Algoritmo Haar Cascade Aplicado na Detecção das Placas de Parada Obrigatória e de Velocidade Máxima Permitida

Anne Livia da Fonseca Macedo
Faculdade de Computação
Universidade Federal do Pará – UFPA
Castanhal, PA, Brasil
annelivia16@gmail.com

Igor Ruiz Gomes
Faculdade de Computação
Universidade Federal do Pará – UFPA
Castanhal, PA, Brasil
ruiz.igor@gmail.com

ABSTRACT

Systems able to assist drivers in the safe driving of vehicles provide several advantages, such as the reduction of traffic accidents, mostly with fatalities, normally caused by human failures, whether for distractions or even problems related to lighting or climate change. Based on this, this research aims to present a computational model capable of detect stop signs and speed limit signs, so that it contributes to the development of progressively intelligent vehicles. The system was implemented in Phyton programming language, with the support of OpenCV library, and it was divided into two steps: firstly it was performed the training and classification of the objects through Haar Cascade classification method, and in the second step, in order to improve the results, colors relevant to the object were identified using the HSV color space. During the experiments, the proposed algorithm presented satisfactory results, with a hit rate of 91% for speed limit signs and 93% for stop signs. In order to refine the proposed solution, it is intended for the next steps to include traffic sign information recognition, to either describe the specified speed on the detected objects and further reduce false positives.

KEYWORDS

Haar Cascade, Detecção de Sinais de Trânsito, Visão Computacional, Veículos Inteligentes

1 INTRODUÇÃO

O aumento do tráfego de automóveis em rodovias e zonas urbanas provoca o crescimento dos acidentes no trânsito, em sua maioria com vítimas fatais, representando um grave problema de saúde pública global [1], responsável por altos prejuízos econômicos, ambientais e sociais [2]. De acordo com o Observatório Nacional de Segurança Viária (ONSV), 90% dos acidentes de trânsito são causados por falhas humanas, seja por fatores relacionados à condição física e mental do motorista ou até mesmo o desrespeito à legislação [3-4]. Acidentes como esses, são classificados pelo Ministério da Saúde, como causas externas e não intencionais que podem ser evitados apesar de toda essa dinâmica complexa e multicausal [5]. Com fundamento nas circunstâncias supracitadas, a integração de inteligência computacional em automóveis, capazes de apoiar condutores nas tomadas de decisões, são fundamentais para evitar acidentes que muitas das vezes são motivados por distrações [6], ou por condições climáticas e de luminosidade.

Os sinais de trânsito são atributos obrigatórios na organização do tráfego de veículos, com o objetivo de alertar acerca de trechos perigosos, indicar a direção do tráfego, proibir ou dar o direito de passagem, obrigar a reduzir a velocidade e diversas outras instruções que auxiliam os motoristas na condução segura de seus automóveis [7]. O reconhecimento desses sinais é um problema desafiador do mundo real de alta relevância industrial [8] e um dos campos de estudo mais importantes em sistemas de transporte inteligentes [9]. Sistemas com essa funcionalidade podem notificar os condutores da presença de determinados sinais de trânsito, alertando-os e ajudando-os a respeitar por exemplo: o limite máximo de velocidade estabelecido em determinado trecho da estrada e a indicação de parada obrigatória [7].

A detecção de objetos é um dos elementos mais importantes em diversas áreas de visão computacional, com o objetivo principal de encontrar objetos de uma determinada classe em imagens estáticas ou frames de vídeos, podendo ser efetuada através da extração de características como bordas, regiões de cores, texturas e contornos das imagens, e então aplicado algumas heurísticas para encontrar configurações ou combinações dessas características inerentes aos objetos que se deseja localizar [10].

Diante dos fatos apresentados, esta pesquisa tem por objetivo apresentar o desenvolvimento de um sistema capaz de detectar placas de velocidade máxima permitida e de parada obrigatória, através de técnicas computacionais pertencentes as áreas de visão computacional e inteligência artificial, para extração e análise dos padrões. A detecção dessas placas é feita através do método de classificação *Haar Cascade* e da identificação das colorações utilizando o espaço de cores HSV.

O restante deste artigo está organizado como segue: a sessão 2 traz alguns trabalhos relacionados ao tema da pesquisa. Na sessão 3 é sumarizada a metodologia empregada na pesquisa e

descrita as etapas seguidas para o desenvolvimento do sistema. Os experimentos e resultados são explanados na sessão 4. Por fim, na sessão 5 são apresentadas as considerações finais e os trabalhos futuros.

2 TRABALHOS CORRELATOS

Com o avanço da tecnologia e a chegada da internet das coisas e dos automóveis inteligentes, estudos que envolvem o reconhecimento e a detecção de sinais de trânsito tem ocorrido com bastante frequência, principalmente devido estes sistemas desempenharem um papel primordial na redução dos acidentes.

No trabalho de Silva et al. [11], foi proposto um modelo de classificação de sinais de trânsito com baixo custo computacional, utilizando o método *Random Forest* para classificação e o algoritmo de otimização por colmeias para obtenção das características mais adequadas. A taxa de acerto base do sistema proposto, com adição de análises de correlação e entropia, foi próximo de 87,5%.

No trabalho de Duraes, Maciel e Barros [12], foi apresentado um sistema para detecção de placas de parada obrigatória através da segmentação e detecção de segmentos de retas, com o uso de algoritmos como *Canny* e a transformada de *Hough*. Como resultado, foram obtidos 70% de acertos.

Ellahyani, El Ansari e Jaafari [13], propuseram um método para detecção e reconhecimento de sinais de trânsito em três etapas: segmentação baseado em coloração utilizando o espaço de cores HSI, para extração de regiões de interesses; detecção dos formatos circulares, retangulares e triangulares presentes nessas regiões através de um método de momentos geométricos invariantes; reconhecimento das informações através da combinação das características HOG e das características LSS para formar um novo descritor, com a aplicação do método *Random Forest* no reconhecimento dos formatos detectados. A área sob a curva (AUC) foi de 94.21% utilizando o *dataset* GTSDB, em uma taxa de processamento de 8 a 10 frames.

Foi apresentado por Wen e Jo [14], um método para classificação de sinais de trânsito por meio da conversão da imagem para o formato YUV, extração das características e classificação das placas utilizando redes neurais convolucionais (CNN). Para treinamento e teste, foram utilizadas as imagens presentes no *dataset* GTSRB. A acurácia do algoritmo foi de 99.66%.

3 METODOLOGIA

Para alcançar o objetivo principal que consiste no desenvolvimento de um processo de software capaz de detectar com precisão, placas de parada obrigatória e de velocidade máxima permitida em diferentes ambientes, foram realizados levantamentos bibliográficos relativos as técnicas computacionais pertinentes as áreas de visão computacional e inteligência artificial projetadas para a detecção, identificação e o reconhecimento de objetos em imagens digitais.

Com base nesse estudo, foi desenvolvido um programa que efetua a detecção dessas duas placas de regulamentação, através do método de aprendizagem de máquina *Haar Cascade*, baseado no algoritmo *Viola-Jones*, que foi desenvolvido primeiramente para o reconhecimento de faces em tempo real, embora possa ser treinado para a detecção de outros objetos [15]. Além disso, com o objetivo de aumentar a precisão na detecção e reduzir os possíveis falsos positivos, foi realizada a identificação da coloração vermelha pertencente a essas placas utilizando o espaço de cores HSV, sendo esta a principal contribuição deste trabalho.

O programa foi implementado através da linguagem de programação *Python*, com o apoio da biblioteca de código aberto *OpenCV* 4.1.0 [16], que agrupa técnicas otimizadas de aprendizagem de máquina e visão computacional. O treinamento do classificador foi realizado com base no trabalho de Rezaei [17].

3.1 Algoritmo Haar Cascade

Segundo Pereira [18], o método de classificação *Haar Cascade*, oferece grande liberdade de escolha do alvo de acordo com a aplicação, por possuir técnicas de treinamento inteligente e fornecer algoritmos rápidos e de baixo custo computacional para detecção em tempo real. Além do mais, é considerado um método de grande robustez, por permitir o treinamento para detecção de alvos em condições adversas como a baixa iluminação. Este classificador introduz quatro conceitos primordiais para efetuar a detecção dos objetos: as características de Haar; as imagens integrais; o algoritmo de aprendizagem supervisionado *Adaboost* e os classificadores em cascata.

3.1.1 Características de haar

As características de Haar, popularmente utilizadas para a extração de padrões e texturas, são dois ou mais retângulos unidos de coloração branca e preta, utilizadas como entrada para o classificador em cascata. Cada uma dessas características resulta em um único valor obtido através da subtração da soma dos pixels nos retângulos brancos pela soma dos pixels nos retângulos pretos [19]. A Fig. 1 apresenta alguns exemplos de características de Haar, onde as letras (a), (b), (c) e (d) são características de linha, as letras (e) e (f) são características de borda, a letra (g) é a característica de linha diagonal e a letra (h) é a característica center-surround. As outras características rotacionadas em 45 graus, são uma extensão das características de Haar básicas, proposta por Lienhart e Maydt [20], com o objetivo de aumentar a performance na detecção.

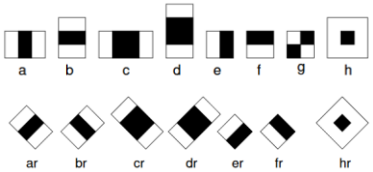

Figura 1. Características de Haar básicas e adicionais giradas em 45º.

3.1.2 Imagens integrais

Diversas características são utilizadas em um classificador, o que tornaria o processo de soma dos pixels em diferentes retângulos, algo bastante lento e não apropriado para aplicações em tempo real. Para solucionar esse problema, são utilizadas as imagens integrais, consideradas representações intermediárias das imagens originais, que permite que as características de Haar possam ser computadas em qualquer escala e localização em tempo constante.

Cada pixel na imagem integral equivale a soma de todos os pixels situados acima e a esquerda do pixel em questão na imagem original [10]. Sua fórmula matemática pode ser descrita como:

$$ii\,(x,y) = \sum_{x' \le\, x,\;\; y' \le\, y} i(x',y'), \qquad (1)$$

Onde ii(x, y) é a imagem integral e i(x, y) representa a imagem digitalizada. Podendo ainda ser computada em uma única varredura através dos pares de recorrência:

$$s(x,y) = s\,(x,y-1) + i(x,y) \qquad (2)$$

$$ii(x,y) = ii\,(x-1,y) + s(x,y) \qquad (3)$$

Sendo s(x, y) a soma acumulada de uma linha e s(x, -1) e ii(-1, y) definidos com o valor 0.

Uma vez obtida a imagem integral, as operações podem ser realizadas a partir da soma dos vértices de uma região retangular em ii(x, y), representando a soma de todos os pixels da área retangular equivalente na imagem original [21]. Viola e Jones [15] demonstraram que através da imagem integral, é possível obter a soma de todos os pixels de uma região retangular utilizando somente quatro valores. A Fig. 2 ilustra este processo, onde existem 4 áreas retangulares A, B, C e D definidas pelos pontos 1, 2, 3 e 4, sendo o ponto 1 equivalente a soma dos pixels no retângulo A, o ponto 2 é a soma dos pixels nos retângulos B e A, o ponto 3 equivale a soma de todos os pixels em A e C, e o ponto 4 é referente a soma de todos os quatro retângulos. Portanto, a operação necessária para obter a soma dos pixels no retângulo D seria: D − (B + C) + A. Desse modo, o tempo de processamento requerido para calcular as características não dependeria do seu tamanho [10].

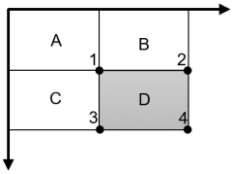

Figura 2. Ilustração de áreas retangulares em uma imagem integral.

3.1.3 Algoritmo Adaboost

Para garantir uma rápida classificação, foi proposto por Viola e Jones [15], a utilização de uma variação do algoritmo de aprendizagem supervisionado *Adaboost* que permite além de treinar o classificador, selecionar as características que melhor separam os conjuntos de imagens positivas e negativas.

O algoritmo *Adaboost* (*Adaptative Boosting*), é um método de classificação que combina vários classificadores fracos até formar um classificador forte [22]. Conforme Gonzales e Velásquez [19], esta ideia está embasada na afirmação de que vários classificadores fracos, sendo estes ligeiramente melhores que simples escolhas arbitrárias, podem combinar-se de forma que se transforme em um classificador de maior precisão, sempre que houver um número suficiente de amostras para o treinamento.

Para cada característica, o classificador determina a função de classificação com limiar ótimo, de forma que o menor número possível de exemplos seja classificado incorretamente [18], podendo o classificador fraco ser descrito matematicamente como:

$$h_j(\mathrm{x}) = \begin{cases} 1 \; se \; p_j f_j(x) < \; p_j \theta_j \\ 0 \quad caso \; contrário \end{cases}, \qquad (4)$$

Onde, para cada imagem x, normalmente de tamanho 24x24, o classificador h_j consiste de uma característica de Haar f_j, um limiar θ_j, que determinará se a imagem será classificada como positiva ou negativa, e uma paridade p_j, que indica a direção de desigualdade.

3.1.4 Classificadores em cascata

Na etapa final do algoritmo é realizada a combinação de classificadores em cascata, de modo que as regiões irrelevantes, que não possuírem características semelhantes a procurada são descartadas imediatamente, para então reservar os classificadores mais específicos para regiões mais propensas a conterem o objeto de interesse [21]. O procedimento em cascata permite obter altas taxas de detecção, onde um objeto somente será detectado com êxito, se a subjanela ao qual pertence for aprovada por todos os classificadores da cascata [22]. A Fig. 3 ilustra um processo de detecção em cascata com N estágios.

Figura 3. Cascata com N estágios para verificar se a subjanela de entrada apresenta uma placa de velocidade máxima permitida ou não.

3.2 Treinamento e implementação

Para realizar o treinamento do classificador, foram utilizadas as aplicações *objectmaker.exe, createsamples.exe, haartraining.exe e haarconv.exe*, conforme demonstrado no trabalho de Rezaei [17].

3.2.1 Amostras positivas e negativas

São necessários para o treinamento um conjunto de imagens que não contém o objeto que se deseja detectar (imagens negativas) e um conjunto de amostras positivas que possuem o objeto e capturam as mais variadas posições, escalas e iluminações possíveis.

Em conformidade com Uddin e Akhi [10], o conjunto para treinamento deve ser escolhido de forma que não confunda o algoritmo de aprendizagem, desse modo, foram utilizadas ao todo 6194 imagens, onde: 2114 possuem somente as placas de velocidade máxima permitida, 1372 contém apenas placas de parada obrigatória e 2708 não possuem nenhum dos dois objetos de interesse. As imagens positivas são oriundas da internet e dos bancos de imagens: *German Traffic Sign Recognition Benchmarck* (GTSRB) [8], *The MASTIF dataset* [23] e *BelgiumTS dataset* [24]. As Fig. 4 e 5 são algumas das imagens positivas utilizadas no treinamento do classificador.

Figura 4. Imagens positivas contendo placas de velocidade máxima permitida.

Figura 5. Imagens positivas contendo placas de parada obrigatória.

As amostras negativas são compostas de imagens arbitrárias adquiridas na internet e imagens de zonas urbanas e rodovias obtidas através dos bancos de dados: *The KITTI Vision Benchmark Suite* [25] e *Karlsruhe Dataset: Labeled Objects (Cars + Pedestrians)* [26], de maneira que o background estivesse o mais próximo possível dos locais onde as placas poderiam ser encontradas.

3.2.2 Processo de treinamento do classificador

O *Haar Cascade* é um algoritmo de aprendizagem supervisionado, sendo por este motivo, necessário informar ao computador quais imagens são positivas e negativas [10]. Para o conjunto de imagens negativas, basta criar um arquivo de texto que contenha a localização de cada uma das imagens. Quanto aos exemplos positivos, é necessário informar em um arquivo de texto, a localização de cada uma das imagens, a quantidade de objetos presentes nessas imagens e para cada objeto, informar as coordenadas do canto superior esquerdo, a largura e a altura em pixels.

Os *datasets* utilizados fornecem anotações descrevendo as coordenadas, largura e altura de cada um dos objetos, porém, para extrair essas informações das imagens adquiridas na internet, foi utilizada a aplicação *objectmaker.exe*, que permite desenhar um retângulo ao redor de cada objeto presente na imagem, criando automaticamente após o processo, um arquivo de texto contendo todas as informações requeridas.

Após gerar os arquivos de texto, é necessário criar um arquivo de vetores contendo as coordenadas de cada um dos objetos em formato binário e redimensionada para um determinado tamanho [10], neste caso foi especificado 24x24. Este arquivo foi obtido através da aplicação *createsamples.exe*. Após a preparação dos dados, foi efetuado o treinamento do classificador utilizando a aplicação *haartraining.exe*, onde foi estabelecida a quantidade máxima de 15 estágios e, finalizado o treinamento, foi gerado para cada um dos estágios, um arquivo de texto, que em seguida foram combinados em um único arquivo em formato XML através da aplicação *haarconv.exe*.

O processo de preparo dos dados e treinamento do classificador foram efetuados duas vezes para a obtenção de dois arquivos distintos, um para a detecção das placas de velocidade máxima permitida e o outro para a detecção das placas de parada obrigatória.

3.2.3 Detecção com a classe CascadeClassifier

Através da classe *CascadeClassifier* pertencente a biblioteca *OpenCV*, foi possível detectar as placas utilizando os arquivos XML obtidos no período de treinamento. Após instanciar a classe, foi utilizado o método *detectMultiscale*, que implementa a classificação em cascata percorrendo a imagem de entrada em vária escalas e retorna uma lista de retângulos, com suas respectivas coordenadas, indicando os locais onde os objetos foram encontrados com sucesso [27].

Antes da definição dos valores para os parâmetros do método, foi realizada a conversão da imagem de entrada para escala de cinza, e então aplicada a função *equalizeHist*, que efetua a equalização do histograma através da distribuição dos valores de brilho. Após o pré-processamento das imagens, foram especificados como argumentos, tanto para a detecção das placas de parada obrigatória, quanto para as de velocidade máxima permitida, o valor 1.03 para o fator de escala (*scaleFactor*), que determina o valor de redimensionamento de uma imagem em cada uma das escalas, e o valor 5 para a quantidade mínima de vizinhos (*minNeighbors*), que objetiva evitar falsas detecções, fazendo com que neste caso, somente os objetos que tiverem pelo menos 5 detecções sobrepostas, é que serão indicados como placa [27].

3.2.4 Identificação da coloração vermelha

Concluído o processo de detecção das placas com o método *detectMultiscale*, foi constatado após a execução do programa, que muitos dos falsos positivos não englobavam a coloração vermelha, que é uma das características das placas de sinalização vertical de regulamentação. Portanto, objetivando reduzir ao máximo a detecção de falsos positivos, foi analisado se os objetos identificados durante a execução do algoritmo abrangem uma porcentagem suficiente da coloração vermelha.

Para efetivar essa identificação, foram selecionadas regiões de interesse (ROI) através das coordenadas dos retângulos retornados pela função de classificação, contendo apenas os objetos detectados. Logo após a obtenção das ROIs, com o propósito de facilitar a seleção das colorações e reduzir a influência da luminosidade, foi efetivada a conversão das imagens do formato RGB para o HSV (*hue*, *saturation*, *value*), devido este espaço de cores fornecer uma separação natural das informações cromáticas e de intensidade [28].

Para concluir o processo de identificação das colorações, foi utilizado o método *InRange* pertencente a biblioteca *OpenCV*, que retorna uma imagem binária onde os pixels de coloração branca, representam as cores especificadas por um determinado limite superior e inferior [27]. Dois limites superiores e inferiores foram definidos para identificar a coloração vermelha, gerando duas imagens binárias que em seguida foram unificadas em uma só imagem através do operador lógico OR. No que diz respeito as placas de parada obrigatória, foi também verificado os pixels que não possuíam tonalidade entre os limites especificados para a identificação da coloração vermelha. A Tab. 1 exibe os limites definidos para a representação dessas colorações.

	Vermelho 1	Vermelho 2	Outras Cores
H	0 até 10	160 até 180	11 até 159
S	90 até 255	80 até 255	0 até 255
V	50 até 255	50 até 255	0 até 255

Tabela 1. limites superiores e inferiores para cada cor.

Efetuada a identificação das colorações, foi aplicada sobre as imagens binárias, as operações morfológicas de erosão e dilatação, que são popularmente utilizadas no pré e pós processamento de imagens, pela possibilidade de remover ruídos, ocasionar o afinamento e o espessamento dos objetos, o que permite unificar os componentes de uma grande região, que poderiam estar separados em múltiplas partes, devido ruídos, sombras ou outros efeitos similares [27].

Logo após o pós-processamento, foi verificada a quantidade de pixels brancos em cada uma das imagens binárias através da função *CountNonZero* e então determinado experimentalmente, após diversas análises sobre diferentes imagens que, quando se trata das placas de velocidade máxima permitida, se a quantidade de pixels brancos retornados após a identificação da cor vermelha, for maior ou igual a 100, de modo que placas em diferentes distâncias fossem agregadas, é provável que o objeto detectado seja de fato uma placa de velocidade máxima.

Quanto as placas de parada obrigatória, se a quantidade de pixels brancos retornados após a identificação da coloração vermelha for maior ou igual a 300, para também compor placas em diferentes distâncias, e a quantidade de pixels de tonalidade vermelha for maior ou igual ao número de pixels que não estão entre os limites especificados para a coloração, é porque existe a possibilidade de que o objeto em questão possa realmente ser uma placa de parada obrigatória, já que sua cor predominante neste caso seria o vermelho. Entretanto, existem casos que mesmo que o objeto detectado seja uma placa, a quantidade de pixels vermelhos pode ser inferior aos das outras colorações, devido problemas referentes a sombras ou a resolução da imagem por exemplo, todavia com diferença mínima. Logo, para contornar problemas como esse e evitar falsos negativos, foi definido que, para que um objeto possa ser detectado como placa de parada obrigatória, caso não possua a cor vermelha como predominante, deve apresentar diferença entre as duas colorações presentes na imagem, menor ou igual a 1000 pixels.

4 EXPERIMENTOS E RESULTADOS

Após a implementação do programa, foram efetuados os experimentos utilizando ao todo 298 imagens oriundas da internet, com placas em diferentes distâncias, ambientes e iluminações possíveis, buscando atender ao máximo os diferentes cenários que elas poderiam ser encontradas. Dentre essas imagens, existem 162 com uma ou mais placas de velocidade máxima permitida, sendo ao todo 206 placas, e 136 imagens com pelo menos uma placa de parada obrigatória, alcançando um total de 147 placas.

Os parâmetros do método *detectMultiscale* são primordiais para o classificador, afetando diretamente a qualidade da detecção. Em geral, quanto maior for o valor para a quantidade mínima de vizinhos, maior é a acurácia do algoritmo, no entanto, menos objetos são detectados. Durante a execução do programa, constatou-se que os valores especificados para o fator de escala e a quantidade mínima de vizinhos apresentaram resultados satisfatórios na detecção, embora ainda apareça alguns falsos positivos e negativos. Dentre os falsos negativos, se encontram algumas placas em condições noturnas, em razão da baixa iluminação, das quais as de parada obrigatória foram as mais afetadas por este problema. As Fig. 6 e 7 apresentam os resultados obtidos na detecção das placas de velocidade máxima permitida e de parada obrigatória respectivamente, com processo de identificação das colorações incluído.

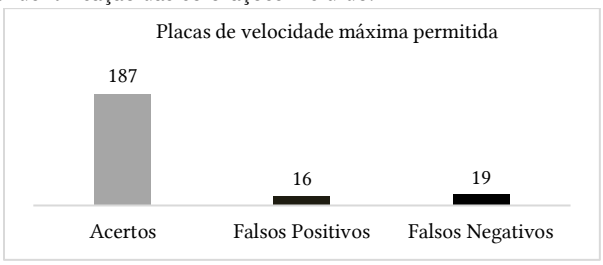

Figura 6. Acertos e erros na detecção das placas de velocidade máxima permitida.

Figura 7. Acertos e erros na detecção das placas de parada obrigatória.

A combinação do algoritmo de classificação com a identificação das colorações proporcionou a diminuição do número de falsos positivos, uma vez que a grande maioria dessas detecções não possuíam a coloração vermelha em quantidade suficiente. A Fig. 8 é um exemplo de imagens com falsas detecções que foram solucionadas com o processo de identificação da coloração.

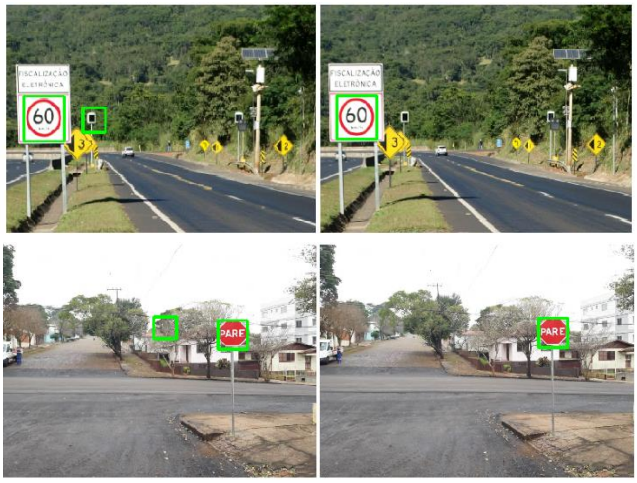

Figura 8. Antes da identificação das colorações à esquerda e após a identificação das colorações à direita.

As Tab. 2 e 3, exibem uma comparação entre a quantidade de verdadeiros positivos (VP), falsos positivos (FP) e falsos negativos (FN), obtidos durante a execução do programa na detecção das placas de velocidade máxima permitida e de parada obrigatória respectivamente, antes e após o processo de identificação das colorações.

Identificação das colorações	VP	FP	FN
Detecções antes	190	52	16
Detecções após	187	16	19

Tabela 2. Comparação dos resultados antes e após a identificação das colorações para as placas de velocidade máxima permitida.

Identificação das colorações	VP	FP	FN
Detecções antes	140	164	7
Detecções após	137	13	10

Tabela 3. Comparação dos resultados antes e após a identificação das colorações para as placas de parada obrigatória.

Conforme apresentado nas Tab. 2 e 3, o processo de identificação das colorações foi fundamental para que o sistema obtivesse um melhor desempenho. O número de falsos positivos foi reduzido de maneira significativa, principalmente para as placas de parada obrigatória, que obteve um decréscimo percentual de 92%. Entretanto, a quantidade de acertos diminuiu moderadamente, onde três placas de velocidade máxima permitida e três placas de parada obrigatória que antes eram detectadas passaram a não ser, devido falha no reconhecimento da cor vermelha. Apesar disso, o sistema continuou apresentando resultados satisfatórios, com percentual de 91% de acertos para placas de velocidade máxima permitida e 93% para placas de parada obrigatória. As Fig. 9 e 10 são exemplos das detecções de algumas placas, bem como o resultado da identificação da coloração vermelha pertencente a cada uma delas.

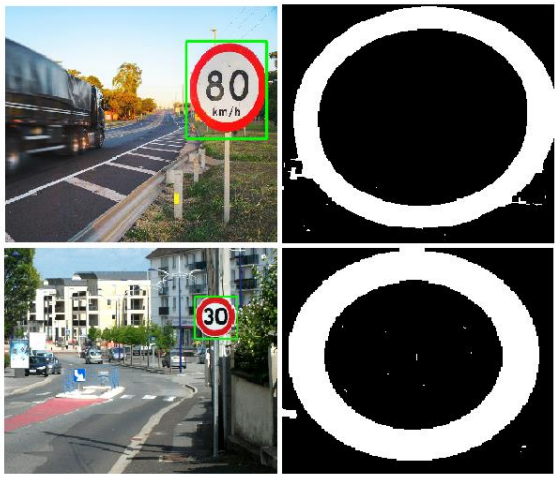

Figura 9. Detecção de placas de velocidade máxima permitida e identificação da coloração vermelha.

Figura 10. Detecção de placas de parada obrigatória e identificação da coloração vermelha.

5 CONCLUSÃO

Diversos estudos que propõem a classificação de sinais de trânsito vêm sendo desenvolvidos utilizando diferentes abordagens, devido à grande relevância industrial que sistemas que fornecem essas funcionalidades desempenham na atualidade. Dessa forma, objetivando contribuir com trabalhos na área, esta pesquisa descreveu o desenvolvimento de um sistema capaz de detectar placas de velocidade máxima permitida e de parada obrigatória, através de um algoritmo de classificação em cascata baseado em características Haar e da identificação das colorações utilizando o espaço de cores HSV. O modelo proposto apresentou resultados satisfatórios, principalmente levando em consideração

que as placas nas imagens de teste, estavam em diferentes distâncias, ambientes e iluminações, apresentando percentual de acertos equivalente a 91% para placas de velocidade máxima permitida e 93% para placas de parada obrigatória. No entanto, durante os experimentos constatou-se que algumas placas de parada obrigatória no período noturno não foram detectadas com facilidade pelo classificador, devido problemas relacionados com as condições de iluminação. Por este motivo, faz-se necessário aprimorar o método proposto para satisfazer essa condição. Como trabalhos futuros pretende-se ainda, incluir o reconhecimento das informações presentes nas placas para possibilitar o reconhecimento das velocidades, bem como minimizar a quantidade de falsos positivos.

AGRADECIMENTOS - ACKNOWLEDGMENTS

Os autores agradecem à Faculdade de Computação da Universidade Federal do Pará – Campus Castanhal pelo incentivo à produção científica.

REFERÊNCIAS

[1] Débora Regina de O. M. Abreu, Eniuce M. de Souza, and Thais Aidar de F. Mathias. Impacto do Código de Trânsito Brasileiro e da Lei Seca na Mortalidade por Acidentes no Trânsito. *Cadernos de Saúde Pública*, 34(8), 2018. doi: 10.1590/0102-311X00122117. URL https://doi.org/10.1590/0102-311X00122117.

[2] Victor Levi R. Rodrigues, Josevan C. Santos, and Felipe S. Nery. Mortes de Motociclistas em Sergipe: tendência temporal de 2000 a 2016. *Congresso Nacional de Enfermagem – CONENF*, 1(1), 2018.

[3] Daniela Cristina D. Teixeira. O comportamento do brasileiro no trânsito e seu impacto para um trânsito menos seguro no Brasil. Master's thesis, FGV Escola de Administração de Empresas de São Paulo - EAESP, São Paulo, Brasil, 2016.

[4] Observatório Nacional de Segurança Viária (ONSV). 90% dos acidentes são causados por falhas humanas, alerta o observatório, 2015. URL http://www.onsv.org.br/90-dos-acidentes-sao-causados-por-falhas-humanas-alerta-observatorio.

[5] Ministério da Saúde. Painel de indicadores do SUS nº 5, 2008. URL http://bvsms.saude.gov.br/bvs/publicacoes/painel_indicadores_sus_n5_p1.pdf.

[6] Caroline Bianca S. T. Molina. Controle veicular autônomo (CVA): um sistema para prevenir acidentes no contexto de veículos autônomos. Master's thesis, Universidade de São Paulo, São Paulo, Brasil, 2018.

[7] Natalia Kryvinska, Aneta Poniszewska-Maranda, and Michal Gregus. An Approach towards Service System Building for Road Traffic Signs Detection and Recognition. *Procedia Computer Science*, 141(1):64-71, 2018. doi: 10.1016/j.procs.2018.10.150. URL https://doi.org/10.1016/j.procs.2018.10.150.

[8] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *Proceedings of the IEEE International Joint Conference on Neural Networks – IJCNN*, pages 1453-1460, San Jose, CA, USA, 2011. IEEE. doi: 10.1109/IJCNN.2011.6033395. URL https://doi.org/10.1109/IJCNN.2011.6033395.

[9] Hasan Fleyeh. Traffic sign recognition without color information. In *2015 Colour and Visual Computing Symposium – (CVCS)*, pages 1–6, Gjovik, Norway, 2015. IEEE. doi: 10.1109/CVCS.2015.7274886. URL https://doi.org/10.1109/CVCS.2015.7274886.

[10] Mohammad Salah Uddin and Afroza Yesmin Akhi. Horse detection using haar like features. *International Journal of Computer Theory and Engineering (IJCTE)*, 8(5):415-418, October 2016. doi: 10.7763/IJCTE.2016.V8.1081. URL https://doi.org/10.7763/IJCTE.2016.V8.1081.

[11] José Cleyton Silva, Felipe C. Farias, Victor C. F. Lima, Victor L. B. Silva, Letícia M. Seijas, and Carmelo J. A. Bastos Filho. Classificação de Sinais de Trânsito Usando Otimização por Colmeias e Random Forest. In Anais do *XII Congresso Brasileiro de Inteligência Computacional*, pages 1-6, Curitiba, PR, Brasil, 2015. ABRICOM. doi: 10.21528/CBIC2015-166. URL https://doi.org/10.21528/CBIC2015-166.

[12] Thiago de Jesus O. Duraes, Lucas S. Maciel, and Wagner F. de Barros. Processamento Digital de Imagens Aplicado a Identificação Automática de Placas de Trânsito. *REVISTA CEREUS*, 10(2):240-251, 2018. doi: 10.18605/2175-7275/cereus.v10n2p240-251. URL https://doi.org/10.18605/2175-7275/cereus.v10n2p240-251.

[13] Ayoub Ellahyani, Mohamed El Ansari, and Ilyas El Jaafari. Traffic sign detection and recognition based on random forests. In *Applied Soft Computing*, 46(1):805-815, September 2016. doi: 10.1016/j.asoc.2015.12.041. URL https://doi.org/10.1016/j.asoc.2015.12.041.

[14] Lihua Wen and Kang-Hyun Jo. Traffic sign recognition and classification with modified residual networks. In *2017 IEEE/SICE International Symposium on System Integration (SII)*, pages 835-840, Taipei, Taiwan, 2017. IEEE. doi: 10.1109/SII.2017.8279326. URL https://doi.org/10.1109/SII.2017.8279326.

[15] Paul Viola and Michael Jones. Rapid Object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition – CVPR*, pages I-I, Kauai, HI, USA, 2001. IEEE. doi: 10.1109/CVPR.2001.990517. URL https://doi.org/10.1109/CVPR.2001.990517.

[16] OpenCV 4.1.0. URL https://opencv.org/releases/.

[17] Mahdi Rezaei. Creating a cascade of haar like classifiers Step by step, 2014. URL https://www.cs.auckland.ac.nz/~m.rezaei/Tutorials/Creating_a_Cascade_of_Haar-Like_Classifiers_Step_by_Step.pdf.

[18] Rafael C. Pereira. Técnica de rastreamento e perseguição de alvo utilizando o algoritmo Haar Cascade aplicada a robôs terrestres com restrições de movimento. Master's thesis, Universidade Federal do Rio Grande do Norte, Natal, Brasil, 2017.

[19] Hirvin Gonzalez and Sergio Velásquez. Reconocimiento facial utilizando Viola-Jones y patrones binários. *Universidad Ciencia y Tecnología*, 23(92):57-63, June 2019.

[20] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings International Conference on Image Processing*, pages I-I, Rochester, NY, USA, 2002. IEEE. doi: 10.1109/ICIP.2002.1038171. URL https://doi.org/10.1109/ICIP.2002.1038171.

[21] Alexandre Zaghetto, Cauê Zaghetto, Luiz H. M. Aguiar, Mateus Mendelson, and Flávio de B. Vidal. Facial Biometrics Using SIFT/SURF in Comparison to Eigenfaces. *XIII Workshop de Visão Computacional*, Natal, RN, Brasil, 2017.

[22] Luciana Maiara Q. de Santana, Fábio G. Rocha, and Thiago S. R. Santos. Processo de detecção Facial utilizando Viola; Jones. *Interfaces Científicas-Exatas e Tecnológicas*, 1(1):35-40, February 2015. doi: 10.17564/2359-4942.2015v1n1p35-40. URL http://dx.doi.org/10.17564/2359-4942.2015v1n1p35-40.

[23] Siniša Šegvic, Karla Brkic, Zoran Kalafatic, Vladimir Stanisavljevic, Marko Ševrovic, Damir Budimir, and Ivan Dadic. A computer vision assisted geoinformation inventory for traffic infrastructure. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 66-73, Funchal, Portugal, 2010. IEEE. doi: 10.1109/ITSC.2010.5624979. URL http://doi.org/10.1109/ITSC.2010.5624979.

[24] Radu Timofte, Karel Zimmermann, and Luc Van Gool. Multi-view traffic sign detection, recognition, and 3D localisation. *Machine Vision and Applications*, 633-647, 2011. doi: 10.1007/s00138-011-0391-3. URL https://doi.org/10.1007/s00138-011-0391-3.

[25] Jannik Fritsch, Tobias Kühnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *16th International IEEE Conference on Intelligent Transportation Systems – ITSC 2013*, pages 1693-1700, The Hague, Netherlands, 2013. IEEE. doi: 10.1109/ITSC.2013.6728473. URL http://doi.org/10.1109/ITSC.2013.6728473.

[26] Andreas Geiger, Christian Wojek, and Raquel Urtasun. Joint 3d estimation of objects and scene layout. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 1467-1475, Red Hook, NY, USA, 2011. Curran Associates Inc. doi: 10.5555/2986459.2986623. URL https://dl.acm.org/doi/10.5555/2986459.2986623.

[27] Adrian Kaehler and Gary Bradski. Learning OpenCV 3: Computer vision in C++ with the OpenCV library. "O'Reilly Media, Inc.", 2016.

[28] Yangyang Hu, Bingshu Wang, Ying Wang, and Yong Zhao. Moving Shadows Removal using HSV Color Space and Texture Analysis. *Advanced Science and Technology Letters*, 122(1):240-246, 2016.