UNIVERSIDADE SAGRADO CORAÇÃO

FABRÍCIO DE OLIVEIRA CARAPELLI

DESENVOLVIMENTO DE ALGORTIMO PARA DETECÇÃO DE PLACAS DE VEÍCULOS UTILIZANDO PROCESSAMENTO DE IMAGENS

BAURU
2013

FABRÍCIO DE OLIVEIRA CARAPELLI

DESENVOLVIMENTO DE ALGORITMO PARA DETECÇÃO DE PLACAS DE VEÍCULOS UTILIZANDO PROCESSAMENTO DE IMAGENS

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação, sob orientação da Profª. Dra. Patricia Bellin Ribeiro.

BAURU
2013

FABRÍCIO DE OLIVEIRA CARAPELLI

DESENVOLVIMENTO DE ALGORITMO PARA DETECÇÃO DE PLACAS DE VEÍCULOS UTILIZANDO PROCESSAMENTO DE IMAGENS

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação, sob orientação da Profª. Dra. Patricia Bellin Ribeiro.

Banca examinadora:

_____

Profª. Dra. Patricia Bellin Ribeiro.
Universidade Sagrado Coração

_____

Profº. Me. Patrick Pedreira Silva
Universidade Sagrado Coração

_____

Profº. Me. Marcio Henrique Cardim
Universidade Sagrado Coração

Bauru, 20 de dezembro de 2013.

DEDICATÓRIA

Dedico este trabalho em primeiro lugar a Deus, pois Ele é o motivo de estarmos aqui, batalhando e vencendo sempre. Sem Ele nada seríamos, e nada teríamos.

Dedico este trabalho também a minha esposa que desde o início do curso sempre esteve ao meu lado me apoiando.

AGRADECIMENTOS

Gostaria de agradecer aos meus amigos Nicolas, Kleber, Vinícius, Eder, Renata, Claudia e Carlos, que desde o início do curso sempre permanecemos unidos e formando uma verdadeira equipe, ajudando uns aos outros.

Deixo aqui também os meus agradecimentos a minha orientadora Professora Dra. Patricia Bellin Ribeiro, Professor Me. Patrick Pedreira Silva e ao Professor Me. Marcio Henrique Cardim, que me ajudaram neste trabalho.

RESUMO

Este trabalho teve como objetivo desenvolver um aplicativo capaz de detectar placas de veículos. Realizar o processamento das imagens capturadas utilizando técnicas de processamento de imagens, como binarização e segmentação. Integrar ao ambiente de desenvolvimento Java as bibliotecas de processamento de imagens OpenCV e ImageJ. O algoritmo desenvolvido para localizar a placa do veículo deve um bom desempenho, visto que, o banco de imagens utilizado para teste continha 100 imagens, destas 100 imagens processadas, o algoritmo consegui localizar a placa do veículo em 70 imagens, totalizando 70% de acerto. Considerando que o objetivo principal do trabalho foi cumprido, conclui-se que o resultado do projeto é encorajador necessitando apenas de melhoria no algoritmo de localização da placa para que o mesmo consiga realizar o processamento das imagens em um menor tempo.

Palavras-chave: Processamento de imagens, Reconhecimento de caracteres, Java, JavaCV, OpenCV.

ABSTRACT

The purpose of this work was to develop an application able of detecting vehicle license plates. Undertake the processing of captured using techniques of image processing, such as binarization and segmentation images. Integrate Java development environment libraries for image processing OpenCV and ImageJ. The algorithm developed to locate the license plate should perform well since the stock images used for testing contained 100 images, 100 of these processed images, the algorithm was able to locate the license plate in 70 files, totaling 70% correct . Whereas the main objective of the work was completed, it was concluded that the project outcome is encouraging requiring only improvement in plate location algorithm so that it can perform image processing in a shorter time.

key-words: Image processing, Character recognition, Java, JavaCv, OpenCV, Java.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 – Introdução

Nas últimas décadas, é notório observar o crescimento expressivo na área de processamento de imagens. Tal fato se explica porque esta área viabiliza quase todos os campos da atividade humana. São múltiplas e distintas as aplicações no campo da medicina, robótica, meteorologia, segurança pública e privada, dentre outros.

Para realizar a localização da placa do veículo na imagem foi implementado dois algoritmos, um utilizando a biblioteca OpenCV, que não se mostrou muito eficiente quando utilizada com fotos de radares, isso devido à qualidade das imagens, então para obter um resultado melhor foi desenvolvido um algoritmo próprio que percorresse toda imagem para localizar a placa, esse algoritmo se mostrou mais eficiente porém levou um pouco mais de tempo para processar cada imagem.

Integram neste trabalho o desenvolvimento de um aplicativo capaz de localizar a placa de um veículo através de uma imagem previamente obtida utilizando técnicas de processamento digital de imagens.

Este trabalho está estruturado da seguinte forma, a saber: No capítulo 1 contem a introdução deste projeto, os objetivos e a justificativa para elaboração do mesmo. No capítulo 2 foi elaborado todo referencial teórico e tecnológico que embasa todo o projeto, já o capítulo 3 é descrito a metodologia utilizada no desenvolvimento do projeto, no capítulo 4 contém a implementação do aplicativo e análise dos resultados e por último no capítulo 5 a conclusão deste trabalho e sugestões para futuros projetos.

1.1 – Objetivos

Desenvolver um aplicativo que seja capaz de identificar a placa de automóveis obtidas através de imagens feitas por radares utilizados no controle de velocidade das vias públicas de trânsito e também imagens capturadas em tempo real por meio de uma câmera de monitoramento.

1.1.2 – Objetivo Específico

Realizar o processamento de imagens capturadas utilizando técnicas de processamento de imagens, como binarização, segmentação e outras técnicas necessárias para localizar a placa do veículo. Instalação do ambiente de desenvolvimento java, Jdk e a Ide Netbeans. Integrar as bibliotecas de processamento de imagens OpenCV e ImageJ ao ambiente de desenvolvimento Java e ao final do trabalho, verificar a eficiência do sistema em relação ao percentual de acerto.

2 – Processamento de Imagem

Uma imagem pode ser definida como uma função bidimensional, f (x, y), em que x e y são coordenadas espaciais (plano), e a amplitude de f em qualquer par de coordenadas (x, y) é chamada de intensidade ou nível de cinza da imagem nesse ponto. Assim podemos dizer que uma imagem é digital quando x, y e os valores de intensidade de f são quantidades finitas e discretas, uma imagem digital é composta de um número finito de elementos, cada um com localização e valores específicos (GONZALES et al. 2010, p. 1).

Processamento de imagem digital consiste em um conjunto de técnicas para capturar, representar e transformar imagens com o auxílio de um computador, e o emprego dessas técnicas permite extrair e identificar informações das imagens e melhorar a qualidade visual de certos aspectos estruturais, facilitando a percepção humana e a interpretação automática por meio de máquinas (PEDRINI et al. 2008, p. 1).

Para o autor Conci et al. (2008, p. 5) a visão computacional (Processamento de Imagem) nós últimos anos teve um grande desenvolvimento, é uma área que trata da extração de informações das imagens e da identificação e classificação de objetos presente na imagem, e define visão computacional como, “ o domínio da ciência da computação que estuda e aplica métodos que permitam aos computadores “compreenderem” o conteúdo de uma imagem”.

2.1 – Áreas de aplicação

Nos dias de hoje são muitas as áreas que utilizam técnicas de processamento ou análise de imagens, existe tecnologia para processar, analisar, classificar e coletar uma grande quantidade de informação a partir de imagens, sendo assim, podemos utilizar técnicas de processamento de imagens considerando diferentes origens, tamanhos e assuntos. Existem sistemas de segurança capaz de reconhecer

faces ou impressões digitais, alguns exames de sangue já podem ser feitos por meio de imagens da lâmina captadas por uma câmera acoplada ao microscópio. Também é possível classificar tipos de minerais contidos em amostras de rochas, ou ainda identificar áreas de desmatamento na Amazônia usando imagens geradas por satélites.

O crescente avanço da tecnologia digital, associado ao desenvolvimento de novos algoritmos, tem permitido um número de aplicações cada vez maior, e o autor exemplifica o uso de aplicações utilizando técnicas de processamento de imagens que incluem resolver problemas na área da medicina, biologia, automação industrial, sensoriamento remoto, astronomia, microscopia, artes, área militar, arqueologia, segurança e vigilância (PEDRINI et al. 2008, p. 2).

Conci et al. (2008, p. 5) diz que os sistemas de processamento digital de imagens são usados em reconhecimento de pessoas, de assinaturas e de objetos, inspeção de peças em linhas de montagens, orientações de robôs em indústrias automatizadas, e em muitas outras.

Hoje em dia, não existe praticamente mais nenhuma área de empreendimento técnico que não seja impactada de uma forma ou de outra pelo processamento digital de imagens. Na área médica as imagens formadas por raios gamas podem ser utilizadas para localizar doenças ósseas, como infecções ou tumores (GONZALES et al. 2010 p. 6).

A Figura 1, mostra algumas imagens utilizada pela área médica, a saber: **(a)** Escaneamento ósseo, **(b)** Tomografia por emissão de pósitrons (Pet), **(c)** Cygnus Loop[1] e a imagem **(d)** Radiação gama (ponto luminoso) de uma válvula de um reator.

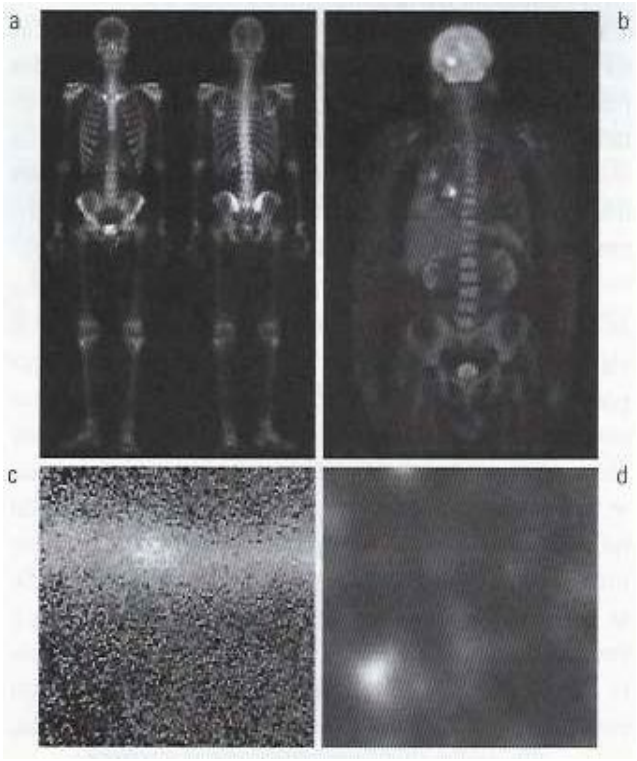

Figura 1 – Imagens utilizada na área médica
Fonte: (GONZALES et al. 2010, p. 6).

Imagens formada por raio X, são utilizadas no diagnóstico médico e também na indústria e em outras áreas como a astronomia (GONZALES, 2010, p. 6).

A Figura 2 também é formada por várias imagens: **(a)** Radiografia de tórax, **(b)** Angiograma da aorta, **(c)** Tomografia computatorizada da cabeça, **(d)** Placa de circuito impresso e a imagem **(e)** Cygnus Loop[1].

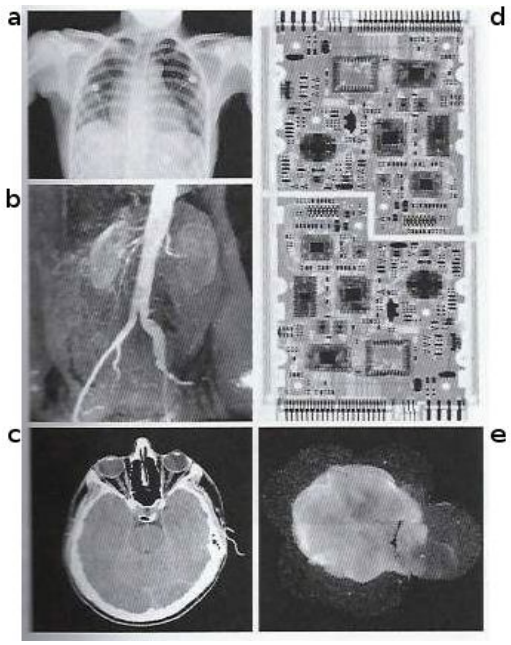

Figura 2 – Imagens utilizada na medicina, indústria e astronomia.
Fonte: (GONZALES et al. 2010, p. 7).

[1] **Cygnus Loop** – É uma Nebulosa que se localiza a aproximadamente 1.500 anos-luz de distância e é a parte remanescente de uma supernova que surgiu depois da explosão de uma estrela massiva ocorrida entre 5.000 e 8.000 anos atrás. <http://cosmonovas.blogspot.com.br/2012/03/nebulosa-cygnus-loop.html> Acessado

em 5 de Jun. de 2013.

A Figura 3 são imagens utilizadas na indústria e na área da segurança publica que representam respectivamente: **(a)** Impressão digital do polegar, **(b)** Nota de dinheiro, **(c) e (d)** Leitura automática de identificação de veículos.

Figura 3 – Imagens utilizadas na indústria e na área de segurança pública
Fonte: (GONZALES et al. 2010, p. 11)

2.2 – Etapas de um sistema de processamento de Imagens

Para o autor Pedrini et al. (2008, p. 3) um sistema de processamento digital de imagens é constituído por um conjunto de etapas, capazes de produzir um resultado a partir do domínio do problema. A seguir, são descritas resumidamente:

Aquisição de Imagens: etapa onde a imagem é capturada por meio de um dispositivo ou sensor, e assim transforma a imagem em uma representação adequada para o processamento digital, nos dias atuais existem vários dispositivos para aquisição de imagens, como câmeras de vídeo e câmeras fotográficas.

Pré processamento: no processo decorrente da aquisição ou captura da imagem podem ocorrer imperfeições ou degradação da imagem, decorrente de

iluminação inadequada ou características dos equipamento de captura, assim esta etapa visa melhorar a qualidade da imagem por meio de técnicas para redução de ruídos ou correção de contraste e brilho.

Segmentação: recorta-se a imagem em regiões disjuntas com algum significado para a aplicação, no caso do reconhecimento de caracteres de placas de veículos, é uma etapa importante. Segundo PEDRINI et al. (2008, p. 4) a etapa de segmentação realiza a extração e identificação de áreas de interesse contidas na imagem, essa etapa é geralmente baseada na detecção de descontinuidades (bordas) ou de similaridades (regiões) na imagem.

Representação e Descrição: estruturas adequadas de representação devem ser utilizadas para armazenar e manipular os objetos de interesse extraídos da imagem, o processo de descrição visa à extração de características ou propriedades que possam ser utilizadas na descrição entre classes de objetos. Essas características são, em geral, descritas por atributos numéricos que formam um vetor de características (PEDRINI et al. 2008, p. 4).

Reconhecimento e Interpretação: reconhecimento ou classificação é o processo que atribui um identificador ou rótulo aos objetos da imagem, baseado nas características providas pelos seus descritores, já o processo de interpretação consiste em atribuir um significado ao conjunto de objetos reconhecidos (PEDRINI et al. 2008, p. 4).

A Figura 4 ilustra essas etapas.

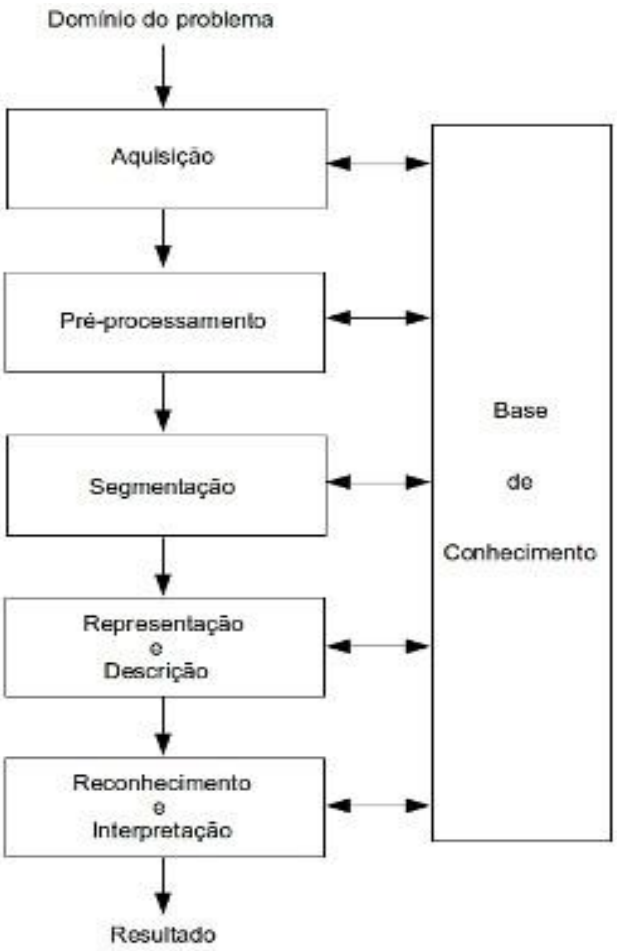

Figura 4 – etapas de um sistema de processamento digital de imagens.
Fonte: (PEDRINI et al. 2008, p. 4).

2.3 – Componentes de um sistema de processamento de imagens

Segundo Gonzales et al. (2010, p. 18) em meados da década de 1980, diversos modelos de sistemas de processamento de imagens vendido ao redor do mundo consistiam em dispositivos periféricos bastante substanciais, e era utilizado em computadores, *host*, igualmente substanciais. Já no final da década de 1980, o mercado se transferiu para um hardware de processamento de imagens na forma de uma placa única, que foi desenvolvido para ser compatível com os padrões de barramento da indústria e também para poder ser utilizados em estações de trabalho e em computadores pessoais.

Com a migração para uma placa única de processamento de imagens e a utilização de estações de trabalho e computadores pessoais, foi possível reduzir os custos de produção, essa transição também serviu como um catalisador para novas empresas especializadas no desenvolvimento de software específico para o processamento de imagens (GONZALES et al. 2008, p. 18).

Os dispositivos desempenham um papel importante em um sistema de processamento de imagens, que podem ser utilizados para aquisição, armazenamento, processamento e exibição das mesmas.

Com o crescente avanço tecnológico e a demanda de determinadas áreas de aplicação, os dispositivos tiveram uma evolução significativa nas últimas décadas. Os parâmetros de funcionalidade e desempenho dos dispositivos são dependentes, em grande parte, das áreas que os utilizam (PEDRINI et al. 2008, p. 5).

A Figura 5, ilustra os componentes de um sistema de processamento de imagens.

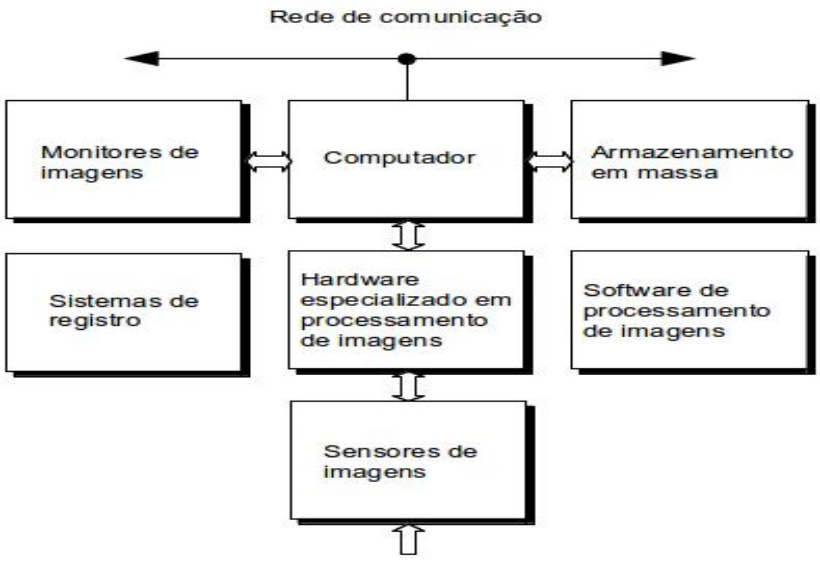

Figura 5 – Componentes de um sistema de processamento de imagens de uso geral.
Fonte: (GONZALES AT AL, 2010, p. 18)

A seguir, uma breve descrição dos componentes de um sistema de processamento de imagens para uso geral:

Sensores de imagens: dois elementos são necessários para aquisição de imagens digitais, o primeiro é um dispositivo físico sensível à energia irradiada pelo objeto cuja imagem desejamos capturar (GONZALES et al. 2010, p. 18). O segundo, chamado de digitalizador é um dispositivo utilizado para converter a saída do dispositivo físico de sensoriamento em formato digital. (PEDRINI et al. 2008, p. 5), ressalta que dentre os diversos tipos de dispositivos existentes, os mais comuns são câmeras de vídeo, tomógrafos médicos, digitalizadores (*scanners*) e satélites.

Hardware especializado: normalmente consiste em um digitalizador, além de um hardware capaz de desempenhar outras operações primárias, como uma unidade lógica e aritmética (*ALU, arithmetic logic unit*), um exemplo de como uma ALU pode ser utilizada está no cálculo da média de uma imagem, à medida que esta

é digitalizada, com o propósito de redução de ruídos (GONZALES et al. 2010, p. 18).

Computador: o computador em um sistema de processamento de imagens é uma máquina de uso geral, que pode variar de um computador pessoal a um supercomputador, para uso geral um computador bem equipado é o suficiente, mas quando queremos atingir um alto nível de desempenho, precisamos de um supercomputador (GONZALES et al. 2010, p. 19).

Software de processamento: os programas para processamento de imagens consistem em rotinas ou módulos específicos para realizar uma determinada tarefa. Bibliotecas podem ser integradas em códigos desenvolvidos por usuários ou em pacotes sofisticados de processamento de imagens (PEDRINI et al. 2008, p. 7).

Armazenamento em massa: a capacidade de armazenamento em massa é indispensável em aplicações de processamento de imagens, as mesmas requerem alta capacidade de armazenamento, por exemplo, uma imagem colorida de 1024 x 1024 *pixe*l, cada *pixel* representado por 24 *bits*, requer 3 *Mbytes* para seu armazenamento sem compressão. Um vídeo com duração de 1 minuto, formado por imagens de 512 x 512, exibidas a uma taxa de 30 imagens por segundo, cada pixel representando 24 bits, requer aproximadamente 1.4 *Gbytes* para seu armazenamento (PEDRINi et al. 2008, p. 5).

Monitores de imagem: os monitores de vídeos são os principais dispositivos de saída utilizado em sistema de processamento de imagens. os monitores de imagem utilizados hoje em dia são, em sua maioria, monitores de TV em cores (preferencialmente de tela plana) , e são controlados pelas placas de vídeo (gráficas ou de imagens), que são parte integral de um sistema computacional (GONZALES et al. 2010, p. 19).

Sistemas de registro: o sistema de registro para as imagens incluem impressoras a laser, filmes fotográficos, impressora térmica, impressoras a jato de tinta e mídias digitais, como os discos óticos e de CD-ROM. O filme proporciona a mais alta resolução possível, mas o papel é o meio preferido para o material escrito. A reprodução fotográfica possui alta qualidade. Outra possibilidade é o uso de papel

sensível à temperatura, muito difundido em equipamentos de fax (PEDRINI et al. 2008, p. 6).

Rede de comunicação: a rede de comunicação é quase um componente padrão de qualquer sistema computacional em uso hoje em dia. Em razão do grande volume de dados inerentes às aplicações de processamento de imagens, a principal preocupação na transmissão de imagens é a largura de banda. Em redes dedicadas, isso normalmente não constitui um problema, mas as comunicações com sites remotos pela Internet nem sempre são eficientes. Felizmente, essa situação está melhorando rapidamente como resultado do advento da fibra ótica e de outras tecnologias de banda larga (GONZALES et al. 2010, p. 19).

2.4 – Filtros

Os filtros aplicados nas imagens objetiva, principalmente, melhorar a qualidade da imagem através de processos que compreende, ampliação de contraste, eliminação de padrões periódicos ou aleatórios (ruídos ou imperfeições das imagens provenientes do processo de aquisição) e melhoria no seu foco e acentuação de características. É necessário ter atenção em relação aos valores resultante do processo de filtragem, pois os mesmos podem ser negativos ou estar acima do valor de tom máximo da imagem, podendo simplesmente deixar de ser representados ou ser representados por valores inadequados, provocando *overflow* ou saturação da imagem (CONCI et al. 2008, p. 5)

2.4.1 – Ruídos

Aquisição da imagem, seja ela por radar ou adquirida em tempo real por uma câmera, podem sofrem várias interferências por diversos fatores, e essas interferências são chamadas de ruídos, e podem causar sua inutilização no processo de localização da placa.

Gonzales et al. (2010, p. 205) diz que as principais fontes de ruídos em imagens digitais são causados na sua aquisição por diversos fatores, como condições ambientais, iluminação, afetam o desempenho dos sensores de aquisição e até mesmo a qualidade dos equipamentos podem causar ruídos na imagem.

Para o autor Nascimento (2012, p.41) o processo de remoção de ruídos deve ser realizado de uma maneira eficiente:

> […] o ruído deve ser atenuado nas regiões homogêneas da imagem, mas as bordas não devem ser suavizadas. Para tal, é importante que se conheça a localização das bordas. Por outro lado, a detecção das bordas requer uma imagem filtrada, pois a detecção das mesmas em imagens ruidosas apresenta resultados errôneos.

2.4.2 – Detecção de bordas

Neste projeto detectar as bordas da placa corretamente é fundamental para localização da mesma na imagem, existem vários métodos para detecção de borda, mas neste projeto será apresentado somente o operador de Canny.

“Uma borda é o limite ou fronteira entre duas regiões com propriedades relativamente distintas de nível de cinza” (Pedrini et al. 2008, p. 153).

2.4.3 – Operador Canny

Para realizar a detecção de borda, o operador Canny inicialmente suaviza a imagem por meio de um filtro Gaussiano, em seguida, a magnitude e a direção do gradiente são calculadas utilizando aproximações baseadas em diferenças finitas para as derivadas parciais. Após esse cálculo, a borda é localizada tomando-se apenas os pontos cuja magnitude seja localmente máxima na direção do gradiente (Pedrini et al. 2008, p.166).

Mas as bordas podem ainda conter fragmentos espúrios causados pela presença de ruídos, para resolver esse problema Canny usa dois limiares diferentes T^1 e T^2, com $T^2 > T^1$, essa etapa é chamada de limiarização com histerese, se o

ponto da borda que possuem gradiente maior que T^2 são mantidos como pontos de borda, qualquer ponto conectado a esses pontos da borda é considerado como pontos de borda se a magnitude de seu gradiente for mair que T^1 (PEDRINI et al. 2008, p. 166).

O operador de Canny baseia-se em três objetivos básicos, baixa taxa de erro, os pontos de bordas devem estar bem localizados e retornar um único ponto de borda(Gonzales et al. 2010, p. 474)

A Figura 6 na próxima página, mostra um exemplo de aplicação do operador de Canny.

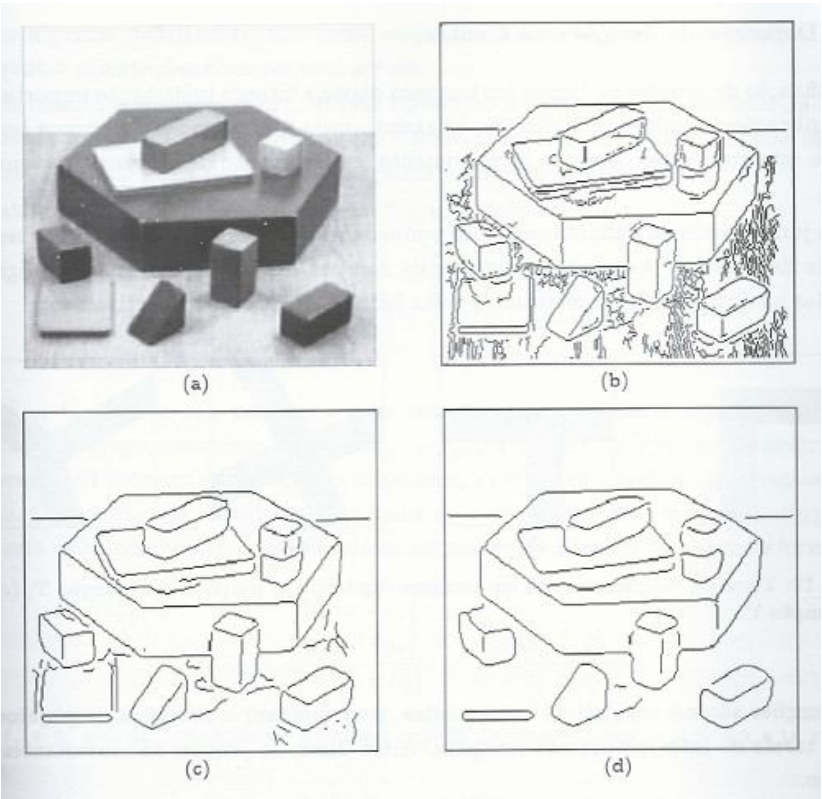

Figura 6 – Exemplo de aplicação do operador de Canny, a saber: (a) imagem original; (b) □ = 0.5; (c) □ = 1.0; e (d) □ 2.0.
Fonte: (PEDRINI et al. 2010, p. 167)

2.5 – Operações de Erosão e Dilatação da imagem

Duas operações simples, mas são fundamentas para o processo de segmentação das letras na placa do veículo.

Dilatação: é aplicado um elemento estruturante sobre um conjunto definido de *pixels* (brancos e pretos) sobre a imagem original, de maneira que o elemento estruturante adicione informação sobre a vizinhança deste ponto, ou seja, a dilatação é o processo que “aumenta” ou “engrossa” os objetos em uma imagem binária (GONZALES et al. 2010, p.419). Assim, para cada pixel preto, se o número de vizinhos brancos for maior do que um valor limiar, o pixel é invertido.

Erosão: a operação de erosão da imagem é o inverso da dilatação, ocorre a aplicação de um elemento estruturante sobre um conjunto de *pixels* (brancos e pretos), porém ao invés do elemento adicionar informação, o mesmo retira informação do objeto. De forma que os objetos pequenos tendem a ser eliminados e os objetos maiores terão suas áreas reduzidas.

2.6 – Bibliotecas

A seguir uma breve introdução sobre as bibliotecas utilizadas na construção do aplicativo.

2.6.1 – OpenCV

OpenCV é uma biblioteca de visão computacional de código fonte aberto distribuída sob BSD-licensed.

Foi projetada para o desenvolvimento de aplicações que são executadas em tempo real, mantendo a eficiência computacional. E tem como um de seus objetivos, fornecer uma infraestrutura de visão computacional para ajudar pessoas a construir aplicações bastante sofisticadas e com processamento rápido. Nela estão contidas mais de 500 funções, as quais abrangem muitas áreas na visão computacional, incluindo inspeção de produtos industriais, imagem para avaliação médica, segurança, interface de usuário, calibração de câmera, visão estéreo e robótica. Em seu desenvolvimento foram utilizadas as linguagens C e C++, mas já possui compatibilidade com outras linguagens como Python, Ruby, Matlab, Java e outras linguagens. Também pode ser utilizada em diversos sistemas operacionais como Linux, Windows, Mac OS X e Android (FEITOSA 2012, p. 41).

2.6.2 – JavaCV

O JavaCV foi projetado para utilizar a maior parte das funções do OpenCV, mas sendo executado por uma codificação desenvolvida em Java, exatamente da mesma forma como a biblioteca OpenCV funciona em códigos C/C++. Também é compatível com a linguagem utilizada pelo sistema operacional Android, a biblioteca possui suporte à aceleração de hardware para exibição de imagem em tela cheia e também inclui métodos para executar código em paralelo em múltiplos núcleos, permite calibração geométrica e de cores em câmeras e projetores, entre outras funcionalidades (FEITOSA 2012, p. 41).

2.7 - Java

Java é a base de praticamente todos os tipos de aplicativos em rede, e é o padrão global para desenvolvimento e fornecimento de aplicativos para celular, jogos, conteúdo on-line e software corporativo. Com mais de 9 milhões de desenvolvedores em todo o mundo, o Java permite desenvolver e implantar

aplicativos e serviços incríveis de maneira eficiente. Com ferramentas abrangentes, um ecossistema sólido e um desempenho eficiente, o Java oferece a portabilidade de aplicativos mesmo entre os ambientes computacionais mais diferentes.

Algumas vantagens:

Independência de plataforma: O Java é executado na maioria dos hardware e plataformas de sistema operacional principais com software JVM diretamente da Oracle, por meio de um dos muitos parceiros no ecossistema Java ou como parte da comunidade *OpenJDK.*

Alto desempenho: *HotSpot* e *Jrockit são exemplos de tecnologias* de máquina virtual imediatas comprovadas que fazem do Java um dos ambientes de programação mais rápidos, otimizações incorporadas para ambientes *multithread* o deixam ainda mais rápido.

Fácil de aprender: Java é a linguagem de programação escolhida em universidades e instituições de ensino em todo o mundo. O modelo do Java para gerenciamento de memória, *multithreading* e tratamento de exceção fazem dessa uma linguagem eficiente para desenvolvedores tanto novatos quanto experientes.

Com base em padrões: A linguagem Java e a tecnologia relacionada evoluem por meio do Java *Community Process*, um mecanismo para o desenvolvimento de especificações técnicas para tecnologia Java.

3 – Metodologia

Neste capítulo será detalhado o desenvolvimento do aplicativo, bem como a instalação e configuração das ferramentas utilizadas para localização da placa veicular e o reconhecimento dos caracteres.

Foi utilizada a linguagem de programação Java em razão das vantagens apresentada no capítulo anterior como: independência de plataforma, alto desempenho e menor curva de aprendizado.

Conforme proposta do projeto segue abaixo descrição das etapas do aplicativo:

Aquisição da imagem: o aplicativo foi construído de forma que as imagens possam ser adquiridas de duas formas, em tempo real via webcam ou qualquer outro dispositivo que possa capturar a imagem em tempo real, ou ainda utilizar imagens capturas previamente; por exemplo, imagens feitas por radares moveis ou estáticos.

Pré-processamento: nessa etapa são utilizadas algumas técnicas para tratamento da imagem, melhorando suas condições antes da localização do objeto. São utilizadas técnicas de detecção de bordas, conversão de formato, RGB (com três canais de cores – vermelho-verde-azul) para escala de cinza de um único canal e binarização.

Localização da placa: a localização da placa é realizada de duas maneiras, quando a imagem é obtida em tempo real, são localizados os contornos da imagem através dos métodos *cvFindContours, CvPoint e o cvRectangle* que gera um retângulo vermelho em torno do objeto localizado. Agora, para as imagens previamente capturadas a localização é realizada percorrendo toda imagem com um templete com as dimensões aproximada do objeto.

Validação: para realizar a validação do objeto localizado, tanto nas imagens capturadas em tempo real, como nas imagens previamente capturadas, é utilizado parâmetros com as dimensões aproximadas da placa, como altura e largura.

Segmentação: nesta etapa é realizada processos de morfologia matemática, como erosão e dilatação da imagem a fim de remover as áreas de ruídos presente na imagem, após estes processos é realizado a segmentação da imagem. Para segmentar a imagem foi criado um algoritmo que varre a imagem procurando os intervalos entre os objetos, quando esses intervalos são localizados as coordenadas são armazenadas em um vetor para serem utilizada no recorte do objeto.

3.1 – Cenário e ferramentas utilizadas

Para realizar o desenvolvimento deste aplicativo, foi utilizado um notebook da marca Dell com as seguintes configurações:

Processador intel core i5 de terceira geração, memória ram de 4gb, placa de vídeo ATI HD Radeon Série 7500M, monitor de t4" led, disco rígido híbrido sendo: ssd 32GB e um convencional de 500gb e sistema operacional Ubuntu Desktop 13.04 de 64 bits.

3.1.1 – Ferramentas utilizadas

Como já mencionado acima, o sistema operacional utilizado para desenvolvimento do aplicativo foi o Linux – Ubuntu 13.04 de 64 bi juntamente com o ambiente de desenvolvimento java, no Apêndice D será mostrado o processo de instalação das ferramentas utilizadas, partindo do princípio que o sistema operacional já esteja instalado e configurado (funcionando corretamente).

4 – Resultados

Neste item será descrito os procedimentos mais importantes realizados para a implementação do aplicativo proposto e os resultados obtidos neste trabalho. No apêndice B, contém o código fonte completo do aplicativo, no apêndice C, contém o *script* para instalação do OpenCV no Ubuntu 13.04 64Bits.

4.1 – Implementação do aplicativo

A interface do aplicativo é composto por três janelas de imagens, onde são exibidas as imagens em diferentes etapas do processamento, desde a sua captura até sua segmentação. Possui uma tabela onde após a imagem ser processada são inseridos os dados coletado nas diversas etapas do aplicativo, por exemplo, nome do arquivo quando a imagem já existir ou no lugar do nome arquivo o aplicativo coloca no nome do dispositivo de captura (webcam), coordenadas encontradas na etapa de localização e validação do objeto e a data e a hora que o arquivo foi processado.

A Figura 7 mostra a interface do aplicativo.

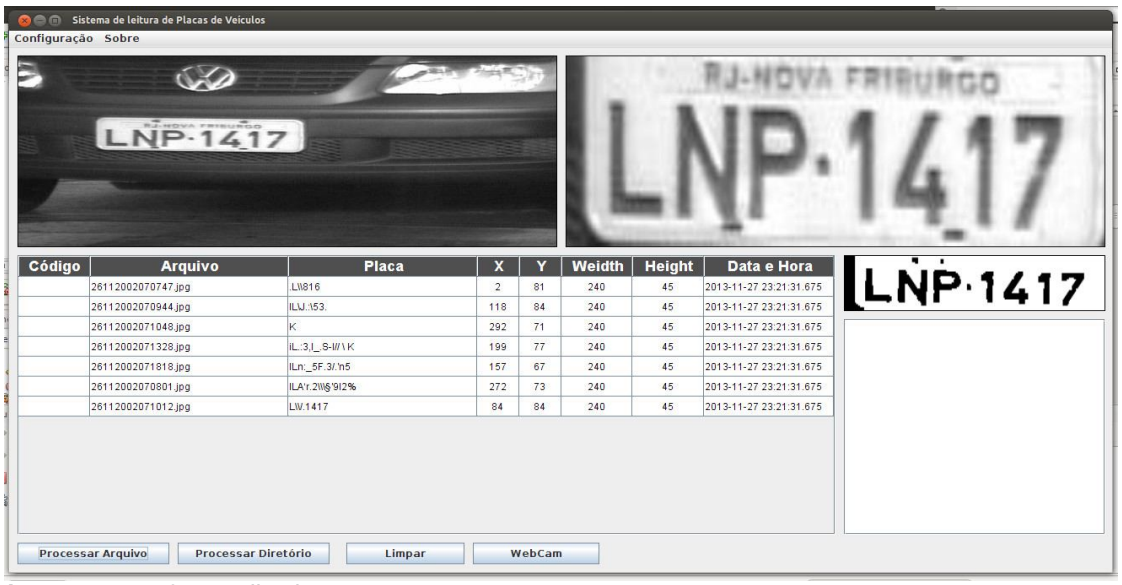

Figura 7 – Interface aplicativo.
Fonte: Autor.

A captura da imagem pode ser realizada de duas maneiras, a primeira é via *webcam* ou qualquer outro dispositivo de captura de vídeo instalado e configurado no aplicativo, a outra maneira é entrar com uma imagem previamente capturada e armazenada no computador.

A Figura 8 mostra o trecho de código responsável por capturar a imagem via *webcam* e passar os *frames capturados via parâmetros para serem processados.*

```
try {
  // inícia a captura do video via webcam
  grabber.start();

  while (wCam) {
     // configura as dimensões da janela que mostra o video em tempo real
    canvas.setCanvasSize(grabber.getImageWidth(), grabber.getImageHeight());

     // atribui o vídeo capturado para variável img (IplImage - OpenCv)
     img = grabber.grab();

     // verifica se a imagem capturada não é nula
     if (img != null) {

        // método CvFlip realiza o espelhamento da imagem
        //cvFlip(img, img, 1);

                  // cria uma cópia da imagem na memória
        imgCopy = img.clone();

        // o método showImage é responsável por apresentar a imagem capturada na tela
       // e o método findPlate é responsável por realizar o pré-processamento da imagem
        canvas.showImage(findPlate(img));
     }
```

Figura 8 - Trecho de código responsável pela captura imagem via *webcam.*
Fonte: Autor

O pré-processamento da imagem é realizado em dois métodos distintos, um é responsável por preprocessar as imagens capturadas em tempo real via *webcam* public IplImage findPlate(IplImage img), o segundo método é responsável por realizar o pré-processamento das imagens adquiridas previamente e armazenadas no computador public searchPlate(IplImage img).

Figura 9 mostra o trecho de código que faz parte do método findPlate(IplImage img):

```
public IplImage findPlate(IplImage img) {
          // método CVSize - OpenCv é responsável por pegar o tamanho da imagem recebida
          // como parâmetro
          CvSize cvSize = cvGetSize(img);

          // cvCreateImage uma nova área na mémoria para armazenar a imagem
          // convertida para tons de cinza com o método cvCvtColor
          IplImage gry = cvCreateImage(cvSize, img.depth(), 1);
          cvCvtColor(img, gry, CV_RGB2GRAY);

          // método cvThreshold binariza a imagem
          cvThreshold(gry, gry, 200, 255, CV_THRESH_BINARY);

          // cria uma área na memória para ser usada no próximo método
          CvMemStorage storage = CvMemStorage.create();
          CvSeq contours = new CvContour(null);

          // método cvFindContours é responsável por procurar os contornos na imagem, com essas
          // informações é possível localizar a placa na imagem.
      int noOfContors = cvFindContours(gry, storage, contours, Loader.sizeof(CvContour.class),
CV_RETR_CCOMP, CV_CHAIN_APPROX_NONE, new CvPoint(0, 0));
```

Figura 9 - Trecho de código que faz parte do método findPlate(IplImage img).
Fonte: Autor.

Na Figura 10, trecho de código responsável por percorrer toda imagem para localizar a placa, método utiliza um templete nas dimensões aproximada da placa para executar esta tarefa. Este método se mostrou muito eficaz na localização da placa (em um banco de imagens de 100 placas, conseguiu localizar o objeto em 70% das imagens ao contrário do mesmo método utilizado para preprocessar as imagens em tempo real, que somente conseguiu encontrar 40 % dos objetos nas imagens), porém por ter que percorrer toda imagem leva um tempo mair para preprocessar as mesmas (em torno de 20 segundos).

```
// laço responsável por percorrer toda  imagem para tentar localizar a placa
for (int y = 1; y < cvSize.height(); y++) {       // image height
    for (int x = 1; x < cvSize.width(); x++) {   // image wedth
        if ((y + HEIGHT) <= cvSize.height()) {
            if ((x + WIDTH) <= cvSize.width()) {
                IplImage copyImg = tpt.clone();

                // métoto cvSetImageRoi realiza o recorde da imagem nas dimensões
                // pré-configuradas e em diversas coordenas no decorrer das interações do for

                cvSetImageROI(img, cvRect(x, y, WIDTH, HEIGHT));
                cvCopy(img, copyImg);

                // método utilizado para armazenar no número de pixel encontrado em cada
                // recorde da imagem
                numberPixel(copyImg, x, y);

                //cvSaveImage("t" + x + y + ".jpg", copyImg);
            } else {
                x = cvSize.width(); // finaliza interação x
            }
        } else {
            y = cvSize.height(); // finaliza interação y
        }
    }
}
```

Figura 10 - Trecho de código responsável por percorrer toda extensão da imagem.
Fonte: Autor

A Figura 11, mostra o trecho de código utilizado na segmentação da imagem após realizar o pré-processamento, para que o resultado da segmentação seja o melhor possível, está rotina depende muito do resultado das etapas anteriores.

```
// segmentação da placa
bfImg = gry.getBufferedImage();
Raster raster = bfImg.getRaster();

// instância um vetor de inteiro para armazenar as coordenadas de corte que será utilizadas
// para separar as letras
int[] col = new int[15];
int aux = 0, i = 0, coluna = 0;


// varre imagem para localizar os pontos de segmentação
// e armazenar as coordenadas em um vetor de inteiros
for (int w = 0; w < bfImg.getWidth(); w++) {
    for (int h = 0; h < bfImg.getHeight(); h++) {
        if (raster.getSample(w, h, 0) == PMIN) {
            if (aux == 0) {
                col[i] = w;
                i++;
                aux = 1;
             }
        } else {
            coluna++;
        }
    }

    // verifica se todos os pixel da coluna processada é = 255
    //  se for, isso indica que é intervalo entre os objetos.
    if (coluna == bfImg.getHeight()) {
        aux = 0;
    }
    coluna = 0;
}

// define valor para o último índice, para finalizar processamento do
// último objeto.
col[i]= bfImg.getWidth();

// cria imagem
BufferedImage imgOut;
WritableRaster rasterOut;

//  rotina realiza a segmentação da imagem com base nas coordenadas localizada na
// etapa anterior.
for(int j = 0; j < i; j++){       // col[i+1] - col[i], h
   imgOut = new BufferedImage(col[j+1] - col[j], bfImg.getHeight(), BufferedImage.TYPE_BYTE_GRAY);
   rasterOut = imgOut.getRaster();

    for(int h = 0; h < bfImg.getHeight(); h++){
        for(int w = col[j]; w < col[j+1]; w++){

            rasterOut.setSample(w - col[j], h, 0, raster.getSample(w, h, 0));
         }
    }
```

Figura 11 - Trecho de código responsável pela segmentação da imagem.

Fonte: Autor

4.2 – Análise dos resultados

O objetivo deste projeto foi desenvolver um aplicativo para localizar placas de veículos em imagens capturadas de radares utilizando processamento de imagens.

Para realizar a localização da placa foi implementado dois algoritmos, um utilizando a biblioteca openCv e outro utilizando a biblioteca imageJ.

O primeiro algoritmo implementado utilizando a biblioteca openCv, conseguiu localizar as placas nas imagens em um menor tempo, porém no banco de imagens utilizado para teste ele não se mostrou muito eficiente conseguindo localizar em um total de 100 imagens de veículos apenas 30 placas totalizando 30% de acerto.

Para obter uma taxa de acerto melhor, foi implementado um algoritmo utilizando outra biblioteca (imageJ), que para localizar a placa na imagem, utiliza um templete nas dimensões aproximada da placa dos veículos e percorre toda imagem para localizar o objeto, como o algoritmo tem que percorrer toda a imagem ele lava um tempo maior para processar cada imagem em relação ao primeiro algoritmo, porém se mostrou muito eficiente, no mesmo banco de teste contento 100 imagens de veículos conseguiu localizar 70 placas totalizando uma taxa de acerto de 70%.

A Figura 12, mostra a placa do veículo localizada utilizando algoritmo criado utilizando a biblioteca imageJ.

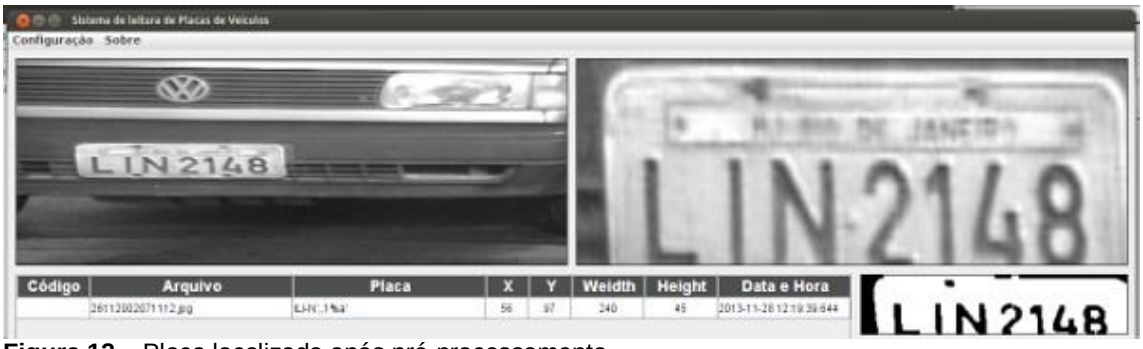

Figura 12 – Placa localizada após pré-processamento.
Fonte: Autor.

5 – Conclusão

Analisando as bibliografias utilizadas e a metodologia para desenvolvimento deste projeto, percebemos que os estudos na área de processamento digital de imagens tem sido cada vez mais explorados em benefício humano, e também por apresentarem soluções especializadas, especialmente quando falamos em processamento de imagens utilizando placas de veículos.

Este estudo teve a finalidade de apresentar um aplicativo que permita localizar placa de veículos a partir de uma imagem. Para este processo, as imagens originais passam por várias etapas até a localização da placa, pois as imagens obtidas por radares nem sempre está em boas condições.

Com base nos resultados obtidos, verificou-se que o aplicativo teve um bom desempenho conseguindo uma taxa de acerto de 70%, levando em consideração que a base imagens utilizada para teste continha 100 imagens de veículos e o aplicativo conseguiu localizar em 70 imagens as placas dos veículos. Apesar do bom resultado obtido, ainda é possível melhorar o aplicativo, para que o mesmo seja capaz de localizar a placa do veículo e reconhecer os caracteres contidos nela.

É importante ressaltar que fatores externos também interferem no resultado, como exemplo, a qualidade da imagem da placa, as condições meteorológicas em regiões que apresentam neblina, o próprio movimento do veículo quando estiver em alta velocidade e devido à ausência de iluminação adequada.

Considerando que o objetivo principal citado no início do trabalho foi cumprido, conclui-se que o resultado final do projeto é encorajador necessitando apenas de melhorias do algoritmo para que o mesmo possa localizar a placa, segmentar a imagem da placa e reconhecer os caracteres segmentados.

5.1 – Trabalhos Futuros

Em trabalhos futuros, melhorar o aplicativo implementando os métodos para reconhecimento de caracteres utilizando reconhecimento óptico de caracteres – OCR.

Desenvolver versões para dispositivos móveis, como celulares ou tablets que utilizem o sistema operacional Android, já que a biblioteca OpenCV possui suporte para esse sistema.

6 – Referências

Conci, Aura; Eduardo Azevedo, Fabiana R. Leta. **Computação gráfica teoria e prática.** Rio de Janeiro: Elsevier, 2008.

Eikvil, Line. **Optical Character Recognition.** Norsk Regnesentral, 1993.

Feitosa, Francisco Coelho Citó; **Um estudo prático para contagem volumétrica automática de veículos.** Goiás: Universidade Federal de Goiás – Instituto de Informática, 2012.

Gonzales, Rafael C.; Richard E. Woods. **Processamento digital de imagens.** São Paulo: Pearson Education do Brasil Ltda., 2010.

Lins, Luiz Fernando Miranda Vieira. **Reconhecimento ótico de Caracteres (OCR) E análise de sistemas ocr baseados em código aberto.** São Paulo: Faculdade de Tecnologia de São Paulo, 2012.

Marques Filho, Ogê; Hugo Vieira Neto. **Processamento Digital de Imagens.** Rio de Janeiro: Brasport, 1999.

Miranda, José Iguelmar. **Processamento de Imagens Digitais : Prática Usando Java.** Campinas: Empraba Informática Agropecuária, 2006.

Nascimento, Jean Dias. **Detecção de Reconhecimento de placa automotiva com baixo custo.** Brasília: Centro Universitário de Brasília – UniCEUB, 2012.

Pedrine, Hélio.; William Robson Schwartz. **Análise de imagens digitais: princípios, algoritimos e aplicações**. São Paulo: Thomson Learning, 2008.

Apêndice A – Instalação das ferramentas

Antes de realizar a instalação da plataforma *Java Standard Edition* e o Netbeans é necessário realizar o download dos mesmos, que pode ser feito em um único pacote de instalação ou separadamente em pacotes distintos. Na sequência, realizar a instalação em modo padrão, a versão do java jdk utilizada foi 7u45 e a do netbeans foi 7.4.

Link para *download:*

- http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html
- https://netbeans.org/downloads/

Após realizar o download dos arquivos de instalação é preciso dar permissão de execução para o instalador com o comando **chmod +x <nome arquivo de instalação>,** para iniciar a instalação, executar **./<nome arquivo de instalação>.**

A Figura 13 exibe os comandos necessários para instalação e a interface do instalador.

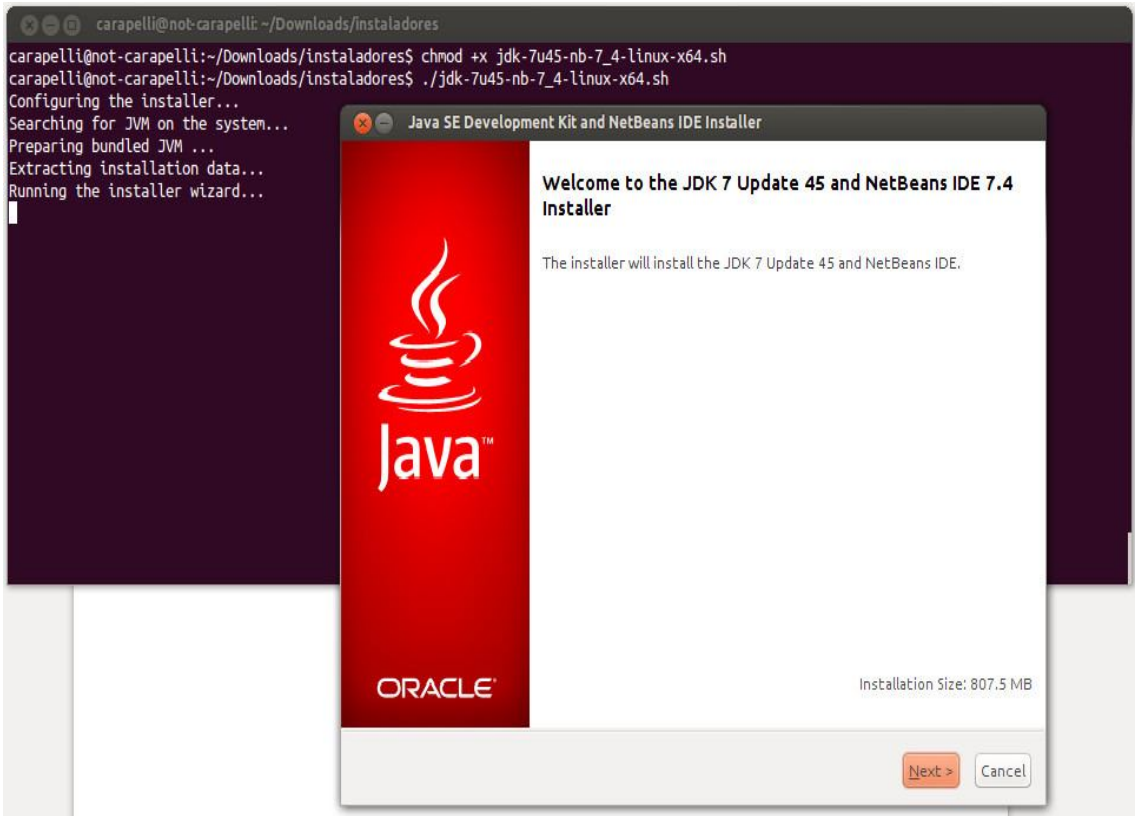

Figura 13 - Comandos e instalador java JDK e netbeans.
Fonte: Autor.

Instalação do OpenCV

Antes da instalação da Biblioteca OpenCV, é necessário instalar o *framework* multiplataforma Qt, isto se faz necessário porque o OpenCV utiliza as bibliotecas Qt, a versão utilizada foi 5.1.1 de 64bits.

Link para *Download* : http://download.qt-project.org/official_releases/qt/5.1/5.1.1/qt-

linux-opensource-5.1.1-x86_64-offline.run

Para instalação do *framework Qt,* entrar na pasta de download, dar permissão de execução para o instalador com o comando **chmod +x <nome arquivo de instalação>,** agora para iniciar a instalação executar **./<nome do arquivo de instalação>.**

Após realizar a instalação do *framework Qt,* localizar as pastas de bibliotecas do Qt com o seguinte comando: **sudo find / -name Qt5Core.**

Abrir o arquivo com o *script* de instalação do OpenCv, procurar pela linha que contenha o seguinte comando: **cmake,** adicionar o endereço das bibliotecas do *framework Qt* com os seguintes parâmetros:

- **-D Qt5Core_DIR=<endereço biblioteca Qt>**
- **-D Qt5Gui_DIR=<endereço biblioteca Qt>**
- **-D Qt5Widgets_DIR=<endereço biblioteca Qt>**
- **-D Qt5Test_DIR=<endereço biblioteca Qt>**
- **-D Qt5Concurrent_DIR=<endereço biblioteca Qt>**
- **-D Qt5OpenGL_DIR=<endereço biblioteca Qt>**
- **-D Qt5Widgets_DIR=<endereço biblioteca Qt>**

Salvar o *script* de instalação da biblioteca, dar permissão de execução para o mesmo com o seguinte comando: **chmod +x <nome do arquivo de *script*>,** para rodar o script de instalação executar: **./<nome do arquivo de *script*>.**

A Figura 14 exibe os comandos necessários para instalação e a interface do instalador.

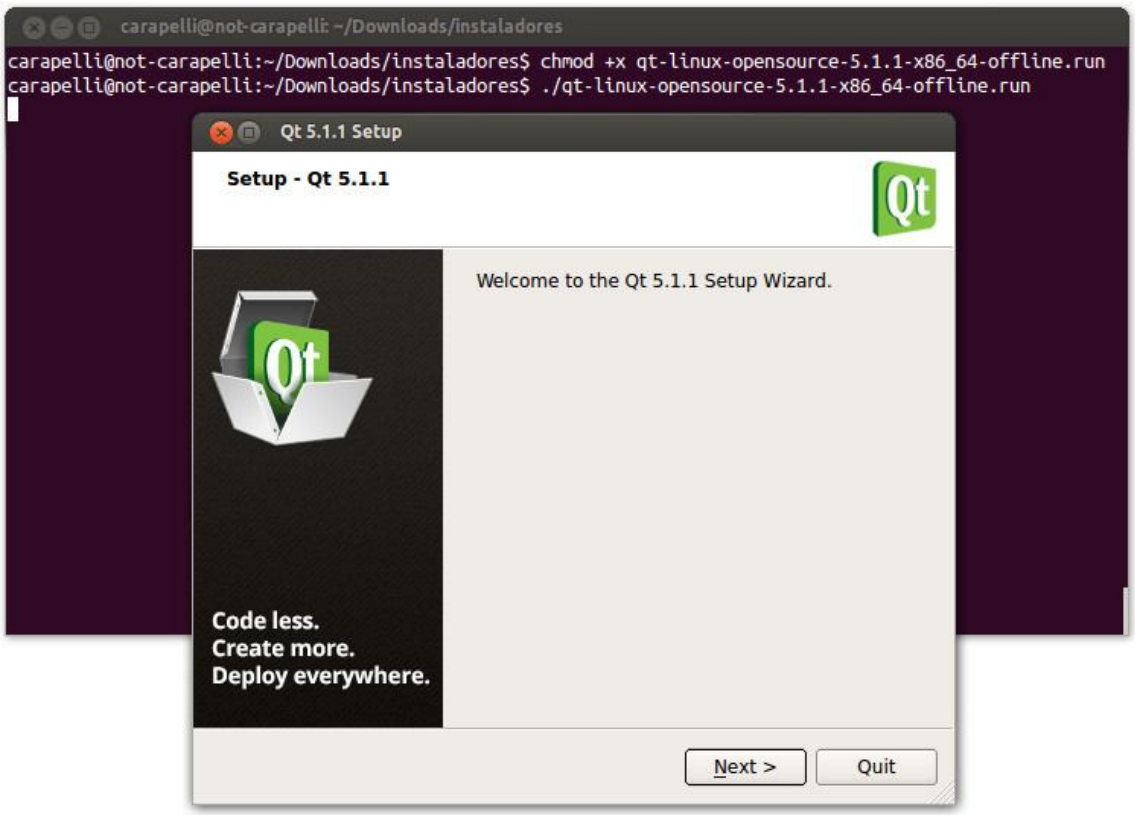

Figura 14 - Comandos e instalador do *framework* Qt5.
Fonte: Autor.

Integrar bibliotecas ao netbeans

Para poder utilizar os recursos do OpenCV e o ImageJ no Java é necessário realizar algumas configurações:

O OpenCV é uma biblioteca desenvolvida para aplicações C/C++ e para que possamos utilizar ela no java existe um projeto chamado JavaCV que é um *wrapper* que permite acessar diretamente os recursos da biblioteca de dentro do Java Virtual Machine.

Para configurar a biblioteca JacaCV no netbeans, é necessário antes fazer o *download* da mesma no seguinte *link*: https://code.google.com/p/javacv/downloads/detail?name=javacv-0.6-bin.zip.

Após baixar o arquivo, é necessário descompacta-lo com o seguinte comando: **unzip javacv-0.6-bin.zip,** feito isso, abrir o netbeans acessar Ferramentas no menu principal, escolher a opção Biblioteca, na janela que se abrirá, clicar no botão Nova biblioteca, colocar o nome de javaCv, clicar no botão adicionar Jar/Diretório, entrar no diretório que você descompactou a biblioteca JavaCv e adicionar os arquivos abaixo:

- **javacv.jar**
- **javacpp.jar**
- **javacv-linux-x86_64.jar**

Biblioteca ImageJ

Baixar a biblioteca no seguinte *link*: http://rsbweb.nih.gov/ij/download/linux/ij147-linux64.zip. Descompactar o arquivo com o comando: **unzip ij147-linux64.zip.**

Após descompactar os arquivos da biblioteca, realizar o mesmo procedimento utilizado na biblioteca OpenCv, para adicionar a biblioteca ImageJ, mudando apenas o nome da biblioteca para imageJ.

Figura 15 mostra a configuração da biblioteca JavaCV no netbeans.

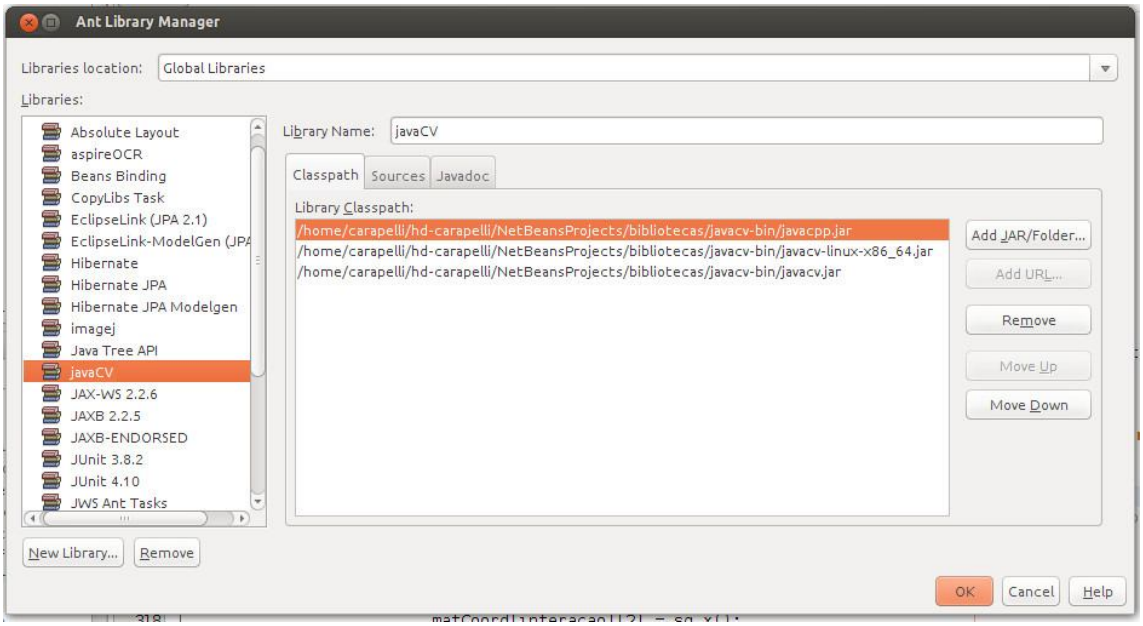

Figura 15 – Configuração biblioteca JavaCv.
Fonte: Autor.

Figura 16 mostra a configuração da biblioteca ImageJ no netbeans.

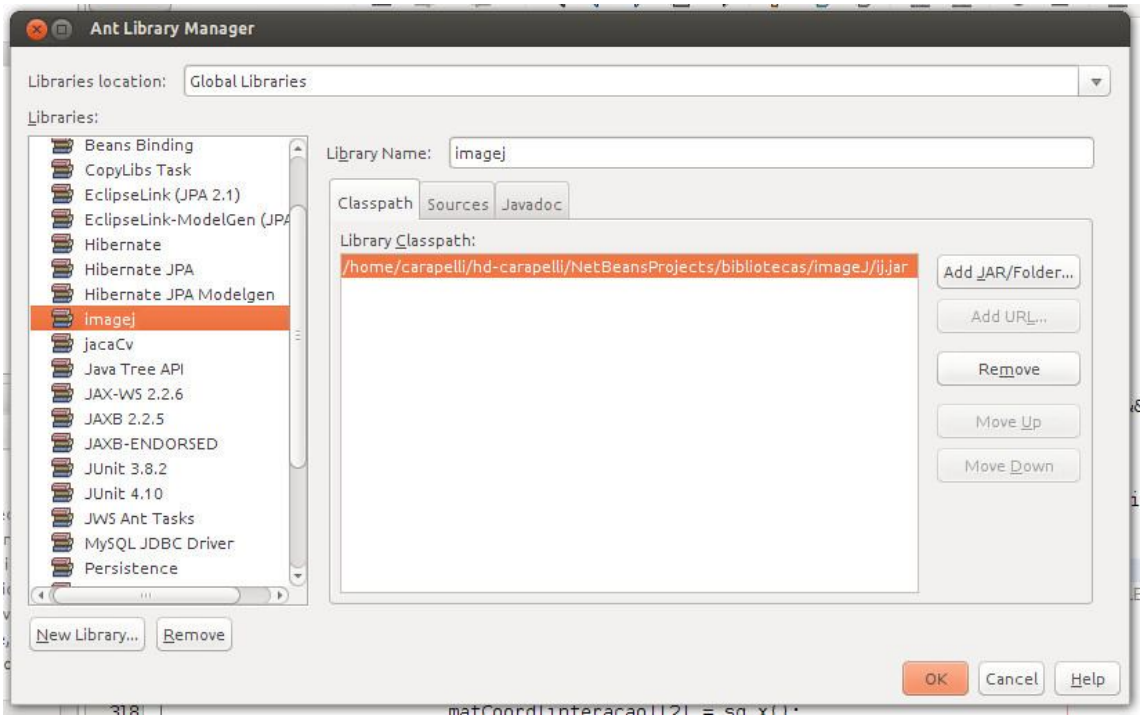

Figura 16 – Configuração biblioteca ImageJ.
Fonte: Autor.

Depois de realizar todos esses procedimentos o ambiente java está configurado e pronto para utilizar todos os recursos das bibliotecas adicionadas.

Apêndice B – Código fonte aplicativo.

Slpv.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package slpv;

/**
 *
 * @author carapelli
 */
public class Slpv {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new menuPrincipal().setVisible(true);
            }
        });
    }
```

menuPrincipal.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package slpv;

import javax.swing.table.DefaultTableModel;
import control.MyRenderer;
import control.imagem;
import ij.ImagePlus;
import ij.process.ImageProcessor;
import java.awt.Font;
import java.awt.image.BufferedImage;
import java.awt.image.Raster;
import java.awt.image.WritableRaster;
import java.io.File;
import java.sql.Timestamp;
import java.util.GregorianCalendar;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.table.JTableHeader;
import com.googlecode.javacpp.Loader;
import com.googlecode.javacv.CanvasFrame;
```

```
 import com.googlecode.javacv.FrameGrabber;
import com.googlecode.javacv.OpenCVFrameGrabber;
import static com.googlecode.javacv.cpp.opencv_core.*;
import static com.googlecode.javacv.cpp.opencv_imgproc.*;
import static com.googlecode.javacv.cpp.opencv_highgui.*;
import control.ocrFiles;
import control.searchPlate;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.TimeZone;
import javax.swing.JProgressBar;
import javax.swing.filechooser.FileNameExtensionFilter;



/**
 *
 * @author carapelli
 */
public class menuPrincipal extends javax.swing.JFrame {

    //private static volatile boolean isRunning = true;
    static final String[] colunas = {"Código", "Arquivo", "Placa", "X","Y","Weidth","Height","Data e
Hora"};
    static Object[] linha = new Object[8];
    final int PMAX = 255, PMIN = 0;

    //ImageIcon imageIcon = null;
    //ImageProcessor imgPcs = null;
    //ImageProcessor imgSaida = null;

    int tipo, nRow = -1, seqArq = 0;

    //BufferedImage bufferedImg = null;
    //BufferedImage imgOut = null;
    //Raster raster = null;
   //WritableRaster rasterOut = null;

   GregorianCalendar calendar = new GregorianCalendar();

   MyRenderer myRenderer = new MyRenderer();

   boolean wCam = false;

   DefaultTableModel modelo = new DefaultTableModel(null, colunas) {
       @Override
       public boolean isCellEditable(int rowIndex, int mColIndex) {
           return false;
       }
   };
```

```
public static IplImage gray(IplImage src) {

    //The RGB color space conversion to BGR color space 8 3 channels
    IplImage pImg = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 3);
    /*src is the source image;
    dst is converted image;
    flags is the conversion mode, you can take 0: no change;
      1: vertical flip, flip that along the x-axis; 2: swap the red and blue channels;
     */
    cvConvertImage(src, pImg, 2);

    //Will be converted to RGB Gray degrees Figure
    IplImage pGrayImg = cvCreateImage(
            cvGetSize(pImg),
            IPL_DEPTH_8U,
            1);
    cvCvtColor(pImg, pGrayImg, CV_RGB2GRAY);
    cvReleaseImage(pImg);
    return pGrayImg;
    //cvSaveImage("D:\\IBM\\gray.jpg",pGrayImg);
}

// Sobel
public IplImage sobel(String filename) {
    CvMat src, src_gray;
    CvMat grad = null;
    int scale = 1;
    int delta = 0;
    int ddepth = CV_16S;
    int c;
    src = cvLoadImageM(filename);

    // GaussianBlur (src, src, cvSize (3,3), 0, 0, BORDER_DEFAULT);
    cvSmooth(src, src, CV_GAUSSIAN, 3);
    src_gray = gray(src.asIplImage()).asCvMat();

    CvMat grad_x = null, grad_y = null;
    CvMat abs_grad_x = null, abs_grad_y = null;

    /// Gradient X
    // Scharr (src_gray, grad_x, ddepth, 1, 0, scale, delta, BORDER_DEFAULT);
    grad_x = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1).asCvMat();
    cvSobel(src_gray, grad_x, 1, 0, 3);
    abs_grad_x = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1).asCvMat();

    /**
     * Src: original number pairs dst: export number pairs (depth Acts 8u).
     * scale: proportionality factor. shift: original number 组元 prime
```

```
          * example 按比 缩放 after addition basis 值
         */
          cvConvertScaleAbs(grad_x, abs_grad_x, 1, 0);

        /// Gradient Y
        // Scharr (src_gray, grad_y, ddepth, 0, 1, scale, delta, BORDER_DEFAULT);

        grad_y = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1).asCvMat();
        abs_grad_y = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1).asCvMat();

        cvSobel(src_gray, grad_y, 0, 1, 3);

        int N = 3;
        int aperature_size = N;
        double lowThresh = 50; //50
        double highThresh = 70; //70
        cvCanny(src_gray, grad_y, lowThresh, highThresh, aperature_size);

        cvConvertScaleAbs(grad_y, abs_grad_y, 1, 0);

        /// Total Gradient (approximate)
        grad = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1).asCvMat();

        cvAddWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0, grad);
        //cvSaveImage("D: \ \ IBM \ \ sobel.jpg", grad);

        //cvSaveImage(System.getProperty("user.dir") + "/canny/"+
filename.substring(filename.lastIndexOf("/") + 1), grad);

        return grad.asIplImage();
    }

    public void findPlate(String filename) {

        jLabImg2.setText(null);
        jLabImg3.setText(null);

        IplImage img = cvLoadImage(filename);

        CvSize cvSize = cvGetSize(img);

        IplImage gry = cvCreateImage(cvSize, img.depth(), 1);
        cvCvtColor(img, gry, CV_RGB2GRAY);

       //cvAdaptiveThreshold(gry, gry, 255, CV_ADAPTIVE_THRESH_MEAN_C,
CV_THRESH_BINARY_INV, 11, 5);
       cvThreshold(gry, gry, 200, 255, CV_THRESH_BINARY);

       CvMemStorage storage = CvMemStorage.create();
       CvSeq contours = new CvContour(null);
```

```
        int noOfContors = cvFindContours(gry, storage, contours, Loader.sizeof(CvContour.class),
CV_RETR_CCOMP, CV_CHAIN_APPROX_NONE, new CvPoint(0, 0));
        //int noOfContors = cvFindContours(gry, storage, contours, Loader.sizeof(CvContour.class),
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, new CvPoint(0, 0));

         searchPlate sp = new searchPlate(gry);

        // create template license     w    h
        IplImage lc = IplImage.create(255, 35, IPL_DEPTH_8U, 1);

        cvSaveImage("tpt.jpg", lc);

        // load template
        IplImage tpt = cvLoadImage(System.getProperty("user.dir") + "/tpt.jpg");


        // image cutter              X                Y                w    h
        cvSetImageROI(img, cvRect(sp.getCoordnadas()[0], sp.getCoordnadas()[1], 255, 35));
        cvCopy(img, tpt);

        // deleta arquivo
        File file = new File(System.getProperty("user.dir") + "/tpt.jpg");
        file.delete();

        // show image processed
        BufferedImage bfimg = tpt.getBufferedImage();
        ImageIcon imgIc = new ImageIcon(bfimg);
        jLabImg2.setIcon(new ImageIcon(imagem.redimensionar(imgIc.getImage(), jLabImg2)));

        cutPlate(tpt, filename,sp.getCoordnadas());

       }

    public IplImage findPlate(IplImage img) {

        CvSize cvSize = cvGetSize(img);

        IplImage gry = cvCreateImage(cvSize, img.depth(), 1);

       cvCvtColor(img, gry, CV_RGB2GRAY);

       cvThreshold(gry, gry, 200, 255, CV_THRESH_BINARY);
       //cvAdaptiveThreshold(gry, gry, 255, CV_ADAPTIVE_THRESH_MEAN_C,
CV_THRESH_BINARY_INV, 11, 5);

       CvMemStorage storage = CvMemStorage.create();
       CvSeq contours = new CvContour(null);

       int noOfContors = cvFindContours(gry, storage, contours, Loader.sizeof(CvContour.class),
CV_RETR_CCOMP, CV_CHAIN_APPROX_NONE, new CvPoint(0, 0));
```

```
        //  int noOfContors = cvFindContours(gry, storage, contours, Loader.sizeof(CvContour.class),
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, new CvPoint(0, 0));

        // verifica se o numero de contorno é maior que zero para não gerar
        // na captura via webcam
        if (noOfContors > 0) {

            jLabImg1.setIcon(null);
            jLabImg2.setIcon(null);
            jLabImg3.setIcon(null);

            CvSeq ptr = new CvSeq();

            int count = 1;
            CvPoint p1 = new CvPoint(0, 0), p2 = new CvPoint(0, 0);

            int matCoord[][] = new int[contours.total()][4];
            int interacao = 0;

            jTextArea1.setText(null);

            for (ptr = contours; ptr != null; ptr = ptr.h_next()) {

                CvScalar color = CvScalar.BLUE;

                try {

                    CvRect sq = cvBoundingRect(ptr, 0);

                    p1.x(sq.x());
                    p2.x(sq.x() + sq.width());
                    p1.y(sq.y());
                    p2.y(sq.y() + sq.height());

                    if ((390 < sq.width() && sq.width() < 410) && (100 < sq.height() && sq.height() < 150)) {

                    jTextArea1.append("Contour No =" + count + "\n");
                    jTextArea1.append("X =" + sq.x() + " Y=" + sq.y() + "\n");
                    jTextArea1.append("Height =" + sq.height() + " Width =" + sq.width() + "\n");
                    jTextArea1.append("\n\n");

                    cvRectangle(img, p1, p2, CV_RGB(255, 0, 0), 2, 8, 0);
                    //cvDrawContours(img, ptr, color, CV_RGB(0, 0, 0), -1, CV_FILLED, 8, cvPoint(0, 0));

                    // matriz get coordenadas x,y and width, height the image
                    matCoord[interacao][0] = sq.width();
                    matCoord[interacao][1] = sq.height();
                    matCoord[interacao][2] = sq.x();

                    matCoord[interacao][3] = sq.y();

                    interacao++;
```

```
                    // cute plate
                        cutPlate(img, matCoord, interacao, "WebCam");

                        // stop thred webcam
                        wCam = false;
                    }else if(sq.width() > 380 && sq.height() > 90){
                        //show rectangle width > 380 and height > 90

                        cvRectangle(img, p1, p2, CV_RGB(255, 0, 0), 2, 8, 0);
                        cvDrawContours(img, ptr, color, CV_RGB(0, 0, 0), -1, CV_FILLED, 8, cvPoint(0, 0));
                    }

                    count++;
                } catch (Exception ex) {
                    jTextArea1.append("findPlate(IplImage) \n" + ex.getMessage());
                    return img;
                }
            }
        }

        return img;
    }


    /**
     * *
     * @param iplImg
     * @param matCoord - 0 = weidth, 1 = heigth, 2 = x and 3 = y
     * @param interacao
     */
    public void cutPlate(IplImage iplImg, int matCoord[][], int interacao, String fName) {
        int i;
        String caracter = null;

        for (i = 0; i <= interacao - 1; i++) {

            // create template license
            IplImage lc;
            lc = IplImage.create(matCoord[i][0], matCoord[i][1], IPL_DEPTH_8U, 1);

          cvSaveImage("tpl.jpg", lc);

          // load template
          IplImage tpt = cvLoadImage(System.getProperty("user.dir") + "/tpl.jpg");
         tTry {

              // image cutter              X                  Y             w                h
              cvSetImageROI(iplImg, cvRect(matCoord[i][2], matCoord[i][3], matCoord[i][0],
matCoord[i][1]));
```

```
                cvCopy(iplImg, tpt);

            } catch (Exception ex) {

                jTextArea1.append("CutPlate \n" + ex.getMessage());
            }

            // clone image tpt
            //imgClone = tpt.clone();
            CvSize cvSize = cvGetSize(tpt);
            IplImage gry = cvCreateImage(cvSize, tpt.depth(), 1);
            cvCvtColor(tpt, gry, CV_RGB2GRAY);

            cvThreshold(gry, gry, 200, 255, CV_THRESH_BINARY); //;

            cvDilate(gry, gry, null, 1);
            cvErode(gry, gry, null, 2);

            // localiza quantidade de contorno na imagem
            // se for menor que 7 descarta imagem
            //findCharacter(gry);

            IplImage tmpGry = gry.clone();

            CvMemStorage storage = CvMemStorage.create();
            CvSeq contours = new CvContour(null);

            int noOfContors = cvFindContours(tmpGry, storage, contours,
Loader.sizeof(CvContour.class), CV_RETR_CCOMP, CV_CHAIN_APPROX_NONE, new CvPoint(0,
0));
            //  int noOfContors = cvFindContours(gry, storage, contours, Loader.sizeof(CvContour.class),
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, new CvPoint(0, 0));



            //coordRetangCharacter crch = new coordRetangCharacter();

            if (noOfContors >= 7) {
                //gry = crch.GetImagem().clone();

                // show image processed

                BufferedImage bfimg = gry.getBufferedImage();
                ImageIcon imgIc = new ImageIcon(bfimg);
                jLabImg3.setIcon(new ImageIcon(imagem.redimensionar(imgIc.getImage(), jLabImg3)));

                cvSaveImage(fName.substring(fName.lastIndexOf("/") + 1), gry);

              //ImageIcon imc = new ImageIcon(System.getProperty("user.dir") + "/plate.jpg");
              //img3.setIcon(new ImageIcon(imagem.redimensionar(imc.getImage(), img3)));
```

```
                caracter = reconheceCaracter(System.getProperty("user.dir") +
fName.substring(fName.lastIndexOf("/")));

                File file = new File("tpl.jpg");
                file.delete();
            }
        }


        // add data in table
        linha[0] = "";
        linha[1] = fName.substring(fName.lastIndexOf("/") + 1);   // file name
        linha[2] = caracter;                                // character recognized
        linha[3] = matCoord[i-1][2];                        // x image
        linha[4] = matCoord[i-1][3];                        // y image
        linha[5] = matCoord[i-1][0];                        // width image
        linha[6] = matCoord[i-1][1];                        // height image
        linha[7] = new Timestamp(calendar.getTimeInMillis()); // hour
        modelo.addRow(linha);
    }

    /**
     *
     * @param iplImg
     * @param fName
     * @param xy
     */
    public void cutPlate(IplImage iplImg, String fName, int[] xy) {
        try {
            String caracter = "";

            cvSaveImage(System.getProperty("user.dir") + fName.substring(fName.lastIndexOf("/")),
iplImg);

            IplImage img1 = cvLoadImage(System.getProperty("user.dir") +
fName.substring(fName.lastIndexOf("/")));

  CvSize cvSize = cvGetSize(img1);

           IplImage gry = cvCreateImage(cvSize, img1.depth(), 1);
           cvCvtColor(img1, gry, CV_RGB2GRAY);

           //img1 = IplImage.create(cvSize.width(),cvSize.height(), IPL_DEPTH_8U, 1);
           //cvSmooth(gry, t, CV_GAUSSIAN,  11 );
           //cvEqualizeHist(gry, img1);

           // bufferImage to imageProcessor
           BufferedImage bfImg = gry.getBufferedImage();
           ImagePlus imgPlus = new ImagePlus(null, bfImg);
           ImageProcessor imgPcs = imgPlus.getProcessor();
           int threshold = imgPcs.getAutoThreshold();
```

```
    cvThreshold(gry, gry, threshold, 255, CV_THRESH_OTSU); //;

    cvDilate(gry, gry, null, 1);
    cvErode(gry, gry, null, 1);

    // show image processed
    BufferedImage bfimg = gry.getBufferedImage();
    ImageIcon imgIc = new ImageIcon(bfimg);
    jLabImg3.setIcon(new ImageIcon(imagem.redimensionar(imgIc.getImage(), jLabImg3)));

    // segmentação da placa
    bfImg = gry.getBufferedImage();
    Raster raster = bfImg.getRaster();

    int[] col = new int[15];
    int aux = 0, i = 0, coluna = 0;

    /* varre imagem para localizar os pontos de segmentação
     * e armazenar as coordenada em um vetor de inteiros
     */
    for (int w = 0; w < bfImg.getWidth(); w++) {
        for (int h = 0; h < bfImg.getHeight(); h++) {
            if (raster.getSample(w, h, 0) == PMIN) {
                if (aux == 0) {
                    col[i] = w;
                    i++;
                    aux = 1;

                }
            } else {
                coluna++;
            }
        }
        /*
         * verifica se todos os pixel da coluna processada é = 255
         * se for, isso indica que é intervalo entre os objetos.
         */
        if (coluna == bfImg.getHeight()) {

         aux = 0;
      }
      coluna = 0;
  }

  /* define valor para o último índice, para finalizar processamento do
  * último objeto.
  */
  col[i]= bfImg.getWidth();
```

```
      // cria imagem
          BufferedImage imgOut;
          WritableRaster rasterOut;

            // segmenta a imagem
            for(int j = 0; j < i; j++){       // col[i+1] - col[i], h
                imgOut = new BufferedImage(col[j+1] - col[j], bfImg.getHeight(),
BufferedImage.TYPE_BYTE_GRAY);
                rasterOut = imgOut.getRaster();

                for(int h = 0; h < bfImg.getHeight(); h++){
                    for(int w = col[j]; w < col[j+1]; w++){

                        rasterOut.setSample(w - col[j], h, 0, raster.getSample(w, h, 0));

                    }
                }

               // ImageIO.write(imgOut, "jpg", new File(System.getProperty("user.dir") +
fName.substring(fName.lastIndexOf("/"))));

                img1 = IplImage.createFrom(imgOut);

                cvSaveImage(System.getProperty("user.dir") +
fName.substring(fName.lastIndexOf("/")),img1);

                caracter = caracter + reconheceCaracter(System.getProperty("user.dir") +
fName.substring(fName.lastIndexOf("/")));

            }

            // add data in table
            linha[0] = "";
            linha[1] = fName.substring(fName.lastIndexOf("/") + 1);   // file name
            linha[2] = caracter;                                      // character recognized
            linha[3] = xy[0];                                        // x image
            linha[4] = xy[1];                                        // y image
            linha[5] = "240";                                         // width image
            linha[6] = "45";                                         // height image
            linha[7] = new Timestamp(calendar.getTimeInMillis());     // hour

          modelo.addRow(linha);

          File f = new File(System.getProperty("user.dir") + fName.substring(fName.lastIndexOf("/")));
          f.delete();

      } catch (Exception ex) {

          System.out.println("CutPlate \n" + ex.getMessage());
      }
  }
```

```
    public String reconheceCaracter(String fName){

        List<File> nfile = new ArrayList<File>();

        File imageFile = new File(fName);

        nfile.add(imageFile);


        try{

           // BufferedImage bfimg = ImageIO.read(new File(System.getProperty("user.dir") +
"/plate1.png"));


            //Runtime.getRuntime().exec("tesseract " + fName + "t");

            ocrFiles ocrEngine = new ocrFiles(System.getProperty("user.dir") + "/tesseract-ocr", "8");


            String caracter = ocrEngine.recognizeText(nfile, "eng");

            //jTextArea1.append("Reconhecimento letras ********** \n" + caracter);

            return  caracter;

        } catch(Exception e){
           jTextArea1.append("Reconhecimento ********** \n" + e.getMessage());
           return e.getMessage();
        }
    }

    public boolean verificaracter(String caracter){
        //pega o código asc do caracter
        int cod = caracter.charAt(1);

      // verifica se o código asc é um caracter: 0 - 9
       if(cod >= 48 && cod <= 57 ) return true;

       // verifica se o código asc é um caracter: A - Z
       if(cod >= 65 && cod <= 90) return true;

       // verifica se o código asc é um caracter: a - z
       if(cod >= 97 && cod <= 122) return true;

       // se chegou até aqui, porque não é um caracter válido.
       return false;

}
```

```
    public BufferedImage bImage(BufferedImage bImg){
        // binarização
        tipo = BufferedImage.TYPE_BYTE_GRAY;
        Raster raster = bImg.getRaster();
        BufferedImage imgOut = new BufferedImage(bImg.getWidth(), bImg.getHeight(), tipo);
        WritableRaster rasterOut = imgOut.getRaster();

        for (int h = 0; h < bImg.getHeight(); h++) {
            for (int w = 0; w < bImg.getWidth(); w++) {
                if (h == 65 || h == 135) {
                    rasterOut.setSample(w, h, 0, PMAX);
                } else if (h > 65 & h <= 135) {
                    rasterOut.setSample(w, h, 0, raster.getSample(w, h, 0));
                }
            }
        }

        // bufferImage to imageProcessor
        ImagePlus imgPlus = new ImagePlus(null, imgOut);
        //ImageProcessor imgPcs = imgPlus.getProcessor();

        return imgPlus.getBufferedImage();
      }

    public final void listFile() {
        File diretorio = new File(System.getProperty("user.dir") +
"/LPR_ImageDataBase_Sample/imagens");

        File ldiretorio[] = diretorio.listFiles();

        for (int i = 0; i < ldiretorio.length; i++) {
            linha[0] = i;
            linha[1] = ldiretorio[i].getName();
            linha[3] = new Timestamp(calendar.getTimeInMillis());

            modelo.addRow(linha);
        }
    }

    public final void iniciar() {
        jLabImg1.setText(null);
        jLabImg2.setText(null);
        jLabImg3.setText(null);

        jLabelProcessando.setVisible(false);
        System.load(System.getProperty("user.dir") + "/libAspriseOCR.so");

        btnProcessarArq.requestFocus();
    }
```

```
    /**
     * Creates new form menuPrincipal
     */
    public menuPrincipal() {
        initComponents();

        this.setTitle("Sistema de leitura de Placas de Veículos");

        java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width - 1310) / 2, (screenSize.height - 705) / 2, 1310, 705);

        // sets table
        tabela.setDefaultRenderer(Object.class, myRenderer);
        tabela.setModel(modelo);

        // sets title table
        JTableHeader titulo = tabela.getTableHeader();
        titulo.setFont(new Font("arial", Font.BOLD, 18));
        titulo.setBackground(new java.awt.Color(79, 79, 79));
        titulo.setForeground(new java.awt.Color(255, 255, 255));

        // configura tamanho default linha
        tabela.setRowHeight(25);

        // configura tamanho default coluna
        tabela.getColumnModel().getColumn(0).setPreferredWidth(50);
        tabela.getColumnModel().getColumn(1).setPreferredWidth(200);
        tabela.getColumnModel().getColumn(2).setPreferredWidth(190);
        //tabela.getColumnModel().getColumn(2).setCellRenderer(new MyRenderer());
        tabela.getColumnModel().getColumn(3).setPreferredWidth(5);
        tabela.getColumnModel().getColumn(4).setPreferredWidth(5);
        tabela.getColumnModel().getColumn(5).setPreferredWidth(50);
        tabela.getColumnModel().getColumn(6).setPreferredWidth(50);
        tabela.getColumnModel().getColumn(7).setPreferredWidth(120);

        modelo.setNumRows(0);

       // ativa sort por coluna
       //tabela.setAutoCreateRowSorter(false);
       //tabela.getTableHeader().setReorderingAllowed(false);

       iniciar();

       //listFile();
    }
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
private void initComponents() {

    jPanel1 = new javax.swing.JPanel();
    jLabImg1 = new javax.swing.JLabel();
    jPanel2 = new javax.swing.JPanel();
    jLabImg2 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    tabela = new javax.swing.JTable();
    btnProcessarArq = new javax.swing.JButton();
    btnProcessarDir = new javax.swing.JButton();
    btnLimpar = new javax.swing.JButton();
    jScrollPane2 = new javax.swing.JScrollPane();
    jTextArea1 = new javax.swing.JTextArea();
    btnWebCam = new javax.swing.JButton();
    jPanel3 = new javax.swing.JPanel();
    jLabImg3 = new javax.swing.JLabel();
    jLabelProcessando = new javax.swing.JLabel();
    jMenuBar1 = new javax.swing.JMenuBar();
    jMenu1 = new javax.swing.JMenu();
    jMenu2 = new javax.swing.JMenu();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    getContentPane().setLayout(null);

    jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));
    jPanel1.setLayout(new java.awt.GridLayout(1, 0));

    jLabImg1.setText("jLabel1");
    jPanel1.add(jLabImg1);

    getContentPane().add(jPanel1);
    jPanel1.setBounds(10, 10, 640, 240);

    jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));
    jPanel2.setLayout(new java.awt.GridLayout(1, 0));

   jLabImg2.setText("jLabel1");
   jPanel2.add(jLabImg2);

   getContentPane().add(jPanel2);
   jPanel2.setBounds(660, 10, 640, 240);

   tabela.setModel(new javax.swing.table.DefaultTableModel(
      new Object [][] {

      },
      new String [] {

      }
   ));
```

```
        tabela.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                tabelaMouseClicked(evt);
            }
        });
        jScrollPane1.setViewportView(tabela);

        getContentPane().add(jScrollPane1);
        jScrollPane1.setBounds(10, 260, 970, 350);

        btnProcessarArq.setText("Processar Arquivo");
        btnProcessarArq.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnProcessarArqActionPerformed(evt);
            }
        });
        getContentPane().add(btnProcessarArq);
        btnProcessarArq.setBounds(10, 620, 180, 27);

        btnProcessarDir.setText("Processar Diretório");
        btnProcessarDir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnProcessarDirActionPerformed(evt);
            }
        });
        getContentPane().add(btnProcessarDir);
        btnProcessarDir.setBounds(200, 620, 180, 27);

        btnLimpar.setText("Limpar");
        btnLimpar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnLimparActionPerformed(evt);
            }
        });
        getContentPane().add(btnLimpar);
        btnLimpar.setBounds(400, 620, 140, 27);

        jTextArea1.setColumns(20);

       jTextArea1.setRows(5);
       jScrollPane2.setViewportView(jTextArea1);

       getContentPane().add(jScrollPane2);
       jScrollPane2.setBounds(990, 340, 310, 270);

       btnWebCam.setText("WebCam");
       btnWebCam.addActionListener(new java.awt.event.ActionListener() {
           public void actionPerformed(java.awt.event.ActionEvent evt) {
               btnWebCamActionPerformed(evt);
           }
       });
```

```
        getContentPane().add(btnWebCam);
        btnWebCam.setBounds(550, 620, 150, 27);

        jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));
        jPanel3.setLayout(new java.awt.GridLayout(1, 0));

        jLabImg3.setText("jLabel1");
        jPanel3.add(jLabImg3);

        getContentPane().add(jPanel3);
        jPanel3.setBounds(990, 260, 310, 70);

        jLabelProcessando.setFont(new java.awt.Font("Ubuntu", 1, 18)); // NOI18N
        jLabelProcessando.setForeground(new java.awt.Color(240, 11, 11));
        jLabelProcessando.setText("Processando ...");
        getContentPane().add(jLabelProcessando);
        jLabelProcessando.setBounds(1000, 620, 190, 21);

        jMenu1.setText("Configuração");
        jMenuBar1.add(jMenu1);

        jMenu2.setText("Sobre");
        jMenuBar1.add(jMenu2);

        setJMenuBar(jMenuBar1);
    }// </editor-fold>

    private void btnProcessarArqActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        File file;

        JFileChooser fileChooser = new JFileChooser(".");
        fileChooser.setDialogTitle("Choose Image");
        FileNameExtensionFilter filter = new FileNameExtensionFilter("JPG & PNG", "jpg", "png");

        fileChooser.setFileFilter(filter);

       int result = fileChooser.showOpenDialog(null);

       if (result == JFileChooser.APPROVE_OPTION) {
          file = fileChooser.getSelectedFile();
          try {

              ImageIcon imageIcon = new ImageIcon(file.toString());
              jLabImg1.setIcon(new ImageIcon(imagem.redimensionar(imageIcon.getImage(),
jLabImg1)));

              jLabelProcessando.setVisible(true);

              // find plate in image
              findPlate(file.toString());
```

```
                jLabelProcessando.setVisible(false);

            } catch (Exception ex) {
                jLabelProcessando.setVisible(false);
                JOptionPane.showMessageDialog(null, ex.getMessage(), "Usc", 1);
            }
        }
    }

    private void btnProcessarDirActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        File file;

        DateFormat dfo = new SimpleDateFormat("HH:mm:ss");
        //formato UTC do TimeZone
        dfo.setTimeZone(TimeZone.getTimeZone("UTC"));

        Calendar hora;
        Calendar horaInicio = Calendar.getInstance();

        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        fileChooser.setDialogTitle("Choose Directory");

        int result = fileChooser.showOpenDialog(null);
        if (result == JFileChooser.APPROVE_OPTION) {
            file = fileChooser.getSelectedFile();

            jTextArea1.append("Directory: " + file.getPath()+"\n\n");

            jTextArea1.append("********************************** \n");
            jTextArea1.append("Begin: " + dfo.format(horaInicio.getTimeInMillis()) + "\n");
            jTextArea1.append("********************************** \n\n");

            try {

             File ldiretorio[] = file.listFiles();

             for (int i = 0; i < ldiretorio.length; i++) {
                ImageIcon imageIcon = new ImageIcon(ldiretorio[i].getName());

                jLabImg1.setIcon(new ImageIcon(imagem.redimensionar(imageIcon.getImage(),
jLabImg1)));

                jLabelProcessando.setVisible(true);

                //plate search in image
                findPlate(file.getPath() + "/" + ldiretorio[i].getName());
```

```
hora = Calendar.getInstance();

                    jTextArea1.append(ldiretorio[i].getName() + "\nTime: " +
dfo.format(hora.getTimeInMillis()) + "\n\n");

                }

                jLabelProcessando.setVisible(false);

                Calendar horaFim = Calendar.getInstance();

                jTextArea1.append("********************************** \n");
                jTextArea1.append("Time: " + dfo.format(horaFim.getTimeInMillis() -
horaInicio.getTimeInMillis()) + "\n");
                jTextArea1.append("End: " + dfo.format(horaFim.getTimeInMillis()) + "\n");
                jTextArea1.append("********************************** \n\n");

            } catch (Exception ex) {
                jLabelProcessando.setVisible(false);
                JOptionPane.showMessageDialog(null, "Erro ao carregar imagem \n\n" +
ex.getMessage(), "Usc", 1);
            }
        }
    }

    private void btnLimparActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        jLabImg1.setIcon(null);
        jLabImg2.setIcon(null);
        jLabImg3.setIcon(null);
        jTextArea1.setText(null);

        nRow = -1;

        modelo.setRowCount(0);
        btnProcessarArq.requestFocus();
    }

    private void tabelaMouseClicked(java.awt.event.MouseEvent evt) {
    }

    public class progressBar{

        public void progressBar(){
          JProgressBar  jProgressBar = new JProgressBar();
          jProgressBar.setBounds(new java.awt.Rectangle(1000,622,300,20));
          jProgressBar.setMinimum(0);
          jProgressBar.setMaximum(100);
          jProgressBar.setStringPainted(true);

        }
```

```
    }

    public class webCam extends Thread {

        @Override
        public void run() {
            if (wCam == true) {
                //Create canvas frame for displaying webcam.
                CanvasFrame canvas = new CanvasFrame("Webcam - Processed image");

                //Set Canvas frame to close on exit
                //canvas.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);

                //Declare FrameGrabber to import output from webcam
                FrameGrabber grabber = new OpenCVFrameGrabber(0);

                //Declare img as IplImage
                IplImage img = null;
                IplImage imgCopy = null;

                try {

                    //Start grabber to capture video
                    grabber.start();

                    while (wCam) {

                        //Set canvas size as per dimentions of video frame.
                        canvas.setCanvasSize(grabber.getImageWidth(), grabber.getImageHeight());

                        //inser grabed video fram to IplImage img
                        img = grabber.grab();

                        if (img != null) {

                            //Flip image horizontally
                            //cvFlip(img, img, 1);
                            imgCopy = img.clone();

                            // Show processed image
                            canvas.showImage(findPlate(img));
                        }
                    }

                    // show new image
                    BufferedImage bfimg = imgCopy.getBufferedImage();
                    ImageIcon imgIc = new ImageIcon(bfimg);
                    jLabImg1.setIcon(new ImageIcon(imagem.redimensionar(imgIc.getImage(),
jLabImg1)));
```

```
                grabber.stop();
                canvas.dispose();

            } catch (Exception e) {
                jTextArea1.append("Problema na webcam \n" + e.getMessage());
                try{
                    grabber.stop();
                    canvas.dispose();
                    wCam = false;
                } catch(Exception e1){

                }
            }
        }
    }
}


private void btnWebCamActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    // verifica se ativa webcam ou não
    // wCam = true  - ativa webcam
    // wCam = false - desativa webcam

    if (wCam == false) {
        wCam = true;

        webCam wb = new webCam();
        wb.start();

    } else {
        wCam = false;
    }

}

// Variables declaration - do not modify
private javax.swing.JButton btnLimpar;
private javax.swing.JButton btnProcessarArq;
private javax.swing.JButton btnProcessarDir;
private javax.swing.JButton btnWebCam;
private javax.swing.JLabel jLabImg1;
private javax.swing.JLabel jLabImg2;
private javax.swing.JLabel jLabImg3;
private javax.swing.JLabel jLabelProcessando;
private javax.swing.JMenu jMenu1;
private javax.swing.JMenu jMenu2;
private javax.swing.JMenuBar jMenuBar1;
```

```
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private static javax.swing.JTextArea jTextArea1;
    private javax.swing.JTable tabela;
   // End of variables declaration
}
```

ocrFiles.java

```
package control;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author carapelli
 */
public class ocrFiles {

    private final String LANG_OPTION = "-l";

    private final String PSM_OPTION = "-psm";
    private final String HOCR_OPTION = "hocr";
    private final String EOL = "\n";
    private String tessPath;
    private String psmOption;
    final static String OUTPUT_FILE_NAME = "TessOutput";
    final static String FILE_EXTENSION = ".txt";
    final static String HTMLFILE_EXTENSION = ".html ";
```

```
/**
     * Creates a new instance of OCR
     *
     * @param tessPath
     * @param psmOption
     */
    public ocrFiles(String tessPath, String psmOption) {
        this.tessPath = tessPath;
        this.psmOption = psmOption;
    }

     public String recognizeText(final List<File> tiffFiles, final String lang) throws Exception {
         File tempTessOutputFile = File.createTempFile(OUTPUT_FILE_NAME, FILE_EXTENSION);
         String outputFileName = tempTessOutputFile.getPath().substring(0,
tempTessOutputFile.getPath().length() - FILE_EXTENSION.length()); // chop the .html/.txt extension

         List<String> cmd = new ArrayList<String>();
         cmd.add(tessPath + "/tesseract");
         cmd.add(""); // placeholder for inputfile
         cmd.add(outputFileName);
         cmd.add(LANG_OPTION);
         cmd.add(lang);
         cmd.add(PSM_OPTION);
         cmd.add(psmOption);

         //if (this.isHocr()) {
         //    cmd.add(HOCR_OPTION);
         //}

         ProcessBuilder pb = new ProcessBuilder();
         pb.directory(new File(System.getProperty("user.home")));
         pb.redirectErrorStream(true);

         StringBuilder result = new StringBuilder();

         for (File tiffFile : tiffFiles) {
             cmd.set(1, tiffFile.getPath());
             pb.command(cmd);
             Process process = pb.start();

             // any error message?

           // this has become unneccesary b/c the standard error is already merged with the standard
output
           //StreamGobbler errorGobbler = new StreamGobbler(process.getErrorStream());
           //errorGobbler.start();
           // any output?
           StreamGobbler outputGobbler = new StreamGobbler(process.getInputStream());
           outputGobbler.start();

             int w = process.waitFor();
             //System.out.println("Exit value = " + w);
```

```
            if (w == 0) {
                BufferedReader in = new BufferedReader(new InputStreamReader(new
FileInputStream(tempTessOutputFile), "UTF-8"));

                String str;

                while ((str = in.readLine()) != null) {
                    result.append(str).append(EOL);
                }

                int length = result.length();
                if (length >= EOL.length()) {
                    result.setLength(length - EOL.length()); // remove last EOL
                }
                in.close();
            } else {
                tempTessOutputFile.delete();
                String msg = outputGobbler.getMessage(); // get actual message from the engine;
                if (msg.trim().length() == 0) {
                    msg = "Errors occurred.";
                }
                throw new RuntimeException(msg);
            }
        }

        tempTessOutputFile.delete();
        return result.toString();
    }

    /**
     * When Runtime.exec() won't.
     * http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-traps.html
     */
    class StreamGobbler extends Thread {

        InputStream is;
        StringBuilder outputMessage = new StringBuilder();

        StreamGobbler(InputStream is) {

            this.is = is;
        }

        String getMessage() {
            return outputMessage.toString();
        }
```

```
        @Override
        public void run() {
            try {
                InputStreamReader isr = new InputStreamReader(is);
                BufferedReader br = new BufferedReader(isr);
                String line = null;
                while ((line = br.readLine()) != null) {
                    System.out.println(line);
                    outputMessage.append(line).append("\n");
                }
            } catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
    }
}
```

searchPlate.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package control;

import com.googlecode.javacv.cpp.opencv_core.CvSize;
import static com.googlecode.javacv.cpp.opencv_core.IPL_DEPTH_8U;
import com.googlecode.javacv.cpp.opencv_core.IplImage;
import static com.googlecode.javacv.cpp.opencv_core.cvCopy;
import static com.googlecode.javacv.cpp.opencv_core.cvGetSize;
import static com.googlecode.javacv.cpp.opencv_core.cvRect;
import static com.googlecode.javacv.cpp.opencv_core.cvSetImageROI;
import static com.googlecode.javacv.cpp.opencv_highgui.cvLoadImage;
import static com.googlecode.javacv.cpp.opencv_highgui.cvSaveImage;
import static com.googlecode.javacv.cpp.opencv_imgproc.CV_THRESH_BINARY;
import static com.googlecode.javacv.cpp.opencv_imgproc.cvThreshold;
import ij.ImagePlus;
import ij.process.ImageProcessor;
import java.awt.image.BufferedImage;
import java.awt.image.Raster;
import java.io.File;
/**
 *
 * @author carapelli
 */
public class searchPlate {
    private int tPixel;
    private static int[] coordenadas;
```

```
private final int  WIDTH = 255; // 270
private final int HEIGHT = 35;  // 60
private final int BLACK  = 0;

public searchPlate(IplImage img) { //(String fileName){

    //IplImage img = cvLoadImage(System.getProperty("user.dir") + "/canny-contours/" + fileName);

    CvSize cvSize = cvGetSize(img);
    coordenadas = new int[2];

    this.tPixel = 0;

    //save image for create rgb
    cvSaveImage("img.jpg", img);
    //load image
    img = cvLoadImage(System.getProperty("user.dir") + "/img.jpg");

    // deleta arquivo
    File file = new File(System.getProperty("user.dir") + "/img.jpg");
    file.delete();

    // bufferImage to imageProcessor
    BufferedImage bfImg = img.getBufferedImage();
    ImagePlus imgPlus = new ImagePlus(null, bfImg);
    ImageProcessor imgPcs = imgPlus.getProcessor();
    int threshold = imgPcs.getAutoThreshold();

    // binariza imagem
    cvThreshold(img, img, threshold, 255, CV_THRESH_BINARY);

    //cvSaveImage("original.jpg", img);

    // create template license
    IplImage lc;
    lc = IplImage.create(WIDTH, HEIGHT, IPL_DEPTH_8U, 1);

    cvSaveImage("tpt.jpg", lc);
     //load template
    IplImage tpt = cvLoadImage(System.getProperty("user.dir") + "/tpt.jpg");

    // apaga arquivo
    file = new File(System.getProperty("user.dir") + "/tpt.jpg");
    file.delete();

   for (int y = 1; y < cvSize.height(); y++) {       // image height
      for (int x = 1; x < cvSize.width(); x++) {   // image wedth
          if ((y + HEIGHT) <= cvSize.height()) {
              if ((x + WIDTH) <= cvSize.width()) {
```

```
                    IplImage copyImg = tpt.clone();

                    cvSetImageROI(img, cvRect(x, y, WIDTH, HEIGHT));
                    cvCopy(img, copyImg);
                    numberPixel(copyImg, x, y);

                    //cvSaveImage("t" + x + y + ".jpg", copyImg);
                } else {
                    x = cvSize.width(); // finaliza interação x
                }
            } else {
                y = cvSize.height(); // finaliza interação y
            }
        }
    }

    /**
    System.out.println("x ->" + coordenadas[0]);
    System.out.println("y ->" + coordenadas[1]);
    System.out.println("total pixel ->" + tPixel);


    cvSetImageROI(img, cvRect(coordenadas[0], coordenadas[1], WIDTH, HEIGHT));
    cvCopy(img, tpt);
    cvSaveImage("placa.jpg", tpt);
    */

}

private void numberPixel(IplImage npImg, int cx, int cy) {
    BufferedImage bfImg = npImg.getBufferedImage();
    Raster raster = bfImg.getRaster();
    int nPixel = 0;

    for (int y1 = 1; y1 < bfImg.getHeight(); y1++) {       // image height
        for (int x1 = 1; x1 < bfImg.getWidth(); x1++) {   // image wedth

            if (raster.getSample(x1, y1, 0) > BLACK) {

                nPixel++;
            }
        }
    }

    /**
    System.out.println("x ->" + cx);
    System.out.println("y ->" + cy);
    System.out.println("total pixel ->" + nPixel);
    */
```

```
        if (nPixel > this.tPixel) {
            this.tPixel = nPixel;
            coordenadas[0] = cx;
            coordenadas[1] = cy;
        }
    }

    public int[] getCoordnadas() {
        return coordenadas;
    }

}
```

imagem.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package control;

import ij.ImagePlus;
import ij.process.ImageProcessor;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.RenderingHints;
import java.awt.image.BufferedImage;
import javax.swing.ImageIcon;
import javax.swing.JLabel;

/**
 *
 * @author fabricio
 */
public class imagem {

    public static ImageIcon resizeImg(ImageProcessor ip, JLabel label){

        ImagePlus iplus = new ImagePlus(null, ip);

        /*
         * cria imagem em miniatura (thumbnail)
         */
        //double scale = (double) label.getHeight() / iplus.getProcessor().getHeight();
        //int sizeW = (int) (iplus.getProcessor().getWidth() * scale);

        /*
         * cria nova imagem ImagePlus e redimensiona (resize) conforme escala e altura
         */
```

```
        //ImagePlus iplusThumbnail = new ImagePlus(null, iplus.getProcessor().resize(sizeW,
label.getHeight()));
        ImagePlus iplusThumbnail = new ImagePlus(null, iplus.getProcessor().resize(label.getWidth(),
label.getHeight()));

        /*
         * retorna imagem redimencionada
         */

        return new ImageIcon(iplusThumbnail.getBufferedImage());
    }


    //   Método retirado da url: http://www.portaljava.com/forum/posts/list/29657.page
    //   Método para redimensionar imagens (criar thubmnails)
    //   Adaptado por Fabrício Carapelli para o projeto rvp
    public static BufferedImage redimensionar(Image image, JLabel label){

        int imageWidth =  label.getWidth(); //image.getWidth(null) / 2;
        int imageHeight = label.getHeight(); //image.getHeight(null) / 2;

        BufferedImage thumbImage = new BufferedImage(imageWidth, imageHeight,
BufferedImage.TYPE_INT_RGB);

        Graphics2D graphics2D = thumbImage.createGraphics();

        graphics2D.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
RenderingHints.VALUE_INTERPOLATION_BILINEAR);

        graphics2D.drawImage(image, 0, 0, imageWidth, imageHeight, null);

        return thumbImage;
    }
}
```

myRenderer.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package control;
import java.awt.Component;
import java.awt.Font;
//import javax.swing.ImageIcon;
import javax.swing.JTable;
```

```
import static javax.swing.SwingConstants.CENTER;
import javax.swing.table.DefaultTableCellRenderer;

/**
 *
 * @author carapelli
 */
public class MyRenderer extends DefaultTableCellRenderer {

        @Override
        public Component getTableCellRendererComponent(JTable table, Object value, boolean
isSelected, boolean hasFocus, int row, int column) {
        Component c = super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row,
column);

        if (isSelected) {
            c.setFont(new Font("arial", Font.BOLD, 12));
            c.setBackground(new java.awt.Color(156, 156, 156));
            c.setForeground(new java.awt.Color(139, 0, 0));
        } else {
            c.setFont(new Font("arial", Font.PLAIN, 12));
            c.setBackground(new java.awt.Color(255, 255, 255));
            c.setForeground(new java.awt.Color(0, 0, 0));
        }
       if (column == 0 || column == 3 || column == 4 || column == 5 || column == 6) {
            setHorizontalAlignment(CENTER);
        }else{
            setHorizontalAlignment(LEFT);
        }

        /**
        // implemet image in cell jtable
        if (value!=null && column == 2) {
            ImageIcon imgIc = (ImageIcon)value;
            this.setIcon(new ImageIcon(imgIc.getImage()));
        }
        */
       return c;
    }
```

Apêndice C - Script de instalação OpenCv no Ubuntu.

install_opencv.sh

```
version="$(wget -q -O - http://sourceforge.net/projects/opencvlibrary/files/opencv-unix | egrep -m1 -o '\"[0-9](\.[0-9])+' | cut -c2-)"

echo "Installing OpenCV" $version

mkdir OpenCV
cd OpenCV

echo "Removing any pre-installed ffmpeg and x264"

apt-get remove ffmpeg x264 libx264-dev

echo "Installing Dependenices"

apt-get install libopencv-dev build-essential checkinstall cmake pkg-config yasm libtiff4-dev libjpeg-dev libjasper-dev libavcodec-dev libavformat-#dev libswscale-dev libdc1394-22-dev libxine-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev python-dev python-numpy libtbb-#dev libqt4-dev libgtk2.0-dev libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils ffmpeg

echo "Downloading OpenCV" $version

wget -O OpenCV-$version.tar.gz http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/$version/opencv-"$version".tar.gz/download

echo "Installing OpenCV" $version

tar -xvf OpenCV-$version.tar.gz

cd opencv-$version
mkdir build

cd build

cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D Qt5Core_DIR=/opt/Qt5.1.1/5.1.1/gcc_64/lib/cmake/Qt5Core -D Qt5Gui_DIR=/opt/Qt5.1.1/5.1.1/gcc_64/lib/cmake/Qt5Gui -D  Qt5Widgets_DIR=/opt/Qt5.1.1/5.1.1/gcc_64/lib/cmake/Qt5Widgets -D Qt5Test_DIR=/opt/Qt5.1.1/5.1.1/gcc_64/lib/cmake/Qt5Test -D Qt5Concurrent_DIR=/opt/Qt5.1.1/5.1.1/gcc_64/lib/cmake/Qt5Concurrent -D
```

```
Qt5OpenGL_DIR=/opt/Qt5.1.1/5.1.1/gcc_64/lib/cmake/Qt5OpenGL -D
WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D
INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D
BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..

make -j 4
sudo make install

sh -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'

ldconfig

echo "OpenCV" $version "ready to be used"
```

Apêndice D – Tabela com informações sobre localização das placas

Tabela 1 - informações referentes a localização das placas na imagem.

Código	Arquivo	Placa	X	Y	Weidth	Height	Data e Hora
	26112002071731.jpg	...1_'-'_1	88	79	240	45	2013-11-28 03:35:17.814
	26112002071636.jpg	H'	273	51	240	45	2013-11-28 03:35:17.814
	26112002071239.jpg	II/1\	192	128	240	45	2013-11-28 03:35:17.814
	26112002070917.jpg	_.....\\,-\\..	231	94	240	45	2013-11-28 03:35:17.814
	26112002071100.jpg	3?-.K_..._..'-E-*	59	98	240	45	2013-11-28 03:35:17.814
	26112002071825.jpg	Lu/..s://\	197	83	240	45	2013-11-28 03:35:17.814
	26112002071806.jpg	"	68	50	240	45	2013-11-28 03:35:17.814
	26112002071209.jpg	L\F/PJ..I13.///.	246	74	240	45	2013-11-28 03:35:17.814
	26112002071336.jpg	§	172	119	240	45	2013-11-28 03:35:17.814
	26112002071507.jpg	//.''../////.	255	100	240	45	2013-11-28 03:35:17.814
	26112002071942.jpg	I.II_W'2/.6.'-/..	101	108	240	45	2013-11-28 03:35:17.814
	26112002071544.jpg	II/-..'-.O4..1.	123	92	240	45	2013-11-28 03:35:17.814
	26112002070913.jpg	\\\\	189	60	240	45	2013-11-28 03:35:17.814
	26112002071831.jpg	////7:._'/.	29	39	240	45	2013-11-28 03:35:17.814
	26112002071454.jpg	j	129	87	240	45	2013-11-28 03:35:17.814
	26112002071108.jpg	/-.-	364	44	240	45	2013-11-28 03:35:17.814
	26112002071228.jpg	|-.:..n_n-3-1	332	99	240	45	2013-11-28 03:35:17.814
	26112002071818.jpg	ILn:_5F.3/.'n5	157	67	240	45	2013-11-28 03:35:17.814
	26112002071532.jpg	....Eff	1	204	240	45	2013-11-28 03:35:17.814
	26112002071325.jpg	IL/.?\\1§\	113	96	240	45	2013-11-28 03:35:17.814
	26112002071923.jpg	.LII']'='-.'§.s.'»	147	32	240	45	2013-11-28 03:35:17.814
	26112002071814.jpg	Z / _	310	72	240	45	2013-11-28 03:35:17.814
	26112002070904.jpg	!KR';311a|	95	96	240	45	2013-11-28 03:35:17.814
	26112002071153.jpg	nfjfi	50	89	240	45	2013-11-28 03:35:17.814
	26112002070750.jpg	JI..\Q,.1.u/I/2.	197	120	240	45	2013-11-28 03:35:17.814
	26112002070810.jpg	I'	229	139	240	45	2013-11-28 03:35:17.814
	26112002071936.jpg	ILNZ./0292.	282	96	240	45	2013-11-28 03:35:17.814
	26112002071214.jpg	IL\Z.3\77.	163	71	240	45	2013-11-28 03:35:17.814
	26112002070747.jpg	.L\\816	2	81	240	45	2013-11-28 03:35:17.814
	26112002071142.jpg	LNO.3./£3	254	92	240	45	2013-11-28 03:35:17.814
	26112002071740.jpg	IA\Y\\.,9'?	153	68	240	45	2013-11-28 03:35:17.814
	26112002071737.jpg	ILCTN7\24V	108	75	240	45	2013-11-28 03:35:17.814
	26112002071323.jpg	IL\\SBT7	161	93	240	45	2013-11-28 03:35:17.814
	26112002070756.jpg	II_NZ/-75	9	86	240	45	2013-11-28 03:35:17.814
	26112002071743.jpg	IGZA/47//.	105	78	240	45	2013-11-28 03:35:17.814
	26112002071315.jpg	ILII-_/CI_.	128	132	240	45	2013-11-28 03:35:17.814
	26112002070855.jpg		384	85	240	45	2013-11-28 03:35:17.814
	26112002071157.jpg	Ki???	70	113	240	45	2013-11-28 03:35:17.814
	26112002071103.jpg	_ _	166	77	240	45	2013-11-28 03:35:17.814
	26112002071012.jpg	L\V.1417	84	84	240	45	2013-11-28 03:35:17.814
	26112002071539.jpg	—	281	53	240	45	2013-11-28 03:35:17.814
	26112002071112.jpg	ILI-N'.,1%a'	56	97	240	45	2013-11-28 03:35:17.814
	26112002071054.jpg	I_IVr\"-JKJJY	272	71	240	45	2013-11-28 03:35:17.814
	26112002071823.jpg	/uII\.3.//3.	214	111	240	45	2013-11-28 03:35:17.814
	26112002071751.jpg	\.\7/	133	118	240	45	2013-11-28 03:35:17.814
	26112002071442.jpg	_.fi/T7'/«	1	190	240	45	2013-11-28 03:35:17.814

Fonte: Autor.

Tabela 1 - 2 - informações referentes a localização das placas na imagem.

Código	Arquivo	Placa	X	Y	Weidth	Height	Data e Hora
	26112002071812.jpg	II.._1%?	263	85	240	45	2013-11-28 03:35:17.814
	26112002071800.jpg	Lk	384	78	240	45	2013-11-28 03:35:17.814
	26112002071348.jpg	IL=J.\6..J	147	90	240	45	2013-11-28 03:35:17.814
	26112002071609.jpg	I4/\\V.52-33	337	114	240	45	2013-11-28 03:35:17.814
	26112002071748.jpg	..	1	183	240	45	2013-11-28 03:35:17.814
	26112002071524.jpg	IAAVI:19'7	254	42	240	45	2013-11-28 03:35:17.814
	26112002071712.jpg	._K;M/\\\'fie\	70	71	240	45	2013-11-28 03:35:17.814
	26112002071048.jpg	K	292	71	240	45	2013-11-28 03:35:17.814
	26112002071425.jpg	I\\\\\.9\/\.	7	94	240	45	2013-11-28 03:35:17.814
	26112002071216.jpg	IINI\6£9.S/	216	93	240	45	2013-11-28 03:35:17.814
	26112002071029.jpg	% % E	333	39	240	45	2013-11-28 03:35:17.814
	26112002071502.jpg	IL\\EUV//-	120	64	240	45	2013-11-28 03:35:17.814
	26112002071246.jpg	._f	290	55	240	45	2013-11-28 03:35:17.814
	26112002071504.jpg	L//.—'_:_._-.'.//g.\\\\	121	40	240	45	2013-11-28 03:35:17.814
	26112002071328.jpg	iL.:3,I_.S-I//\K	199	77	240	45	2013-11-28 03:35:17.814
	26112002071621.jpg		384	116	240	45	2013-11-28 03:35:17.814
	26112002070846.jpg	ILKI6%4£-94	205	103	240	45	2013-11-28 03:35:17.814
	26112002071121.jpg		83	78	240	45	2013-11-28 03:35:17.814
	26112002071911.jpg	ILC.7.//5.	121	83	240	45	2013-11-28 03:35:17.814
	26112002071827.jpg	'rg__§JT-'31'	211	31	240	45	2013-11-28 03:35:17.814
	26112002071810.jpg	'-1	312	100	240	45	2013-11-28 03:35:17.814
	26112002070759.jpg	.jj	37	50	240	45	2013-11-28 03:35:17.814
	26112002071305.jpg	_	323	65	240	45	2013-11-28 03:35:17.814
	26112002071450.jpg	IL\K	194	123	240	45	2013-11-28 03:35:17.814
	26112002071928.jpg	L__I	1	80	240	45	2013-11-28 03:35:17.814
	26112002071634.jpg	Ej	1	194	240	45	2013-11-28 03:35:17.814
	26112002070944.jpg	IL\J.:\53.	118	84	240	45	2013-11-28 03:35:17.814
	26112002071559.jpg		230	61	240	45	2013-11-28 03:35:17.814
	26112002071525.jpg	IL\A-79'9.-.2	2	68	240	45	2013-11-28 03:35:17.814
	26112002071933.jpg		177	82	240	45	2013-11-28 03:35:17.814
	26112002071723.jpg		190	38	240	45	2013-11-28 03:35:17.814
	26112002071043.jpg	/_.../	256	40	240	45	2013-11-28 03:35:17.814
	26112002071527.jpg	__	369	34	240	45	2013-11-28 03:35:17.814
	26112002071302.jpg	LKS"4"5"%	287	33	240	45	2013-11-28 03:35:17.814
	26112002070801.jpg	ILA'r.2\\\§'9I2%	272	73	240	45	2013-11-28 03:35:17.814
	26112002071612.jpg	2*:-*I/I	199	97	240	45	2013-11-28 03:35:17.814
	26112002071458.jpg	I\\\\..2///3.	193	111	240	45	2013-11-28 03:35:17.814
	26112002071128.jpg	T	384	70	240	45	2013-11-28 03:35:17.814
	26112002070939.jpg		207	61	240	45	2013-11-28 03:35:17.814
	26112002070825.jpg	ILNIJ.7:35\	195	84	240	45	2013-11-28 03:35:17.814
	26112002071726.jpg	_.'''._._.___.___.'''u4	384	1	240	45	2013-11-28 03:35:17.814
	26112002071547.jpg	KN_.//I33,/.	184	90	240	45	2013-11-28 03:35:17.814
	26112002071052.jpg	QJ»I.F)////	207	87	240	45	2013-11-28 03:35:17.814
	26112002071146.jpg	7-	25	126	240	45	2013-11-28 03:35:17.814
	26112002071026.jpg	-	105	112	240	45	2013-11-28 03:35:17.814
	26112002071754.jpg	ILA_7\\D7.	278	80	240	45	2013-11-28 03:35:17.814

Fonte: Autor.

Tabela 1 - 3 - informações referentes a localização das placas na imagem.

Código	Arquivo	Placa	X	Y	Weidth	Height	Data e Hora
	26112002071212.jpg	ILN3.%\16.	156	72	240	45	2013-11-28 03:35:17.814
	26112002071718.jpg	L/C)./2.3O	308	61	240	45	2013-11-28 03:35:17.814
	26112002071011.jpg	I\\\\//.2.\\/A.	80	96	240	45	2013-11-28 03:35:17.814

Fonte: Autor.

Apêndice E – Artigo Científico

Reconhecimento de Caracteres em Placas de Veículos Utilizando Processamento de Imagens e Ocr

Fabrício de Oliveira Carapelli, Profª Dra Patricia Bellin Ribeiro, Profº. Me. Patrick Pedreira Silva, Profº. Me. Marcio Henrique Cardim

Curso de Ciência da Computação - Centro de Ciências Exatas e Sociais Aplicadas - Universidade Sagrado Coração (USC) - Bauru – SP

fabriciocarapelli@gmail.com, patriciabellin@yahoo.com.br, patrick.silva@usc.br, mcardim@usc.br

Abstract. *The purpose of this work was to develop an application able of detecting vehicle license plates. Undertake the processing of captured using techniques of image processing, such as binarization and segmentation images. Integrate Java development environment libraries for image processing OpenCV and ImageJ.*

Resumo. Este trabalho teve como objetivo desenvolver um aplicativo capaz de detectar placas de veículos. Realizar o processamento das imagens capturadas utilizando técnicas de processamento de imagens, como binarização e segmentação. Integrar ao ambiente de desenvolvimento Java as bibliotecas de processamento de imagens OpenCV e ImageJ.

1. Introdução

Nas últimas décadas, é notório observar o crescimento expressivo na área de processamento de imagens. Tal fato se explica porque esta área viabiliza quase todos os campos da atividade humana. São múltiplas e distintas as aplicações no campo da medicina, robótica, meteorologia, segurança pública e privada, dentre outros.

Para realizar a localização da placa do veículo na imagem foi implementado dois algoritmos, um utilizando a biblioteca OpenCV, que não se mostrou muito eficiente quando utilizada com fotos de radares, isso devido à qualidade das imagens, então para obter um resultado melhor foi desenvolvido um algoritmo próprio que percorresse toda imagem para localizar a placa, esse algoritmo se mostrou mais eficiente porém levou um pouco mais de tempo para processar cada imagem.

2. Processamento de Imagem

Uma imagem pode ser definida como uma função bidimensional, f (x, y), em que x e y são coordenadas espaciais (plano), e a amplitude de f em qualquer par de coordenadas (x, y) é chamada de intensidade ou nível de cinza da imagem nesse ponto. Assim podemos dizer que uma imagem é digital quando x, y e os valores de intensidade de f são quantidades finitas e discretas, uma imagem digital é composta de um número finito de elementos, cada um com localização e valores específicos (GONZALES, 2010).

Processamento de imagem digital consiste em um conjunto de técnicas para capturar, representar e transformar imagens com o auxílio de um computador, e o emprego dessas técnicas permite extrair e identificar informações das imagens e melhorar a qualidade visual de certos aspectos estruturais, facilitando a percepção humana e a interpretação automática por meio de máquinas (PEDRINI, 2008).

Para o autor Conci (2008) a visão computacional (Processamento de Imagem) nós últimos anos teve um grande desenvolvimento, é uma área que trata da extração de informações das imagens e da identificação e classificação de objetos presente na imagem, e define visão computacional como, “ o domínio da ciência da computação que estuda e aplica métodos que permitam aos computadores “compreenderem” o conteúdo de uma imagem”.

2.1. Etapas de um sistema de processamento de Imagens

Para o autor Pedrini (2008) um sistema de processamento digital de imagens é constituído por um conjunto de etapas, capazes de produzir um resultado a partir do

domínio do problema. A seguir, são descritas resumidamente:

- **Aquisição de Imagens:** etapa onde a imagem é capturada por meio de um dispositivo ou sensor, e assim transforma a imagem em uma representação adequada para o processamento digital, nos dias atuais existem vários dispositivos para aquisição de imagens, como câmeras de vídeo e câmeras fotográficas.
- **Pré processamento:** no processo decorrente da aquisição ou captura da imagem podem ocorrer imperfeições ou degradação da imagem, decorrente de iluminação inadequada ou características dos equipamento de captura, assim esta etapa visa melhorar a qualidade da imagem por meio de técnicas para redução de ruídos ou correção de contraste e brilho.
- **Segmentação:** recorta-se a imagem em regiões disjuntas com algum significado para a aplicação, no caso do reconhecimento de caracteres de placas de veículos, é uma etapa importante. Segundo Pedrini (2008) a etapa de segmentação realiza a extração e identificação de áreas de interesse contidas na imagem, essa etapa é geralmente baseada na detecção de descontinuidades (bordas) ou de similaridades (regiões) na imagem.
- **Representação e Descrição:** estruturas adequadas de representação devem ser utilizadas para armazenar e manipular os objetos de interesse extraídos da imagem, o processo de descrição visa à extração de características ou propriedades que possam ser utilizadas na descrição entre classes de objetos. Essas características são, em geral, descritas por atributos numéricos que formam um vetor de características (PEDRINI. 2008).
- **Reconhecimento e Interpretação:** reconhecimento ou classificação é o processo que atribui um identificador ou rótulo aos objetos da imagem, baseado nas características providas pelos seus descritores,

já o processo de interpretação consiste em atribuir um significado ao conjunto de objetos reconhecidos (PEDRINI, 2008).

A Figura 1 ilustra essas etapas.

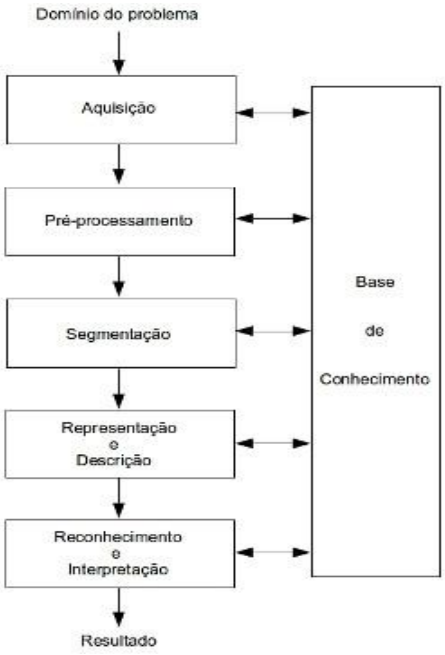

Figura 1 – etapas de um sistema de processamento digital de imagens.

3. Metodologia

Neste capítulo será detalhado o desenvolvimento do aplicativo, bem como a instalação e configuração das ferramentas utilizadas para localização da placa veicular e o reconhecimento dos caracteres.

Foi utilizada a linguagem de programação Java, já que o mesmo permite desenvolver e implantar aplicativos e serviços incríveis de maneira eficiente. Com ferramentas abrangentes, um ecossistema sólido e um desempenho eficiente, o Java oferece a portabilidade de aplicativos mesmo entre os ambientes computacionais mais diferentes.

Conforme proposta do projeto segue abaixo descrição das etapas do aplicativo:

- **Aquisição da imagem:** o aplicativo foi construído de forma que as imagens possam ser adquiridas de duas formas, em tempo real via *webcam* ou qualquer outro dispositivo que possa capturar a imagem em tempo real, ou ainda utilizar imagens capturas previamente; por exemplo, imagens feitas por radares moveis ou estáticos.

- **Pré-processamento:** nessa etapa são utilizadas algumas técnicas para tratamento da imagem, melhorando suas condições antes da localização do objeto. São utilizadas técnicas de detecção de bordas, conversão de formato, RGB (com três canais de cores – vermelho-verde-azul) para escala de cinza de um único canal e binarização.

- **Localização da placa:** a localização da placa é realizada de duas maneiras, quando a imagem é obtida em tempo real, são localizados os contornos da imagem através dos métodos *cvFindContours, CvPoint e o cvRectangle* que gera um retângulo vermelho em torno do objeto localizado. Agora, para as imagens previamente capturadas a localização é realizada percorrendo toda imagem com um templete com as dimensões aproximada do objeto.

- **Validação:** para realizar a validação do objeto localizado, tanto nas imagens capturadas em tempo real, como nas imagens previamente capturadas, é utilizado parâmetros com as dimensões aproximadas da placa, como altura e largura.

- **Segmentação:** nesta etapa é realizada processos de morfologia matemática, como erosão e dilatação da imagem a fim de remover as áreas de ruídos presente na imagem, após estes processos é realizado a segmentação da imagem. Para segmentar a imagem foi criado um algoritmo que varre a imagem procurando os intervalos entre os objetos, quando esses intervalos são localizados as coordenadas são armazenadas em um vetor para serem

utilizada no recorte do objeto.

- **Leitura da placa:** aqui os objetos recortados na segmentação são passados um por vez como parâmetros para a biblioteca Tesseract OCR para que a mesma possa reconhecer os caracteres e retorná-los no formato alfanumérico.

4. Análise dos resultados

O objetivo deste projeto foi desenvolver um aplicativo para localizar placas de veículos em imagens capturadas de radares utilizando processamento de imagens.

Para realizar a localização da placa foi implementado dois algoritmos, um utilizando a biblioteca openCv e outro utilizando a biblioteca imageJ.

O primeiro algoritmo implementado utilizando a biblioteca openCv, conseguiu localizar as placas nas imagens em um menor tempo, porém no banco de imagens utilizado para teste ele não se mostrou muito eficiente conseguindo localizar em um total de 100 imagens de veículos apenas 30 placas totalizando 30% de acerto.

Para obter uma taxa de acerto melhor, foi implementado um algoritmo utilizando outra biblioteca (imageJ), que para localizar a placa na imagem, utiliza um templete nas dimensões aproximada da placa dos veículos e percorre toda imagem para localizar o objeto, como o algoritmo tem que percorrer toda a imagem ele lava um tempo maior para processar cada imagem em relação ao primeiro algoritmo, porém se mostrou muito eficiente, no mesmo banco de teste contento 100 imagens de veículos conseguiu localizar 70 placas totalizando uma taxa de acerto de 70%.

A Figura 1, mostra a placa do veículo localizada usando algoritmo criado utilizando a biblioteca imageJ.

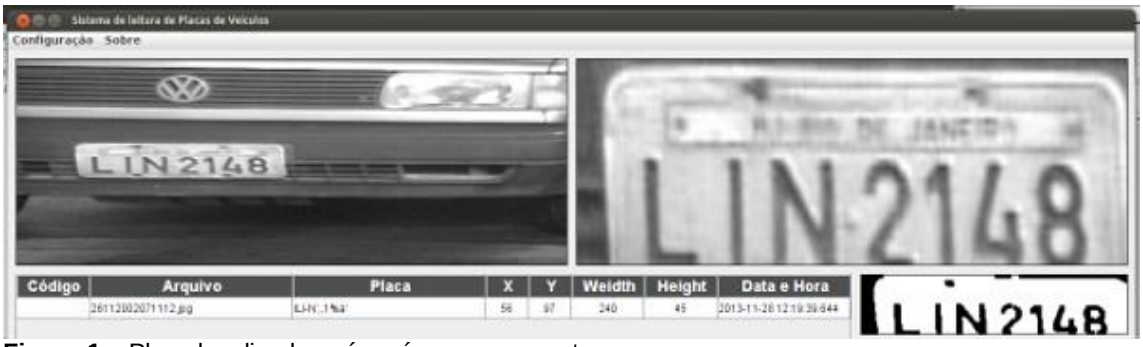

Figura 1 – Placa localizada após pré-processamento.

5 . Conclusão

Este estudo teve a finalidade de apresentar um aplicativo que permita localizar placa de veículos a partir de uma imagem. Para este processo, as imagens originais passam por várias etapas até a localização da placa, pois as imagens obtidas por radares nem sempre está em boas condições.

Com base nos resultados obtidos, verificou-se que o aplicativo teve um bom desempenho conseguindo uma taxa de acerto de 70%, levando em consideração que a base imagens utilizada para teste continha 100 imagens de veículos e o aplicativo conseguiu localizar em 70 imagens as placas dos veículos. Apesar do bom resultado obtido, ainda é possível melhorar o aplicativo, para que o mesmo seja capaz de localizar a placa do veículo e reconhecer os caracteres contidos nela.

É importante ressaltar que fatores externos também interferem no resultado, como exemplo, a qualidade da imagem da placa, as condições meteorológicas em regiões que apresentam neblina, o próprio movimento do veículo quando estiver em alta velocidade e devido à ausência de iluminação adequada.

Considerando que o objetivo principal citado no início do trabalho foi cumprido, conclui-se que o resultado final do projeto é encorajador necessitando apenas de melhorias do algoritmo para que o mesmo possa localizar a placa, segmentar a

imagem da placa e reconhecer os caracteres segmentados.

6. Referências

Conci, Aura; Eduardo Azevedo, Fabiana R. Leta. **Computação gráfica teoria e prática.** Rio de Janeiro: Elsevier, 2008.

Eikvil, Line. **Optical Character Recognition.** Norsk Regnesentral, 1993.

Gonzales, Rafael C.; Richard E. Woods. **Processamento digital de imagens.** São Paulo: Pearson Education do Brasil Ltda., 2010.

Nascimento, Jean Dias. **Detecção de Reconhecimento de placa automotiva com baixo custo.** Brasília: Centro Universitário de Brasília – UniCEUB, 2012.

Pedrine, Hélio.; William Robson Schwartz. **Análise de imagens digitais: princípios, algoritimos e aplicações**. São Paulo: Thomson Learning, 2008.