UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

HENRIQUE WEBER

Um protótipo móvel para detecção automática de placas veiculares brasileiras

Trabalho de Graduação.

Prof. Dr. Cláudio Rosito Jung
Orientador

Porto Alegre, dezembro de 2013

“If I had a chance for another try
I wouldn’t change a thing
It’s made me all who I am inside“
— Tom DeLonge, Angels & Airwaves

AGRADECIMENTOS

Gostaria de agradecer enormemente aos meus pais e minha irmã pelo carinho e apoio incondicional durante todos estes anos. Nada disso seria possível sem a ajuda de vocês!

A minha namorada, que esteve junto a mim todo esse tempo e me motivou a sempre dar o melhor de mim. Por ter vivenciado e auxiliado, com carinho e paciência, em todos os detalhes deste trabalho, tornando-o uma experiência inesquecível.

Aos meus amigos de faculdade e de infância, pela parceria de sempre e saudável competição. Espero contar com vocês pelo resto da vida!

Ao meu professor orientador Cláudio Jung por todo o auxílio e excelente colaboração para a realização deste trabalho.

A UFRGS e seus funcionários pela grande oportunidade de expandir meus horizontes em um ambiente rico em experiências e com um ensino superior de qualidade.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

GCC	*GNU Compiler Collection*
GNU	*GNU is Not Unix*
MinGW	*Minimalist GNU for Windows*
OpenCV	*Open Source Computer Vision*
ALPR	*Automatic License Plate Recognition*
OCR	*Optical Character Recognition*

LISTA DE FIGURAS

RESUMO

Detecção automática de placas de veículos consiste em uma técnica de identificação de uma região em uma imagem que contém uma placa de automóvel. Esta informação, após análise de um reconhecedor de caracteres, pode ser utilizada para diversos fins, como patrulhamento de ruas a fim de identificar veículos em locais irregulares e localização de veículos roubados ou com infrações que o impossibilitem de trafegar.

Este trabalho tem como objetivo o desenvolvimento de um protótipo de baixo custo para detecção automática de placas de automóvel, funcionando durante a locomoção do veículo ao qual estiver acoplado. Tal protótipo deve estar posicionado no para-brisa de um carro e ser capaz de armazenar a porção da imagem onde há uma placa, caso haja uma ou mais delas. Este *patch* da imagem deve então ser enviado a um servidor externo, o qual pode processar a informação e fazer uso desta de acordo com suas necessidades. Com isto, pretende-se oferecer uma alternativa mais viável, economicamente, para o rastreamento de carros para as finalidades explicitadas acima. Os resultados alcançam uma taxa de acerto de 35% das placas visíveis, com um tempo médio de detecção de 0,09 segundos.

Palavras-chave: Detecção Automática de Placas de Automóvel, visão computacional.

A mobile prototype for automatic brazilian license plate detection

ABSTRACT

Automatic license plate detection consists of a technique for identification of a region in an image that contains a license plate. This information, after an Optical Character Recognition analysis can be used for various purposes such as patrolling the streets in order to identify vehicles at irregular location, which have being stolen or have violated traffic laws.

The goal of this project is to develop a low cost prototype for automatic license plate detection working during locomotion of the vehicle to which it is attached. Such prototype must be positioned at the vehicle windshield and must be capable of store the image portion where there is a plate, if there is one. This image patch must be sent to an external server, which can process the information and make use of it according to its needs. Thus, the aim is to offer a more economically viable alternative for tracking cars for the purposes explained above. Results show a hit rate of 35% with an average detection time of 0.09 seconds.

Keywords: Automatic License Plate Detection, computer vision.

1 INTRODUÇÃO

O monitoramento de veículos que trefegam em vias públicas é uma necessidade constante e que apresenta diversos desafios. Um deles é o aumento da frota de automóveis, que no mundo pode alcançar 1.7 bilhões de unidades em 2035 (o dobro da frota atual), de acordo com a Agência Internacional de Energia[1]. Já no Brasil, a frota cresceu 119% entre 2000 e 2010, de acordo com o Departamento Nacional de Trânsito (Denatran)[2]. Números tão expressivos colocam em xeque muitas das práticas utilizadas para o combate e apreensão de veículos roubados ou proibidos de trafegar, como a realização de *blitzes* ou a inspeção manual dos dados de identificação de veículos suspeitos.

Uma alternativa que vem sendo aplicada para o aumento da eficiência deste tipo de monitoramento é a automação do serviço, fazendo uso de tecnologias que combinam dados capturados em vias de tráfego de veículos com as informações de automóveis tidos como roubados ou proibidos legalmente de trafegar. A principal forma de obtenção desses dados se dá pelo uso de câmeras, tanto fixas quanto móveis. Elas identificam os veículos principalmente pela detecção de suas placas, pois esse é um item obrigatório em todos os automóveis particulares e os caracteriza de forma única.

Apesar do fato de que sistemas de detecção de placas fixos e móveis compartilham objetivos semelhantes, os desafios, por sua vez, são muito diferentes. O sistema fixo possui a vantagem de permitir a remoção do fundo da imagem, separando mais facilmente veículos do pavimento. Ele também não sofre trepidação, o que torna a imagem mais estável e nítida. Outra vantagem é saber com bastante precisão a região do espaço onde a placa mais provavelmente aparecerá, o que reduz o campo de busca e, por conseguinte, melhora a eficiência do algoritmo de detecção. A grande desvantagem do sistema diz respeito à sua imobilidade, pois desta forma infratores podem evitar o trajeto no qual o sistema está instalado.

O sistema móvel, por sua vez, permite o monitoramento de forma praticamente imperceptível, além de possibilitar que determinadas regiões sejam vigiadas temporariamente, caso seja necessário. Estes dados podem ser utilizados para a detecção de padrões de fuga, além de inúmeros outros propósitos de inteligência. Estas vantagens trazem, porém, uma série de novos desafios para a detecção de placas. Uma delas é a variação quase que constante das características do ambiente, como iluminação, localização da placa na imagem, inclinação da pista, ruídos, entre outros. A melhora da tecnologia empregada neste tipo de equipamento, entretanto, vem em um ritmo crescente nos últimos anos, tornando as soluções propostas cada vez mais robustas a estas adversidades.

Porém, estas mesmas soluções móveis possuem um preço proibitivo em diversos ce-

[1] `http://www.iea.org/newsroomandevents/speeches/Amb_Jones_WEO2011_Oslo-1.pdf`

[2] `http://www.denatran.gov.br/frota.htm`

nários, como o Brasil. Mesmo que este valor seja justificado pela alta eficiência destes produtos, a carência de opções economicamente viáveis faz com que muitos serviços de segurança com baixo poder aquisitivo não possam adquirir nenhum produto do ramo. Neste aspecto, alternativas menos robustas e mais baratas podem ser uma alternativa atraente, capaz de inserir estes "consumidores" no mercado de vigilância automática de placas.

1.1 Objetivo

O objetivo deste trabalho consiste em desenvolver algoritmos de baixo custo computacional para detecção de placas, fazendo uso de câmera veicular embarcada. O protótipo a ser criado deve ter um valor acessível, utilizando um *Raspberry Pi Model B* com a câmera *Raspberry Pi Camera* 69W0689, acoplada no para-brisa do veículo. O software será escrito em C++, utilizando a biblioteca *OpenCV*.

Para que o sistema funcione da forma mais eficaz possível, é altamente recomendável que este seja utilizado sob determinadas circunstâncias. A primeira delas diz respeito às condições climáticas no momento da filmagem. A melhor performance foi observada em dias nublados ou com o Sol posicionado de tal forma que não gere reflexos no para-brisa. Em dias de chuva, o limpador do para-brisa interfere no processo, enquanto a água que escorre pelo vidro torna a imagem "borrada".

Em relação à instalação do equipamento, o principal cuidado se dá com o posicionamento da câmera. Ela deve ser disposta de forma a capturar o horizonte da via, fazendo com que este fique próximo à margem superior da imagem. Em veículos de passeio, o local ideal para a colocação do aparelho situa-se entre o para-brisa e o espelho retrovisor.

Para que a placa detectada possua uma resolução que possibilite seu posterior reconhecimento, é necessário que os veículos a serem monitorados estejam a no máximo 10 metros de distância do automóvel com o aparelho de detecção. A distância mínima é de aproximadamente 2 metros, de acordo com os testes realizados experimentalmente. O sistema é capaz de detectar veículos que estejam tanto na mesma faixa que o automóvel rastreador quanto nas faixas imediatamente à esquerda e à direita (desde que sejam respeitadas as regras acima). Também é importante ressaltar que placas de motocicletas não são capturadas. Como última restrição, não podem haver objetos que ocultem partes da placa ou do carro a ser analisado.

1.2 Organização do texto

O trabalho encontra-se dividido em seis capítulos. O capítulo 2 aborda as principais opções de equipamentos semelhantes ao proposto por este trabalho, bem como seu uso no Brasil e no mundo. Também é feita uma revisão do que tem sido proposto pela academia em relação aos algoritmos de detecção de placas. O capítulo 3 apresenta as características físicas do equipamento, além do *hardware* necessário e o custo de cada peça. A implementação dos algoritmos é demonstrada no capítulo 4. O capítulo 5 mostra os resultados obtidos a partir de um conjunto de quadros, enquanto o capítulo 6 refere-se às conclusões obtidas e ideias para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo aborda o cenário atual de tecnologias de detecção de placas de automóveis. Na seção 2.1, são apresentados alguns produtos oferecidos por empresas privadas, com informações à respeito do seu funcionamento e preço, quando este for divulgado. A seguir, a seção 2.2 retrata o uso do sistema em diversos lugares do mundo, bem como os resultados obtidos depois de sua implementação. Na seção 2.3, é discutido o atual estágio de adoção dessa tecnologia no Brasil. A seção 2.4 detalha uma série de algoritmos propostos para a área pelo meio acadêmico. Por fim, é feita uma discussão na seção 2.5 à respeito do estado atual da área.

2.1 Sistemas comerciais

Atualmente, existem no mercado uma série de dispositivos móveis de detecção de placas. Um deles é o sistema ALPR da Motorola[1]. O sistema funciona acoplado a um servidor que fica instalado dentro do veículo, processando os dados adquiridos e alertando apenas quando uma placa de interesse é encontrada. De acordo com o fabricante, a solução é capaz de capturar placas mesmo em condições de tempo e luminosidade adversas, através do uso de câmeras infra-vermelho. O custo do produto varia entre 12.250 a 18.700 dólares nos Estados Unidos, de acordo com o número de câmeras.

Outro modelo disponível é oferecido pela empresa 3M[2]. A empresa possui inclusive a patente da tecnologia *Triple Flash*, a qual promete suprimir efeitos luminosos da cena provocados por faróis de outros veículos ou pelo pôr do Sol. Uma terceira opção é oferecida pela empresa Coban[3]. De acordo com o fabricante, o modelo pode ter seu banco de dados atualizado de forma *wireless*, além de contar com a marcação da posição geográfica onde a placa suspeita se encontra. O preço do produto parte de 9.500 dólares.

No Reino Unido, a empresa *MAV Systems Limited* oferece o sistema *Rapier Mobile ANPR Camera*[4]. De acordo com a empresa, o produto funciona mesmo quando o veículo de vigilância está acima das 200mph. Também é afirmado que a solução proposta é capaz de realizar o processo de reconhecimento até 50 vezes por segundo através do uso de obturadores de alta velocidade, o que elimina *motion blur*. O preço inicial é de 950 libras.

[1] http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Software+and+Applications/Public+Sector+Applications/Automatic\%20License\%20Plate\%20Recognition_US-EN

[2] http://solutions.3m.com/wps/portal/3M/en_US/NA_Motor_Vehicle_Services_Systems/Motor_Vehicle_Industry_Solutions/product_catalog/3m-automatic-license-plate-recognition/mobile-alpr-camera-systems/

[3] http://www.cobantech.com/www/ALPR.html

[4] http://www.anprcameras.com/mobile_cameras.php

2.2 Detecção de placas no mundo

Nos Estados Unidos, 23% das agências de segurança entrevistadas pelo Instituto Nacional de Justiça já possuíam sistemas de reconhecimento de placas de automóveis em 2009[5]. Dentre estas agências, 95% utilizavam versões móveis do aparelho, e eram aplicados principalmente em roubos de carros (69%), monitoramento de veículos e trânsito (28%) e investigações (25%). Menos da metade (48%) criou regras para o uso do sistema. Em questão de retenção dos dados, 40% dos entrevistados afirmam retê-los por 6 meses ou menos, enquanto 13% os mantém indefinidamente. Já 6% indicaram que a retenção depende da capacidade do equipamento instalado. Em um empreendimento na cidade de Sacramento, Califórnia, o roubo de veículos, que era de 77 em 2007, caiu para 12 um ano após o uso do sistema[6].

Na Europa, a aplicação de sistemas de reconhecimento de placas de automóveis é feita em diversos países. No Reino Unido, um grande projeto de vigilância prevê o uso de vans equipadas com o sistema e utilizado pela polícia[7]. Já na Hungria, as autoridades fazem uso de um *software* chamado *Matrix Police*[8], o qual pode capturar placas de veículos mesmo em alta velocidade. O programa, utilizado pela polícia local desde 1994, pode ser instalado em computador e câmera genéricos.

2.3 Detecção de placas no Brasil

No Brasil, não há registros de agências de segurança que utilizem sistemas de reconhecimento de placas de automóvel atualmente. Há, porém, um sistema em testes em São Paulo, acoplado a ônibus que trafegam sobre faixas exclusivas para o transporte coletivo, e que monitora a via em que o veículo se encontra[9]. A ideia é fiscalizar o tráfego ilegal de veículos particulares que se movimentam na faixa exclusiva.

A Prefeitura de São Paulo, por sua vez, apresentou em 2011 um projeto de rede de câmeras OCR a ser instalado na cidade[10], tanto de forma fixa quanto embarcada. O sistema deve, de acordo com o previsto pelo projeto, informar sobre o licenciamento e os bloqueios administrativos ou judiciais que o veículo possua, sendo estes dados disponíveis *online* para consulta na sede do órgão demandante ou em terminais embarcados.

2.4 Técnicas acadêmicas

Esta seção trata da análise das práticas existentes na academia para extração de placas de automóveis de uma imagem. A principal diferença entre elas é a característica da placa na qual elas focam, como cores específicas ou o contorno da placa. Estas técnicas também levam em consideração a constante mudança do ambiente em que a placa se

[5] https://www.theiacp.org/PublicationsGuides/ContentbyTopic/tabid/216/Default.aspx?id=1895&v=1

[6] http://investigations.nbcnews.com/_news/2013/07/19/19548772-license-plate-data-not-just-for-cops-private-companies-are\-tracking-your-car

[7] http://www.independent.co.uk/news/science/surveillance-uk-why-this-\revolution-is-only-the-start-520396.html

[8] http://www.anpr.hu/matrix-police/

[9] http://www1.folha.uol.com.br/cotidiano/2013/09/1341903-sp-testa-onibus-que-fala-e-flagra-quem-invade-a-faixa.shtml

[10] http://www.prefeitura.sp.gov.br/cidade/secretarias/upload/seguranca_urbana/arquivos/rededecamerasocr.pdf

encontra, como iluminação, reflexo e trepidação da câmera.

2.4.1 Extração de placas de automóveis utilizando informação de borda

O formato retangular da placa é uma informação importante que pode ser utilizada para extraí-la da imagem. A forma mais comum de detectar esta característica se dá através da detecção de borda.

Em [13], o filtro Sobel é utilizado para a detecção de bordas em imagens preto e branco. Com isto, é possível detectar as bordas da placa devido à diferença de suas cores com as do corpo do automóvel. Porém, o artigo justifica que o filtro é aplicado apenas na vertical (ou seja, apenas ocorre a detecção de bordas que estejam na horizontal) pois foi observado que a maioria dos veículos possuem mais linhas horizontais do que verticais, o que prejudica o desempenho do algoritmo. Estas bordas verticais são então combinadas formando retângulos candidatos. Estes, por fim, são considerados como sendo uma placa de automóvel apenas se mantiverem as proporções de uma placa.

Em [1], foi proposto outro algoritmo de detecção de bordas verticais (*Vertical Edge Detection Algorithm*, ou VEDA). De acordo com os autores, o mesmo é de sete a nove vezes mais rápido do que o filtro Sobel.

Em [10], a transformada Hough foi utilizada para detectar as linhas que delimitam a placa. A imagem é pré-processada com o filtro Sobel e pode detectar placas com até 30% de inclinação. Porém, o seu custo computacional é alto.

Apesar de simples e relativamente rápidos se comparados a outros métodos, os algoritmos de detecção de borda dependem também da continuidade das bordas [4]. Para contornar este problema, em [9] é proposta uma análise morfológica e estatística que elimina bordas indesejadas, aumentando, assim, a taxa de acerto do algoritmo.

2.4.2 Extração de placas de automóveis utilizando características de textura

Este método faz a análise da variação de intensidade dos tons de cinza entre os caracteres e o plano de fundo da placa. Esta variação gera áreas com bordas de alta intensidade, as quais podem ser detectadas por uma série de diferentes técnicas.

Em [16], é feita a análise de uma certa amostra de linhas horizontais da imagem (em escala de cinza), a fim de melhorar o desempenho do algoritmo. Sobre estas linhas, é aplicado o filtro Sobel, dando ênfase a regiões de transição de intensidade de tom de cinza. Para cada linha, é gerado então um perfil de borda, de onde é possível extrair os picos de transição de tom de cinza. A análise da distância entre estes picos, bem como a quantidade deles é comparada com as proporções dos caracteres, o que então resulta nas possíveis regiões da imagem onde há uma placa.

A irregularidade presente na textura de uma placa também é analisada por [3]. A técnica descreve irregularidades "locais"na imagem através de estatísticas como desvio padrão e valor médio. Neste trabalho, é apresentada uma técnica de segmentação adaptativa baseada nessas estatísticas e chamada de “Janela Deslizante Concêntrica”. O algoritmo foi desenvolvido através de dois passos:

1. Duas janelas deslizantes A e B de tamanho $(2X_1) \times (2Y_1)$ *pixels* e $(2X_2) \times (2Y_2)$ *pixels*, respectivamente, percorrem a imagem.

2. Para cada posição das janelas deslizantes, é calculada a média e desvio padrão.

A segmentação da imagem ocorre da seguinte forma: se a proporção entre as medidas estatísticas calculadas para o par de janelas deslizantes A e B for maior do que

um limiar(L) adaptativo estabelecido pelo usuário, então o *pixel* central das janelas será considerado como pertencente a uma Região de Interesse (RI).

Considerando (x, y) como sendo as coordenadas do *pixel* a ser analisado na imagem I_1, o valor do *pixel* $I_1(x, y)$ é setado como 0 (não faz parte da RI) ou 1 (faz parte da RI) de acordo com

$$I_1(x,y) = \begin{cases} I_{AND}(x,y) = 0 & \text{se } \frac{M_B}{M_A} \leq L(x,y) \\ I_{AND}(x,y) = 1 & \text{se } \frac{M_B}{M_A} > L(x,y) \end{cases} \tag{2.1}$$

onde I_{AND} é a imagem binarizada resultante e M a medida estatística (desvio padrão ou valor médio). Os parâmetros X_1, X_2, Y_1, Y_2 e L podem ser adaptados manualmente de acordo com alguma aplicação específica, mas de acordo com os autores do artigo, os melhores resultados foram encontrados quando as dimensões das janelas concêntricas possuíam proporção semelhante ao objeto a ser encontrado (neste caso, a placa). Em relação ao limiar local adaptativo L, foi escolhido o de Sauvola [14], sendo seu valor calculado como

$$L(x,y) = m(x,y) + \left[1 + k.\left(\frac{\sigma(x,y)}{R} - 1\right)\right] \tag{2.2}$$

com $m(x, y)$ e $\sigma(x, y)$, sendo a média aritmética local e a variância, respectivamente. Savoula sugere os valores $k = 10$ e $R = 128$, os quais são utilizados no algoritmo proposto. Em seguida, é feita uma análise dos componentes conexos da imagem, como proporção entre suas dimensões, área, orientação, entre outros, sendo então gerado o candidato à placa.

Uma desvantagem de algoritmos baseados exclusivamente em detecção de textura é que sua aplicação em cenas complexas dificilmente gera bons resultados, principalmente em imagens com bordas que geram um alto valor de variância, como o para-choque da figura 2.1 ou o fundo do cenário. Neste último caso, todavia, é possível criar uma fase prévia que exclua o fundo da imagem, evitando com que ele interfira nos cálculos estatísticos exigidos pela técnica. Além disso, a combinação de partes destes algoritmos com outras abordagens pode gerar resultados satisfatórios e com bom desempenho, se comparados a outros métodos.

Figura 2.1: Padrão de para-choque que gera uma alta variância.

2.4.3 Extração de placas de automóveis utilizando características dos caracteres

Métodos de detecção baseados nos caracteres também foram propostos. Eles examinam a imagem em busca de caracteres, e caso esses sejam encontrados, a região na qual eles se encontram é considerada como sendo uma placa.

Em [6], a primeira identificação é feita por meio do tamanho dos caracteres e pela diferença entre a cor de fundo da placa e o caractere. A placa é então extraída encontrando-se a distância inter-caracteres na região da placa. De acordo com os autores, os experimentos obtiveram uma taxa de extração de placas de 99,5%.

A procura por placas através dos caracteres é feita em [2] analisando-se objetos binarizados que possuam, no mínimo, 30 *pixels* e sejam proporcionais às dimensões de uma

placa. É aplicada então a transformada Hough na parte superior e inferior de cada um destes objetos à procura de um segmento de reta nesta região, e caso sejam localizadas duas retas paralelas e com um certo número de objetos conexos entre elas, a região em questão é considerada uma placa veicular.

Os algoritmos que procuram por textos na imagem possuem a desvantagem de demandar alto poder computacional, pois processam, na maioria das vezes, todos os componentes conexos em busca de caracteres. Outra característica das cenas que pode provocar um alto índice de erros é a presença de outros textos na imagem, os quais podem ser erroneamente considerados como sendo parte de uma placa veicular[8].

2.4.4 Extração de placas de automóveis utilizando detecção de veículo

Na área de assistência ao motorista, é abundante a variedade de algoritmos que detectam outros veículos para evitar colisões e prever eventos futuros. Estes algoritmos também podem ser utilizados para a detecção de placas, pois, uma vez que é sabida a localização da parte de um carro, por exemplo, é possível inferir a região onde provavelmente uma placa se encontra.

Em [11], é proposto um algoritmo que procura pela sombra dos veículos que é projetada na pista. Tendo em vista que a área abaixo do carro é distintamente mais escura do que o resto da via[12], o algoritmo procura por transições verticais do cinza claro para o cinza escuro (escaneando de baixo para cima). As bordas com transições verticais são definidas como potenciais bordas de uma Região de Interesse (RI) e são armazenadas para posterior análise, dentre elas, a procura por um eixo de simetria. Por fim, é utilizado o classificador *Adaboost* para filtrar os candidatos.

Uma abordagem semelhante é feita em [18]. Inicialmente, é feita a identificação da distribuição de intensidade da superfície da pista. Esta análise é feita a partir de um algoritmo de detecção do espaço livre para dirigir (*free-driving-space*, ou f-d-s). O f-d-s é definido pela via observada diretamente à frente da câmera. Para delimitar esta área, é feito inicialmente o reconhecimento de bordas na imagem. Posteriormente, o espaço é determinado pela menor região homogênea central. O valor resultante é utilizado para a detecção de *pixels* que possuem uma transição vertical do cinza claro para o cinza escuro. A partir desta seleção, é feita a procura por pixels conectados horizontalmente. Apenas segmentos de linha maiores do que um limite preestabalecido são então escolhidos. Acima de cada linha potencialmente conectada a um veículo, é definida uma Região de Interesse, e seu tamanho é correspondente ao comprimento do segmento de linha horizontal.

Tendo a RI definida, é feita uma análise da entropia da porção da imagem. Se há um veículo na RI, então é provável que as suas bordas laterais - bem como outras variações de intensidade na direção horizontal entre estas bordas - provocarão uma alta entropia (na horizontal). A ausência de um veículo na RI e a presença de maior área homogênea geram uma baixa entropia. Posteriormente, é feita a análise da simetria da RI, tendo em vista que automóveis, em geral, possuem um eixo vertical de simetria bem definido.

2.4.5 Extração de placas de automóveis a partir de suas cores

Métodos baseados nas cores de placas também foram propostos na literatura. A hipótese inicial é de que as combinações de cores presentes nas placas ocorrem, na maioria dos casos, apenas nas placas[15]. Em [7], placas da Coreia do Sul são extraídas da imagem com base nos seus diferentes arranjos de cores. No método proposto, a imagem é inicialmente convertida para o sistema de cores HSI (*Hue, Saturation and Intensity*).

Este sistema descreve as cores a partir de sua matiz, saturação e intensidade, sendo estes valores utilizados para binarizar a imagem de acordo com a cor da placa que se deseja encontrar. São gerados então componentes conexos na imagem resultante, os quais são analisados isoladamente levando em conta suas propriedades geométricas e a presença ou não de caracteres, que por sua vez pode ser verificada através de histogramas de intensidade.

Uma abordagem semelhante é utilizada por [19]. A procura por placas se dá no espaço de cores RGB, onde a cor de cada *pixel* é comparada com um intervalo de cores tido como característico de uma placa. Quando a comparação acusa que a cor pertence ao intervalo, são verificados os *pixels* vizinhos, e caso estes também estejam dentro do intervalo, a posição do *pixel* inicial é marcada em uma nova imagem como sendo ponto de borda horizontal ou vertical, dependendo de qual dos seus vizinhos também pertencia ao intervalo. Essa nova imagem é então verificada, entre outros métodos, por um limiar que estipula um número mínimo de bordas para uma certa região ser considerada candidata.

A procura por placas através da cor tem a vantagem de sofrer pouca interferência da orientação ou nitidez das placas. Entretanto, esses mesmos algoritmos normalmente tem seu desempenho afetado quando o veículo e a placa tiverem cores semelhantes. A constante variação da iluminação também faz com que os tons de cores fiquem instáveis. Este problema é mais evidente quando se utiliza o sistema RGB para identificar as cores, enquanto o sistema HSI apresenta melhores resultados. Este último, contudo, é sensível a ruídos[8].

2.5 Discussão

As técnicas de detecção apresentadas neste capítulo são de difícil comparação entre si devido ao fato de não existir um banco de dados de placas veiculares padronizado atualmente. Assim o são também as opções comerciais, que divulgam dados de desempenho baseados em testes distintos para cada fabricante. Optou-se, então, por escolher as técnicas principalmente pela sua eficiência do ponto de vista de custo computacional (mesmo que isto signifique menor taxa de acerto) e robustez necessárias para o propósito inicial.

Dentre as técnicas apresentadas, as utilizadas serão a de detecção de traseira de veículo proposta por [18] e posteriormente a busca pela placa através de suas características de textura. A motivação para o uso destes dois métodos vem do fato de que o primeiro deles elimina boa parte da imagem que possui altos valores da variância, além de ser rápido e diminuir consideravelmente a região de busca. Já o segundo método foi selecionado por ser robusto à ausência de variação entre a borda da placa e o veículo, o que é característica comum de carros de cor branca ou assemelhado.

3 HARDWARE

Este capítulo detalha os componentes utilizados na confecção do protótipo. A primeira seção detalha a estrutura de suporte criada para comportar o hardware na posição ideal de captura de imagens. A segunda seção descreve o computador que processa os dados, enquanto a terceira seção caracteriza a câmera utilizada no protótipo. Por fim, a quarta seção apresenta uma tabela com os valores de cada componente.

3.1 Estrutura de suporte

A fim de posicionar o protótipo no para-brisa do veículo, foi desenvolvido um suporte que oferece estabilidade e padronização no ângulo da câmera, sendo necessária apenas a correção da inclinação do aparelho, dependendo da inclinação do para-brisa.

Como observado na figura 3.1, a fixação é feita por meio de ventosas. Há uma cavidade para a colocação da câmera e para os demais dispositivos de entrada e saída do computador.

Figura 3.1: Estrutura de suporte.

3.2 Raspberry Pi Model B

O computador utilizado no protótipo trata-se de um *Raspberry Pi Model B*[1]. Do tamanho de um cartão de crédito, ele foi desenvolvido objetivando o estímulo do ensino de Ciência da Computação em escolas. Possui um processador ARM11 de 700 MHz, com memória de 512MB, GPU Broadcom VideoCore IV de 250 MHz, porta USB 2.0 e porta

[1] `http://www.raspberrypi.org`

Ethernet 10/100. O *Raspberry Pi* é compatível com sistemas operacionais baseados em *Linux*. É possível visualizar o computador na figura 3.2.

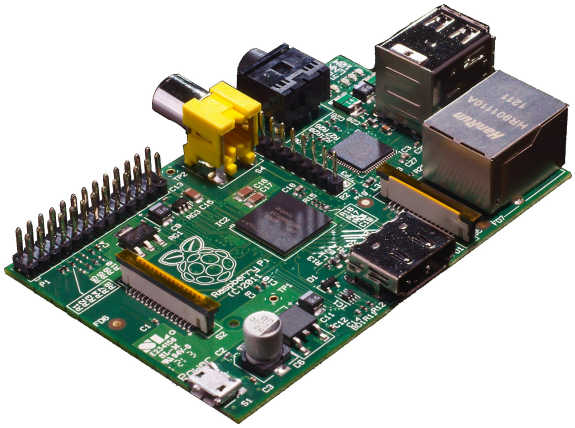


Figura 3.2: Raspberry Pi Model B.

Para alimentar o dispositivo, foi adquirido uma bateria *Sony CP-EL* de 2000mAh. Ela é conectada ao *Raspberry Pi* por um cabo USB *Samsung U2*. Opcionalmente, o aparelho pode ser alimentado diretamente pela energia do veículo através de um carregador universal veicular USB *MOX MO-U58*. Este também é conectado ao dispositivo pelo cabo USB. Estes componentes podem ser visualizados nas figuras 3.3, 3.4 e 3.5.

Figura 3.3: Bateria *Sony CP-EL*.

Figura 3.4: Cabo USB *Samsung U2*.

Figura 3.5: Carregador Universal veicular USB *MOX MO-U58*.

3.3 Câmera

A câmera utilizada no protótipo é a *Raspberry Pi Camera* 69W0689[2]. Ela foi desenvolvida especialmente para o *Raspberry Pi*, utilizando, de forma otimizada, o potencial de sua GPU. Ela possui 5 *megapixels* com lente de foco fixo. É capaz de gerar imagens estáticas de 2592 x 1944 *pixels*, suportando vídeos de 1080p30, 720p60 e 640x480p60. É acoplada ao *Raspberry Pi* através de um soquete na placa e utiliza uma interface *CSi* dedicada, criada especialmente para realizar a interface com câmeras. É possível visualizar a câmera na figura 3.6.

3.4 Custo total dos componentes

A tabela 3.1 traz os valores de cada componente utilizado na confecção do protótipo. Todos foram adquiridos em 2013.

[2] `http://www.raspberrypi.org/camera`

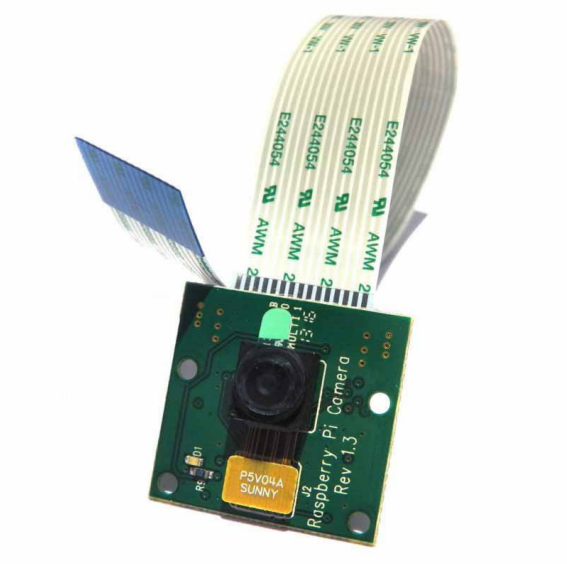


Figura 3.6: Câmera para o Raspberry Pi.

Tabela 3.1: Custo individual dos componentes e custo total do protótipo.

Componente	**Preço**
Raspberry Pi Model B	R$ 195,80
Raspberry Pi Camera	R$ 175,02
Estrutura de suporte	R$ 10,00
Cabos (de energia, de dados)	R$ 50,00
Total	**R$ 430,82**

4 IMPLEMENTAÇÃO

A detecção de placas em uma imagem demanda uma série de análises sobre diferentes características. A eficiência em termos de tempo é crucial em cada uma destas etapas e, portanto, é um fator limitante. As etapas iniciais, em especial, são as que mais merecem atenção em relação a este aspecto, pois são elas que irão fornecer os dados a serem processados pelas fases subsequentes. Além disso, quanto mais precisas forem as hipóteses iniciais, melhor será a qualidade e eficiência do processo de detecção de placas.

Neste capítulo, será detalhada cada etapa necessária para a obtenção das placas de automóveis presentes na imagem até seu envio para um servidor externo. Antes da explicação propriamente dita, é feita uma breve descrição da placa de trânsito brasileira na seção 4.1. Já em relação à implementação, a primeira etapa, detalhada na seção 4.2, consiste em capturar a imagem através da câmera, juntamente com determinadas modificações para torná-la ideal para os processamentos subsequentes. A etapa seguinte descreve como é feita a procura por veículos na imagem. Esta busca é feita a partir do rastreamento da sombra dos veículos pois, como demonstrado por [12], a sombra pode ser utilizada como um padrão de busca para a detecção de automóveis. O tom de cinza da via é um dos dados necessários para a obtenção de tal informação, sendo sua obtenção detalhada na seção 4.3. O passo seguinte, de detecção da sombra dos veículos, é apresentando na seção 4.4. Na seção 4.5 é feita a extração da placa da traseira do veículo, sendo esta porção da imagem o dado final enviado ao servidor externo, enquanto a seção 4.6 trata do ambiente de desenvolvimento para a criação do *software*.

4.1 A placa veicular brasileira

A placa veicular brasileira é atualmente regulamentada pelas Resoluções 241 e 231[1] (em vigor desde 1 de janeiro de 2008) do Conselho Nacional de Trânsito (CONTRAN). Entre as padronizações mais importantes estão as dimensões (400 milímetros de largura por 130 milímetros de altura) e a fonte *Mandatory*. As diversas cores de fundo e de fonte que a placa pode assumir, de acordo com o tipo de veículo, não interferem no processo de detecção. A figura 4.1 ilustra um modelo de placa de veículo particular.

4.2 Captura da Imagem

Apesar da câmera permitir captura de vídeos de até 1080p30, optou-se pela resolução de 640x480p60. O principal motivo é a grande diferença de desempenho entre as duas resoluções, além do fato de que a resolução maior abrangia partes onde é pouco provável

[1] `http://www.denatran.gov.br/download/Resolucoes/RESOLUCAO_231.pdf`

Figura 4.1: Placa veicular brasileira.

a presença de um veículo. Um exemplo de captura pode ser visto na figura 4.2.

Figura 4.2: Imagem capturada pela câmera.

Devido ao reflexo provocado pelo painel do veículo, optou-se pela eliminação desta porção da imagem na procura por placas de automóveis. A imagem também é, nesta fase, convertida para tons de cinza, o que agiliza o processo de detecção e elimina informação desnecessária para as próximas etapas. A imagem gerada ao fim do processo é visualizada na figura 4.3.

4.3 Obtenção do tom de cinza da via

O objetivo desta etapa é obter um valor que possa ser utilizado como limite superior para a determinação das sombras na imagem, ou seja, apenas valores abaixo deste calculado serão considerados posteriormente. O algoritmo de obtenção do tom de cinza da via é uma adaptação do algoritmo proposto por [17]. Inicialmente, a imagem gerada pela etapa anterior é diminuída para 30% do seu tamanho inicial. Esse redimensionamento visa agilizar o processo de obtenção do tom de cinza da via, além do fato de que tal ação

Figura 4.3: Imagem após a eliminação da região refletida pelo painel e transformação para tons de cinza.

pouco interfere no resultado do algoritmo.

Em seguida, aplica-se o detector de bordas *Canny*[5] sobre a imagem. Este detector funciona basicamente analisando o vetor gradiente de cada *pixel*. Antes de aplicá-lo, porém, é realizada uma operação de suavização da imagem, onde cada *pixel* passa a receber o valor médio do tom de cinza de seus vizinhos, de forma a eliminar ruídos. Para isso, utiliza-se um kernel baseado na derivada de uma Gaussiana (cujo desvio padrão σ pode ser ajustado). Abaixo, um exemplo de filtro Gaussiano com $\sigma = 1.4$, onde $*$ representa a operação de convolução.

$$\mathbf{K} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{Img}. \tag{4.1}$$

Para obter a intensidade do vetor gradiente G, inicialmente são aplicados dois *kernels* sobre a imagem (na direção x e y:)

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \tag{4.2}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{4.3}$$

e em seguida G é obtido por

$$G = \sqrt{G_x^2 + G_y^2} \tag{4.4}$$

Entretanto, é necessário estipular um limite superior e inferior do valor do gradiente dos *pixels* que podem ser considerados como pertencentes a uma borda. Tais valores, porém, não podem ser calculados previamente, pois dependem de características do momento da filmagem como a iluminação, que por sua vez depende da intensidade da luz do Sol e, portanto, é inconstante. Por este motivo, não é possível utilizar um limite fixo nesta situação. Para contornar este problema, o detector de bordas *Canny* é aplicado diversas

Figura 4.4: Imagem após a aplicação do filtro *Canny*.

vezes sobre a cena, até que o próximo passo do algoritmo (de detecção da região livre para trafegar) acuse um percentual mínimo de 20% da imagem como sendo pertencente à via, o qual foi obtido experimentalmente. O resultado é visualizado na figura 4.4.

O passo seguinte visa uma aproximação grosseira da região livre para trafegar. O limite superior é delimitado, no pior caso, pelo horizonte. Inicialmente é feita uma varredura na imagem resultante da aplicação de *Canny*, de baixo para cima, até encontrar uma borda. O algoritmo desta etapa pode ser visualizado no pseudo-código 4.1.

```
1  for(lin=0; lin<imgOriginal.lin; lin++)
2  {
3    col = imgOriginal.col;
4    while(col!=0 && imgOriginal[lin][col]!=borda)
5    {
6      col--;
7      regiaoLivreTrafegar[lin][col] = imgOriginal[lin][col];
8    }
9  }
```

Código 4.1: detecção da região livre para trafegar.

O que se pode visualizar na figura 4.5 é a porção da imagem original que foi considerada região livre para trafegar, de acordo com o algoritmo (região com tons de cinza). Já a região escura são os *pixels* que foram excluídos por esta filtragem.

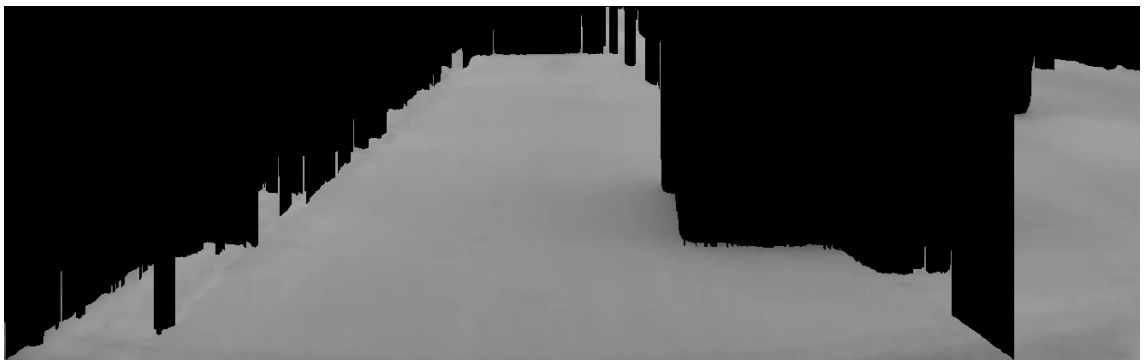


Figura 4.5: Extração da região livre para trafegar na imagem.

Após este procedimento, calcula-se a mediana dos pixels restantes. A escolha pela mediana deve-se ao fato de ela ser mais robusta a pixels com valores atípicos do que a média, por exemplo.

Devido ao fato de que a obtenção do tom de cinza da via é um procedimento custoso computacionalmente (pelo fato de utilizar o detector de bordas *Canny*), optou-se por realizá-lo apenas na inicialização do sistema do protótipo e nos intervalos onde não é constatada a detecção de nenhuma placa. Este lapso de captura de placas pode indicar que o tom atual da via está muito diferente do calculado previamente, sendo portanto necessário um recálculo deste valor.

4.4 Detecção da traseira de veículos a partir de suas sombras

Obtido o valor mediano do tom de cinza da via, é possível agora localizar as sombras dos veículos presentes na imagem. Para isto, de acordo com [17], é necessário extrair as bordas horizontais que correspondem à transição entre a via e a sombra abaixo do veículo. O primeiro passo nesta etapa consiste em varrer a imagem analisando o tom de cinza de cada *pixel*. Caso o *pixel* atual possua um tom mais escuro do que o valor calculado para a via (com uma margem de 20% de tolerância para tolerar ruídos), e seu vizinho inferior (uma linha abaixo) possuir um tom mais claro que o *pixel* atual, ele será marcado para avaliação na fase seguinte. O algoritmo pode ser visualizado no pseudo-código 4.2. Na figura 4.6, estes *pixels* estão representados pela cor branca.

```
1  for(lin=imgOriginal.lin; lin>0; lin--)
2  {
3      for(col=0; col<imgOriginal; col++)
4      {
5          pixelAtual = imgOriginal[lin,col];
6          pixelVizinhoAbaixo = imgOriginal[lin+1,col];
7
8          if((pixelAtual<medianaViaLivre*0.8)
9           &&(pixelVizinhoAbaixo>pixelAtual))
10             imgResultado[lin,col]=corBranca;
11         else
12             imgResultado[lin,col]=corPreta;
13     }
14 }
```

Código 4.2: detecção de *pixels* com potencial de pertencer a sombras de automóveis.

Figura 4.6: *Pixels* marcados como candidatos a pertencerem à sombra de veículos.

O próximo passo do algoritmo proposto visa criar componentes conexos que tenham características esperadas de uma sombra de veículo. Esta busca é feita analisando-se a imagem de baixo para cima e da esquerda para a direita, selecionando apenas os *pixels*

candidatos da etapa anterior e que contenham, no mínimo, um vizinho (consideram-se os 8 *pixels* que estão a sua volta) também candidato. Ao mesmo tempo, um contador do tamanho do segmento é atualizado, até que se encontre um *pixel* que não satisfaça as condições anteriores. Neste ponto, o tamanho do segmento é analisado, e caso ele seja menor do que o tamanho mínimo estipulado, o segmento é apagado da imagem. Também é analisada a relação entre o tamanho do segmento com sua posição na imagem, ou seja, quanto mais próximo o carro estiver, maior será sua sombra, e quanto mais distante estiver, menor a sombra será. Para realizar esta verificação, é levada em conta a posição no eixo y na qual a sombra se encontra. O algoritmo pode ser visto no pseudo-código refsegmSombra. As linhas resultantes são exibidas na figura 4.7.

```
for(lin=imgOriginal.lin; lin>0; lin--)
{
  for(col=0; col<imgOriginal; col++)
  {
    if(PixelAtualPossuiVizinhoCandidato(imgOriginal[lin][col]))
    {
      tamanhoSegmento++;
      imgResultado[lin][col]=corBranca;
    }
    else
    {
      //se o segmento for menor do que um tamanho mínimo ou não for
          proporcional à sua localização no eixo y, ele deve ser
          excluído
      if((tamanhoSegmento<limiteMinimo)
      ||(((tamanhoSegmento>col*1.35) || (tamanhoSegmento<col*0.65))))
      {
        ApagarSegmento(tamanhoSegmento,lin,col);
      }
      tamanhoSegmento=0;
    }
  }
}
```

Código 4.3: filtragem dos pixels que pertencem a segmentos candidatos à sombra.

Figura 4.7: Resultado da aplicação do código 4.3 à figura 4.6, com os *Pixels* que pertencem a um componente conexo destacados com a cor branca.

Estes segmentos de reta são então analisados a partir dos pontos (x_1, y_1) e (x_2, y_2), correspondentes ao ponto mais à esquerda e mais à direita do segmento, respectivamente. Como observa-se na figura 4.7, foram gerados diversos segmentos de reta que não representam a sombra do veículo. Para eliminar estes resultados incorretos, os candidatos à

sombra são analisados aos pares da seguinte forma:

```
1  if((sombra1.x1 > sombra2.x1 && sombra1.x1 < sombra2.x2)
2  && (sombra1.x2 > sombra2.x1 && sombra1.x2 < sombra2.x2)
3  && (sombra2.x1 > sombra1.x1 && sombra2.x1 < sombra1.x2)
4  && (sombra2.x2 > sombra1.x1 && sombra2.x2 < sombra1.x2))
5  {
6    apagarSombraMaisAcima(sombra1,sombra2);
7  }
```

Código 4.4: filtragem dos pixels que pertencem a segmentos candidatos à sombra.

O teste analisa os segmentos apenas no eixo x, procurando por intersecções entre eles. Caso encontre uma intersecção, a linha que possui o menor y, ou seja, está mais próxima da borda superior da imagem, é apagada. Como a análise é feita iniciando pelos segmentos com os maiores y, os candidatos à sombra que estiverem mais abaixo na imagem são favorecidos.

Para calcular a altura da traseira dos veículos, é utilizado um fator de escala referente à posição no eixo y na qual a sombra se encontra, pois constatou-se que os automóveis mais distantes encontram-se, na maioria dos casos, próximos à borda superior da imagem. A seguir, a fórmula que relaciona altura com o eixo y:

$$altura = 0.6 * y \tag{4.5}$$

A região de interesse final é então mapeada para a imagem original, a fim de eliminar a redução na imagem realizada no início da execução deste algoritmo. O resultado é visível na figura 4.8.

Figura 4.8: Região detectada como traseira de veículo.

4.5 Extração da placa da traseira do veículo

A região de interesse gerada pelas etapas anteriores poderia já ser analisada diretamente por um algoritmo de reconhecimento ótico de caracteres. Porém, devido ao limitado poder de processamento do *Raspberry Pi*, o desempenho ainda compromete o bom funcionamento do algoritmo. Por este motivo, optou-se por restringir ainda mais a área em torno dos caracteres, compreendendo apenas a placa em si.

Para atingir este objetivo, foram feitas diversas tentativas que procuravam reconhecer a placa a partir de diferentes características. Uma delas foi a busca por retângulos, a partir

da aplicação de *Canny* e a posterior combinação das retas até que se encontrasse um padrão retangular entre as candidatas, que deveria corresponder à placa. Os resultados, contudo, geravam uma série de falsos positivos, o que motivou a busca por alternativas.

Decidiu-se, por fim, utilizar a informação da irregularidade da textura da placa para rastreá-la na cena. Isto é feito através da busca por regiões que apresentam um valor do desvio padrão amostral local acima de um limiar previamente calculado, ocasionado pela transição brusca no tom de cinza entre letras e a cor de fundo das placas.

Para efetuar este cálculo, é necessária a criação de uma matriz que contenha o desvio padrão amostral de cada *pixel* em relação à sua vizinhança, para que seja possível, a partir dela, localizar regiões de alta intensidade do desvio padrão. Esta matriz será chamada de dpa.

Posteriormente, aplica-se o filtro *Sobel* sobre a imagem apenas no sentido horizontal, a fim de se obter a intensidade do vetor gradiente dos *pixels*. Tal procedimento evidencia a presença de linhas verticais, a qual é uma das características da placa que a distingue das demais regiões da imagem. A aplicação do filtro na vertical não se mostrou útil, uma vez que diversos elementos dos veículos, como o para-choque, possuem longos traços horizontais, interferindo no processo de detecção. A aplicação do filtro na horizontal resulta na figura 4.9. Já a aplicação na vertical é exibida na figura 4.10.

Figura 4.9: Imagem após a aplicação do filtro *Sobel* na horizontal.

Figura 4.10: A aplicação do filtro *Sobel* na vertical evidencia diversos outros elementos além da placa.

Obtido o valor do gradiente horizontal de cada *pixel*, pode-se agora calcular o desvio padrão amostral (dpa) local dos gradientes utilizando uma janela deslizante sobre a imagem. Já o cálculo do desvio padrão amostral de cada janela deslizante pode ser feito através da fórmula

$$dpa_{i_c,j_c} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{I}\sum_{j=1}^{J}(x_{i,j}-\overline{x})^2} \tag{4.6}$$

onde N corresponde ao número de *pixels* dentro da janela deslizante, i_c e j_c o valor de i e j do ponto central da janela deslizante e I e J a largura e altura da janela, respectivamente. Entretanto, o custo computacional para efetuar os cálculos do desvio padrão amostral da forma representada acima é alto. Por isto, optou-se por uma alternativa mais eficiente que faz uso de imagens integrais. O cálculo passa a ser feito da seguinte forma:

$$dpa_{i_c,j_c} = \frac{\sqrt{N * s_2 - {s_1}^2}}{N} \tag{4.7}$$

com

$$\begin{aligned} s_1 = {} & \mathtt{som}(i_c - I/2, j_c - I/2) + \mathtt{som}(i_c + I/2, j_c + I/2) \\ & - \mathtt{som}(i_c - I/2, j_c + I/2) - \mathtt{som}(i_c + I/2, j_c - I/2) \end{aligned} \tag{4.8}$$

$$\begin{aligned} s_2 = {} & \mathtt{som2}(i_c - I/2, j_c - I/2) + \mathtt{som2}(i_c + I/2, j_c + I/2) \\ & - \mathtt{som2}(i_c - I/2, j_c + I/2) - \mathtt{som2}(i_c + I/2, j_c - I/2) \end{aligned} \tag{4.9}$$

Já som e $som2$ são o somatório e o somatório quadrado dos tons de cinza da imagem que, para cada ponto (X, Y), são calculados da forma abaixo:

$$\mathtt{som}(X,Y) = \sum_{x=1}^{X}\sum_{y=1}^{Y}\mathtt{img}(x,y) \tag{4.10}$$

$$\mathtt{som2}(X,Y) = \sum_{x=1}^{X}\sum_{y=1}^{Y}\mathtt{img}(x,y)^2 \tag{4.11}$$

O ganho de tempo vem do fato de que o cálculo de som e $som2$ é feito para toda a imagem somente uma vez com complexidade $O(n)$ cada (sendo n o número de *pixels*), mas o cálculo do desvio padrão amostral em cada janela deslizante tem complexidade $O(1)$. O mesmo não ocorre com a fórmula inicial, pois à medida que a janela deslizante cresce de tamanho, cresce também o custo de calcular seu desvio padrão amostral (complexidade $O(mn)$, com m sendo a área da janela deslizante), além de haver maior número de cálculos de divisão, os quais são caros computacionalmente.

Em relação ao tamanho da janela deslizante, é importante lembrar que ele deve possuir certa relação com o tamanho da placa em coordenadas de imagem, afinal, caso a placa ocupar 50% da região de interesse destacada da traseira do veículo, uma janela deslizante muito pequena pode não acusar a variação no desvio padrão local da placa. Por este motivo, a geração da matriz dpa final é feita a partir do valor médio da aplicação do algoritmo com janelas de diferentes tamanhos. O pseudo-código 4.5 demonstra esta e as demais etapas apresentadas até agora nesta seção.

```
1 for(numJanela=0;numJanela<3;numJanela++)
2 {
3   tamJanela = tamanhos[numJanela];
4   numPixels = (2*tamJanela+1)*(2*tamJanela+1);
5   for(col=1+tamJanela; col<imagem.lin-tamJanela-1; col++)
```

```
6     {
7       for(lin=tamJanela; lin<imagem.col-tamJanela-1;lin++)
8       {
9         s =  somatório(col+tamJanela+1,lin+tamJanela+1)
10             +somatório(col-tamJanela, lin-tamJanela)
11             -somatório(col-tamJanela, lin+tamJanela+1)
12             -somatório(col+tamJanela+1, lin-tamJanela);
13
14        s2 = somatórioQuadrado(col+tamJanela+1,lin+tamJanela+1)
15             +somatórioQuadrado(col-tamJanela, lin-tamJanela)
16             -somatórioQuadrado(col-tamJanela, lin+tamJanela+1)
17             -somatórioQuadrado(col+tamJanela+1, lin-tamJanela);
18        dpas(col, lin, numJanela)=sqrt(numPixels*s2-s*s)/numPixels;
19      }
20    }
21  }
22  //calculando o valor médio entre as janelas deslizantes de tamanhos
        diferentes
23  for(col=0;col<imagem.col;col++)
24  {
25    for(lin=0;lin<imagem.lin;lin++)
26    {
27      dpa(col,lin)=(dpas(col,lin,0)+dpas(col,lin,1)+dpas(col,lin,2))/3;
28    }
29  }
```

Código 4.5: cálculo do desvio padrão local para janelas de tamanho variável.

Com a matriz de desvio padrão amostral local calculada, é possível agora filtrar as regiões onde este valor é mais alto. Definir um limiar fixo é uma tarefa complicada, pois os desvios padrão local dependem de vários fatores externos, como iluminação e ruído da câmera. Assim, optou-se pela escolha de um limiar adaptativo, dado por:

$$\begin{aligned} Limiar = 0.5 * (&elemDesvioPadraoOrdenados(tamanhoVetor * 0.5) \\ &+elemDesvioPadraoOrdenados(tamanhoVetor * 0.9)); \end{aligned} \tag{4.12}$$

O que tem-se neste cálculo é a média entre o percentil 50% e o percentil 90% da distribuição. Para obter este valor, os elementos são dispostos em ordem crescente no vetor $elemDesvioPadraoOrdenados$, e então o elemento que está no meio da lista consiste no percentil 50%. Já o percentil 90%, calculado sobre a mesma lista ordenada, é uma forma de evitar com que alguns elementos presentes na traseira de veículos (principalmente adesivos) interfiram no cálculo deste limiar. Para comparar a distribuição do desvio padrão de uma imagem com e sem adesivo, foram criadas as imagens 4.11 e 4.12. Os valores estão normalizados e ordenados de forma crescente e dispostos no eixo x, com o eixo y representando o valor de cada um dos desvios padrões:

Nas figuras, as linhas vermelhas representam o percentil 50%, enquanto as linhas amarelas representam o percentil 90%. Percebe-se que na imagem 4.11 há apenas uma pequena porção da amostra com um valor de desvio padrão consideravelmente maior do que o restante. Em tese, esta pequena porcentagem corresponde à placa, por ser o único elemento que gera valores altos em relação ao restante da imagem. A figura 4.12, por sua vez, apresenta uma porção maior de desvios padrão com um valor elevado. Isso provavelmente ocorre pela presença do adesivo, o qual gera altos valores de desvio padrão. Após este cálculo, pode-se utilizar o limiar para restringir as regiões da imagem onde o desvio padrão pertence ao intervalo calculado, gerando a imagem chamada de $resBinario$ da seguinte forma:

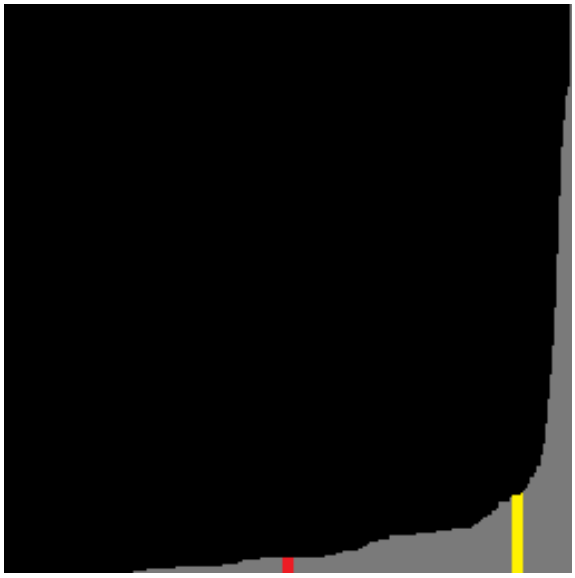


Figura 4.11: distribuição dos valores do desvio padrão de uma imagem sem adesivo.

Figura 4.12: distribuição dos valores do desvio padrão de uma imagem com adesivo.

$$resBinario(col, lin) = \begin{cases} 1 & imagem(col, lin) > Limiar \\ 0 & \text{caso contrário} \end{cases} \quad (4.13)$$

O que tem-se agora são regiões, destacadas em branco na figura 4.13, que possuem uma variação do desvio padrão acima do limiar previamente calculado. Para analisar cada uma destas regiões individualmente, optou-se por fazer a busca por componentes conexos. A tarefa foi facilitada através do uso de uma biblioteca para *OpenCV*, conhecida como *cvblob*[2]. Ela é capaz de reconhecer os componentes conexos, devolvendo suas coordenadas como centróide, altura, largura, além de rotular cada um deles. Outra informação corresponde à área em *pixels* ocupadas pelo componente conexo, a qual é utilizada para eliminar de antemão regiões consideradas pequenas demais para conterem caracteres legíveis.

Para tratar cada um dos componentes conexos, a ideia geral consiste em definir a menor região que se enquadra em um tamanho mínimo definido como sendo de uma placa

[2] `http://code.google.com/p/cvblob/`

Figura 4.13: Imagem após a aplicação de limiarização do desvio padrão local.

(caso haja alguma), sendo esta análise feita em duas etapas. A primeira consiste em uma varredura horizontal da região que cobre todo o componente conexo, calculando-se o número de *pixels* de cada coluna pertencentes ao componente conexo em questão. Em seguida, é feita uma varredura vertical e o cálculo de *pixels* é feito por linha. O pseudo-código 4.6 demonstra como isto pode ser feito:

```
1  //acumulado de cada coluna
2  for(col=componenteConectado.miny;col<componenteConectado.maxy;col++)
3  {
4     for(lin=componenteConectado.minx;lin<componenteConectado.maxx;lin++)
5     {
6           if(imagemCompConectado[lin][col]==componenteConectado.rótulo)
7                 acumuladoColuna[col]++;
8     }
9  }
10
11 //acumulado de cada linha
12 for(lin=componenteConectado.minx;lin<componenteConectado.maxx;lin++)
13 {
14    for(col=componenteConectado.miny;col<componenteConectado.maxy;col++)
15    {
16          if(imagemCompConectado[lin][col]==componenteConectado.rótulo)
17                acumuladoLinha[lin]++;
18    }
19 }
```

Código 4.6: Cálculo do número de *pixels* de cada coluna e linha pertencentes ao componente conexo.

A partir dos vetores $acumuladoColuna$ e $acumuladoLinha$, é possível filtrar os valores da seguinte forma:

$$resBinarioColuna(col) = \begin{cases} 1 & acumuladoColuna(col) > LC \\ 0 & \text{caso contrário} \end{cases} \tag{4.14}$$

e

$$resBinarioLinha(lin) = \begin{cases} 1 & acumuladoLinha(lin) > LL \\ 0 & \text{caso contrário} \end{cases} \tag{4.15}$$

onde LC e LL correspondem a valores mínimos esperados de comprimento para coluna e linha, respectivamente. Posteriormente, é feita a busca pelo maior componente conexo

dentro dos vetores $resBinarioColuna$ e $resBinarioLinha$, e seu tamanho definirá as dimensões do candidato final à placa. Um exemplo que parte de um componente conexo até o cálculo das dimensões finais é apresentado abaixo. A tabela 4.1 representa a região de um componente conexo de rótulo 3:

Tabela 4.1: Delimitação de um componente conexo de rótulo 1.

0	0	**1**	**1**	**1**	**1**	**1**	**1**	0
0	0	0	**1**	**1**	**1**	**1**	**1**	0
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0
0	0	0	**1**	0	0	0	0	0

Para cada uma das linhas, é calculado o acumulado de *pixels* pertencentes ao componente conexo:

Tabela 4.2: Valor de *pixels* acumulado de cada linha.

índice da Linha	0	1	2	3	4
acumuladoLinha	6	5	9	7	1

Em seguida, os valores são filtrados com um limiar de 6, e por fim é encontrado o maior componente conexo:

Tabela 4.3: Exemplo de localização do componente conexo dentro do vetor $resBinarioLinha$.

resBinarioLinha	1	0	1	1	0
componenteConexo	0	0	1	1	0

Para as colunas, o acumulado de *pixels* fica como demonstrado na tabela 4.4.

Tabela 4.4: Valor de *pixels* acumulado de cada coluna.

índice da Coluna	0	1	2	3	4	5	6	7	8
acumuladoColuna	1	2	3	5	4	4	4	4	1

Em seguida, os valores são filtrados com um limiar de 3, e por fim é encontrado o maior componente conexo:

Tabela 4.5: Exemplo de localização do componente conexo dentro do vetor $resBinarioColuna$.

resBinarioColuna	0	1	1	1	1	1	1	1	0
componenteConexo	0	1	1	1	1	1	1	1	0

Mapeando os pontos extremos dos componentes conexos de $resBinarioColuna$ e $resBinarioLinha$ para o índice das colunas e linhas, obtemos $minx = 1, maxx = 8, miny = 2, maxy = 3$, resultando na região em verde da tabela 4.6.

Tabela 4.6: Região final definida como candidata à placa veicular destacada em verde.

0	0	**1**	**1**	**1**	**1**	**1**	**1**	0
0	0	0	**1**	**1**	**1**	**1**	**1**	0
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0
0	0	0	**1**	0	0	0	0	0

Aplicando o mesmo algoritmo na figura 4.13 (com alteração no valor dos limiares), obtemos o resultado da figura 4.14.

Figura 4.14: Placa extraída ao final do algoritmo.

4.6 Ambiente de desenvolvimento

Para desenvolver o *software* de reconhecimento de placas, foi utilizada a linguagem C/C++, além do compilador MinGW/GCC. Também foi utilizada a biblioteca de visão computacional *OpenCV*[3]. O sistema operacional instalado no *Raspberry Pi* é o *Raspbian*[4]. É um sistema operacional grátis, baseado no *Debian* e otimizado para o *hardware* do *Raspberry Pi*.

[3] `http://www.opencv.org`
[4] `http://www.raspbian.org/`

5 RESULTADOS

Este capítulo apresentará de que forma os dados gerados pelo protótipo foram criados e analisados, bem como o impacto nos resultados de acordo com a variação de alguns parâmetros configuráveis. Para isso, são geradas tabelas comparativas, seguidas de uma explicação dos motivos que acarretaram os diferentes resultados.

5.1 Geração de dados

Para que fosse possível analisar os resultados de uma forma mais detalhada, foram realizadas gravações de trajetos percorridos pelo veículo no qual o protótipo estava instalado. Estas gravações foram feitas a partir do próprio protótipo, simulando as condições reais de funcionamento.

Foram realizadas 3 gravações distintas, as quais foram avaliadas sem cortes. Isto significa que há determinados trechos do vídeo que não possuem placas a serem avaliadas, o que representa as condições reais nas quais o protótipo terá de funcionar. Outro detalhe importante em relação a estas gravações é de que elas foram salvas em formato H264, ou seja, elas sofreram processo de compactação. Acredita-se que a análise das imagens sem esta compactação traria resultados ainda melhores.

Posteriormente, foi feita uma seleção manual de cada placa na qual fosse possível reconhecer os caracteres através da visão humana. O ideal seria, neste caso, que a seleção fosse feita beseando-se em um algoritmo de reconhecimento ótico de caracteres (OCR), porém a dificuldade em encontrar um algoritmo que tivesse um desempenho aceitável no cenário avaliado fez com que se optasse pela seleção manual. Foi constatado que essas possuem a largura mínima de 45 *pixels*, e altura mínima de 15 *pixels*. A marcação foi feita marcando-se com o *mouse* o canto superior esquerdo e inferior direito da placa, de forma a selecionar apenas os caracteres nela presentes. Após esta marcação, que ocorria quadro a quadro, era calculada a altura e largura da respectiva placa. Foi então gerado um *log* da posição e dimensão de cada uma delas, juntamente com o número do quadro a qual ela pertence.

Os vídeos foram então processados pelo protótipo, ao mesmo tempo em que os dados de desempenho eram coletados. Vale destacar que a leitura do vídeo ocorria a partir da memória a uma velocidade de 5 quadros por segundo. Entretanto, a leitura das imagens geradas diretamente pela câmera era feita a uma taxa média de 11 quadros por segundo. Esta diferença de desempenho vem do fato de que a câmera possui barramento especial com a GPU, enquanto a leitura de arquivo era feita utilizando-se o barramento de dados comum à memória do *Raspberry Pi*.

5.2 Métricas de avaliação

Os resultados foram avaliados a partir do uso de determinadas métricas que revelam diferentes aspectos dos dados obtidos. A seguir, considere $PR(p, q)$ o espaço ocupado pela placa *p* selecionada manualmente no quadro *q*, e $PA(p, q)$ o espaço ocupado pela placa *p* selecionada pelo algoritmo no quadro *q*. As comparações somente são feitas entre a placa $PR(p_1, q_1)$ e $PA(p_2, q_2)$ quando $q_1 = q_2$. Caso haja duas ou mais $PA(p, q)$ ou $PR(p, q)$ com o mesmo q, a comparação será feita entre os pares de placas cujos centróides estiverem mais próximos. Quando não houver $PR(p, q)$ correspondente para $PA(p, q)$, $PR(p, q)$ será considerada um espaço de dimensões 0×0. O mesmo vale para o caso de não haver $PA(p, q)$ correspondente para $PR(p, q)$.

A primeira métrica, chamada de Placas Parcialmente Selecionadas (PPS), é calculada por

$$PPS = \sum_{q=1}^{Q} \sum_{p=1}^{tPR(q)} I(p, q) \tag{5.1}$$

com $tPR(q)$ sendo o número total de placas reais no quadro q e

$$I(p, q) = \begin{cases} 1 & PA(p, q) \bigcap PR(p, q) > PR(p, q) \times M \\ 0 & \text{caso contrário} \end{cases} \tag{5.2}$$

com M correspondendo à porcentagem mínima de $PR(p, q)$ que deve ser abrangida por $PA(p, q)$ para que a detecção seja considerada bem-sucedida. Quando $M = 1$, $PR(p, q)$ deve estar completamente abrangida por $PA(p, q)$.

A segunda métrica, chamada de Erro por Área (EA), calcula o erro para uma dada combinação de $PA(p, q_1)$ e $PR(p, q_2)$, com $q_1 = q_2$ através de

$$EA = \frac{\#((PA(p, q) - PR(p, q)) \bigcup (PR(p, q) - PA(p, q)))}{\#(PA(p, q) \bigcup PR(p, q))} \tag{5.3}$$

onde $\#A$ denota o número de elementos de um conjunto A. Este valor sempre fica no intervalo de 0 a 1, sendo que 0 significa que o algoritmo abrangeu única e exclusivamente a placa marcada manualmente. Qualquer valor maior do que 0 e menor do que 1 significa que $PA(p, q)$ e $PR(p, q)$ possuem área em comum, porém $PA(p, q)$ ou $PR(p, q)$ possuem áreas não cobertas pelo seu respectivo par. Quando o valor de EA é 1, significa que $PA(p, q)$ e $PR(p, q)$ são conjuntos disjuntos.

Outra métrica calculada é a quantidade de falsos positivos (FP) através de

$$FP = tPA - PPS \tag{5.4}$$

onde tPA é o número total de placas geradas pelo algoritmo. Já o número de falsos negativos (FN) é calculado como

$$FN = tPR - PPS \tag{5.5}$$

5.3 Resultados obtidos

Nesta seção, serão apresentados os dados obtidos com o processamento de 3 diferentes vídeos a partir do protótipo. A tabela 5.1 apresenta as características básicas de cada um deles.

Tabela 5.1: Descrição dos vídeos utilizados para validação do algoritmo.

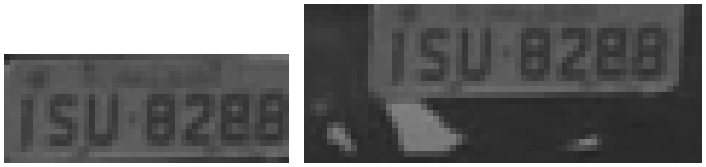
	N°de quadros	N°de placas reais
Vídeo 1	26500	2975
Vídeo 2	4481	782
Vídeo 3	3069	633

A seguir, a tabela 5.2 exibe o número de placas capturadas pelo algoritmo (nPA), juntamente com as métricas Falsos Positivos (FP), Falsos Negativos (FN) e Placas Parcialmente Selecionadas (PPS) com $M = 1$ e $M = 0.8$. A escolha por $M = 0.8$ provém de experimentos nos quais foi verificado que este valor é uma fração de intersecção que ainda permite o reconhecimento visual da placa. É também exibido o Erro Médio (EM), que corresponde à média do Erro por Área (EA) do conjunto de placas geradas pelo algoritmo.

Tabela 5.2: Resultados para cada uma das métricas propostas.

	nPA	FP	FN	PPS (M=1)	EM (M=1)	PPS (M=0.8)	EM (M=0.8)
Vídeo 1	2208	998	1754	1098	0.56	1229	0.55
Vídeo 2	295	35	522	235	0.68	239	0.66
Vídeo 3	533	262	362	260	0.62	265	0.61

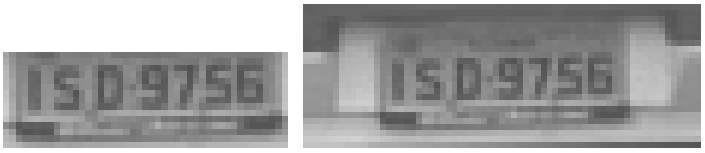
Percebe-se, pelos dados acima, que o algoritmo conseguiu, para o vídeo 1, uma eficiência de 36% para PPS sendo calculado com $M = 1$. Com $M = 0.8$, esse valor foi para 41%. Para o vídeo 2, essa porcentagem foi de 30% para $M = 1$ e $M = 0.8$, enquanto para o vídeo 3 foi de 41% para $M = 1$ e $M = 0.8$. As figuras 5.1a e 5.1b demonstram um caso em que uma placa gerada pelo algoritmo, mesmo que cobrindo menos de 80% da região definida como placa pelo processo manual, continua legível.



(a) (b)

Figura 5.1: Placa selecionada manualmente (a) e pelo algoritmo (b).

Em relação ao Erro Médio, tem-se nas figuras 5.2a e 5.2b um exemplo que ilustra a porção da imagem selecionada manualmente e sua respectiva extração pelo algoritmo, onde EA=0.6.



(a) (b)

Figura 5.2: Placa selecionada manualmente (a) e pelo algoritmo (b).

Em relação ao tempo de execução no *Raspberry Pi*, constatou-se uma taxa de 11 quadros por segundo, o que corresponde a um tempo médio de 0,09 segundos para cada quadro. Estima-se que, com este desempenho, seja possível implementar futuramente um algoritmo de OCR embarcado mantendo-se uma performance aceitável.

6 CONCLUSÃO

Este trabalho teve por objetivo a construção de um protótipo de baixo custo para detecção de placas veiculares. Através dele foi possível compreender os reais desafios da área, os quais justificam a variada gama de soluções propostas pela academia e por empresas privadas até hoje. Também foi possível provar que é possível construir um algoritmo eficiente e que funcione mesmo em um computador de poder computacional limitado, o que é o caso do *Raspberry Pi*.

Pode-se dizer também que, apesar de obtermos menos da metade das placas disponíveis considerando todos os quadros analisados, basta que poucas imagens de uma determinada placa sejam extraídas para que um reconhecedor ótico de caracteres consiga, provavelmente, identificar o registro do automóvel. Infelizmente não foi possível implementar um deles para validar os dados, pois neste caso haveria uma maior certeza sobre a qualidade das placas detectadas pelo algoritmo.

Em relação ao algoritmo de detecção de veículos através de suas sombras, foi possível comprovar sua eficiência pelo fato de serem poucos os casos onde foram indicadas placas em regiões do plano de fundo da imagem. Sem esta filtragem, certamente haveria um número muito maior de falsos positivos. Como desvantagem, porém, foi constatado que o algoritmo gera diversos resultados errôneos quando o veículo no qual o sistema está instalado encontra-se em curvas acentuadas ou parado em algum cruzamento. No primeiro caso, o meio-fio, que acaba ficando no lugar do horizonte, assemelha-se à sombra de um veículo, confundindo o algoritmo. Em relação ao segundo caso, o problema de cruzamentos consiste em veículos que cruzam a via perpendicularmente ao carro no qual o protótipo está instalado, gerando uma sombra com tom de cinza semelhante à gerada quando se observa veículos de trás, o que também confundia o sistema de detecção.

Quanto ao algoritmo de detecção de placas a partir da traseira dos veículos, foi constatado que este também apresenta uma eficiência e desempenho consideráveis, principalmente pelas operações matemáticas realizadas e que reduziram consideravelmente o tempo de processamento. Em diversas situações, o algoritmo acusou a presença de uma placa mas que, por estar longe demais para ser lida por um OCR, por exemplo, teve de ser descartada. Por outro lado, a sua sensibilidade a adesivos, para-choques de cor intermitente ou qualquer outra característica da traseira que gerasse um alto desvio padrão resultou em alguns falso positivos.

Em trabalhos futuros, pretende-se abordar, primeiramente, o refinamento dos dados enviados para um OCR, por exemplo, pois atualmente uma mesma placa de automóvel é detectada indiscriminadamente pelo algoritmo, sem que haja uma comparação com as placas geradas anteriormente. Há uma série de alternativas que podem resolver este problema, como casamento de padrões ou a exploração da coerência temporal na ocorrência de uma mesma placas em quadros subsequentes.

Outro aspecto e a ser reconsiderado é a detecção de veículos através da sombra. Esta abordagem, apesar de eficiente em termos de custo computacional, não apresenta bons resultados quando o veículo fica muito próximo ao carro onde o dispositivo está acoplado, de forma a ocultar sua sombra. Também há problemas com motocicletas, as quais não são atualmente detectadas pelo algoritmo de identificação de sombras.

Também pretende-se implementar um algoritmo de OCR embarcado, de forma que todo o processo de reconhecimento da placa possa ser feito no próprio protótipo. Desta forma, basta que este receba rotineiramente um banco de dados com placas a serem buscadas e compare com as que ele detectou. Isto o tornaria então um dispositivo com um diferencial de mobilidade considerável, visto que a maioria das outras soluções comerciais fazem uso de computadores de mesa e telas *LCD*.

REFERÊNCIAS

[1] A.M. Al-Ghaili, S. Mashohor, A. Ismail, and A.R. Ramli. A new vertical edge detection algorithm and its application. In *Computer Engineering Systems, 2008. ICCES 2008. International Conference on*, pages 204–209, 2008.

[2] F. Alegria and P.S. Girao. Vehicle plate recognition for wireless traffic control and law enforcement system. In *Industrial Technology, 2006. ICIT 2006. IEEE International Conference on*, pages 1800–1804, 2006.

[3] C.-N.E. Anagnostopoulos, I.E. Anagnostopoulos, V. Loumos, and E. Kayafas. A license plate-recognition algorithm for intelligent transportation system applications. *Intelligent Transportation Systems, IEEE Transactions on*, 7(3):377–392, 2006.

[4] C.-N.E. Anagnostopoulos, I.E. Anagnostopoulos, I.D. Psoroulas, V. Loumos, and E. Kayafas. License plate recognition from still images and video sequences: A survey. *Intelligent Transportation Systems, IEEE Transactions on*, 9(3):377–391, 2008.

[5] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, 1986.

[6] B.K. Cho, S.H. Ryu, D.R. Shin, and J.I. Jung. License plate extraction method for identification of vehicle violations at a railway level crossing. *International Journal of Automotive Technology*, 12(2):281–289, 2011.

[7] K. Deb and Kang-Hyun Jo. Hsi color based vehicle license plate detection. In *Control, Automation and Systems, 2008. ICCAS 2008. International Conference on*, pages 687–691, 2008.

[8] Shan Du, M. Ibrahim, M. Shehata, and W. Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(2):311–325, 2013.

[9] Bai Hongliang and Liu Changping. A hybrid license plate extraction method based on edge statistics and morphology. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 831–834 Vol.2, 2004.

[10] V. Kamat and S. Ganesan. An efficient implementation of the hough transform for detecting vehicle license plates using dsp's. In *Real-Time Technology and Applications Symposium, 1995. Proceedings*, pages 58–59, 1995.

[11] Wei Liu, Xuezhi Wen, Bobo Duan, Huai Yuan, and Nan Wang. Rear vehicle detection and tracking for lane change assist. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 252–257, 2007.

[12] H. Mori and N.M. Charkari. Shadow and rhythm as sign patterns of obstacle detection. In *Industrial Electronics, 1993. Conference Proceedings, ISIE'93 - Budapest., IEEE International Symposium on*, pages 271–277, 1993.

[13] M. Sarfraz, M.J. Ahmed, and S.A. Ghazi. Saudi arabian license plate recognition system. In *Geometric Modeling and Graphics, 2003. Proceedings. 2003 International Conference on*, pages 36–41, 2003.

[14] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[15] Xifan Shi, Weizhong Zhao, and Yonghang Shen. Automatic license plate recognition system based on color image processing. In *Computational Science and Its Applications–ICCSA 2005*, pages 1159–1168. Springer, 2005.

[16] Young Sung Soh, Byung Tae Chun, and Ho Sub Yoon. Design of real time vehicle identification system. In *Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on*, volume 3, pages 2147–2152 vol.3, 1994.

[17] Christos Tzomakas and Werner von Seelen. Vehicle detection in traffic scenes using shadows. In *IR-INI, INSTITUT FUR NUEROINFORMATIK, RUHR-UNIVERSITAT*. Citeseer, 1998.

[18] M.B. Van Leeuwen and F. C A Groen. Vehicle detection with a mobile camera: spotting midrange, distant, and passing cars. *Robotics Automation Magazine, IEEE*, 12(1):37–43, 2005.

[19] Yao-Quan Yang, Jie Bai, Rui-Li Tian, and Na Liu. A vehicle license plate recognition system based on fixed color collocation. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 9, pages 5394–5397 Vol. 9, 2005.