Localização de Placas de Licenciamento Veicular em tempo real utilizando OpenCV com CUDA

Jorge Fernando Silva Pereira Filho*
Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Rua Emídio dos Santos, S/N
Barbalho, Salvador, Bahia
jorgepereira@ifba.edu.br

Antonio Carlos dos Santos Souza†
Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Rua Emídio dos Santos, S/N
Barbalho, Salvador, Bahia
antoniocarlos@ifba.edu.br

RESUMO

A localização automatizada de placas veiculares é parte fundamental de um Sistema de Reconhecimento de Placas de Licenciamento Veiculares (SRPLV). Sua importância vem do fato que a não localização da região que contenha a placa veicular inviabilizará as demais etapas de um SRPLV. A proposta deste trabalho é criar um algoritmo de localização de placas veiculares em tempo real utilizando a biblioteca de código aberto *OpenCV* com a plataforma de computação paralela CUDA. A solução foi desenvolvida utilizando o algoritmo de detecção de bordas de Canny seguido da aplicação de morfologias matemáticas. A partir dos resultados obtidos, foi possível perceber que a utilização de GPU foi fundamental nesse processo para permitir uma redução de 54% do tempo de processamento, se comparada com a implementação do algoritmo utilizando somente funções executadas em CPU. Esse fator possibilitou o uso do algoritmo em situações reais que exijam processamento em tempo real. Além disso, o sistema proposto apresentou uma taxa de acerto de 95,5% na localização de placas veiculares. O *dataset* utilizado para validar os resultados foi composto por 100 fotografias de veículos parados ou em movimento além de 7 filmagens contendo, no total, 101 veículos trafegando em vias públicas de grande fluxo.

Keywords

Localização Automática de Placas Veiculares, Visão Computacional, Processamento de Imagens.

1. INTRODUÇÃO

Segundo relatório do Instituto Nacional de Ciência e Tecnologia [12], nos últimos 10 anos (período de 2001 até 2012) houve um incremento da ordem de 28,5 milhões de veículos à frota brasileira de automóveis e motos, o que representa um crescimento superior a 138,6% se comparados os dois anos (2001 até 2012). Com o crescente aumento da frota nacional, faz-se necessário o estudo de meios cada vez mais eficientes para garantir o controle de tráfego, fiscalização de infrações além de um combate efetivo às ocorrências de furtos e roubos veiculares. Souza e Susin [16] afirmam que os sistemas propostos para solucionar tal problemática se agrupam em dois tipos: os que necessitam de instalação no veículo de equipamentos de identificação e os que fazem o reconhecimento pela placa de licenciamento veicular.

No primeiro grupo, é necessária a instalação de dispositivos para a identificação dos veículos, seja por meio de equipamentos que utilizam o sistema de posicionamento global (GPS), etiquetas de rádio frequência (RFID) ou códigos de barra. Os sistemas participantes deste grupo tem como principal vantagem a alta taxa de precisão na identificação, porém necessitariam ser instalados em toda a frota nacional. Esse último fator inviabilizando sua utilização imediata devido aos elevados custos envolvidos para sua implementação e dificuldade de abranger tamanha quantidade de automóveis e motocicletas.

No segundo grupo estão os sistemas que fazem uso de técnicas computacionais para identificação da placa de licenciamento veicular. Tais sistemas não necessitam de intervenção física, ou seja, instalação de qualquer equipamento de identificação adicional no veículo. Em contrapartida, não gozam da mesma precisão dos sistemas citados anteriormente, já que as imagens das placas que serão identificadas são obtidas em ambientes não controlados, logo, suscetíveis a ações climáticas como a presença de neblina, ocorrência de chuva ou ausência de iluminação adequada para obtenção de imagens passíveis de identificação. Outro fator que poderá dificultar a identificação é o mau estado de conservação das placas de licenciamento, com caracteres apagados, avariados, bem como placas com sujeira em excesso; comumente encontradas nos veículos em circulação. O sistema proposto nesse trabalho de conclusão de curso está inserido nesse último grupo.

Os sistemas de reconhecimento de placas de licenciamento veicular são compostos por 3 fases:

*Aluno do curso de Análise e Desenvolvimento de Sistemas (ADS)

†Mestre em Ciência da Computação e Professor do Curso de Análise e Desenvolvimento de Sistemas

1. Aquisição da imagem
2. Localização e extração da placa de licenciamento veicular
3. Reconhecimento dos caracteres

O foco do presente trabalho é o desenvolvimento de um algoritmo que funcione em tempo real para localização e extração de placas de licenciamento veicular. A opção por desenvolver uma solução em tempo real vem da necessidade de processar dados de maneira instantânea vindos de fluxos automotivos capturados em velocidades variadas, sejam em grandes rodovias, pedágios ou estacionamentos. Em tais situações, pode ser necessário o processamento dos *frames* de um vídeo no intervalo de exibição deles, o que representaria um tempo máximo de processamento de 62ms em uma vídeo capturado a 15 por segundos. Tal algoritmo foi desenvolvido utilizando a biblioteca de código aberto *OpenCV* [2] compilada com o *framework* CUDA [10]. A motivação para a utilização da *OpenCV* com CUDA se dá pela grande capacidade computacional presente nas unidades de processamento gráfico (GPUs) [11]. Tal poder computacional é acessível através da linguagem CUDA e as funções implementadas em tal linguagem e presentes na *OpenCV* facilitam o uso da GPU no processamento de imagens, minimizando assim os esforços de programação [4].

Para efeito de validação, a técnica desenvolvida será analisada em relação ao seu tempo de processamento sob duas perspectivas: desempenho para uma solução que utilize apenas o processamento da CPU e para uma solução híbrida com processamento em CPU e GPU. O objetivo desta análise é avaliar o ganho de desempenho obtido com o uso da GPU na localização das placas veiculares. Além disso o algoritmo será analisado também sob a perspectiva da precisão, comparando os resultados obtidos com o de outros algoritmos presentes na literatura.

O presente trabalho está estruturado da seguinte forma:

Seção 2: apresenta os principais trabalhos relacionados à área de localização e extração de placas de licenciamento veicular;

Seção 3: mostra a fundamentação teórica relativa ao processamento de imagens digitais necessária para o entendimento da solução proposta nesse artigo;

Seção 4: descreve o algoritmo e a estratégia utilizada para sua elaboração;

Seção 5: relata os resultados obtidos pelo algoritmo apresentando os tempos médios de execução, a precisão na localização e o *dataset* utilizado para obtenção de tais resultados;

Seções 6 e 7: apresenta as conclusões, trabalhos futuros e referências utilizadas no projeto.

2. TRABALHOS RELACIONADOS

Os trabalhos seguintes exemplificam pesquisas na área de localização de placas de licenciamento veicular, sendo eles:

Sales [13] desenvolveu uma solução computacional para a localização de placas veiculares utilizando, sequencialmente, as seguintes técnicas: binarização por limiares globais , por detecção de bordas e morfologias matemáticas [7] para a localização de regiões candidatas a placa veicular e, para validação dessas regiões, foi utilizado uma rede neural. Nesse trabalho, foram desenvolvidos um total de dez soluções distintas, sendo o último formado pela utilização dos nove algoritmos desenvolvidos no trabalho, executados em paralelo com o intuito de maximizar a precisão do algoritmo. Este algoritmo atingiu uma alta taxa de acerto, com resultados precisos em 98% das 906 imagens que compuseram o *dataset* utilizado para validação dos resultados. Em contrapartida, o tempo de execução também se tornou elevado, com um tempo médio de processamento por imagem de 1279 ms, o que na prática inviabiliza a utilização de tal abordagem em um sistema de tempo real.

Em Lee e Hung [8] foi desenvolvido um algoritmo de localização de placas fazendo uso de operadores morfológicos, precedido da utilização da *Discrete Wavelet Transform (DWT)* [14] e, por fim, uma rede neural para a validação das regiões candidatas. O algoritmo utilizou como operador de gradiente o algoritmo de Prewitt. A validação do algoritmo foi realizada em uma base de dados composta por 600 imagens e obteve uma taxa de acerto superior a 95% . Assim como o algoritmo citado anteriormente, os resultados de processamentos não foram satisfatórios para um sistema de tempo real já que, neste trabalho, o tempo de processamento médio ultrapassou 2,5 segundos.

Baggio et al., em seu livro *Mastering OpenCV with Practical Computer Vision Projects* [1], propuseram um algoritmo de localização de placas veiculares para veículos espanhóis capturados a partir de câmeras infravermelho. Imagens capturadas dessa maneira possuem menos ruídos ou objetos que possam atrapalhar a localização. O algoritmo utiliza o algoritmo de Sobel para detecção de contornos e a operação morfológica de fechamento [7] para realce de regiões candidatas. Para validação das regiões candidatas, são extraídas somente as regiões que possuem as dimensões proporcionais as de uma placa de licenciamento e validadas em uma rede neural. Por disponibilizar o código fonte do algoritmo para implementação, maiores detalhes sobre o desempenho e precisão desse algoritmo serão mostrados com maiores detalhes na seção 5.6.

Nascimento [9] propôs o desenvolvimento de uma solução de visão computacional para detecção e reconhecimento de placas automotivas capturadas com câmeras de baixo custo. O algoritmo de localização de placas foi desenvolvido com a biblioteca *OpenCV* e o reconhecimento de caracteres com a biblioteca de código aberto *Tesseract OCR* [15]. Na fase de pré-processamento a imagem foi convertida para escala de cinza e em seguida aplicado um filtro Gaussiano para suavização. Na fase de localização de objetos, foi utilizado o algoritmo de detecção de bordas de Canny seguido de um algoritmo de busca de contornos. Por último, foi desenvolvido um algoritmo para validação das regiões candidatas com base nas dimensões de uma placa de licenciamento veicular. Essa solução apresentou tempo médio de processamento de 60ms para vídeos capturados na resolução de 480p e de 95ms a 105ms para vídeos de resolução 1080p. A pre-

cisão na detecção de placas foi de 100% no vídeo de menor resolução e de 63,3% a 70,5% nos vídeos de maior resolução. Apesar de apresentar resultados animadores quanto ao tempo de processamento, foram utilizados somente 49 imagens na composição do *dataset* de teste do algoritmo, evidenciando assim que a amostragem utilizada para obtenção de tais dados não possui quantidade suficiente para validar o desempenho e a precisão de tal algoritmo. Além disso, no vídeo com resolução de 720p foram utilizadas somente duas amostras e nos dois vídeos de 1080p outras 47.

Um sistema de transporte inteligente em tempo real foi proposto por Wen et al. [18]. É utilizado um algoritmo de segmentação chamado *Sliding Concentric Windows (SCW)* em conjunto com uma rede neural para a detecção de caracteres. Testado em condições de capturas distintas, o algoritmo apresentou um desempenho satisfatório na localização das placas veiculares reconhecendo 96.5% das 1334 imagens disponibilizadas em sua base de teste. No entanto, apesar do algoritmo ser proposto para funcionar em tempo real, seu tempo de processamento para extração da região candidata alcançou 111ms o que inviabiliza seu uso na localização de placas veiculares em tempo real.

Freire e Maia [6] propuseram um detector de placas veiculares a partir da adaptação do método de detecção de Viola-Jones [17] implementado na biblioteca *OpenCV*. O algoritmo apresenta alta taxa de acerto na detecção de placas, conseguindo reconhecer 348 das 350 imagens contidas em seu *dataset*, o que corresponde a uma taxa de acerto de 99,4%, em um tempo médio de execução de aproximadamente 91ms . Apesar do tempo de execução ser pequeno, não atende aos requisitos para execução em tempo real, visto que mesmo em gravações a 15 *frames* por segundo, o algoritmo não seria capaz de processar todas as imagens do vídeo e assim placas veiculares poderiam deixar de ser localizadas.

Araujo et al. [5] propuseram um algoritmo de localização de placas veiculares a partir de um esquema robusto de operadores morfológicos. O algoritmo baseou-se em características presentes em placas de licenciamento veicular, como o contraste existente entre os caracteres presentes na placa e o seu fundo ou as dimensões desses caracteres ou do conjunto deles. Essa solução conseguiu extrair corretamente a placa veicular em 88% das 324 imagens presentes na sua base de dados e o tempo de processamento médio para execução do algoritmo não foi citado no artigo. Apesar de 88% ser um percentual elevado, outros trabalhos na literatura se mostraram mais precisos.

Em geral, as soluções propostas na literatura apresentaram um alto grau de precisão na localização de regiões candidatas obtendo, em sua maioria, percentuais superiores a 90% na taxa de acerto. Em contrapartida, é possível observar que o tempo de execução apresentado supera, consideravelmente, os valores aceitáveis para uma aplicação em tempo real.

3. FUNDAMENTAÇÃO TEÓRICA

Nessa seção serão abordados os conceitos necessários para um correto entendimento do algoritmo desenvolvido.

3.1 Definição de Imagem Digital

Uma imagem analógica é definida como uma função bidimensional de intensidade luminosa $f(x, y)$, onde x e y representam as coordenadas espaciais e o valor da função f em qualquer ponto (x, y) representa a cor no determinado ponto [7]. Já uma imagem digital pode ser entendida como uma matriz de MxN pontos, sendo que cada ponto de tal matriz é conhecido com *pixel*. Entende-se como resolução de uma imagem digital as dimensões da matriz que a definem ou a quantidade de *pixels* na vertical e na horizontal. Todas as imagens utilizadas nesse trabalho se enquadram na definição de imagem digital.

3.2 Classificação das Imagens Digitais

Em geral, podemos classificar as imagens digitais a partir da relação do seu número de cores. Essa diferença vem do número de *bits* utilizado no momento da captura da imagem. As imagens podem ser classificadas como em tons de cinza, binárias ou coloridas.

Imagens em Tons de Cinza: Cada *pixel* da matriz é associado a um valor. Esse valor, nas imagens em escala de cinza (monocromáticas), é representado por um número no intervalo que vai de 0 (*pixel* na cor preta) até 255 (*pixel* na cor branca). Um exemplo de imagem em tons de cinza pode ser visto na Figura 1a.

Imagens Binárias: Um tipo especial de imagens monocromáticas são as imagens binárias. Em tais imagens, somente valores binários podem ser assumidos em cada *pixel*, ou seja, 0 para *pixels* pretos e 1 para *pixels* brancos. Um exemplo de imagem binária é dado na Figura 1b.

Imagens Coloridas: Para imagens coloridas no modelo RGB (*red, green, blue*), a cor de cada *pixel* é descrita pelas proporções de vermelho, verde e azul que a compõe. Essa divisão é feita porque a maior parte das cores visíveis pelo olho humano pode ser representada como uma combinação das três cores primárias: vermelho (R), verde (G) e azul (B). Assim, um vetor composto por 3 grandezas é utilizado para representar as cores de cada e a intensidade de cada cor é descrita numa escala que vai de 0 a 255, onde 0 representa a ausência da cor e 255 a cor em sua maior intensidade.

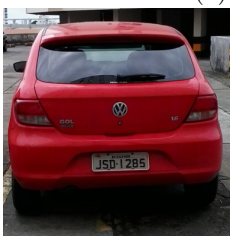
Além do RGB podemos citar outros modelos largamente utilizados como o CMYK, HSB e YUV [7]. Uma imagem colorida é mostrada na Figura 1c.

3.3 Sistemas de Processamento Digital

Entende-se como processamento de imagem digital, a manipulação de uma imagem feita por um computador. Esse processo tem como entrada uma imagem e como saída outra imagem com as devidas melhorias implementadas sobre a entrada. A Figura 2 demonstra os passos fundamentais definidos por Gonzalez e Woods [7] para um sistema de processamento de imagens digitais. Esse passos serão explicados detalhadamente no decorrer dessa seção.

Aquisição: processo que permite a aquisição por meio de um dispositivo ou sensor, de uma cena real tridimensional, e a sua posterior conversão em uma imagem digital. Um exemplo de dispositivos que realizam tal tarefa seriam as câmeras digitais.

(a) Tons de Cinza (b) Binária



(c) Colorida

Figura 1: Classificação das Imagens Digitais

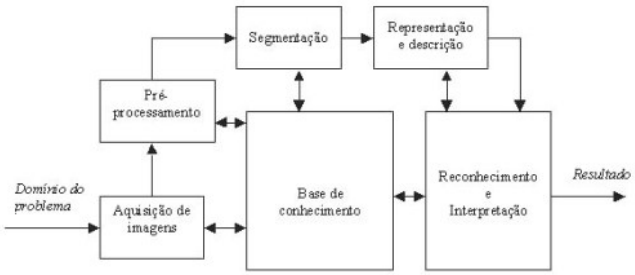


Figura 2: Passos Fundamentais em um Sistema de Processamento de Imagens Digitais [7]

Pré-processamento: nesta fase, procura-se aplicar melhorias com o intuito de aumentar as chances de sucessos nos passos seguintes. É nessa etapa que são aplicadas técnicas para melhorias de contrastes, tratamento de ruídos e isolamento de regiões promissoras [7].

Segmentação: na etapa de segmentação a imagem de entrada é dividida em partes. O processo de segmentação automática é considerada uma das tarefas mais difíceis no processamento de imagens digitais e o desenvolvimento de um algoritmo robusto em tal etapa é de fundamental importância para obter sucesso na resolução do problema proposto [7]. Algoritmos de detecção de pontos, de bordas e de formas geométricas são exemplos de técnicas de segmentação.

Representação e Descrição: após a etapa de segmentação são gerados dados em forma de *pixels*, que podem corresponder tanto às fronteiras de uma região, como todos os pontos contidos nela. A escolha pela representação dada pela fronteira, pelo conteúdo interno ou ambas irá depender do objetivo pretendido. Fronteiras são interessantes quando as características da forma externa, como os cantos ou dimensões são informações relevantes. Já o interesse pela representação da região é adequada quando texturas ou formas internas são importantes [7].

A descrição ou seleção de características procura extrair informações que sejam importantes para discriminar objetos distintos ou que resultem em informações importantes [7].

Um exemplo de representação é a escolha pelas dimensões de largura e altura da borda das regiões retangulares candidatas a serem placas automotivas. Já a descrição pode ser exemplificada no trabalho desenvolvido com a escolha da relação de aspecto entre as dimensões anteriores para descrição dos objetos selecionados.

Reconhecimento e Interpretação: A última etapa envolve o reconhecimento, que nada mais é que a atribuição de um rótulo a um determinado objeto, e interpretação que pode ser entendido como a atribuição de significado a um conjunto de objetos reconhecidos [7].

3.4 Técnicas de Pré-Processamento de Imagem

As técnicas de realce ou filtragem têm como principal objetivo processar uma imagem de maneira a atenuar características importantes para uma aplicação específica. Operações de filtragem podem ser realizadas em dois domínios: espaço e frequência [7]. Domínio espacial se refere ao conjunto de *pixels* que compõem uma imagem e os métodos que agem em tal domínio atuam diretamente sobre estes *pixels*. As técnicas que atuam no domínio da frequência se baseiam na modificação da Transformada de Fourier na imagem. Existem casos em que ambas as abordagens são combinadas. Nesta seção serão abordadas somente as técnicas de filtragem no domínio espacial devido as técnicas utilizadas nesse trabalho atuarem em tal domínio.

3.4.1 Limiarização de Imagens

Limiarização é o processo de conversão de uma imagem em tons de cinza para uma imagem com somente dois níveis de intensidade. Uma imagem limiarizada g é definida como:

$$g(x,y) = \begin{cases} 1 & \text{se } f(x,y) > T \\ 0 & \text{se } f(x,y) \leq T \end{cases}$$

Onde:

f**:** imagem em níveis de cinza;

T**:** limiar que separa os dois valores possíveis;

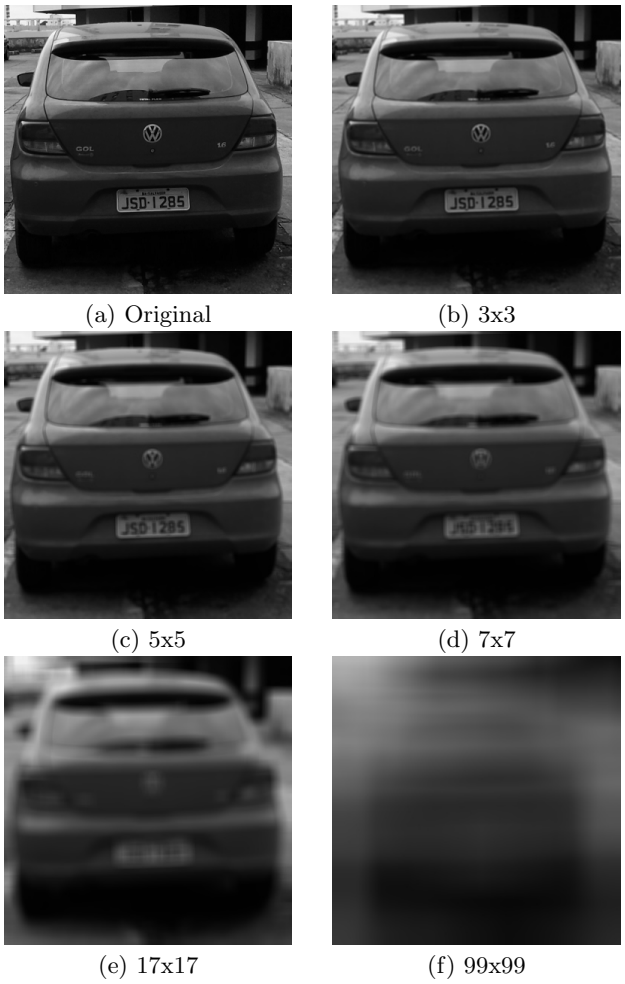
x **e** y**:** coordenadas da coluna e linha respectivamente;

A determinação de T é de fundamental importância para obtenção de bons resultados. A escolha de um limiar incorreto pode gerar a perda de representação de regiões promissoras. A Figura 1b mostra o resultado de um processo de limiarização (*Threshold*) aplicado na Figura 1a.

3.4.2 Suavização de imagens

O objetivo de utilizar filtros de suavização é a obtenção do efeito de borramento e a redução de ruídos. Dentre os variados motivos para seu uso podem ser destacadas: a remoção de pequenos detalhes antes da extração de objetos maiores e a correção de pequenas descontinuidades em linhas ou curvas.

Uma das abordagens utilizadas para a construção de um filtro de suavização é a utilização da média dos *pixels* da vizinhança para substituir o valor do *pixel* selecionado. Para isso são definidas máscaras, que nada mais são que matrizes bidimensionais quadradas, com valores ímpares de dimensão preferencialmente.



(a) Original (b) 3x3

(c) 5x5 (d) 7x7

(e) 17x17 (f) 99x99

Figura 3: Aplicação da suavização mediana por dimensão das máscaras

Quando maior a máscara utilizada mais perceptível fica o efeito de borramento. A Figura 3 mostra a aplicação da suavização em uma imagem com diferentes tamanhos de máscaras.

3.5 Técnicas de Segmentação de Imagens

Segmentar uma imagem se refere ao ato de separar ou subdividir uma imagem em regiões que a compõem. A definição das regiões que serão subdivididas irá depender do contexto que está inserido, ou seja, a segmentação somente deverá parar quando os objetos de interesse estiverem devidamente isolados. Pela dificuldade envolvida no isolamento de tais objetos, a segmentação é uma das atividades que apresentam maior complexidade ao serem automatizadas. Com isso, um processo de segmentação bem sucedido é primordial para a execução com sucesso da etapa de reconhecimento. Todas as técnicas apresentadas em seguida são aplicadas sobre imagens em tons de cinza.

3.5.1 Detecção de pontos isolados

A detecção de pontos isolados pode ser feita medindo a diferença de nível de cinza assumido em um *pixel* R e os 8 *pixels* vizinhos através da delimitação de uma limiar T. Caso $|R| > T$ é dito que o *pixel* foi encontrado. A ideia é que o nível de cinza de um ponto isolado será completamente diferente de sua vizinhança para que ele seja identificado.

3.5.2 Detecção de linhas

As linhas presentes em uma imagem podem ser orientadas horizontalmente, a 45°, verticalmente ou a -45°. Tais linhas podem ser representadas em máscaras, estando uma máscara de detecção de linha horizontal exemplificada na Figura 4a, uma máscara de uma linha a 45° pela Figura 4b, na direção vertical na Figura 4c e com um inclinação de -45° pela máscara apresentada na Figura 4d.

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$$

(a) Horizontal (b) 45°

$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

(c) Vertical (d) -45°

Figura 4: Máscaras para linhas

3.5.3 Detecção de bordas

Embora a detecção de pontos isolados e linhas seja de suma importância quando se fala em segmentação, a detecção de bordas é, sem dúvidas, a abordagem mais significativa para a detecção de descontinuidades. Isso porque, em aplicações práticas que normalmente buscam padrões, pontos e linhas finas isoladas não possuem o mesmo nível de aplicabilidade que a detecção de bordas . Define-se borda como o limite entre duas regiões cujo nível de cinza predominante apresenta razoável diferenciação. Em uma imagem, as bordas caraterizam os contornos dos objetos e por esse motivo são úteis no processo de segmentação de imagens e localização de regiões candidatas. Dentre os muitos operadores disponíveis, dois se destacam pela constante utilização na literatura pesquisada e por esse motivo foram escolhidos para estudo, são eles: Sobel e Canny.

Detecção de Bordas por Sobel : O operador de Sobel consiste em duas máscaras, uma para bordas verticais como visto na Figura 5a e outro para bordas horizontais exemplificado na Figura 5b. Comumente tais máscaras são chamados de operadores de Sobel. Esses operadores atuam aproximando a magnitude do gradiente com a diferença de valores ponderados dos níveis de cinza da imagem.

Detecção de Bordas por Canny: Canny [3] criou um operador com o objetivo de localizar bordas mesmo em ima-

$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$

(a) Máscara vertical (b) Máscara Horizontal

Figura 5: Operadores de Sobel

gens que apresentam ruídos. O filtro proposto por ele é multi-passo e utiliza a convolução gaussiana juntamente com a primeira derivada.

(a) Borda Normal

(b) Uniformização de Gauss

(c) Primeira Derivada

Figura 6: Processo de detecção de borda por Canny

Seus passos podem ser vistos na Figura 6. A Figura 6a representa um exemplo de borda de uma dimensão. Aplicando uma convolução Gaussiana (filtro de suavização) é obtida uma variação contínua do valor inicial e final, esse passo pode ser visto na Figura 6b. Por último, é aplicada a primeira derivada em relação ao eixo x permitindo a obtenção do resultado mostrado na Figura 6c, onde, a inclinação máxima indicará o máximo da nova função em relação a original (Figura 6a).

3.6 Morfologias Matemáticas

Gonzalez e Woods [7] definem morfologia matemática como sendo uma ferramenta para a extração de componentes de imagens que sejam úteis para a representação e descrição da forma de uma região, como as fronteiras pertencentes a ela.

3.6.1 Dilatação

Dilatação é um operador morfológico que tem por objetivo aumentar a área geométrica (dilatar) de um objeto. Esse aumento é regido por um elemento estruturante que desliza sobre um conjunto de pontos, dilatando assim sua vizinhança na proporção do elemento estruturante escolhido. Seu uso pode ser útil para o preenchimento de pequenos intervalos vazios ou lacunas [7].

3.6.2 Erosão

Assim como a dilatação, a erosão também utiliza um elemento estruturante. No entanto, a erosão age de maneira inversa à dilatação, ou seja, ao deslizar sobre um conjunto de pontos são retiradas informações da sua vizinhança (erodindo). Seu uso, obviamente, é o contrário da dilatação, servindo para eliminar pequenos detalhes não desejáveis, como ruídos, ou eliminar ligações estreitas entre regiões maiores [7].

3.6.3 Abertura

Uma erosão seguida de uma dilatação é o que define a operação morfológica denominada abertura. Seu uso geralmente é útil quando se deseja suavizar os contornos de uma objeto, removendo assim contornos estreitos ou pequenas saliências [7].

3.6.4 Fechamento

Invertendo a ordem de execução das operações morfológicas da abertura surge a operação morfológica definida como fechamento, ou seja, a aplicação de uma dilatação seguida de uma erosão. É útil para fundir separações estreitas entre objetos, remover pequenos buracos e preencher lacunas em contornos [7].

4. LOCALIZAÇÃO DE PLACAS VEICULARES EM TEMPO REAL

É abordada nessa seção a metodologia aplicada para solucionar o problema de localização de placas veiculares em tempo real. O algoritmo visa extrair de um vídeo ou imagem, a região que contenha a placa de licenciamento.

Pela característica escolhida de ser um sistema em tempo real, o algoritmo necessitou ser desenvolvido a partir de uma solução simples, precisa e principalmente com baixo tempo de processamento. O tempo de processamento não poderia superar 66.6 ms, tempo médio de exibição de cada *frame* em um vídeos gravado a 15 *frames* por segundos (FPS), preferencialmente sendo executado próximo a 33 ms para ser possível a execução no intervalo de exibição de cada *frame* em um vídeo gravado a 30 FPS.

A partir da análise dos trabalhos relacionados, apresentados na Seção 2, foi possível identificar as diversas metodologias propostas para a localização de placas de licenciamento, assim como o tempo de processamento necessário e o resultado na localização correta de regiões candidatas. A metodologia que apresentou melhor relação entre essas duas últimas características foram as baseadas em operações morfológicas. Por esse motivo, essa abordagem foi a escolhida para o desenvolvimento da solução aqui proposta.

O algoritmo pode ser dividido em três partes, são elas:

1. Pré-processamento
2. Localização de regiões candidatas
3. Validação de regiões candidatas

Uma visão global do algoritmo pode ser vista na Figura 7. O algoritmo detalhado pode ser visto nas seções seguintes.

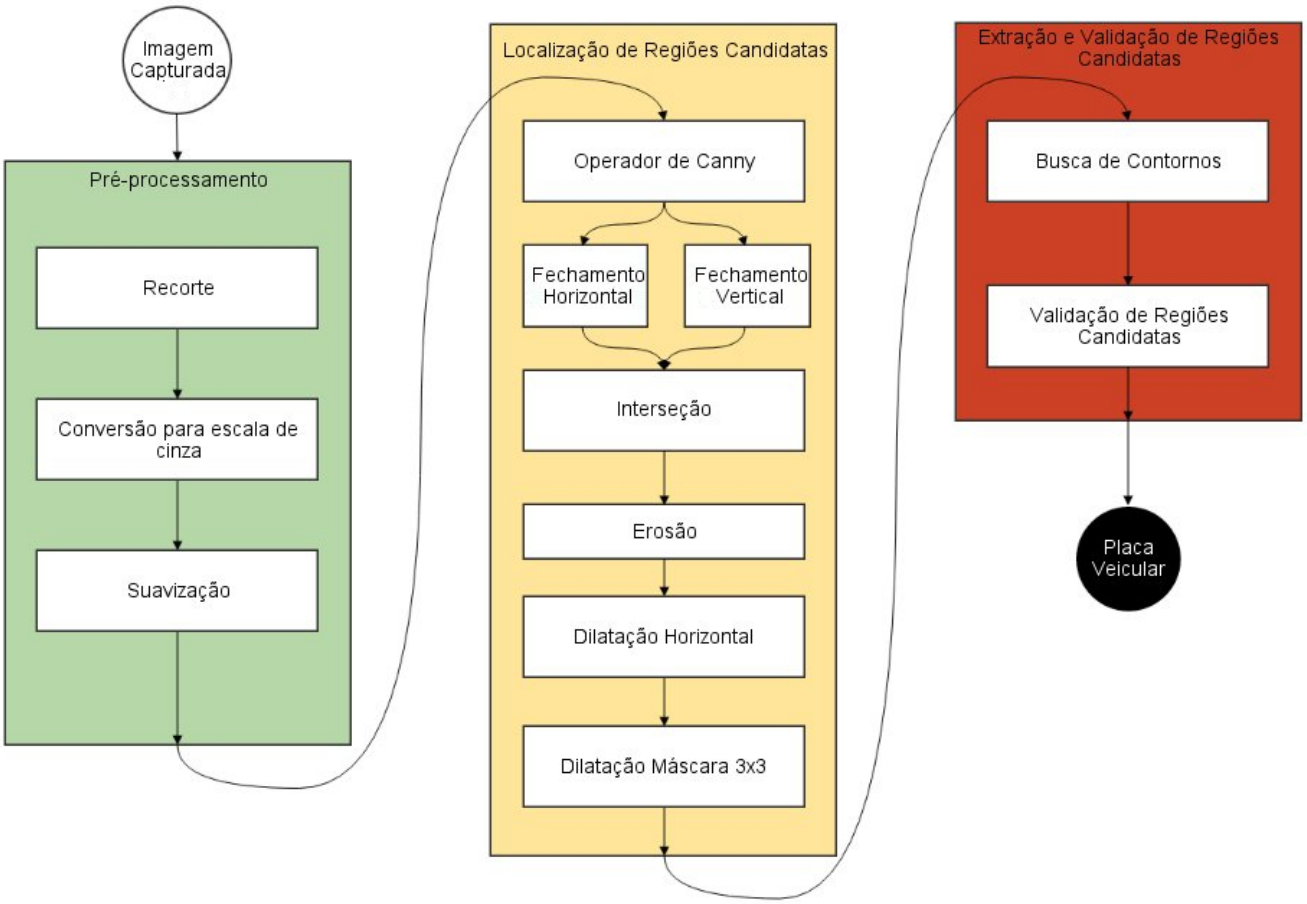


Figura 7: Algoritmo desenvolvido: Fases e suas operações

4.1 Pré-processamento

A etapa de pré-processamento no algoritmo desenvolvido visou redimensionar imagens para melhor enquadramento e, consequentemente, melhorar o tempo de processamento. Além disso, esta etapa procurou realçar características importantes e suprimir ruídos que possam interferir nas etapas de processamento seguinte. As técnicas utilizadas em tal etapa foram: recorte da imagem/vídeo original, conversão para escala de cinza, seguido pela aplicação de um filtro de suavização. As duas primeiras técnicas são opcionais já que podem ser aplicadas na fase de aquisição.

4.1.1 Recorte

O recorte da imagem original é necessário para eliminar regiões que não apresentem possibilidade de conter placas automotivas. A aplicação dessa técnica é importante, conforme explicado anteriormente, para diminuir o tempo de processamento e reduzir a presença de falsos positivos. Como dito antes, essa etapa é opcional, já que as dimensões da região de captura de uma imagem podem ser ajustadas facilmente na etapa de aquisição sem a necessidade do processamento adicional. Essa etapa é configurada de maneira manual e dependerá da posição de captura das imagens para definição da área a ser recortada.

4.1.2 Conversão para escala de cinza

Assim como a fase anterior, o procedimento de transformação para escala de cinza visa reduzir o tempo de processamento. Conforme explicado na seção 3.2, imagens em tons de cinza são representadas em um intervalo único enquanto que em imagens coloridas são necessárias três grandezas. Por esse motivo, o tempo de processamento de imagens coloridas é superior àquelas em tons de cinza. Uma imagem colorida convertida para tons de cinza pode ser vista na Figura 1a.

4.1.3 Aplicação do filtro de suavização

Conforme explanado na seção 3.4.2, a técnica de suavização visa, principalmente, a redução de ruídos, correção de pequenas descontinuidades e a eliminação de objetos pequenos. Pelos motivos anteriormente citados, a aplicação do algoritmo de suavização foi fundamental para melhora nos resultados nas fases seguintes. Em testes realizados, a retirada de tal etapa aumentou significativamente a detecção de regiões desnecessárias, bem como o tempo de processamento da função de busca de contornos. A execução do algoritmo com diversos tamanhos de máscara evidenciou que a função de borramento com máscara $7x7$, como pode ser visto um exemplo na Figura 3d, resulta em um melhor resultado.

4.2 Localização de regiões candidatas

A etapa de localização ou segmentação de regiões candidatas, conforme evidenciado na Seção 3.3, representa o maior desafio em um algoritmo de localização automática de placas veiculares. Essa etapa foi implementada no algoritmo aqui desenvolvido através da seguinte sequência de operações: primeiramente aplicação do algoritmo de detecção de bordas de Canny, em seguida foi executada uma sequência de operações morfológicas e por último foi executado uma algoritmo de busca de contornos.

4.2.1 Aplicação da detecção de bordas de Canny

O algoritmo de detecção de bordas de Canny foi descrito, anteriormente, na seção 3.5.3. A escolha por essa metodologia de detecção de bordas se deu pelo seu alto desempenho na detecção de bordas mesmo em imagens que apresentem ruídos e pela capacidade de evidenciar as bordas presentes, tanto no contorno da placa como um todo quanto nos caracteres contidos dentro dela. Outro fator primordial para escolha de tal algoritmo foi o baixo tempo de processamento se comparado aos tempos de execução obtidos com o algoritmo de Sobel.

A *OpenCV* apresenta uma implementação paralelizada em GPU, utilizando CUDA, do algoritmo de detecção de bordas que apresentou um considerável desempenho comparado com a implementação tradicional (em CPU). Três parâmetros são necessários para execução de tal função: um limiar inferior de binarização, um limiar superior de binarização e por último é necessário a definição de uma máscara para do operador. Analisando a aplicação do algoritmo com diferente limiares mínimos em uma conjunto variado de imagens, foi definido o limiar mínimo como o valor 40. O limiar máximo, conforme recomendado em [3], foi definido na proporção de 2:1, e neste caso assumiu o valor 80. Por último, foi utilizada o tamanho de máscara padrão definido na biblioteca para execução do operador.

A Figura 8b mostra o resultado da execução da função com os parâmetros anteriormente definidos na Figura em tons de cinza 8a.

(a) Imagem Original (b) Imagem após aplicação

Figura 8: Aplicação do algoritmo de detecção de borda por Canny

4.2.2 Aplicação das operações morfológicas

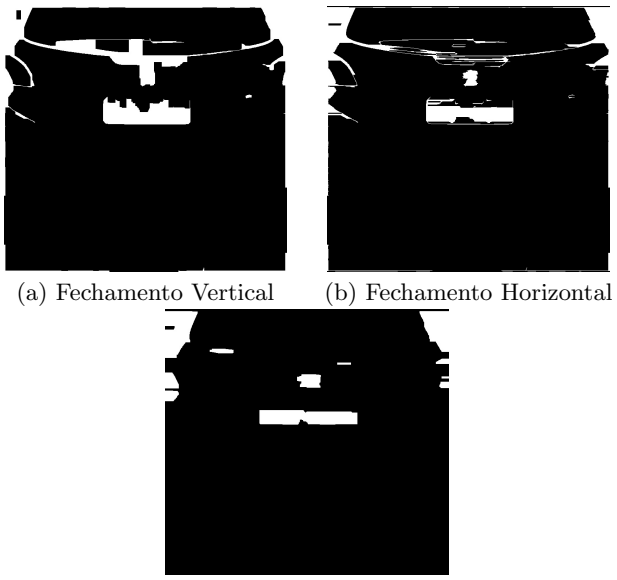
As operações morfológicas foram utilizadas com o intuito de realçar as regiões de interesse. Essa abordagem obteve sucesso em [5] e [13], sendo que no primeiro as dimensões da placa foram utilizadas como parâmetros, e no segundo foram utilizados as dimensões dos caracteres contido na placa. Essa última abordagem foi a escolhida por apresentar menor quantidade de regiões candidatas ao fim do processo.

O primeiro passo para realçar regiões candidatas foi a aplicação da operação de fechamento (seção 3.6.4) na imagem oriunda da operação anterior, com a máscara vertical definida a partir da medida do maior valor de altura dos caracteres da placa. O resultado dessa etapa pode ser visto na Figura 9a.

Assim como o passo anterior, a partir da imagem gerada da aplicação do algoritmo de detecção de Canny é aplicada a operação morfológica de fechamento. No entanto, nessa etapa uma máscara horizontal foi definida com base na máxima distância entre um caractere e outro. Esse processo pode ser visto na Figura 9b.

Os últimos dois passos também foram utilizados por Sales [13], no entanto, no algoritmo aqui desenvolvido, os passos foram aplicados separadamente sobre uma mesma imagem de origem e não sequencialmente como a utilizada pelo autor.

Após a execução das duas últimas etapas, é executada uma operação de interseção entre as imagens geradas nos dois passos anteriores. Essa operação elimina, em sua maioria, grandes regiões que foram destacadas em uma das operações anteriores mas que não aparecem no outro passo. A Figura 9c mostra o resultado ao final dessa etapa.



(a) Fechamento Vertical (b) Fechamento Horizontal

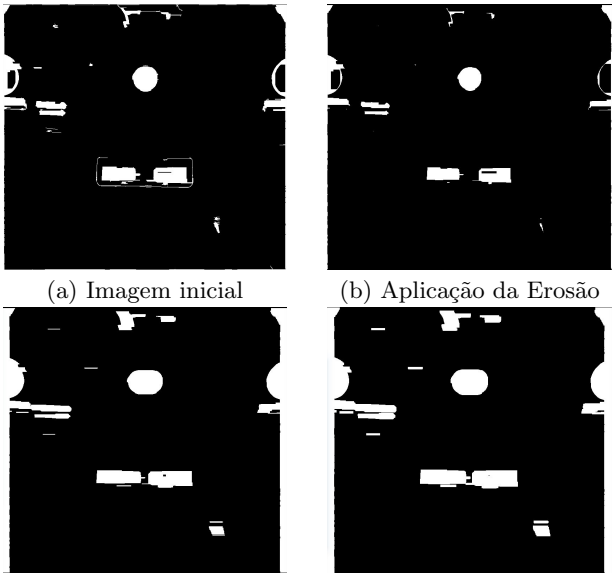
(c) Interseção das duas Imagens

Figura 9: Operações de Fechamento e sua interseção

Apesar de remover uma quantidade considerável de regiões desnecessárias e de evidenciar a região que contém a placa, as imagens obtidas ainda contêm muitas regiões desnecessárias, ou apresentam descontinuidades entre os caracteres da placa. Para solucionar esses problemas faz-se necessária a execução das operações seguintes.

Para eliminação de pequenos blocos que possam surgir na etapa passada, é feita uma operação de erosão com uma pequena máscara para evitar a exclusão de regiões promissoras e eliminar somente pequenas regiões isoladas. Essa operação evita também que pequenos objetos próximos às placas se fundam com ela ao aplicar as operações de dilatação que serão executadas em seguida. A aplicação dessa etapa sobre a Figura 10a, obtida das interseção das operações morfológicas de fechamento descritas anteriormente, pode ser vista na Figura 10b.

Finalizando o conjunto de operações morfológicas, duas operações de dilatação são executadas. Como exemplificado na Figura 10b e explicado anteriormente, comumente as regiões entre caracteres da placa costumam apresentar descontinuidades. Para preencher esses espaços é aplicada uma operação de dilatação com uma máscara horizontal idêntica a utilizada no fechamento horizontal executado em etapas anteriores. O resultado de tal dilatação pode ser vista na Figura 10c. Para corrigir pequenas falhas que possam não ser corrigidas pela execução da dilatação anterior é executada uma segunda operação de dilatação com uma pequena máscara $3x3$. Como pode ser visto na Figura 10c uma pequena descontinuidade se faz presente e é prontamente corrigida com a segunda dilatação exemplificado na Figura 10d.



(a) Imagem inicial (b) Aplicação da Erosão

(c) Dilatação máscara horizontal (d) Dilatação máscara 3x3

Figura 10: Sequência de operações de erosão e dilatação

4.3 Extração e Validação de regiões candidatas

Após a execução das operações morfológicas, a imagem resultante apresenta um conjunto de regiões como exemplificado na Figura 11a. A *OpenCV* disponibiliza uma função para captura de contorno e adaptação em regiões retangulares, essa função chama-se *findContours* e tem como entrada uma imagem binária e após sua execução gera uma vetor contendo os conjuntos de pontos que definem as bordas das regiões retangulares presentes na imagem de entrada.

Como possivelmente mais de uma região candidata pode ter sido inserida no vetor que contem os contornos dessas regiões, é necessário encontrar uma forma eficiente de validar tais regiões. Para solucionar esse problema, o vetor é percorrido verificando a presença de algumas características presentes em regiões que contem placas veiculares.

O primeiro fator é a verificação do aspecto (razão entre largura e altura), caso o tamanho medido na vertical seja maior que o medido na horizontal esse contorno é eliminado pois não corresponde as características de uma região que contenha uma placa veicular. Mesmo que a região seja aprovada no critério anterior é necessário verificar se a razão entre a largura e a altura não supera a proporção aproximada máxima de 6 : 1 considerando somente a região do caracteres ou de no mínimo 3 : 1 considerando toda a região que compõe a placa.

É necessário verificar também os tamanhos máximos que os contornos podem apresentar. Essas dimensões variam a depender da distância que é capturada a imagem a ser processada e a definição precisa de tamanho mínimos e máximos que os contornos poderão apresentar é essencial para a captura precisa das regiões que contem as placas veiculares.

No conjunto de imagens da Figura 11 vemos a imagem de entrada para a função *findContours* na figura 11a e os contornos encontrados, marcados na imagem original, podem ser vistos na Figura 11b. O resultado final da execução das etapas de validação e a consequente localização da região que contem a placa veicular é mostrado na Figura 11c

(a) Imagem resultante da etapa de localização de regiões candidatas (b) Contornos encontrados

(c) Região da placa veicular

Figura 11: Sequência de operações de erosão e dilatação

5. RESULTADOS OBTIDOS

Nesta seção os resultados obtidos são discutidos a partir dos testes realizados em dois cenários distintos: algoritmo executado em imagens e o algoritmo sendo executado em vídeo. Serão verificados, nos dois cenários, a precisão na localização

da placa veicular e o tempo necessário para execução do algoritmo utilizando funções do *OpenCV* executadas em CPU e em GPU e um terceiro utilizando a melhor implementação mesclando as duas abordagens.

5.1 Ambiente de Desenvolvimento

A configuração do ambiente de trabalho utilizado para implementação do algoritmo aqui desenvolvido pode ser visto na tabela 5.1.

Processador	AMD Phenom II x4 3Ghz
Memória RAM	4GB DDR 2 800Mhz
Sistema Operacional	Arch Linux 64Bits
Placa de Vídeo	Nvidia GeForce GTX 650 1GB
CUDA SDK	Versão 5.5
OpenCV	Versão 2.4.7

5.2 Dataset Utilizado

O *dataset* utilizado para validação da solução aqui proposta foi composto por imagens de automóveis de passeio, táxis e ônibus, capturadas a partir de duas fontes: fotografias e vídeos.

As fotografias foram capturadas a partir de câmeras digitais ou de celulares. Essas imagens foram obtidas em diversos ambientes, apresentando diferenças significativas de iluminação e ângulos de captura entre elas. Os veículos foram capturados parados em estacionamentos ou em circulação em vias públicas. O *dataset* de imagem foi composto por 100 imagens. As imagens foram recortadas de maneira a gerar um *dataset* normalizado com a resolução de 1000x1000 *pixels* . Na Figura 12 é possível ver alguns exemplos de veículos que compõem a base de dados, fotografados em condições adversas.

Figura 12: Exemplo de imagens do *Dataset* utilizado

Os vídeos foram capturados exclusivamente em vias públicas com veículos em velocidades variadas. Foram gravados com resolução de 720p com veículos trafegando a aproximadamente 20 metros de distância, e com a câmera posicionada paralela a via e direcionando o foco para somente uma faixa.

Foram feitos 7 vídeos com um total de 101 veículos passando pela região de captura da câmera. Foram desprezados os veículos que apareceram na filmagem mas não estavam na faixa em que o foco do câmera estava ajustado, pois, nestes casos as imagens não apresentaram legibilidade dos caracteres das placas. A Figura 13 apresenta exemplos de *frames* capturados nos vídeos utilizados para teste da aplicação.

Figura 13: Exemplo de *frames* dos vídeos gravado para o *Dataset* utilizado

5.3 Tempo de Execução

Para conseguir obter o menor tempo de processamento possível, a implementação do algoritmo desenvolvido foi analisada através de duas abordagens: a execução de todas as funções utilizando sua versão em CPU e a execução, quando disponível, das funções em sua implementação em GPU com CUDA. Para melhor visualização dos resultados obtidos foram medidos em cada caso o tempo total do algoritmo e os percentuais que cada função representa desse tempo. O algoritmo é idêntico em ambos os casos sendo a única modificação o local de execução.

5.3.1 Algoritmo com funções em CPU

A primeira solução testada foi implementada utilizando somente as funções executadas em CPU. Essa abordagem apresentou um tempo de processamento médio aproximado de 42,4 milessegundos.

Conforme pode ser visto na Figura 14, 13.48ms são gastos somente na execução do operador de Canny. Outros 11.44ms são gastos com as operações de fechamento horizontal e vertical (*morphologyEX*). O conjunto de operações morfológicas de erosão e dilatação ocuparam 6.18ms do tempo total. A atividade de converter a imagem colorida para escala de cinza demorou em média 5,01ms para ser executada. A funções de suavização (*blur*), busca de contornos(*findContours*), a etapa de validação e a operação de interseção (*bitwise_and*) ocuparam, respectivamente, 2.62ms, 2.42ms, 0.77ms e 0.41ms do tempo total de processamento.

5.3.2 Algoritmo com funções em GPU

A segunda implementação foi feita utilizando as funções que apresentavam versão implementada em CUDA. Das funções

disponíveis na *OpenCV* e utilizadas no projeto, somente a função *findContours* não apresentou versão paralelizada em GPU, por esse motivo, foi utilizado a versão disponível em CPU.

O tempo médio para execução completa do algoritmo ficou em 45,89ms, tempo esse bastante próximo a implementação em CPU. Nessa versão, a função que consumiu maior tempo de processamento foi o conjunto de operações morfológicas de fechamento horizontal e vertical, que juntas consumiram 56,6% do tempo total de processamento ou 25,96ms de tempo médio. Isso representa um acréscimo de 14,56ms se comparada com a execução em CPU e a baixa eficiência pode ser explicada pela grande desproporcionalidade presente no formatos das máscara utilizadas na execução, o que torna a execução ineficiente se realizada em paralelo. O mesmo motivo explica o acréscimo 4,6ms de tempo médio, se comparada a execução e CPU, no conjunto de operações morfológicas de erosão e dilatação já que em uma das operações de dilatação é utilizado uma máscara horizontal conforme explicado na seção 4.2.2, esse conjunto de funções representou 23,5% do tempo total ou 10.78ms. A implementação em GPU apresentou melhorias significativas quando a execução do algoritmo de detecção de bordas de Canny foi comparada nas duas versões. Foi possível observar que ao executar a função paralelizada, o tempo de processamento médio caiu para 3.51ms o que representa uma diminuição do tempo de processamento em 3,84 vezes. Outra função que apresentou excelente desempenho, nessa abordagem, foi a de conversão para escala de cinza que foi executada em 0.42ms o que representa uma diminuição de aproximadamente 12 vezes ou de 4,59ms quando comparada as duas situações. Apesar de em ambas as implementações a função de interseção entre imagens, na *OpenCV* chamada de *bitwise_and*, apresentar baixo custo computacional na implementação em GPU houve diminuição de 41,46% passando a ser executada em 0,24ms. O função de suavização *blur* apresentou um tempo de execução médio de 2,35ms o que representa uma melhora de aproximadamente 10% em comparação com a execução em CPU. A função de busca de contorno e a etapa de validação, por serem executadas em CPU, apresentaram resultados semelhantes, com uma pequena redução no tempo de processamento na execução por GPU justificado talvez pela menor utilização do processador em etapas anteriores.

Em geral, as funções que apresentaram baixo desempenho quando executadas em GPU foram motivadas pela utilização de grandes máscaras, o que exige um maior poder computacional cada vez que elas são aplicadas na imagem. As GPUs são caracterizadas pela grande quantidade de núcleos de processamento com poder de processamento muito inferior aos da CPU, que apresentam poucos núcleos com grande capacidade computacional. Assim, atividades que possuem quantidades menores de iterações porém com grande carga de processamento em cada uma delas, são melhores executadas na CPU, e o contrário, grande quantidade de iterações com baixa carga de processamento por iteração, tendem a obter melhora no desempenho quando paralelizadas e executadas em GPU. As funções Sobel e cvtColor são exemplos notáveis do ganho de desempenho que uma função pode adquirir quando paralelizada em GPU.

5.3.3 Algoritmo Final

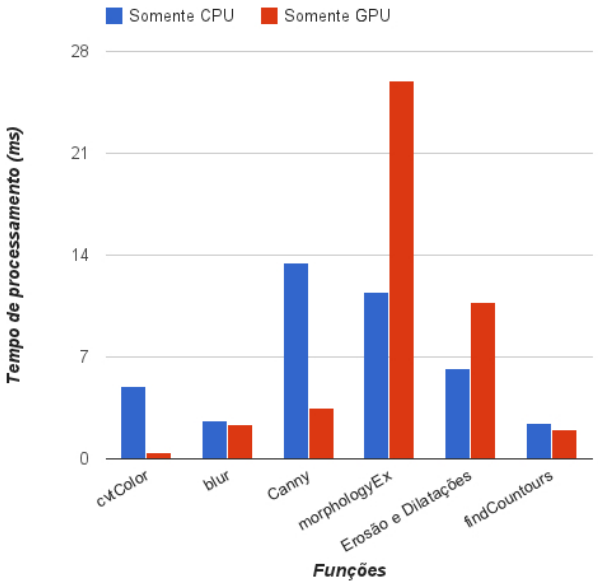


Figura 14: Comparativo do tempo médio de execução em CPU e GPU das funções: cvtColor- Conversão para escala de cinza, blur - filtro de suavização, Canny - detecção de bordas de Canny, MorphologyEx - Operação de morfológia - operações morfológicas de fechamento, conjunto de operações de erosão e dilatações e findCountours - busca de contornos

A partir da análise dos tempos de processamento das etapas anteriores, foi possível identificar os pontos fortes e as deficiências de cada abordagem e assim extrair a melhor solução oriunda da combinação dos dois cenários. A Figura 14 mostra a comparação do tempo de processamento médio de cada função nas duas implementações anteriores.

Analisando a Figura 14 também foi possível observar que as operações morfológicas de erosão, dilatação e fechamento apresentaram uma grande melhora de desempenho quando executadas em CPU. Por esse motivo, no algoritmo final tais funções foram utilizadas seguindo essa abordagem ajudando assim a reduzir consideravelmente o tempo total de execução.

Devido a redução significativa do tempo de processamento das funções cvtColor e Sobel, quando executadas na versão desenvolvida em CUDA, essas funções foram implementadas no algoritmo final em sua versão paralelizada. Por apresentar uma pequena melhora quando executada em paralela no algoritmo final a função blur foi escolhida também em sua implementação em GPU.

Versões	Tempo de Processamento
Em CPU	42,4ms
Em GPU	45,89ms
Solução Final	19,41ms

No final, foi possível reduzir o tempo médio de execução para apenas 19.41ms, conforme visto na tabela 5.3.3, o que repre-

senta uma redução de aproximadamente 23ms se compararmos com a versão em CPU, e de aproximadamente 26,5ms quando comparamos com a versão em GPU. Assim, é possível notar que a abordagem hibrida CPU/GPU apresentou resultados superiores as duas versões anteriores, sendo possível notar uma redução no tempo médio total superior a 50% e se mostrou, do ponto de vista do tempo de execução, uma solução apropriada para uma aplicação em tempo real devido ao baixo tempo médio de execução obtido.

5.4 Precisão

Nessa seção é discutida a precisão obtida ao executar a solução aqui desenvolvida. A precisão foi aferida com o algoritmo executando nas duas bases de dados apresentadas anteriormente na seção 5.2 e visa validar a utilização do algoritmo a partir duas formas possíveis de captura de veículos: fotografias e vídeos.

5.4.1 Precisão em Fotografias

Executando em uma base de dados composta por 100 fotografias, o algoritmo conseguiu êxito na localização de regiões que continham placas de licenciamento veicular em 96 casos, representando uma taxa de acerto de 96%. Na Figura 15 é possível ver exemplos de imagens em que as placas veiculares foram encontradas com êxito.

Figura 15: Placas localizadas

Na base de imagens, em apenas 4 fotografias não foi possível localizar a placa veicular corretamente. A Figura 16 mostra as ocasiões em que o algoritmo não obteve sucesso na localização da placa veicular.

5.4.2 Precisão em Vídeos

Ao executar o algoritmo com o intuito de localizar placas veiculares em vídeos de carros em movimento, o algoritmo logrou êxito em 95% dos casos, o que representa a correta localização em 96 dos 101 veículos passíveis de localização.

A Figura 17 apresenta um conjunto de veículos que tiveram sua placas localizadas com sucesso ao passarem pela faixa que estava sendo filmada.

Somente 5 veículos não tiveram suas placas de licencimento detectadas durante as filmagens, alguns exemplos dessa ocasião podem ser observadas na Figura 18.

Figura 16: Fotografias em que o algoritmo não obteve sucesso na localização

Figura 17: Placas corretamente localizadas no vídeo

5.5 Falsos Positivos

Durante a execução do algoritmo foi possível observar que em alguns casos regiões que não possuíam placas veiculares foram apontadas como válidas. No *dataset* de fotografias, 63 regiões foram apontadas equivocadamente com sendo placas veiculares, em algumas imagens, foram apontadas múltiplas regiões como contendo placas veiculares. A Figura 19 mostra algumas fotografias que apresentaram falsos positivos.

Em muitos casos, regiões que continham logotipos do fabricante, adesivos ou numerações foram apontadas como regiões de placas veiculares, isso ocorre porque o algoritmo

Figura 18: Automóveis não detectados pelo algoritmo no vídeo

Figura 19: Fotografias que apresentaram falsos positivos

utiliza os caracteres da placa como ponto chave para localização, e nestas situações citadas anteriormente os caracteres presentes acabam gerando regiões de dimensões semelhantes às dos caracteres contidos na placa.

Em outras situações, falsos positivos podem ser gerados pela união de contornos retangulares que apresentem proximidade entre eles suficiente para se unirem em uma única região após a execução das operações morfológicas.

Em todo caso, esses falsos positivos podem ser facilmente descartados na fase de reconhecimento de caracteres, etapa posterior em um sistema de reconhecimento de placas de licenciamento veicular, por não apresentarem os caracteres ordenados conforme o padrão definido para as placas de licenciamento nacionais.

5.6 Discussão sobre os resultados

Os resultados demonstrados anteriormente mostram que o algoritmo de localização de placas veiculares aqui desenvolvido atende prontamente as premissas para execução em tempo real assim como a sua precisão alcançou níveis satisfatórios.

Quando comparado os resultados aqui obtidos com as soluções apresentada na seção 2, nenhum trabalho conseguiu apresentar tempo de processamento próximo àqueles obtidos aplicando a metodologia aqui apresentada . Assim, como a precisão do algoritmo se manteve próxima as propostas que se saíram melhor nesse quesito. A tabela 5.6 relaciona cada abordagem a seu tempo de processamento e sua precisão.

Autor(es)	Tempo de Processamento	Precisão
Nosso algoritmo	19ms	95 a 96%
Sales [13]	1279ms	98%
Lee and Hung [8]	>2500ms	95%
Nascimento[9]	95 a 105 ms	63 a 100%
Wen et al. [18]	111ms	86 a 96,5%
de Araújo et al. [5]	Não informado	86%
Freire e Maia [6]	89 a 154 ms	99,5%

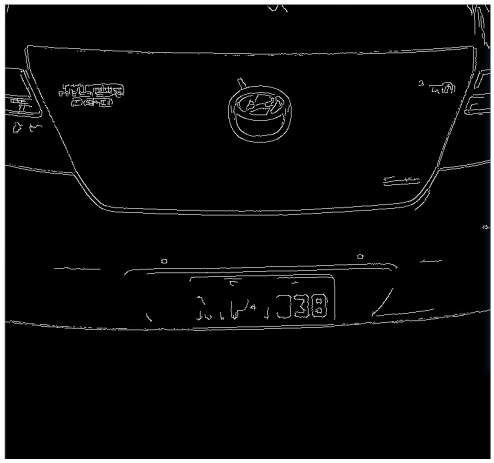
Ao testarmos o algoritmo desenvolvido por Baggio et al [1] no mesmo *dataset* de imagens utilizados nesse trabalho, efetuando as devidas adaptações para as dimensões das placas nacionais, o algoritmo obteve sucesso na localização em 76% dos casos com um tempo médio de processamento de aproximadamente 220ms. Os resultados obtidos mostram a superioridade, tanto do ponto de vista da precisão, quanto do desempenho da solução proposta neste artigo.

Também foi possível constatar que a correta localização da placa veicular está estreitamente ligada a capacidade do algoritmo de Canny de delimitar os contornos dos caracteres presentes nas placas. O uso de limiares fixos, apesar de eficiente na maior parte das ocasiões, em alguns casos não foi totalmente eficaz na delimitação de tais contornos. As Figuras 20a e 20b mostram, respectivamente, um exemplo em que a função Sobel não conseguiu realçar os contornos de todos os caracteres contidos na placa e o resultado final da aplicação das operações morfológicas. Analisando a Figura 20b é possível notar a presença de pequenas descontinuidades entres os 3 blocos que juntos corresponderiam a placa veicular, essas descontinuidades inviabilizaram o reconhecimento da região como placa veicular e foram geradas devido a falha na delimitação do contornos de alguns caracteres conforme explicado anteriormente.

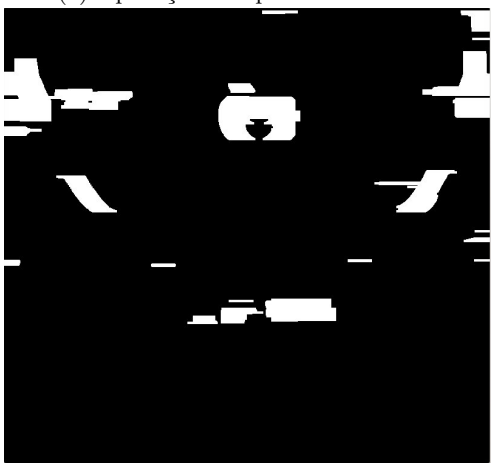
6. CONCLUSÕES

O presente trabalho utilizou técnicas de processamento de imagens digitais com o intuito de desenvolver uma solução computacional automatizada capaz de localizar placas de licenciamento veicular capturadas em fotografias ou filmagens.

Diferente da maior parte dos trabalhos presentes na literatura que utilizaram somente fotografias para validação dos



(a) Aplicação do operador de Sobel



(b) Aplicação das operações morfológicas

Figura 20: Falha do operador de Sobel na definição dos contornos da placa

resultados, a solução aqui desenvolvida foi testada em uma base de dados de imagens de resolução superior as que compõem os demais projetos. Além disso, foi testada a solução em vídeos capturados em via públicas de grande fluxo veicular, o que aproxima os testes realizados de cenários encontrados em aplicações utilizadas em condições reais.

Outro grande diferencial presente na metodologia aqui desenvolvida foi a utilização em parte do algoritmo de funções executadas em GPU. Essa escolha possibilitou a redução dos tempos de processamento e consequentemente a possibilidade de utilização dessa solução em sistemas que funcionam em tempo real, o que fez com que o principal objetivo desse trabalho fosse alcançado que era a localização de placas veiculares em tempo real. A precisão aferida de 95% em vídeos, comprovam a competitividade do sistema desenvolvido quando comparado com outras soluções existentes, abrindo caminhos para sua utilização comercial como parte de um sistema de reconhecimento de placas veiculares. Muitos são os cenários que exigem sistemas localizadores de placas veiculares em vídeos, dentre eles podemos citar:

- Fiscalização de faixas exclusivas;
- Fiscalização de rodízio de veículos;
- Monitoramento de tráfego em tempo real;
- Fiscalização de veículos roubados, com impostos atrasados ou que apresentem outras irregularidades;
- Gerenciamento automatizado de estacionamentos;
- Controle de acesso a locais restritos, como condomínios, órgãos públicos etc;

6.1 Trabalhos Futuros

Apesar de alcançar um nível aceitável de precisão, outros trabalhos na literatura se mostraram mais eficientes neste quesito. Com isso, a adição de outras operações morfológicas, como as utilizadas na solução desenvolvida por Sales [13], ou o estudo aprofundado de outras técnicas utilizadas em outros trabalhos, que possam aumentar ainda mais a precisão sem comprometer o tempo médio de execução do algoritmo.

Outro fato que necessita de uma atenção especial é o elevado número de falsos positivos. Apesar dessas regiões serem possivelmente descartadas em etapas posteriores como explicado na seção 19, melhorias na etapa de validação ou utilização de redes neurais para validação como observado em [13], [18] e [6] podem ajudar a reduzir consideravelmente essas ocorrências.

A solução também necessita ser testada em condições reais, com equipamentos específicos para tal situação, instalados de maneira a se obter o melhor posicionamento possível em relação aos veículos. Como as filmagens feitas para composição do *dataset* utilizado aqui foram realizadas com câmeras amadoras, muitas vezes a identificação da região que continha a placa veicular foi prejudicada pela incapacidade do equipamento de focar os veículos que trafegava no momento ou pela posição de captura não ser a mais adequado para captura-los.

Por último, será necessário o desenvolvimento ou integração com sistemas que realizem o reconhecimento dos caracteres contidos na placa para utilização desse trabalho em condições reais.

7. REFERÊNCIAS

[1] D. L. Baggio, S. Emami, D. M. Escrivá, K. Ievgen, N. Mahmood, J. Saragih, and R. Shilkrot. *Mastering OpenCV with Practical Computer Vision Projects*. Packt Pub., 2012.

[2] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'reilly, 2008.

[3] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[4] D. O. Dantas and J. Barrera. As gpu no processamento de vídeo em tempo real. *Universidade de São Paulo-USP, SP*, 2006.

[5] S. A. de Araújo, D. de Sousa Abreu, W. A. L. Alves, and A. F. H. Librantz. Uma abordagem para localização e reconhecimento de placas de licenciamento veicular por meio de operadores

morfológicos e busca por template. *Exacta*, 9(3):355–362, 2011.

[6] F. A. N. FREIRE and J. E. B. MAIA. Localização automática de placas de veículos em imagens baseada no detector de viola-jones. 2013.

[7] R. C. Gonzalez and R. E. Woods. *Processamento de imagens digitais*. Edgard Blucher, 2000.

[8] R.-C. Lee and K.-C. Hung. An automatic vehicle license plate recognition based on edge detection and artificial neural network. *Open Science Repository Computer and Information Sciences*, (open-access):e23050468, 2013.

[9] J. D. d. Nascimento. Detecção e reconhecimento de placa automotiva com baixo custo. 2012.

[10] C. Nvidia. Programming guide, 2008.

[11] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[12] J. M. Rodrigues. Evolução da frota de automóveis e motos no brasil. 2012.

[13] R. B. Sales. Localização e validação automática de regiões candidatas de placas a partir da aná lise de imagens digitais. 2010.

[14] M. Shensa. The discrete wavelet transform: wedding the a trous and mallat algorithms. *Signal Processing, IEEE Transactions on*, 40(10):2464–2482, 1992.

[15] R. Smith. An overview of the tesseract ocr engine. In *ICDAR*, volume 7, pages 629–633, 2007.

[16] F. Souza and A. Susin. Siav–um sistema de identificação automática de veículos. In *Congresso Brasileiro de Automática*, pages 1377–1380, 2000.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[18] Y. Wen, Y. Lu, J. Yan, Z. Zhou, K. M. von Deneen, and P. Shi. An algorithm for license plate recognition applied to intelligent transportation system. *Intelligent Transportation Systems, IEEE Transactions on*, 12(3):830–845, 2011.