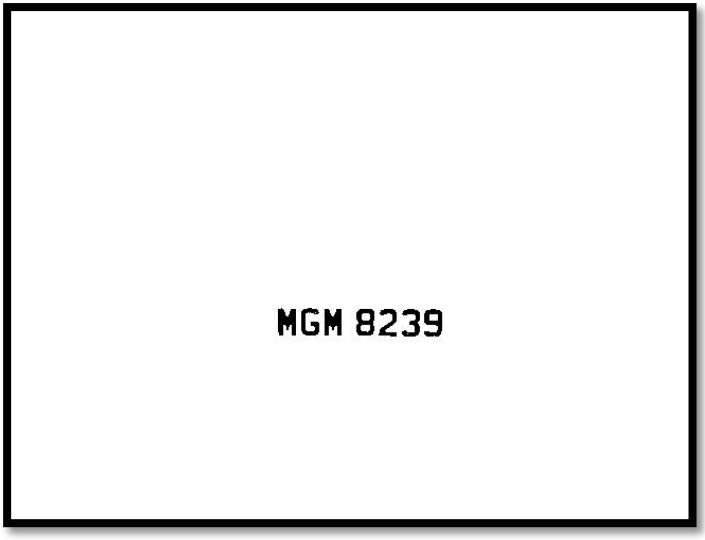
FIGURA 44 - Resultado do Sistema - Foto E. (a) Imagem original. (b) Imagem processada e extraída.

(a)

(b)

FIGURA 45 - Resultado do Sistema - Foto F. (a) Imagem original. (b) Imagem processada e extraída.

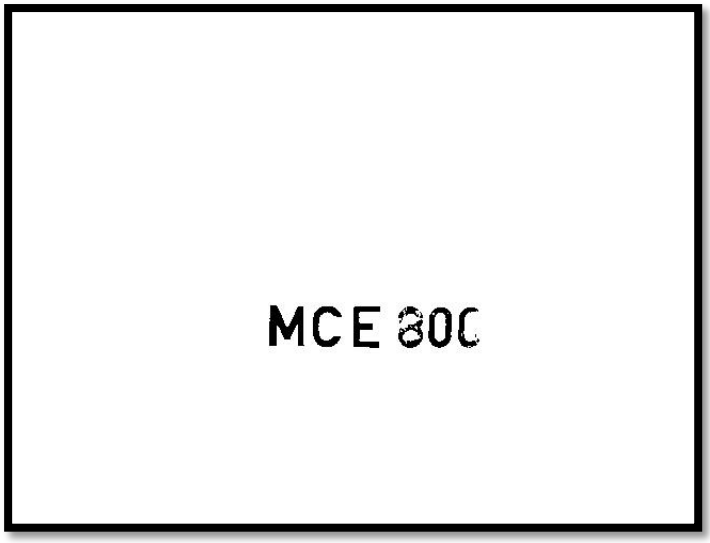
(a)



(b)

FIGURA 46 - Resultado do Sistema - Foto G. (a) Imagem original. (b) Imagem processada e extraída.

(a)



(b)

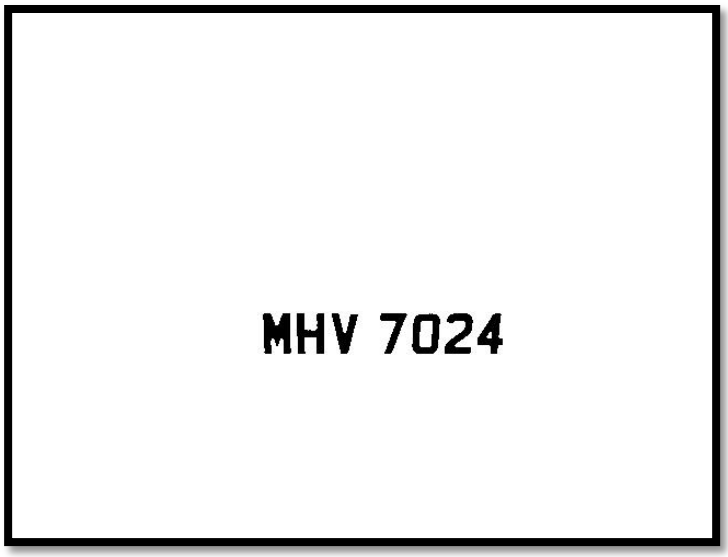
FIGURA 47 - Resultado do Sistema - Foto H. (a) Imagem original. (b) Imagem processada e extraída.

(a)

(b)

FIGURA 48 - Resultado do Sistema - Foto I. (a) Imagem original. (b) Imagem processada e extraída.

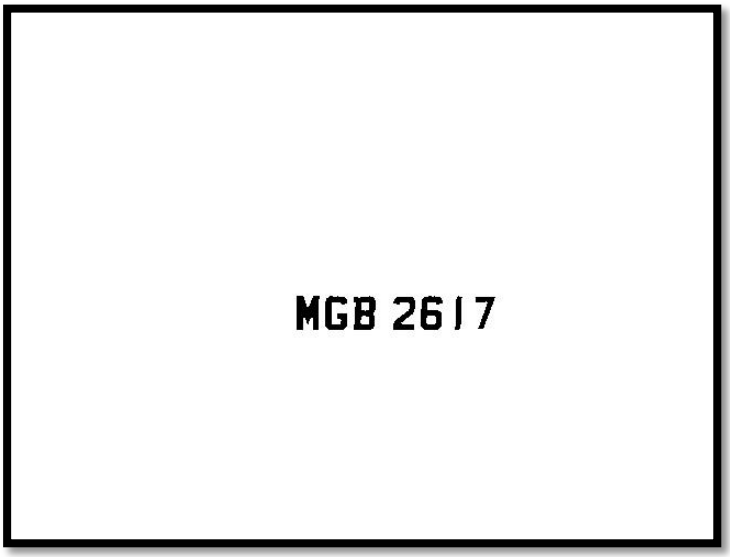
(a)



(b)

FIGURA 49 - Resultado do Sistema - Foto J. (a) Imagem original. (b) Imagem processada e extraída.

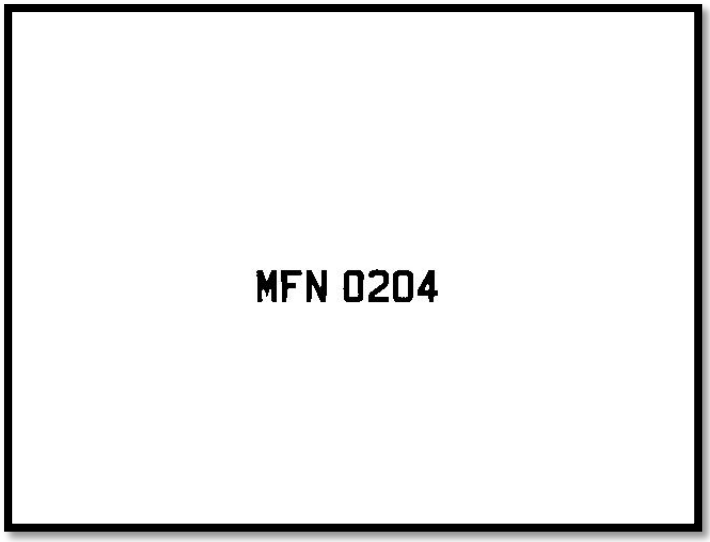
(a)



(b)

FIGURA 50 - Resultado do Sistema - Foto K. (a) Imagem original. (b) Imagem processada e extraída.

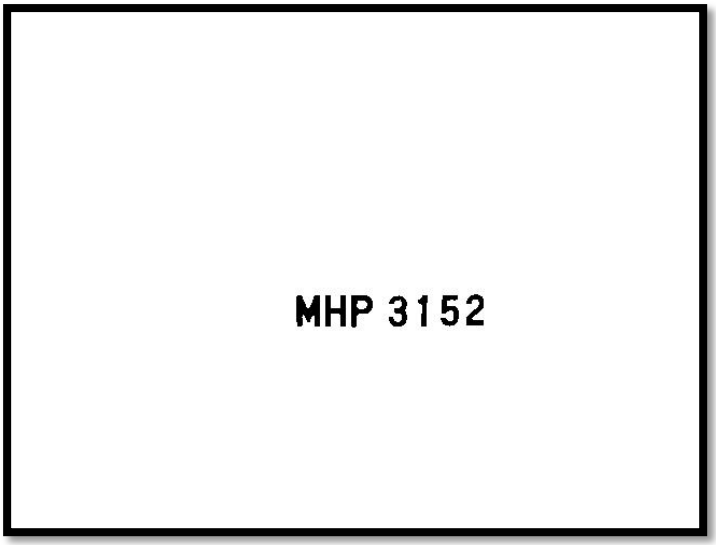
(a)



(b)

Figura 51 - Resultado do Sistema - Foto L. (a) Imagem original. (b) Imagem processada e extraída.

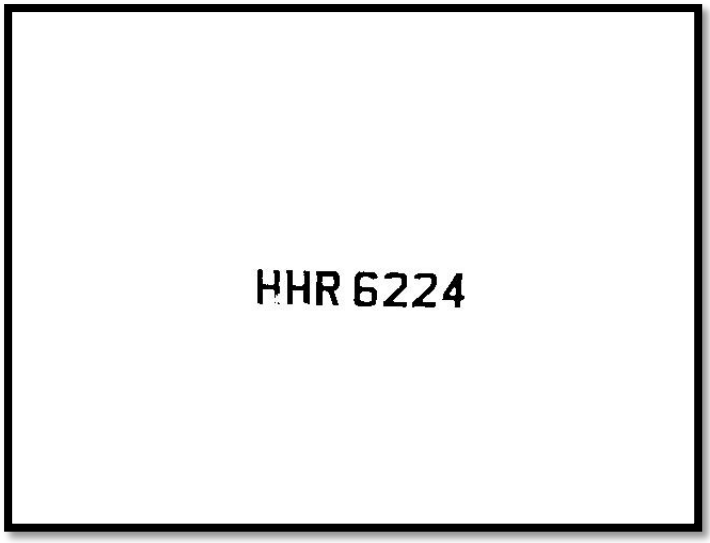
(a)



(b)

FIGURA 52 - Resultado do Sistema - Foto M. (a) Imagem original. (b) Imagem processada e extraída.

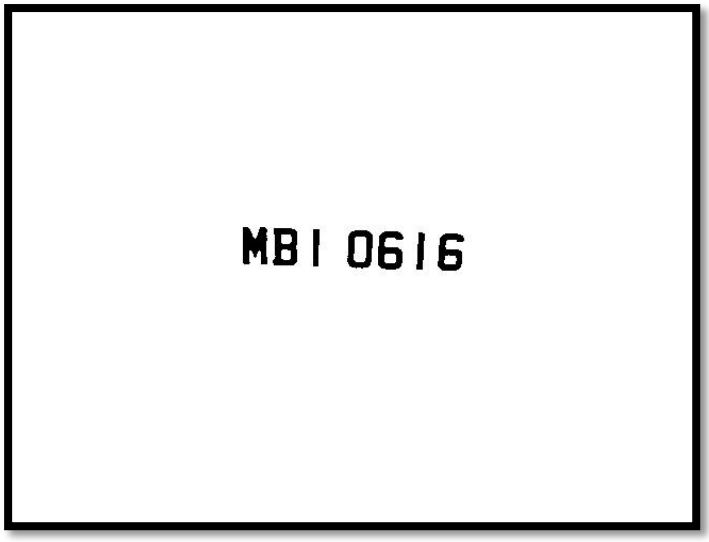
(a)



(b)

FIGURA 53 - Resultado do Sistema - Foto N. (a) Imagem original. (b) Imagem processada e extraída.

(a)



(b)

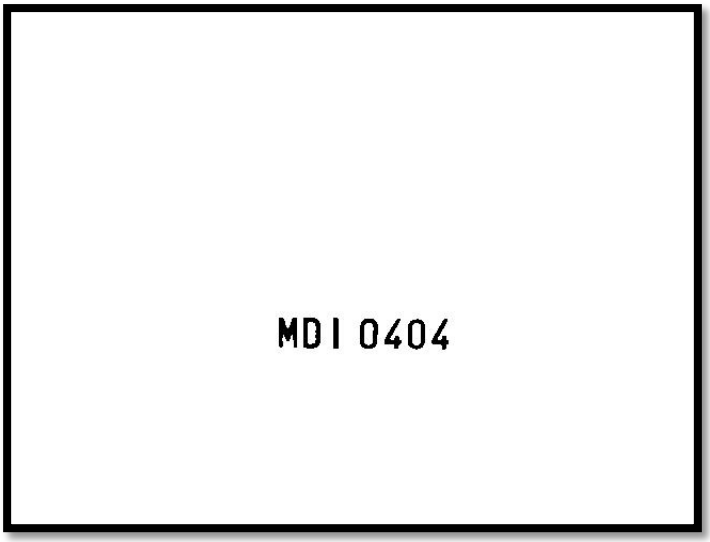
FIGURA 54 - Resultado do Sistema - Foto O. (a) Imagem original. (b) Imagem processada e extraída.

(a)

(b)

FIGURA 55 - Resultado do Sistema - Foto P. (a) Imagem original. (b) Imagem processada e extraída.

(a)



(b)

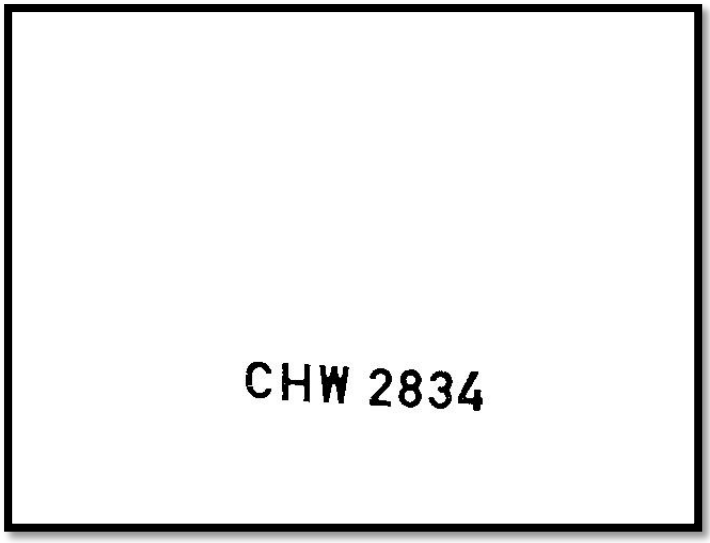
FIGURA 56 - Resultado do Sistema - Foto Q. (a) Imagem original. (b) Imagem processada e extraída.

(a)

(b)

FIGURA 57 - Resultado do Sistema - Foto R. (a) Imagem original. (b) Imagem processada e extraída.

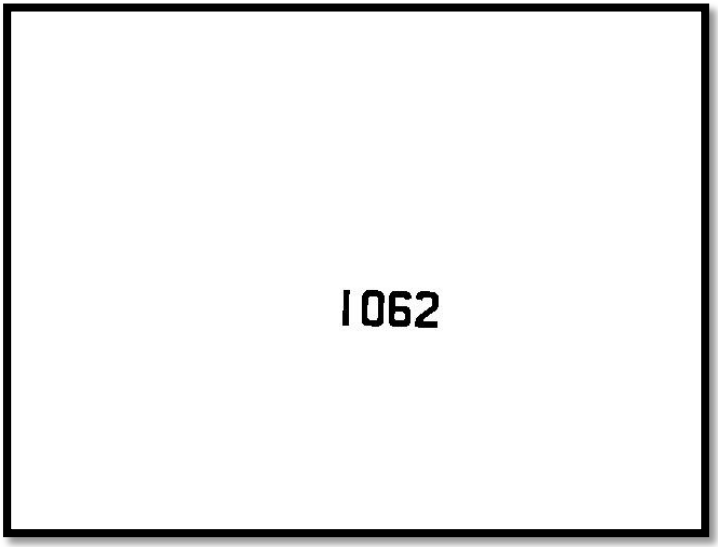
(a)



(b)

FIGURA 58 - Resultado do Sistema - Foto S. (a) Imagem original. (b) Imagem processada e extraída.

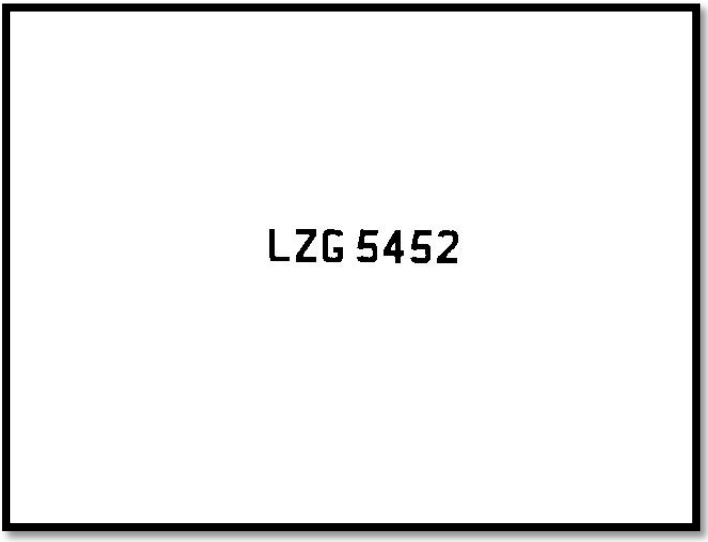
(a)



(b)

FIGURA 59 - Resultado do Sistema - Foto T. (a) Imagem original. (b) Imagem processada e extraída.

(a)



(b)

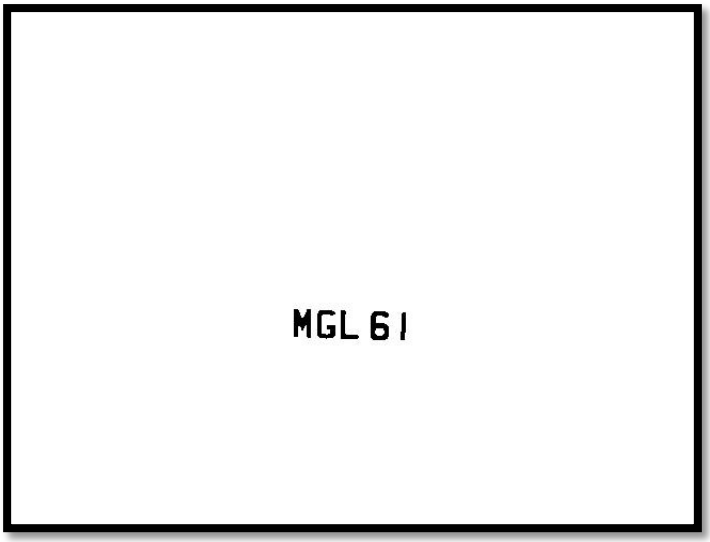
FIGURA 60 - Resultado do Sistema - Foto U. (a) Imagem original. (b) Imagem processada e extraída.

(a)

(b)

FIGURA 61 - Resultado do Sistema - Foto V. (a) Imagem original. (b) Imagem processada e extraída.

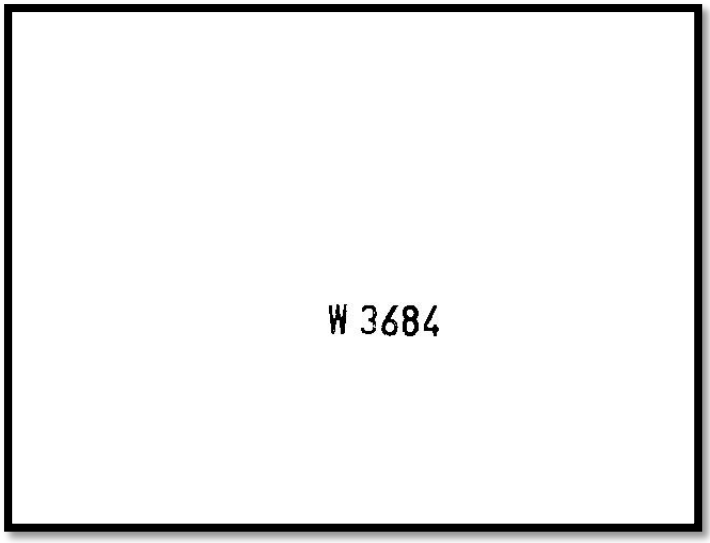
(a)



(b)

FIGURA 62 - Resultado do Sistema - Foto W. (a) Imagem original. (b) Imagem processada e extraída.

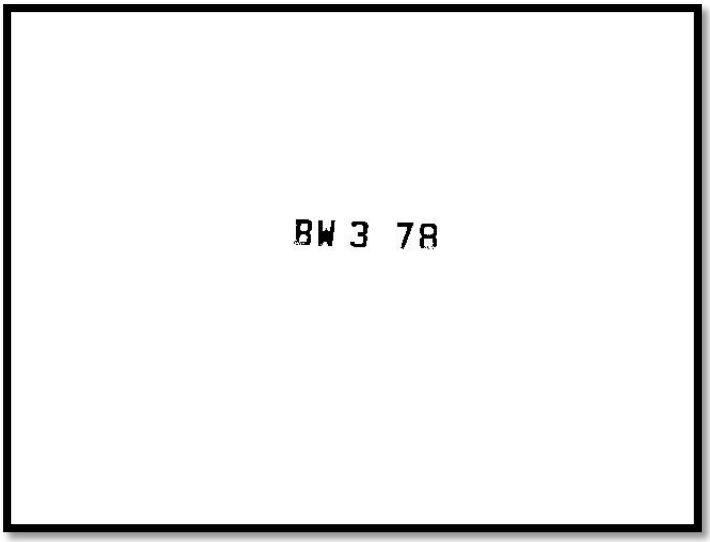
(a)



(b)

FIGURA 63 - Resultado do Sistema - Foto X. (a) Imagem original. (b) Imagem processada e extraída.

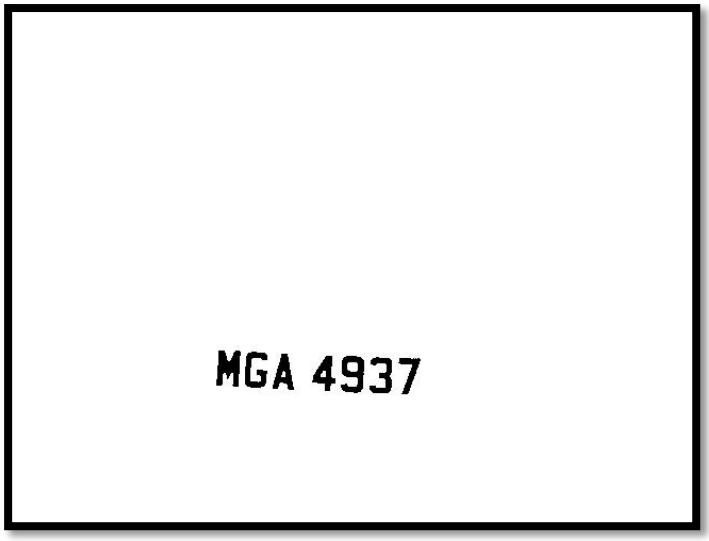
(a)



(b)

FIGURA 64 - Resultado do Sistema - Foto Y. (a) Imagem original. (b) Imagem processada e extraída.

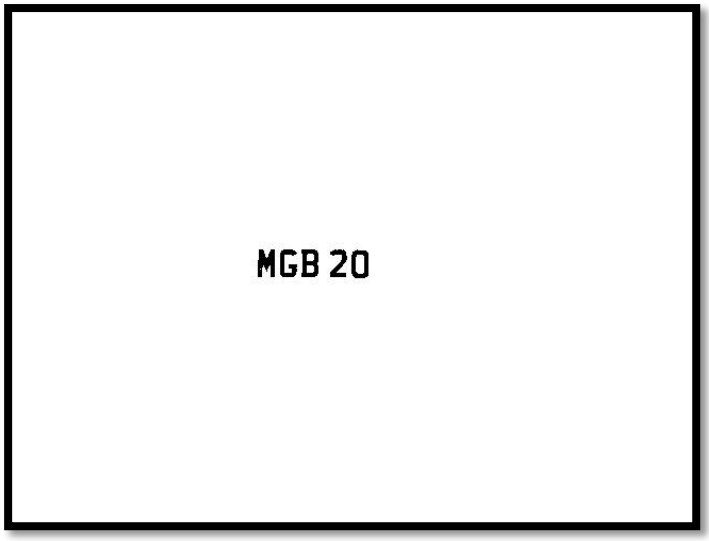
(a)



(b)

FIGURA 65 - Resultado do Sistema - Foto Z. (a) Imagem original. (b) Imagem processada e extraída.

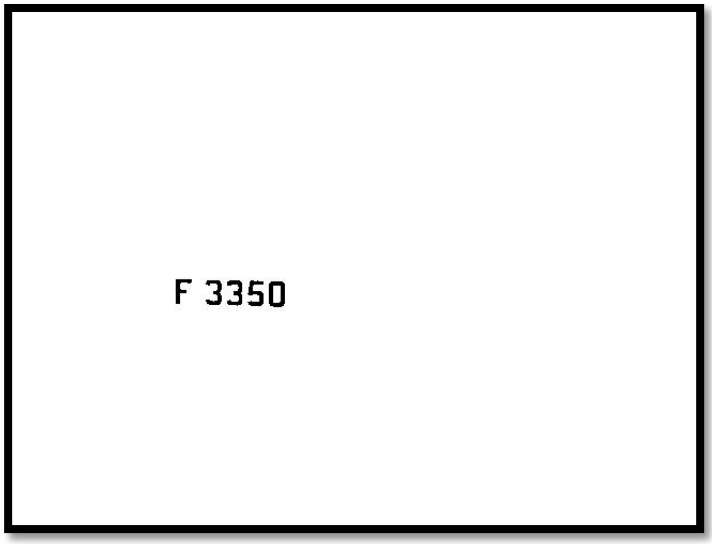
(a)



(b)

FIGURA 66 - Resultado do Sistema - Foto AA. (a) Imagem original. (b) Imagem processada e extraída.

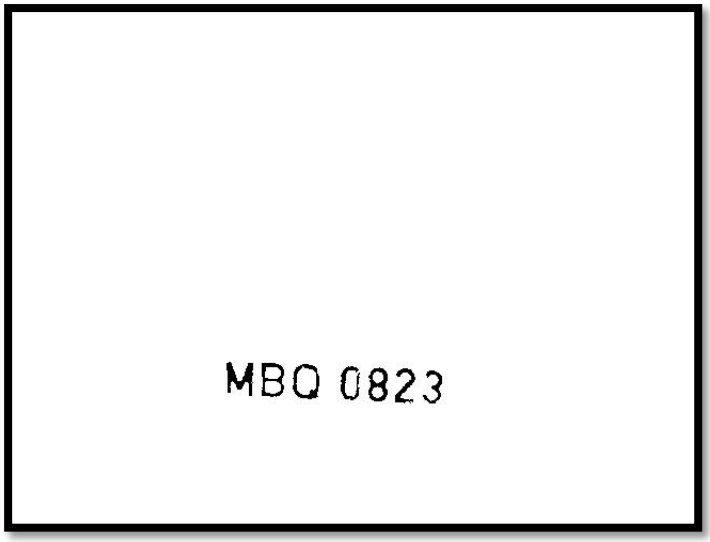
(a)



(b)

FIGURA 67 - Resultado do Sistema - Foto AB. (a) Imagem original. (b) Imagem processada e extraída.

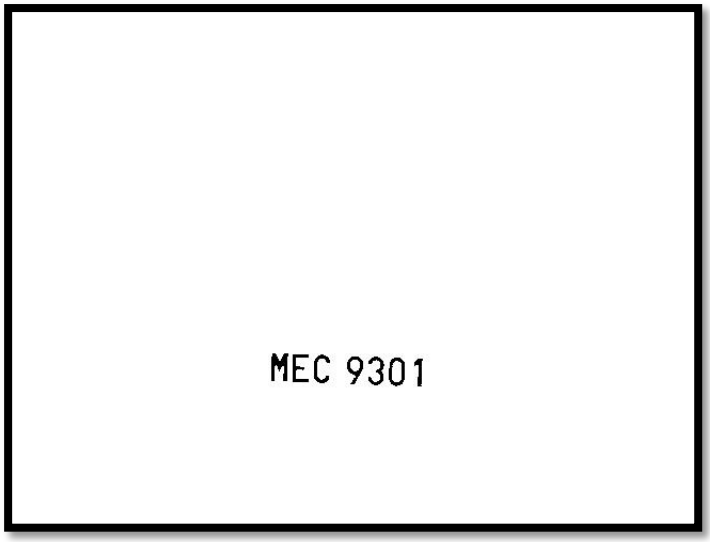
(a)



(b)

FIGURA 68 - Resultado do Sistema - Foto AC. (a) Imagem original. (b) Imagem processada e extraída.

(a)



(b)

FIGURA 69 - Resultado do Sistema - Foto AD. (a) Imagem original. (b) Imagem processada e extraída.