Detecção de Sinais de Trânsito Através do Método de Classificação *Adaboost*

Traffic Sign Detection by Adaboost Classification Method

Willian Augusto Dias dos Reis[a]*

[a]Universidade Norte do Paraná, Curso de Engenharia da Computação, PR, Brasil

*E-mail: willianaugusto@gmail.com

Resumo

A Secretaria de Estado dos Transportes Metropolitanos estima que, por anos, as perdas financeiras com poluição, acidentes de trânsito e engarrafamentos em São Paulo, sejam de 4,1 bilhões de reais por ano. Nesta perspectiva, vê-se que a situação caótica do trânsito é conhecida. Neste artigo, se focou em arrumar alternativas para solucionar, ou diminuir a frequência dos acidentes de trânsito. Normalmente, a maior causa de acidentes não são os problemas mecânicos, ou embriaguez, ou falta de sinalização, mas falta de atenção nos sinais de trânsito existentes em estradas e rodovias do nosso país. Partindo desta premissa, a melhora ou aumento da sinalização de trânsito não evitaria os acidentes, por ser o homem a causa raiz deste mal. Por isso, é necessário que foquemos em chamar atenção do homem quando houver sinalização. Para isso, se realizará treinamentos de cascatas de classificadores que, durante a captura de vídeo em *run-time*, identificarão regiões de interesse que poderão ser placas, e as reconhecerão com o intuito de serem exibidas para o motorista no interior do seu próprio veículo. O classificador *Adaboost* será o responsável por identificar regiões de interesse que contenham placas. Este classificador se baseia em procurar padrões e não uma busca pixel a pixel.

Palavras-chave: Adaboost. Árvores. Classificação. Inteligência Artificial. Classificadores.

Abstract

The State Secretary of Metropolitan Transportation estimates that, the financial losses with pollution, traffic accidents and traffic jams in Sao Paulo is 4.1 billion dollars per year. In this perspective, we see that the chaotic traffic situation is known. This paper focused on trying to solve the problem by decreasing its frequency. Normally, the major cause of accidents is not the mechanical problems, or drunkenness, or lack of signage, but lack of attention to road signs on existing roads and highways of our country. Thus, the signaling improvement would not prevent accidents since man is responsible for that. Thus, it is important to draw man's attention. To do this, cascades training of classifiers were performed, that during video capture run-time, identified the areas of interest and recognized them in order to be displayed to the driver inside of his vehicle. The Adaboost classifier was responsible for identifying regions of interest with traffic plates. This classifier was based on search patterns rather than pixel by pixel searching.

Keywords: *Adaboost. Trees. Classification. Classifiers. Artificial Intelligence.*

1 Introdução

O trânsito, assim como toda instituição universal, sobretudo as que regem a coletividade, necessita de regras e leis para prezar pela ordem e o respeito fraterno (PARISE, 2008). Desde a mais alta até a mais baixa organização faz com que os seus envolvidos se deparem com tais leis e, sobretudo, com que as cumpram.

Para o trânsito não é diferente. Existem diversas leis, regras e boas práticas que regem sua conduta e sua moral. Por ter muitas leis e regras, surge a necessidade de explorar os sentidos do homem para que ele possa melhor interagir e, sobretudo, se compenetrar no que é necessário para que a ordem do trânsito flua no âmbito das ruas e estradas.

Os sinais de trânsito fazem com que os homens e mulheres envolvidos, independente de nacionalidade e cor, respeitem o trânsito. Porém, mesmo tendo esta abrangência, o trânsito ainda não está bom.

As imprudências são muitas, e, sobretudo, o desrespeito à sinalização de trânsito. A imprudência aumentou e a violência extrapolou os limites suportados pela sociedade brasileira. O trânsito é prova disso. Um reconhecimento automático de placas de trânsito traria um conforto para os motoristas, pois o painel de seu veículo sinalizaria, constantemente, as imprudências e erros, para evitar fins trágicos.

Neste trabalho utilizaremos técnicas (LIENHART; KURANOV; PISAREVSKY, 2002; VIOLA; JONES, 2004) que treinarão cascata de classificadores para detecção de objetos, neste caso sinais de trânsito.

Este trabalho teve como objetivo apresentar o uso do algoritmo *adaboost* para detecção de sinais de trânsito, de forma a proporcionar estudos na linha de pesquisa de sistemas de controle autônomo de veículos e possibilitar a criação de um produto embarcado, que proporcione aos seus usuários mais segurança ao dirigirem seus veículos automotores. Por ser uma prova de conceito para demostrar a utilização do algoritmo *adaboost,* os classificadores serão treinados para detectar um somente um objeto: a placa de trânsito PARE.

2 Desenvolvimento

2.1Fundamentação teórica

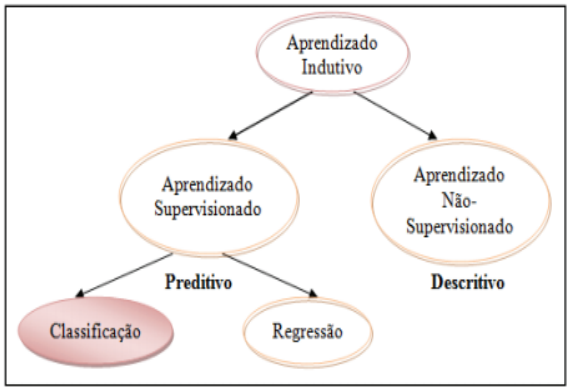
O aprendizado de máquina (do inglês, *Machine Learning*) é um subcampo da inteligência artificial que trata de problemas de aprendizado computacional, desenvolvendo técnicas computacionais, com o intuito de assimilar conhecimento de forma automática (BISHOP, 2004). A função de um sistema de aprendizado é analisar dados e informações e generalizá-las, para a extração de outros novos conhecimentos. Para isso, usa-se um programa de computador para automatizar o processo de aprendizado (MONARD; BARANAUSKAS, 2003).

Inferência lógica ou o princípio da indução é utilizado no aprendizado, com o objetivo de obter genéricas conclusões a partir de um dado conjunto de exemplos. Quando se efetua inferência indutiva sobre o dado conjunto de exemplos, é aprendido um conceito. As hipóteses e resultados gerados por esta inferência podem ser ou não uma verdade.

A boa definição das classes de exemplo fará com que a indução gere um conhecimento novo representativo, obtendo, deste modo, boas hipóteses para um determinado tipo de problema. Por isso, quanto mais exemplos selecionados para o treinamento dos indutos, melhor será treinado o novo conjunto de dados. Um algoritmo de indução, por sua vez, tem como objetivo construir um classificador que tenha poder de determinar a classe que um exemplo não rotulado pertence. Torna-se possível rotular um novo exemplo por causa da generalização.

O aprendizado indutivo pode ser dividido em: supervisionado (utilizado para classificação dos exemplos em classes predefinidas: resolve problemas preditivos) e não-supervisionado (utilizado para agrupamento, agrupando exemplos semelhantes: resolve problemas descritivos). Desta forma, classificação e agrupamento são, respectivamente, exemplos destes dois tipos de aprendizado.

Figura 1: Hierarquia do aprendizado. Adaptado por *Monard* e *Baranauskas*.



Fonte: Monard e Baranauskas (2003).

Método de Comitê é a sub área do aprendizado de máquina, que constrói um grupo de classificadores, denominados classificadores-base, com o objetivo de ser o mais preciso que o melhor dos elementos deste comitê ou grupo (POLIKAR, 2006). Para treinar um classificador-base, pode-se usar o mesmo algoritmo-base, ou ainda, utilizar algoritmos-base diferentes, iterativamente. Uma forma mais simples baseada neste algoritmo é o Voto da Maioria Simples. Nele, a partir de uma estratégia de votos, diversos classificadores são combinados, tendo como resultado final, a resposta que receber mais votos. Esta resposta é considerada a resposta do comitê.

Bagging, ou *Bootstrap Aggregating* é um método de comitê. Nele, dado um conjunto de treinamento A de tamanho t, o *Bagging* gera b novos conjuntos de treinamentos de tamanho t' <= t (BREIMAN, 1996). Os novos conjuntos criados são combinados com o intuito de definir uma resposta final, podendo ser por meio de uma média, quando temos casos de regressão, ou votos, em casos de problemas de classificação. O método *Bagging* é bem simples e tem um desempenho muito melhor quando é combinado com árvores de decisão, diminuindo a possibilidade de sobre-ajuste (*over-fitting*).

Boosting é um meta algoritmo de aprendizado de máquina que combina um conjunto de classificadores-base "fracos", ou com um baixo desempenho, criando um classificador "forte", ou com um alto desempenho (FREUND, 1990; FREUND; SCHEPIRE, 1990). Embora o aprendizado *Boosting* e *Boosting* pela Maioria tenham sido os primeiros algoritmos que utilizaram tal paradigma, não obtiveram muito sucesso, pois não eram adaptáveis aos classificadores-base utilizados (DRUCKER; SCHAPIRE; SIMARD, 1993; FREUND, 1990; FREUND; SCHEPIRE, 1990).

AdaBoost ou *Adaptive Boosting* é um dos algoritmos mais famoso e utilizado dos que são baseados em *Boosting* (FREUND; SCHAPIRE, 1995). Tal fama deve-se a sua capacidade de conseguir adaptar-se aos classificadores-base. Neste algoritmo, classificadores são gerados de forma a favorecer os exemplos erroneamente classificados pelos classificadores anteriores.

2.2 Características Haar

O uso de características, previamente extraídas, de um objeto para sua detecção, é no que se baseiam os algoritmos de aprendizado de máquina. Estas características são responsáveis para distinguir um objeto do outro, pois as características dos objetos são diferentes uma das outras.

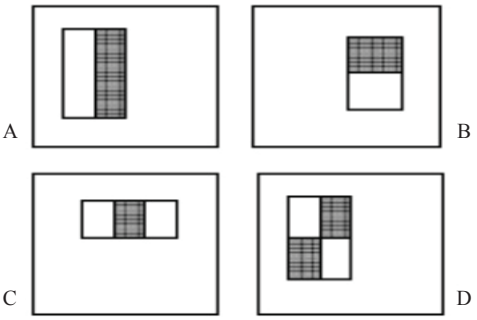
O que motiva o uso de características ao invés de do uso tradicional de pixels, é que a velocidade de análise de um objeto por características é bem melhor que por pixels, uma vez que a quantidade de características, naturalmente, tende a ser menor que a quantidade de pixels de uma imagem.

Com isto em mente, Viola e Jones (2004) propõem a utilização de três formatos de feições (características):

1. Dois Retângulos: Onde a diferença entre a soma dos pixels dentro de duas regiões adjacentes de mesmo tamanho define o seu valor.

2. Três retângulos: Onde a soma de um retângulo central menos a soma de dois retângulos externos define seu valor.
3. Quatro retângulos: Onde a diferença entre os pares diagonais de retângulos define seu valor.

Figura 2: Possíveis formas de características *Haar*.

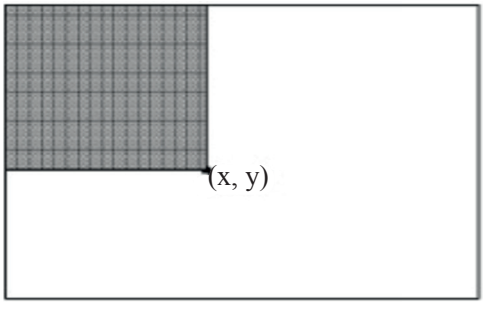


Fonte: Viola e Jones (2004).

Estas características retangulares, conhecidas como "características *Haar*", definidas por *Papageorgio* e *Poggio* em trabalho no que se propõe o uso de *Haar Wavelets* ajudou a definir a proposta do uso de uma representação intermediária da imagem, chamada imagem integral (CONSTANTINE; MICHAEL; POGGIO, 1998; MALLAT, 1989). Esta forma de representação nova aperfeiçoaria os cálculos, sendo que cada ponto x, y da imagem tem o somatório dos pixels da origem da imagem até a sua localização. Uma das características de aperfeiçoamento desta proposta é que em uma única passada pode calcular as feições (características), e acordo com a equação:

$$ii(x, y) = \sum_{x' \leq x,\, y' \leq y} i(x', y'), \qquad (2.1.1)$$

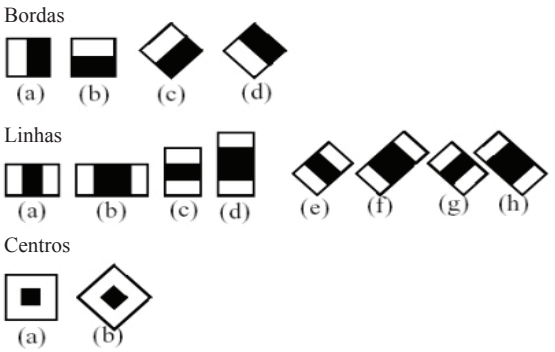
Figura 3: Exemplo de área retangular a ser calculada na imagem integral.



Fonte: Viola e Jones (2004).

Para melhorar o processo de detecção de objetos, a quantidade de características retangulares foi estendida, agora, além das usadas por Viola e Jones (2004), que estariam giradas em 45º (LIENHART; KURANOV; PISAREVSKY, 2002). Estas novas características proporcionaram em torno de 10% de redução de falsos positivos, aumentando a eficiência do processo de detecção.

Figura 4: Características *Haar*



Fonte: Lienhart, Kuranov e Pisarevsky (2002).

Uma função de classificação tem como objetivo principal detectar um determinado objeto a partir de suas características previamente extraídas. Para descobrir estas características, podem-se utilizar diversos meios, dentre os quais se encontram as Redes Neurais e *Support* Vector Machine (MEYER; LEISCH; HORNIK, 2003). Viola e Jones (2004) escolheram uma variação o algoritmo *Adaboost* para esta tarefa (FREUND; SCHAPIRE, 1995).

Quando se trata da qualidade de um classificador, naturalmente, quanto melhor e maior o conjunto de treinamento, maior será sua precisão na detecção do objeto desejado. Sendo assim, no exemplo da figura abaixo, o classificador percorre as imagens, identificando características semelhantes entre elas e definindo as características do objeto em questão. Neste exemplo, uma face humana. Na segunda linha da figura, uma face foi adicionada no plano de fundo das características identificadas e vê-se uma das características é que a região dos olhos é geralmente mais escura que a região das bochechas. E ainda vê-se que há diferenças nas intensidades das regiões dos olhos e do nariz.

Figura 5: Características retangulares mais marcadas de uma face.

Fonte: Viola e Jones (2004).

Como já foi dito, o algoritmo de aprendizado necessita de um conjunto de exemplos para conseguir extrair as características do objeto, neste caso chamamos este conjunto de imagens positivas, ou seja, imagens que contêm o objeto. E ainda um segundo conjunto chamado de imagens negativas, ou seja, que não contenham o objeto de interesse para extrair características opostas ao mesmo. O algoritmo *Adaboost* é

utilizado para selecionar as características e ainda treinar o classificador, fazendo uso dos conjuntos positivo e negativo.

Segue o pseudocódigo do algoritmo de treinamento dos classificadores:

1. Dados os exemplos de imagens $(x_1, Y_1),...,(x_n, Y_n)$ onde $Y_i = 0,1$ para imagens negativas e positivas respectivamente;
2. Inicializar só pesos $w_{ti} = \frac{1}{2_m}, \frac{1}{2_l}$ para $Y_0 = 0,1$, respectivamente onde m e l são o número de imagens negativas e positivas;
3. De t =1 até T repetir:
   a. Normalizar os pesos, $w_{t,i} \leftarrow w_{t,i} / \sum_{j=1}^{n} w_{t,j}$ para w_t seja uma distribuição de probabilidade.
   b. Para cada característica, j, treinar classificador h_j que seja restrito a utilizar apenas uma característica. O erro é calculado respeitando-se $w_t, \varepsilon_j = \sum_i w_i | h_j(x_i) - Y_i|$
   c. Escolher o classificador h_t com o menor erro ε_t;
   d. Atualizar os pesos: $w_{t+1,i} = w_{t,i}\beta_t^{1-\varepsilon_i}$ onde $\varepsilon_i = 1$ caso contrário, é $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
4. Ao término do algoritmo o classificador forte é: $h(x) = \begin{cases} 1 \\ 0 \end{cases} \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t$ onde $\alpha_t = \log \frac{1}{\beta_t}$.

Viola e Jones (2004) conseguiram resultados extraordinários ao usar o algoritmo apresentado. O classificador testado foi criado com 200 características retangulares e uma taxa de detecção de 95%. Nestas condições, o número de falsos positivos em um total de 14.804 imagens testadas foi de apenas um.

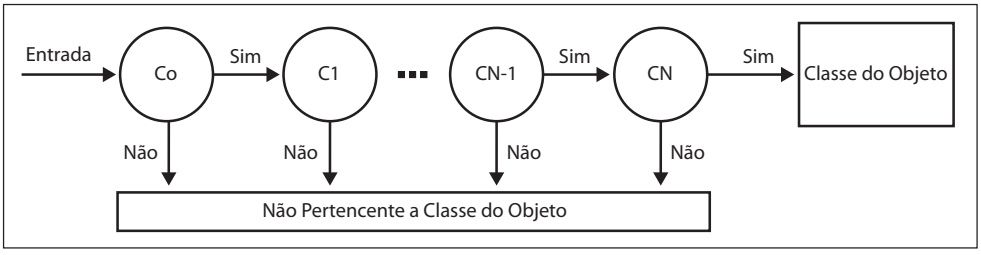
2.3 Árvores generativas de decisão (*decision stump*)

Quando se aumenta a velocidade de classificação, normalmente temos erros que se tornam inevitáveis. O meio mais coerente de diminuir o tempo de classificações é a redução do número de avaliações dos "classificadores fracos" encontrados. Nisto, teríamos uma melhora na velocidade, porém a acurácia do sistema seria prejudicado. Como solução, Viola e Jones (2004) propõem uma estrutura de árvores de decisão, ou ainda, cascata de classificadores.

Viola e Jones (2004) definem árvore de decisão (*decision stump*) como uma estrutura contendo um encadeamento de classificadores do mais genérico ao mais específico, segundo o qual os primeiros níveis da cascata não são muitos precisos, apesar de conseguirem classificar uma grande quantidade de amostras com uma pequena quantidade de características.

Ao usar uma cascata durante o processo de detecção de imagem, caracteriza-se que a maioria das subjanelas analisadas pelo classificador serão rejeitadas. Por isso, nos primeiros passos, existe uma generalização que deve ser suficientemente alta para que evite a passagem para os próximos passos, subjanelas classificadas com falsos positivos.

Figura 6: Cascata de classificadores proposto por Viola e Jones



2.4 Treinamento das árvores de decisão

Conclui-se, portanto, que as árvores de decisão não apenas melhoram a velocidade, mas podem também melhorar a sua acurácia, por meio de uma seleção inteligente dos nós das árvores. A sistemática que deve ser usada é de reduzir as taxas de falsos positivos em cada nó, melhorando a taxa de detecção e, consequentemente, sua eficiência. Segundo Viola e Jones, dada uma árvore de classificação, a taxa de falsos positivos é o produto de todas as taxas de falsos positivos encontrados nos nós de sua árvore.

É possível por meio da cascata de classificadores, adequar uma baixa taxa de falsos positivos para melhorar seus valores. Viola e Jones (2004) descrevem que, para alcançar uma taxa de detecção de 0.9, é necessário apenas dez estágios do classificador. De fato, o que mais é motivador, é que nesse nível a taxa de falso positivo é de 0.3, e isso é demonstrado na aplicação de valores nas equações (2.3.1) e (2.3.2).

Uma das motivações de Viola e Jones (2004) ao usar uma cascata de classificadores deve-se ao fato de seu funcionamento ser de fácil compreensão. Para que tudo funcione conforme o

$$F = \prod_{t=1}^{K} f_{p,i} \qquad (2.3.1)$$

$$D = \prod_{t=1}^{K} d_i \qquad (2.3.2)$$

esperado, são necessários alguns parâmetros, entre eles a taxa de detecção aceitável mínima (d), a taxa de falsos positiva máxima aceitável (f) de cada passo, o conjunto de amostras positivas e negativas (P e N respectivamente), e o valor da taxa de falso positivo requerida no sistema como um todo (f_{alvo}). Através de interações de *boosting* e por meio das informações passadas, cresce a cascata de classificadores. No final, a taxa de falso positivo é comparada com o valor alvo (f_{alvo}) e, se não atingir a meta desejada, mais amostras são carregadas.

2.5 Em busca do objeto

O algoritmo de detecção para buscar o objeto percorrerá várias vezes a imagem analisada em vários tamanhos diferentes, isso chamamos de janela de busca. O tamanho desejado desta janela de busca deve ser maior ou igual ao tamanho com o qual a cascata de classificadores foi inicialmente treinada. Portanto, se o classificador foi treinado com imagens de 24X24 pixels, o tamanho mínimo da janela de busca deve seguir esta resolução. A janela de busca deverá percorrer toda a extensão da imagem, por isso um valor Δ em pixels é escolhido. O deslocamento da janela se dará no eixo x e depois no eixo y, respeitando o valor Δ cadastrado. Alguns experimentos demonstram que um pixel é um valor adequado, pois mesmo sendo mais demorado em grandes imagens, não permitirá perder detalhes em caso de um valor muito alto.

Além de percorrer a imagem horizontalmente\verticalmente, o algoritmo deverá trabalhar o fator de escala δ da janela de tempo. Após a imagem percorrer a imagem com descrito anteriormente, percorrerá novamente e novamente com novos tamanhos de imagem. O objetivo é detectar objetos de vários tamanhos. Viola e Jones (2004) sugerem 1,25 para o fator de escala.

A técnica de imagem integral, citada anteriormente, faz com que a janela de busca, seja eficiente e muito rápida mesmo sendo redimensionada muitas vezes, pois o reescalonamento do detector é feito em tempo constante, através de um cálculo algébrico direto.

3 Material e Métodos

Neste trabalho foram utilizados: um computador, um conjunto de amostras positivas do objeto, um conjunto de amostras negativas, bibliotecas de desenvolvimento, uma *webcam*. Tais matérias e ferramentas serão descritas a seguir:

3.1 *Webcam* e *tablet*

Para detecção em tempo real, a *webcam* utilizada captura imagens em uma resolução de 640X480 e sua definição é de um *Mega* pixel.

O computador utilizado está descrito na Tabela 1 com suas configurações de hardware.

Tabela 1: Configurações do computador.

Nome do computador	Willian-PC
Processador	Intel® Core™ 2 Duo CPU T6500 @ 2.1 GHz
Memória	4,00 GB
Sistema Operacional	Windows 7 Ultimate – 64 bit

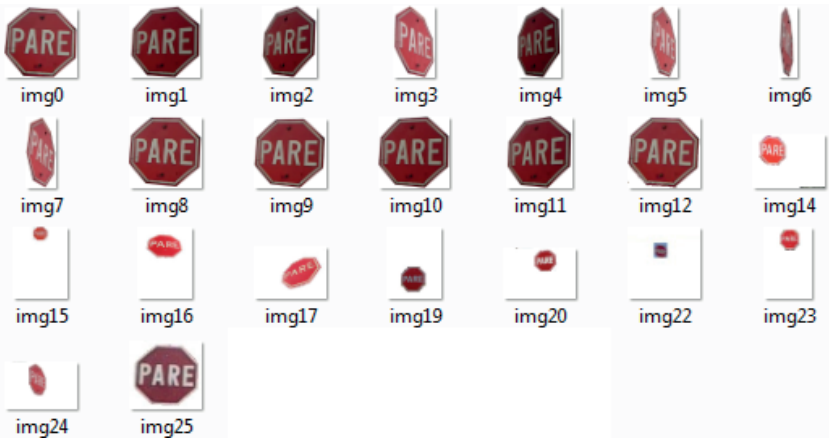
3.2 Conjuntos de imagens positivo e negativo

O conjunto de imagens positivo é importante para o experimento, pois é dele que será extraída toda informação necessária para que o classificador, ou classificadores, possa ser treinado. Eles podem ser criados a partir de uma única imagem com objeto ou por uma coleção de imagens previamente marcadas.

A quantidade de imagens neste conjunto se define pela complexidade de reconhecimento do objeto em questão. Por exemplo, podemos precisar de apenas uma amostra positiva para o objeto absolutamente fácil de ser encontrado, como um *logo*, mas você definidamente precisa de centenas e até milhares de amostras positivas para rosto, por exemplo. No caso de rosto, deve-se considerar toda raça e grupos etários, características e até barba.

Neste trabalho, o objeto em questão é uma placa de trânsito PARE, por isso, segue o conjunto de imagens utilizadas:

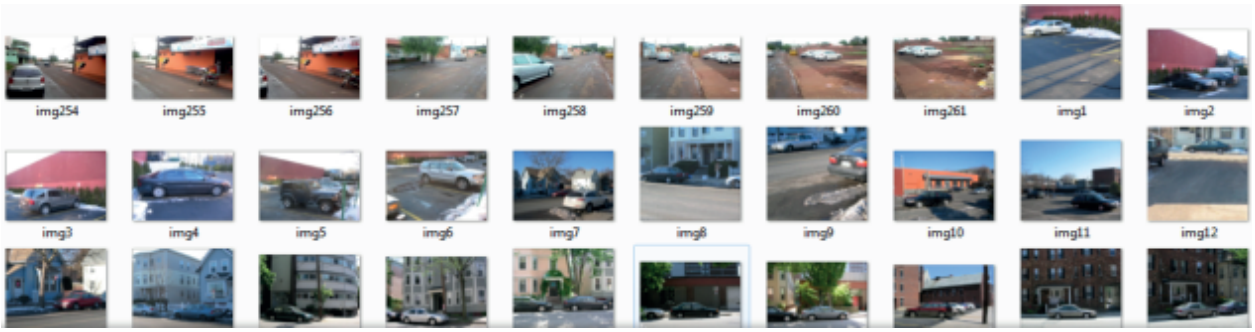
Figura 7: Conjunto de imagens positivo.



O outro conjunto, não menos importante que o conjunto positivo, é o conjunto de imagens negativas. Este conjunto é construído a partir de imagens arbitrárias. Estas imagens não devem conter o objeto que desejamos detectar.

Para este treinamento utilizou-se cerca de 300 imagens negativas. Estas imagens tem sua origem conhecida por todos; a internet. As imagens escolhidas são todas voltadas ao trânsito para que o *background* seja o mais possível parecido com o ambiente no qual a objeto estará.

Figura 8: Parte do conjunto de imagens negativas.



Estes conjuntos apresentados serão utilizados para treinar os classificadores.

3.3 Biblioteca *OpenCV*

OpenCV(Intel Open Source Computer Vision Library) é um conjunto de funções, implementadas em *C\C++*, que implementam algoritmos populares de processamento de imagem.

Neste projeto, utilizaremos algumas funções existentes na biblioteca *OpenCV*, e também algumas de suas ferramentas. Segue nome e resumo de suas funcionalidades:

1. Opencv-createsamples.cpp: preparar os conjuntos de imagens para que os classificadores possam ser treinados. Com esta ferramenta, um grande conjunto de amostras positivas é criado a partir do conjunto de imagens do objeto dado pela rotação aleatória, alterando a intensidade, bem como colocá-lo em um fundo arbitrário.
2. Opencv-haartraining.cpp: utilizado para o treinamento dos classificadores. Algoritmo explicado anteriormente neste artigo.
3. Opencv-performance.cpp: utilizado para realizar teste de desempenho dos classificadores. Esta ferramenta gera uma tabela com o número de objetos detectados, falsos positivos e falsos negativos, possibilitando gerar gráficos de desempenho.

Esta biblioteca contempla mais de 300 funções, e ainda, se necessário, pode-se utilizar bibliotecas externas para melhorar sua aplicabilidade.

Para melhor compreender estas ferramentas, em (OpenCVA) explica-se, com maior detalhes, o que cada ferramenta significa no processo de treinamento de classificadores e demostra os parâmetros necessários para que eles funcionem adequadamente.

3.4 Metodologias aplicadas

Para o projeto ter sucesso, devemos escolher qual objeto iremos detectar; neste caso, o objeto é a placa de PARE. Após a escolha do objeto, formamos o conjunto de imagens positivas e negativas, os quais estão descritos em 3.2.

Para preparar os conjuntos de imagens, utilizou-se a ferramenta *opencv-createsamples,* cuja funcionalidade está descrita em 3.3.

Após a preparação dos dados, a fase de treinamento é feita por meio do aplicativo *opencv-haartraining*, cujo algoritmo está explicado anteriormente.

Foram criados três classificadores, de acordo com as Tabela 2. Estes receberam a nomenclatura de 1, 2 e 3.

Tabela 2: Características dos Classificadores.

	Classificadores		
	1	**2**	**3**
Números positivos	57	57	57
Números negativos	253	277	281
Estágios	9	12	14
Memória	512	512	512
Altura	20	20	20
Largura	20	200	20
Algoritmo	GAB	GAB	GAB
Mínimo acerto	0,5	0,5	0,5
Máximo erro	0,95	0,95	0,95

Cada um destes classificadores foi criado a partir dos parâmetros citados acima, variando o número de estágios de treinamento e número de imagens negativas.

4 Resultados e Discussão

Após o treinamento, a árvore de classificação será testada para avaliar seu desempenho. Após treinar

os classificadores, utiliza-se o aplicativo *opencv-performance*, explicado em 3.2, que gera uma tabela com o número de objetos detectados, falsos positivos e falsos negativos.

Como prova de conceito, foi realizado um teste de um classificador para identificar se ele foi corretamente treinado. A Figura 9 demostra um conjunto de imagens de teste e o resultado da execução do classificador.

Figura 9: Parte do conjunto de imagens negativas.

Como pode ser visto na Figura 10, o número de acertos aumentou de acordo com o número de estágios, que também aumentou. Isto nos mostra que, quanto mais o classificador passou por passos de treinamento, menores foram as generalizações e melhores os resultados.

Na Figura 11, existe um comparativo entre os erros e acertos dos classificadores. Ao analisar o gráfico, veremos que o classificador com maiores acertos e menor número de erros é o último que teve maior número de imagens no conjunto negativo e que passou por maior número de estágios de treinamento.

Figura 10: Gráfico de acertos X estágios

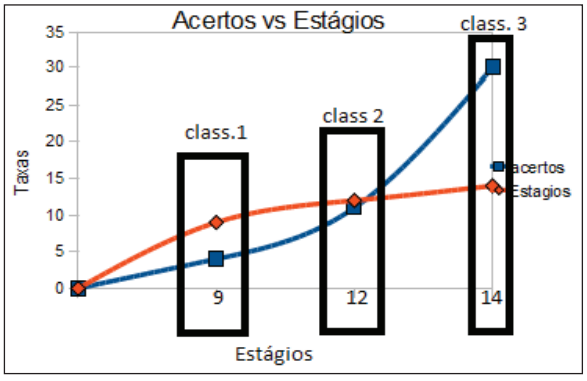
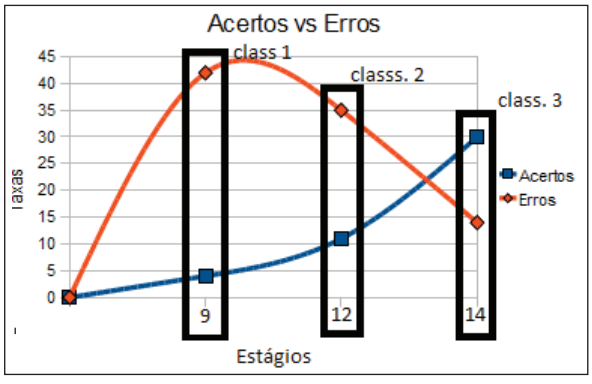


Figura 11: Gráfico de acertos X erros



Uma característica crucial para a melhora do classificador, além do maior número de estágios, foi a diferença no número de imagens no conjunto negativo.

O número de *features,* ou características extraídas do

conjunto positivo, foi de 125199.

Apesar de obterem bons resultados, os classificadores foram treinados em configurações mínimas. O número de imagens no conjunto positivo e negativo, segundo Viola e Jones (2004), deve ultrapassar os 1000, pois através disso teremos melhores resultados em diversas situações.

5 Conclusão

Este trabalho não só nos proporcionou um aprendizado de técnicas relacionadas à detecção, treinamento de classificadores, mas nos permitiu aplicar tais conceitos em uma situação do no nosso dia-a-dia.

Estudos de visão computacional nos proporcionou acesso à biblioteca *OpenCV*, sendo que suas ferramentas foram importantes para a execução e avaliação do aplicativo desenvolvido.

Por ser um classificador de longo prazo de treinamento, um ponto a considerar é que, mesmo usando o mínimo de imagens nos conjuntos (positivo, negativo) que resultou em pouco tempo para treinar os classificadores, conseguimos obter bons resultados para a detecção do objeto desejado.

Por fim, este trabalho, resultando em um aplicativo, é um protótipo, mostrando que é possível a construção de uma ferramenta que permita orientar condutores para evitar, sobretudo, os acidentes de trânsito.

Referências

BISHOP, C.M. Pattern Recognition and Machine Learning. Amazon.com., 2004.

BREIMAN, L. Bagging predictors. *Mach. Learn.*, v.24, n.2, p.123-140, 1996.

CONSTANTINE, P.; PAPAGEORGIOU, M.O.; TOMASO, P. *A general framework for object detection*. *In:* ICCV '98: PROCEEDINGS OF THE SIXTH INTERNATIONAL CONFERENCE ON COMPUTER VISION, p.555, Washington, DC, USA, 1998. IEEE Computer Society.

DRUCKER, H; SCHAPIRE, R.E; SIMARD, P. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, v.7, n.4, p.705-719, 1993.

FREUND, Y. *Boosting a weak learning algorithm by majority. In:* COLT: PROCEEDINGS OF THE WORKSHOP ON COMPUTATIONAL LEARNING THEORY, MORGAN KAUFMANN PUBLISHERS, 1990.

FREUND, Y.; SCHAPIRE, R.E. *A decision-theoretic generalization of on-line learning and an application to boosting. In:* EUROPEAN CONFERENCE ON COMPUTATIONAL LEARNING THEORY, p.23-37, 1995.

LIENHART R; KURANOV, A; PISAREVSKY, V. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Technical report, Microprocessor Research Lab, Intel Labs, December 2002.

MALLAT, S.G. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, v.11, n.07, p.674-693, 1989.

MEYER, D.; LEISCH, F.; HORNIK, K. The support vector machine under test. *Neurocomputing*, v.55, p.169-186, 2003.

MONARD, M.C.; BARANAUSKAS, J.A. *Sistemas inteligentes: fundamentos e aplicações*. São Paulo: Manole, 2003.

(OpenCV) *Open source computer vision library*. Disponível em: <http://docs.opencv.org/. Acesso em: 10 set. 2012.

OPENCVA. *Cascatade Classifier Training*. Disponível em: http://opencv.itseez.com/trunk/doc/user_guide/ug_traincascade.html. Acesso em: 19 jun. 2012.

PARISE, G. *Necessidade da lei*. 2008. Disponível em: http://gutoparise.blogspot.com.br/2008/03/necessidade-da-lei.html. Acesso em: 15 ago. 2012.

POLIKAR, R. Emsemble based systems in decision making. *IEEE Circuits and Systems Magazine*, v.6, n.3, p.21-45, 2006.

SCHAPIRE, R.E. The strength of weak learnability. *Machine Learning*, v.5, p.197-227, 1990.

VIOLA, P.; MICHAEL, J. Robust real-time face detection. *Int. J. Comput. Vision*, v.57, n.2, p.137-154, 2004.