colloquium exactarum

UMA METODOLOGIA PARA DETECÇÃO DE SINAIS DE TRÂNSITO EM VÍDEO

A METODOLOGY FOR DETECTION OF TRAFFIC SIGN IN VIDEO

Antonio Carlos Paes Nascimento[1]; Francisco Assis da Silva[1]; Mário Augusto Pazoti[1]; Danillo Roberto Pereira[1]; Almir Olivette Artero[2]; Marco Antonio Piteri[2]

[1]Universidade do Oeste Paulista – UNOESTE, Faculdade de Informática – FIPP, Presidente Prudente – SP. [2] Universidade Estadual Paulista – UNESP, Faculdade de Ciências e Tecnologia – FCT, Presidente Prudente – SP. E-mail: {almir, piteri}@fct.unesp.br, acpn@unoeste.edu.br, {chico, mario, danilopereria}@unoeste.br

RESUMO - Neste trabalho é realizada a detecção de placas de sinalização de trânsito utilizando *frames* de vídeo de alta resolução, não importando toda a informação do sinal impressa na placa. São buscadas placas considerando apenas as informações como bordas, círculos vermelhos com ou sem as linhas diagonais que representam proibição, placas com fundo amarelo, e placas com fundo vermelho. O método utilizado para realizar a detecção é o método proposto por Viola e Jones (2001), com um treinamento específico para sinais de trânsito. Para alcançar bons resultados, foram realizadas avaliações de cores no espaço de cor HSV para eliminar regiões que não contêm placas, e com isso diminuir a taxa de falsos positivos durante o processo de detecção. Os resultados obtidos com os experimentos realizados mostram que é possível realizar a detecção de placas de trânsito de vários formatos, em um tempo de processamento aceitável para uma aplicação em tempo real.

Palavras-chave: Detecção de Sinais de Trânsito; AdaBoost; Tempo Real.

ABSTRACT - In this work is performed the detection of road traffic signs using video frames of high resolution do not considering all the information printed on plate. Plates are searched considering only the information as edges, red circles with or without the diagonal lines representing prohibition, plates with yellow background, and plates with red background. The method used to perform the detection is proposed by Viola and Jones (2001), with a specific training method for traffic signals. To achieve good results, assessments of colors in the color space HSV were performed to eliminate regions that contain plates, and thus decrease the rate of false positives during the detection process. The results obtained from the experiments show that it is possible to detect traffic plates of various shapes in an acceptable processing time for a real time application.

Keywords: Traffic Signs Detection; Adaboost; Real Time.

Recebido em: 25/05/2014
Revisado em: 10/08/2014
Aprovado em: 23/08/2014_

1 INTRODUÇÃO

O reconhecimento automático de placas de trânsito está se tornando uma área com muitas aplicações na indústria automotiva (MATINOVIĆ, 2010), por exemplo, possibilitando que um motorista seja avisado sobre ações impróprias e situações potencialmente perigosas, o que consiste em uma tarefa essencial para um sistema inteligente de auxílio a motoristas (BARÓ et al., 2009). Para que seja possível o reconhecimento, um método eficiente para detecção é indispensável.

Pesquisas recentes em Visão Computacional e com o avanço da tecnologia, melhor a cada dia, a detecção e reconhecimento de sinais de trânsito em vídeo já é uma realidade. No entanto, ainda há grandes desafios a serem superados, pois como o processamento de imagens requer muito esforço computacional, apenas algumas classes isoladas de sinais de trânsito são detectadas e reconhecidas em vídeo.

Segundo Chen e Hsieh (2008) é difícil detectar os sinais de trânsito diretamente a partir de vídeos, devido a diferentes alterações de condições ambientais, o que pode ter um sinal de trânsito com diferentes mudanças na aparência, incluindo iluminação, cor ou sombras em diferentes dias, estações e climas. Além disso, para a câmera montada na frente de um veículo em movimento, as imagens são obtidas em diferentes perspectivas, fazendo um sinal de trânsito ter diferentes tamanhos, formas, contrastes e borramentos (ocasionado pelo deslocamento do carro). Em alguns casos, a placa com o sinal de trânsito pode ser oclusa por galhos de árvores, folhas ou veículos, e, ainda devido à exposição contínua ao ambiente ocasiona danos à placa, dificultando o processo de detecção.

Este trabalho apresenta uma metodologia para realizar a detecção de sinais de trânsito em vídeo, utilizando técnicas de Visão Computacional. No treinamento adotado neste trabalho, o conjunto de imagens positivas é preparado considerando apenas algumas informações contidas na placa, como cor da borda e cores de fundo. Assim, o algoritmo detecta as placas pelas características que condizem com o formato da placa, abrangendo assim uma maior quantidade de classes de placas. O treinamento é realizado pelo algoritmo AdaBoost em uma estrutura em cascata proposto por Viola e Jones (2001).

Apesar do AdaBoost, em seu método original, ser proposto para detecção de rostos, alguns autores tem aplicado a técnica com sucesso para detecção de sinais de trânsito, como é o caso de Brkić (2010), Matinović (2010), Landeza-Vázquez, Parada-Loira e Alba-Castro (2010) e Timofte, Zimmermann e Van Gool (2011). Porém, de

forma diferente deste trabalho, eles utilizam toda informação contida na placa, assim limitando-se a duas ou três classes de sinais.

O grande desafio na detecção de sinais de trânsito é a detecção de várias classes de sinais, devido principalmente a diferentes condições de iluminação, diferentes formatos e cores dos sinais como mencionado por Chen e Hsieh (2008). Um exemplo desse problema seria detectar dois sinais com informações diferentes, mas, porém com o mesmo formato e cor das bordas. Para detectar esses dois sinais seria necessário realizar, dois treinamentos distintos devido a informação presente no sinal ser diferente. O tempo para preparação das imagens para treino é relativamente alto, e, com o aumento das classes de sinais a serem detectadas são necessárias mais imagens para treinamento. Isto torna o processo demorado, pois alguns treinamentos levam dias para serem concluídos, e se o resultado do treinamento não for suficiente o processo tem que ser refeito.

A principal contribuição apresentada neste trabalho esta na fase de treinamento, em que a informação do sinal de trânsito não é considerada, eliminando assim a necessidade de usar sinais de um mesmo formato, porém com informações diferentes. Da placa de "pare" não foi retirada a informação do sinal, pois como sua forma é singular não se faz necessário retirar a informação, a placa "dê preferência" também não foi preciso realizar tratamento, pois sua forma também é singular. Na Figura 1 estão alguns dos sinais, utilizados para treinamento, como imagens positivas.

Foi utilizado o espaço de cor HSV (*Hue, Saturation, Value*) no algoritmo de detecção, sendo aplicado após se detectar uma região de interesse candidata. Antes de afirmar que a região detectada é um sinal de trânsito, essa região de interesse é convertida para o espaço de cor HSV e as cores são analisadas. Se a cor da região de interesse estiver dentro das cores definidas para o sinal, então o sinal é considerado detectado, caso contrário, essa região de interesse é descartada.

Figura 1. Placas utilizadas para o treinamento como imagens positivas.

As demais seções deste trabalho estão organizadas da seguinte maneira: na Seção 2 são relatados os trabalhos relacionados, usados como base para o desenvolvimento deste trabalho; na Seção 3 é detalhada a fundamentação teórica; na Seção 4 é apresentada a metodologia proposta; a Seção 5 apresenta os experimentos realizados e os resultados obtidos; a Seção 6 apresenta a conclusão

obtida com este trabalho e expõe as considerações finais.

2 TRABALHOS RELACIONADOS

A área de Visão Computacional vem ganhando mais expressão nos últimos anos, devido ao grande avanço tecnológico. Como o processamento de imagens requer um alto custo computacional, máquinas poderosas são exigidas para uma aplicação de sucesso (BRKIĆ, 2010).

Técnicas robustas estão sendo criadas para contornar este alto custo computacional, uma das técnicas mais popular e eficiente no momento é o algoritmo de detecção de faces de Viola e Jones (2001). Esse é um algoritmo eficiente, que pode ser adaptado para detectar qualquer objeto em imagens (BRKIĆ, 2010).

Baseado nos trabalhos de Brkić (2010), Matinović et al. (2010), Landeza-Vázquez, Parada-Loira e Alba-Castro (2010), Timofte, Zimmermann e Van Cool (2011), e no trabalho original de Viola e Jones (2001), o presente trabalho se torna possível e viável, uma vez que, nos trabalhos apresentados, sinais de trânsito de formas retangulares e circulares foram detectados.

Com um método para detectar sinais de trânsito em vídeo, alguns segmentos de mercado obteriam grande êxito, principalmente a indústria automobilística, que, por exemplo, poderia produzir veículos com sistemas inteligentes para auxiliar motoristas nas estradas e rodovias. Além de inúmeros outros ganhos, por exemplo, para empresas que realizam a manutenção da sinalização nas estradas e rodovias, pois o processo torna-se mais rápido e menos propenso a falhas, consequentemente aumentaria a segurança (BRKIĆ, 2010).

No trabalho de Matinović (2010), o treinamento do algoritmo de Viola e Jones (2001), para a detecção de sinais de trânsito em vídeo foi realizado utilizando 5 tipos de sinais de trânsito circulares (Figura 2), com um conjunto de 757 imagens de sinais de trânsito positivas (imagens que contém apenas o sinal). Cada imagem possui apenas um sinal de trânsito, que foi recortado e redimensionado para o tamanho 24x24, que é o tamanho utilizado pelo algoritmo de Viola e Jones (2001). Foram usadas também 3000 imagens negativas (imagens que não contém o sinal de trânsito). O treinamento foi feito, utilizando a biblioteca de Visão Computacional OpenCV. Segundo o autor os resultados utilizando 20 *frames* por segundo foram significativos, apresentando uma taxa de acerto de 73%.

Figura 2. Sinais de trânsito de mesma classe utilizados no trabalho de Matinović (2010).

No trabalho de Landeza-Vázquez, Parada-Loira e Alba-Castro (2010), foram

detectados sinais de trânsito circulares e triangulares, adaptando também o algoritmo de Viola e Jones (2001).

Os testes foram realizados utilizando vídeos de um banco de dados da *Galician Automotive Technology Center* (CTAG), em que foram extraídos 590 sinais de limite velocidade, 921 sinais de perigo e 485 sinais de outras classes. Utilizando um processador de 2.8 GHZ, e com 14 frames por segundo, foi relatado que a detecção dos sinais foi realizada com sucesso em aproximadamente 70.21 ms (milissegundos), em diferentes condições de luminosidade. Na Figura 3, é apresentada a interface de detecção do trabalho, em que o teste foi realizado em ambiente noturno.

Figura 3. Sistema de Landeza-Vázquez, Parada-Loira e Alba-Castro (2010), em funcionamento, em ambiente noturno.

No trabalho de Timofte, Zimmermann e Van Gool (2011), é usado como referência também o detector de faces de Viola e Jones (2001). No trabalho, a detecção dos sinais de trânsito ocorre em aproximadamente 1,5 metros de distância, com 2 frames por segundo, sendo a taxa de detecção de aproximadamente 95%. A Figura 4 apresenta a imagem da detecção realizada.

Figura 4. Imagem com placas detectadas pelo sistema de Timofte, Zimmermann e Van Gool (2011).

3 FUNDAMENTAÇÃO TEÓRICA

O algoritmo que foi utilizado para realizar a detecção das placas de sinalização de trânsito nos *frames* de vídeo é baseado no algoritmo AdaBoost (FREUND; SCHAPIRE, 1999), utilizado com muita eficiência por Viola e Jones (2001) para a detecção de rostos.

O detector de Viola e Jones baseia-se em uma cascata de classificadores boosted Haar-like. Ela combina dois conceitos: (1) AdaBoost e (2) classificadores Haar-like. Classificadores Haar-like são construídos com características retangulares (Haar-features) simples que representam as diferenças de somas de pixels específicos em uma imagem. Cada característica é emparelhada com um limiar, e a decisão do classificador assim construído é determinado por comparação do valor da característica com o limiar (BRKIĆ, 2010).

O algoritmo AdaBoost combina um conjunto de classificadores fracos para formar um classificador forte para detecção de objetos (CHEN; HSIEH, 2008). AdaBoost atribui pesos de classificadores fracos com base na sua qualidade, e o classificador forte resultante é uma combinação linear de classificadores fracos com os pesos apropriados.

Viola e Jones implementaram o AdaBoost em uma estrutura em cascata, assim é definido um número de iterações, cada iteração corresponde a um estágio da cascata, e a saída de cada estagio é um classificador forte. O classificador forte na primeira fase da cascata é escolhido de modo que descarta um número de falsos positivos, enquanto preserva quase todos os verdadeiros positivos do conjunto de treino. O processo de detecção é realizado por uma janela deslizante de detecção através da imagem (*slide-window*). Dentro da janela, a resposta da cascata é calculada. Depois de completar uma passagem sobre a imagem, o tamanho da janela de detecção é aumentado por algum fator de escala (padrão OpenCV 1.1, o que significa que a escala da janela será aumentado em 10%). O tamanho da janela é aumentado até um tamanho pré-definido ser alcançado. Aumentando a janela de detecção por uma porcentagem menor produz as melhores taxas de detecção, mas aumenta o tempo total de processamento e os falsos positivos (BRKIĆ; PINZ; SEGVIC, 2010).

3.1 CARACTERÍSTICAS HAAR

Viola e Jones (2001) propuseram uma metodologia para detecção de faces e objetos que provê uma implementação rápida e robusta. Três contribuições importantes surgiram deste trabalho, sendo:

- Introdução de uma nova representação de imagens denominada imagem integral;
- Construção de um classificador simples e eficiente a partir da seleção de características de Haar (ACHARYA; RAY, 2005), utilizando o algoritmo de aprendizado do tipo boosting (FREUND; SCHAPIRE, 1999), denominado AdaBoost;
- Combinação de classificadores fracos para formar uma estrutura de cascata que aumenta a velocidade da classificação, dando mais foco às áreas mais promissoras da imagem.

O sistema de detecção de faces proposto por Viola e Jones (2001) utiliza características simples para classificar as imagens, baseadas nas funções de Haar.

As características correspondem basicamente de retângulos que possuem duas regiões, sendo uma clara e outra escura. O cálculo do valor correspondente à característica é feito subtraindo o somatório da intensidade dos pixels da região escura do

total acumulado pela intensidade dos pixels da região clara. Este cálculo pode ser feito rapidamente utilizando a representação de imagem integral. Na Figura 5 são apresentadas as características básicas de Haar que foram utilizadas por Viola e Jones (2001).

Figura 5. Características básicas de Haar.

Com o intuito de melhorar ainda mais a etapa de detecção de objetos, Lienhart e Maydt (2002) estenderam a quantidade de tipos possíveis de características retangulares, propondo características novas às previamente sugeridas, mas desta vez rotacionadas em 45 graus. Com estas novas adições ao conjunto, consegue-se reduzir o número de falsos positivos em torno de 10%, aumentando ainda mais a eficiência do processo de detecção. Na Figura 6 é apresentado o conjunto completo com as novas características retangulares propostas.

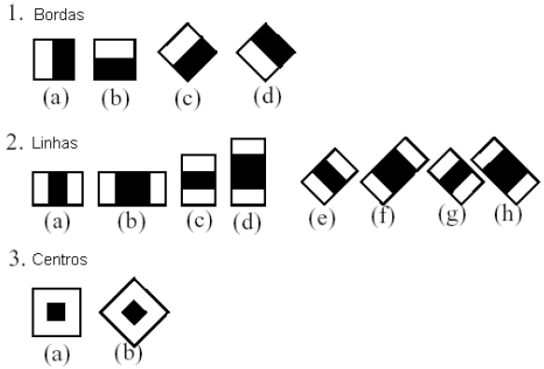

Figura 6. Características estendidas.

3.2 INTEGRAL DA IMAGEM

Ao invés vez de recalcular somas de retângulos para cada característica em cada re-escalonamento podem-se calcular todos os valores no início e guardá-los para cálculos futuros. Isto pode ser feito através da formação de uma tabela de área somada para o *frame* a ser processado, também conhecida como imagem integral.

Dada uma imagem *I*, sua imagem integral *S*(*x*, *y*) contém a soma dos valores de intensidade dos pixels em *I* acumulados da imagem original, iniciando em (0, 0) até o pixel (*x*, *y*), isto é,

$$S(x,y)=\sum_{i=0}^{x}\sum_{j=0}^{y} I(i,j) \qquad (1)$$

A imagem integral pode ser calculada recursivamente, por *S*(*x*, *y*) = *S*(*x*, *y*–1) + *S*(*x*–1, *y*) + *I*(*x*, *y*) – *S*(*x*–1, *y*–1), com a condição de contorno: *I*(–1, *y*) = *I*(*x*, –1) = *I*(–1, –1) = 0.

Claramente, o cálculo de *S*(*x*, *y*) pode ser concluído usando apenas uma varredura sobre a imagem de entrada *I*. Dado um retângulo *R* delimitado por (*l*, *t*, *r*, *b*), a soma do valor de intensidade de pixel pode ser muito eficientemente alcançado ao tirar vantagem da imagem integral *S*. Conforme Figura 7, a soma de *F*(*R*) de intensidade de pixel em *R* pode ser facilmente calculada (CHEN; HSIEH, 2008):

$$\begin{aligned} F(R) &= (A+B+C+R)+A-(A+B)-(A+C) \\ &= S(r,b)+S(l,t)-S(l,b)-S(r,t) \end{aligned} \qquad (2)$$

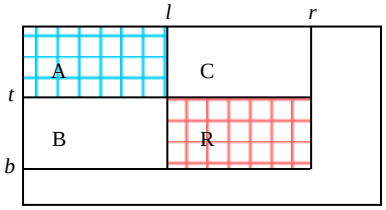

Figura 7. Cálculo da imagem integral.

Como representado na Figura 7, em uma imagem integral, a área para qualquer região retangular na imagem pode ser calculada usando somente quatro acessos à matriz. A Figura 8 mostra um exemplo de cálculo de uma área.

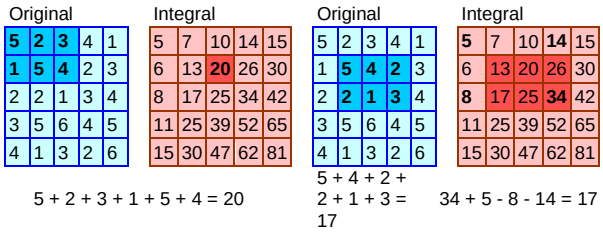

Figura 8. Exemplo de cálculo de uma área a partir da imagem integral.

Na Figura 8, as matrizes azuis representam a imagem original, enquanto que as matrizes vermelhas representam as imagens depois da transformação integral. Se a área mais escura for calculada na primeira imagem, tem-se a soma de todos os pixels individualmente, produzindo o valor 20 depois de seis acessos à matriz. Usando a imagem integral, precisaria de apenas um único acesso (isso somente porque a região escura faz parte da borda). No caso de uma região que não esteja na borda, requer quatro acessos, independente do tamanho da região, reduzindo a complexidade computacional de $O(n)$ para $O(1)$. O cálculo requer apenas uma adição e duas subtrações para recuperar a soma da área escura (Equação 2).

3.3 ADABOOST

Boosting é um método utilizado para aprimorar o desempenho de um algoritmo de aprendizado, não sendo utilizado de forma isolada, e sim sendo aplicado de forma combinada com outras técnicas, como Árvores de Decisão e Redes Neurais. O objetivo é transformar classificadores "fracos" em classificadores "fortes". O método trabalha com a combinação de classificadores gerados por um mesmo algoritmo de aprendizado (algoritmo base ou fraco) e seu funcionamento é ajustado de acordo com os erros cometidos pelo classificador anterior. Assim Boosting busca gerar novos classificadores melhores e mais adequados para o problema, principalmente por corrigirem e aumentarem a eficiência dos classificadores gerados de forma isolada (CHAVES, 2012).

O modelo Boosting, compreende uma família de algoritmos, dentre os quais o algoritmo Adaptive Boosting (AdaBoost) é o mais influente e popular (CHAVES, 2012). Segundo Freund e Schapire (1996), AdaBoost é um algoritmo de aprendizado que combina um conjunto de classificadores fracos para formar um classificador forte. Segundo Chen e Hsieh (2008), um classificador fraco usa um recurso simples para determinar amostras positivas, sendo o resultado apenas um pouco melhor do que o acaso. A cada iteração, um

bom classificador fraco é selecionado e adicionado para formar um classificador forte, que é uma soma ponderada de cada classificador fraco selecionado.

3.3.1 FUNCIONAMENTO DO ALGORITMO ADABOOST

Com base nas informações encontradas em Chaves (2012), originalmente obtidas em Freund e Schapire (1996), em Ramos (2012), é descrito, de forma resumida, o funcionamento do algoritmo AdaBoost.

A primeira etapa para o funcionamento do algoritmo AdaBoost é o treinamento, em que, nesta fase, a entrada é um conjunto de exemplos $S = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$, onde:

- Cada x_i representa um vetor de atributos do sistema, ou seja, um conjunto de dados referentes aos parâmetros (ou propriedades) analisados e que representam um dado instante do sistema;
- Dentre todos os possíveis grupos de classificação predefinidos para o sistema analisado, cada y_i representa o grupo de classificação associado ao x_i;
- O número total de exemplos da amostra de treinamento é igual a m.

O algoritmo base que trabalha junto ao AdaBoost, fornece a cada rodada, uma distribuição de pesos referentes a cada um dos dados de treinamento. Os classificadores iniciam com pesos associados igualmente distribuídos.

$$D_0 = \frac{1}{m} \quad (3)$$

A cada ciclo de aprendizagem, o algoritmo base gera uma hipótese h_t baseada nos pesos atuais com o objetivo de priorizar a classificação correta dos dados que apresentam os maiores pesos associados. Logo, pode-se afirmar que o objetivo do algoritmo base é gerar uma hipótese de forma a minimizar o erro de treinamento e_t.

$$e_t = \sum_{i:h_t(x_t)\neq y_t} D_t(i) \quad (4)$$

Em seguida, esses pesos são reavaliados de forma a aumentar aqueles que são relacionados aos dados incorretamente classificados e uma nova rodada se inicia. Por fim, ao serem realizadas todas as T rodadas determinadas previamente, o AdaBoost combina todas as hipóteses intermediárias h_t de forma a gerar uma única hipótese final $H(x)$. O método para se obter a hipótese final $H(x)$ fornecida pelo AdaBoost é a combinação ponderada das diversas saídas das iterações. Assim, uma votação ponderada das T hipóteses fracas é realizada, onde γt é a importância associada à hipótese h_t.

O esquema mostrado na Figura 9 resume o funcionamento do algoritmo AdaBoost.

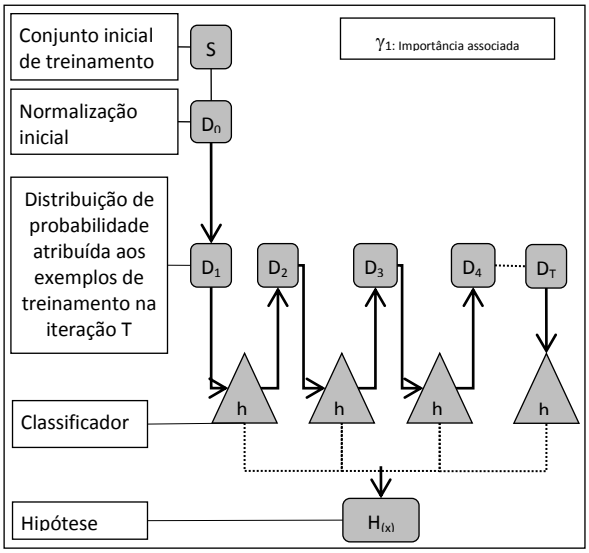

Figura 9. Esquema do funcionamento do algoritmo AdaBoost.

3.3.2 ESTRUTURA CASCATA DE CLASSIFICADORES

O algoritmo AdaBoost melhora o desempenho da classificação por iterativamente adicionar características ao classificador forte. Quando mais características são adicionadas, a técnica de adição irá diretamente aumentar muito o tempo de computação para o classificador projetado para detectar objetos. No entanto, se uma estrutura em cascata é utilizada para treinar o classificador forte, a sua eficiência será significativamente melhorada (CHEN; HSIEH, 2008). Introduzida por Viola e Jones (2001), essa estrutura divide o processo de treinamento em camadas, dessa forma o processo de treinamento é simplificado. Ramos (2012) menciona que a estrutura em cascata, é necessária para eliminar objetos que não correspondem às características procuradas, esses objetos são descartados nos estágios iniciais do processo de classificação, para que os estágios posteriores não despendam processamento desnecessário.

A cascata pode rapidamente descartar a maioria dos objetos negativos, passando somente os objetos positivos, estágio por estágio. Cada classificador na cascata necessita ter uma taxa de detecção muito alta, mas apenas uma taxa moderada de falsos positivos (por exemplo, 50%). Uma região de entrada é passada de h_t para h_t+1 se ela é classificada como um objeto, caso contrário, ela é rejeitada. Se a taxa de alarme falso de cada classificador h_t é η_t (menos de 50%) e independente de outros classificadores, o alarme falso acumulado irá se tornar:

$$\prod_{t=1}^{T} \eta_t \qquad (5)$$

Após T classificadores mais fracos terem sido aplicados. O erro de detecção será significativamente reduzido se mais classificadores h_t são usados. Como a maioria dos objetos negativos são rejeitados rapidamente, o detector projetado pode ser extremamente rápido para detectar os objetos desejados (CHEN; HSIEH, 2008). A Figura 10 mostra a configuração da estrutura em cascata.

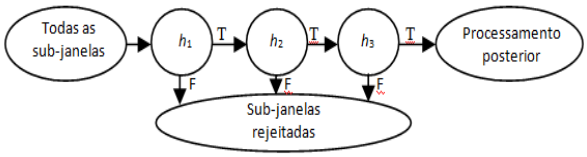

Figura 10. Estrutura cascata do classificador.

3.4 O ESPAÇO DE CORES HSV

O espaço de cores HSV é formado por três componentes, a matiz (*hue*, H), a saturação (*saturation,* S), e o valor (*value,* V). É uma transformação não linear do espaço RGB (RAMOS, 2012).

Considere o espaço RGB normalizado, onde cada uma das componentes varia de 0 a 1.0, sendo que MAX e MIN são, respectivamente, o valor máximo e mínimo desse espaço normalizado (RAMOS, 2012). Para obter as componentes H, S e V têm-se:

$$H = \begin{cases} 60 \times \frac{G-B}{MAX-MIN} + 0, se\ MAX = R\ e\ G \geq B \\ 60 \times \frac{G-B}{MAX-MIN} + 360, se\ MAX = R\ e\ G\ < B \\ 60 \times \frac{B-R}{MAX-MIN} + 120,\ \ se\ MAX = G \\ 60 \times \frac{R-G}{MAX-MIN} + 240,\ \ se\ MAX = B \end{cases} \quad (6)$$

$$S = \frac{MAX-MIN}{MAX} \quad (7)$$

$$V = MAX \quad (8)$$

4 METODOLOGIA PROPOSTA

A metodologia proposta neste trabalho tem como base o algoritmo de Viola e Jones (2001), com um treinamento específico para sinais de trânsito e com as devidas modificações no algoritmo de detecção, além de uma avaliação de cores no espaço HSV, para reduzir os falsos positivos.

4.1 TREINAMENTO

Com a metodologia proposta neste trabalho, é necessário realizar a preparação das imagens antes de iniciar a fase de treino, ou seja, antes de iniciar o treinamento é preciso remover a informação contida na placa. O processo de preparação das imagens para o treinamento foi realizado em duas etapas. A primeira etapa consistiu em remover os sinais inseridos nas placas, utilizando o editor de imagens Adobe Photoshop CS4, em que foi realizada a remoção de cerca de 1.500 sinais inseridos nas placas.

Como o número necessário de imagens para o treinamento é relativamente alto, para obter um bom classificador final como observado em Viola e Jones (2001), Brkić (2010), Martinović (2010), Landeza-Vázquez, Parada-Loira e Alba-Castro (2010) e Timofte, Zimmermann e Van Gool (2011), houve a necessidade de implementar uma aplicação que é capaz, de, a partir, de uma imagem de entrada, gerar uma imagem de saída com aumento ou redução de luminosidade, ou aumento e redução de intensidade da cor. A aplicação também foi escrita em C++ com a biblioteca OpenCV utilizando o espaço de cores HSV (Sessão 3.4). O resultado é um novo conjunto de imagens, que somado ao conjunto obtido na primeira etapa do processo, completa a

quantidade necessária, finalizando a segunda etapa do processo.

A Figura 11 mostra as placas originais e, na Figura 12 é possível observar as mesmas placas sem a informação contida no sinal, a Figura 13 mostra as imagens geradas pelo aplicativo desenvolvido. As imagens das Figuras 12 e 13, representam as imagens utilizadas no treinamento para realização desse projeto.

Figura 11. Placas de trânsito originais.

Figura 12. Placas de trânsito sem informação do sinal.

Figura 13. Placas de trânsito geradas pelo aplicativo desenvolvido.

Com as placas “pare” e com as placas triangulares “dê preferência”, não foi necessário realizar a preparação (primeira etapa), pois suas formas são singulares. Também são necessárias para o treinamento as imagens negativas (não contém o objeto desejado), e geralmente é usado o dobro da quantidade de imagens positivas. A Figura 14 mostra algumas imagens negativas usadas para o treinamento realizado neste trabalho.

Figura 14. Exemplos de imagens usadas como negativas para o treinamento.

Para realizar o treinamento foi utilizada a biblioteca de Visão Computacional OpenCV da Intel, e os aplicativos disponibilizados pela biblioteca escritos em C++ *"opencv_createsamples"* e *"opencv_traincascade"*. O aplicativo *"opencv_createsamples"* tem como entrada um arquivo texto com os nomes das imagens positivas que são utilizadas para treinamento, bem como as devidas coordenadas como largura e altura de cada imagem, após executado, ele gera um arquivo com extensão ".vec" que posteriormente será utilizado no aplicativo *"opencv_traincascade"* que realizará o treinamento.

No aplicativo *"opencv_traincascade"*, além de ser necessário fornecer como parâmetro o arquivo ".vec", também necessita-se ser informado o arquivo texto com os nomes e coordenadas das imagens

negativas, e a quantidade de estágios que serão executados, além da largura e altura das imagens. Vale ressaltar que, esses parâmetros de largura e altura devem ser os mesmos que foram usados no aplicativo "*opencv_createsampes"*, e que o treinamento pode terminar de duas maneiras, ou pelo número de iterações ou se o limiar (*threshold*) for atingido.

Foi utilizado um conjunto com um total de 18.000 imagens em quatro treinamentos distintos. Dentre as quais 9000 são circulares, onde 3000 são imagens positivas e 6000 são imagens negativas. Para as placas de alerta foram utilizadas 3000 imagens, dentre as quais 1000 são positivas, e 2000 são negativas. Para os sinais pare e os triangulares foram utilizadas 1000 imagens positivas e 2000 negativas para cada treinamento.

O treinamento foi realizado em um computador com processador Intel core i5, com 3.1 GHZ. Ignorando os treinamentos que não obtiveram êxito, e adições e remoções das imagens de treinamento, o tempo total gasto para a realização dos treinamentos foi de aproximadamente 43 dias. A Tabela 1 mostra o tempo gasto no treinamento de cada tipo de placa de trânsito, a quantidade de imagens usadas e o tempo total gasto para cada treinamento.

Tabela 1. Dados dos treinamentos realizados para este trabalho.

Placa	Positivas	Negativas	Tempo (horas)
Circular	3.000	6.000	504
Alerta	1.000	2.000	168
Dê preferência	1.000	2.000	150
Pare	1.000	2.000	190
Total	6.000	12.000	1012

Após o término do treinamento, um arquivo XML (*eXtensible Markup Language*) é gerado com o classificador. Esse arquivo de treinamento possui uma estrutura do tipo árvore, que posteriormente é utilizado no algoritmo de detecção. A Figura 15 representa os valores que foram atribuídos para os parâmetros de um estágio de treinamento, e a Figura 16 representa uma característica extraída.

```
<maxWeakCount> 4 </maxWeakCount>
<stageThreshold> -
1.3605325222015381e+000</stageThreshold>
<weakClassifiers>
<_>
<internalNoldes> 0 -1 23 --6.38027771270275116e-
002</internalNodes>
<leafValues> 8.350951671600348e-001 -
7.3932582139968872e-001</leafValues>
</_>
</weakClassifiers>
```

Figura 15. Arquivo XML representando o classificador gerado em um estágio de treinamento.

```
<_>
    <rects>
      <_>5 6 1 6 -1.</_>
      <_>5 9 1 3 2.</_>
    </rects>
    <tilted>0</tilted>
  </_>
<_>
```

Figura 16. Uma das características geradas pelo treinamento.

Os valores apresentados na Figura 14 e 15 são:

- *maxWeakCount*: parâmetro que é relativo a quantidade de classificadores que foram gerados no estágio de treinamento;
- *stageThreshold*: limiar do estágio;
- *weakClassifiers*: parâmetro que informa que as próximas *tags* serão a respeito dos classificadores fracos que foram gerados;
- *internalNodes*: representa os nós internos do classificador;
- *leafValues*: valores da folha do classificador;
- *rects*: parâmetro que define as coordenadas do retângulo que será aplicado na detecção bem como o peso associado a ele;
- *tilted*: define qual o tipo de retângulo de característica (Figura 5 e Figura 6) que foi utilizado. Quando o valor entre as *tags* <*tilted*> e </ *tilted*> for igual a 0, significa que foi utilizado um retângulo de característica simples (Figura 5), quando o valor for 1, significa, que foi utilizado um retângulo de característica rotacionada em 45 graus (Figura 6).

4.2 ALGORITMO DE DETECÇÃO

Um exemplo do algoritmo de detecção, que foi utilizado como base para o algoritmo implementado neste trabalho, é nativo da biblioteca OpenCV. Por padrão, o exemplo nativo é para detecção de rostos, mas alguns métodos podem ser reutilizados para detecção de outros objetos, como por exemplo, para a detecção de placas de trânsito deste trabalho.

O esquema de funcionamento, do algoritmo implementado neste trabalho pode ser observado na Figura 17.

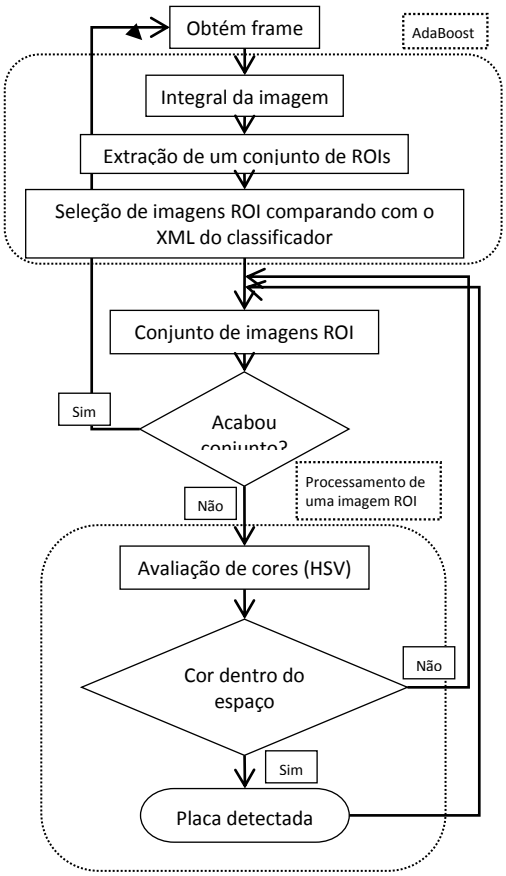

Figura 17. Esquema de funcionamento do algoritmo de detecção proposto.

Como se pode observar na Figura 17, a primeira etapa consiste em obter um *frame* do vídeo, então uma janela deslizante (*slide-window*) com tamanho previamente definido (padrão OpenCV para detecção de rostos, 24x24), é deslocada por todo o frame, e esta janela é incrementada de acordo com um

fator de escala (padrão de escala 1.1 definida no OpenCV), e a janela é incrementada até o tamanho máximo defino for alcançado. Quando o tamanho máximo é alcançado, significa que o frame todo foi processado e um conjunto de possíveis ROIs (*Region Of Interest*) foram extraídas por meio da Integral da Imagem. No próximo passo é realizada a seleção das ROIs. A seleção é feita, por comparação com as características gravadas no XML gerado pelo treinamento. Após a seleção têm-se um conjunto de imagens com possíveis ROI.

Com o conjunto de imagens ROI selecionado é realizado o processamento de cada uma das imagens do conjunto para eliminação dos possíveis falsos positivos, que também foram selecionados. Para diminuir a detecção de falsos positivos foi implementada a avaliação de cores no espaço HSV. Foram definidos três intervalos de cores, um para placas com predominância de cor branca, outro para a cor amarela e o terceiro para a cor vermelha. Se a ROI selecionada para processamento, estiver nos espaços de cores definidos, então a placa é detectada, caso contrário a ROI é descartada. O processo continua enquanto houver uma ROI para ser processada. Ao terminar o processamento do conjunto de ROI extraído do *frame*, o processo reinicia e um novo *frame* será processado, a execução continua enquanto houver *frames* de vídeo.

5 EXPERIMENTOS REALIZADOS

Esta seção apresenta dois experimentos realizados com a proposta apresentada neste trabalho.

Foi gravado um vídeo em um percurso de aproximadamente 4.000 metros, com uma câmera de vídeo JVC Full HD GZ-EX210BUB, fixada sobre o teto de um veículo, e transitando a uma velocidade média de 30 Km/h. O vídeo original foi convertido do formato MTS Full HD para AVI HD. Os experimentos foram realizados utilizando um computador com processador Intel core i5 de 3.1 GHz com 6 GB de memória RAM, o aplicativo foi executado de forma concorrente com outros processos utilizados pelo Sistema Operacional.

O primeiro experimento consistiu em aplicar o algoritmo desenvolvido em um conjunto de 3.915 imagens, extraídas do vídeo gravado com nove minutos de duração, e uma taxa de captura de 29 *frames* por segundo. Foram processados apenas um a cada quatro *frames*, ou seja, aproximadamente 7 *frames* por segundo. O fator de escala utilizado foi de 1.05, ou seja, a janela de detecção (*slide-window*), foi incrementada em cerca de 5%.

O segundo experimento foi realizado utilizando a mesma quantidade de imagens obtidas com os frames do vídeo, mas com o

fator de escala de 1.2 (aproximadamente 20%).

5.1 PRIMEIRO EXPERIMENTO

Este experimento foi realizado com o fator de escala de 1.05 (aproximadamente 5%), como descrito anteriormente. A Figura 18 mostra exemplos de recortes de imagens a partir dos *frames* de vídeo, com detecção de algumas placas.

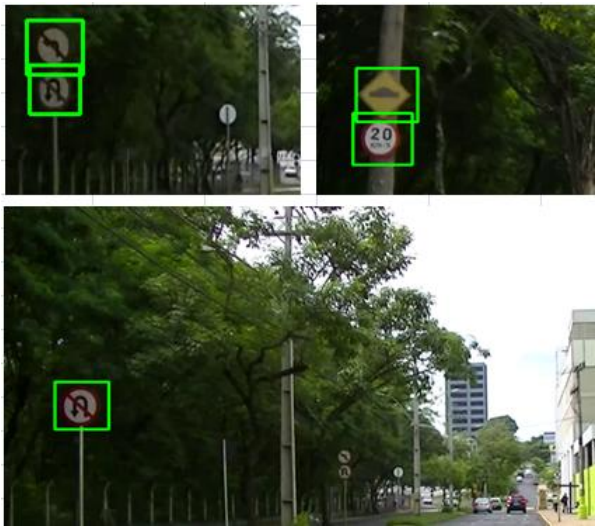

Figura 18. Exemplos de recortes dos *frames* com detecção de algumas placas.

Nos *frames* de vídeo processados, foi concluído que um total de 70 placas de trânsito aparecem nos mesmos, sendo a maioria, placas com formato circular. A Tabela 2 mostra as taxas de acertos e de erros durante o processo de detecção.

Tabela 2. Resultados do processo de detecção com fator de escala em 1.05.

	Acertos	Erros
Circular	90%	10%
Alerta	70%	30%
Pare	80%	20%
Dê Preferência	95%	5%

Neste experimento, foi possível observar, que mesmo com fator de escala menor que o padrão definido no OpenCV, foi possível alcançar altas taxas de detecção e em tempo expressivo. Neste experimento o tempo de processamento de um *frame* em média foi 530 ms (milissegundos).

Apesar das taxas de acertos e tempo de detecção serem expressivos, há também uma alta taxa de falsos positivos (regiões detectadas que não contém placa), mesmo com os filtros HSV não foi possível descartar todos os falsos positivos. A Figura 19 mostra dois falsos positivos detectados.

Figura 19. Detecção de falsos positivos.

5.2 SEGUNDO EXPERIMENTO

Neste experimento o fator de escala foi definido em 1.2 (aproximadamente 20%).

A Figura 20 mostra exemplos de detecção de algumas placas em *frames* de vídeo, utilizando os mesmos *frames* do primeiro experimento.

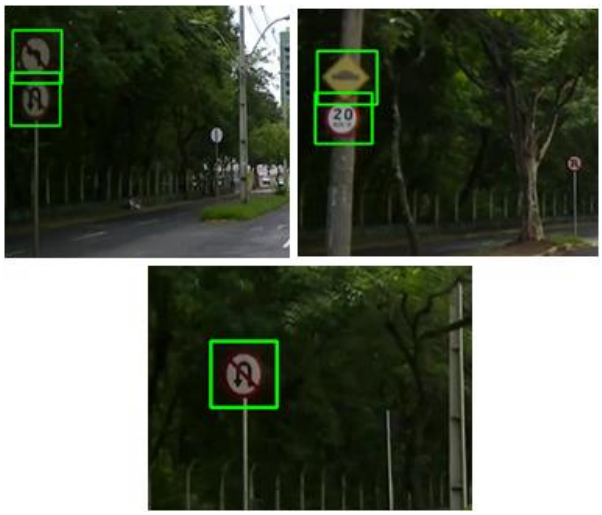

Figura 20. Exemplo de detecção de algumas placas em algumas imagens.

Para comparação dos resultados, foram utilizados a mesma quantidade de *frames* de vídeo usados no primeiro experimento. Portanto, com as mesmas 30 placas, foram obtidos os resultados apresentados na Tabela 3.

Tabela 3. Resultados do processo de detecção com fator de escala em 1.2.

	Acertos	**Erros**
Circular	70%	30%
Alerta	50%	50%
Pare	80%	20%
Dê Preferência	65%	35%

Após a realização deste experimento foi possível perceber a influência do fator de escala no processo de detecção. Comparando com o experimento anterior, observa-se que as placas somente são detectadas quando apresentam tamanhos grandes na imagem, sendo que placas muito ao fundo (pequenas na imagem) não são detectadas.

Como a janela de detecção foi incrementada em cerca de 20%, o tempo total de processamento por *frame* diminuiu drasticamente, em média o tempo total por *frame* caiu de 530 ms (milissegundos) para 190 ms (milissegundos). Apesar da taxa de acertos ter diminuído para algumas placas, com o fator de escala em 1.2, pode-se observar que houve uma grande diminuição no tempo de processamento do *frame*. Também se pode observar na Tabela 3, que com exceção da placa de pare, todas as outras placa foram afetadas negativamente por esse fator de escala.

Acredita-se que uma melhoria na etapa de treinamento, provavelmente resolveria essa queda na taxa de acertos. As placas circulares, que são maioria no vídeo utilizado para os experimentos obtiveram uma taxa de acertos expressivos, maior do que a taxa obtida no primeiro experimento. O mesmo pode ser observado com as placas de "pare" e as de "dê a preferência" (triangulares). Houve ainda uma redução na taxa de falsos positivos, ou seja, aumentando o fator de escala, é possível obter menos falsos positivos, e com isso o processamento se torna mais rápido.

6 CONCLUSÕES

A indústria automobilística nos últimos anos está apresentando veículos cada vez mais automatizados, consequentemente a tecnologia torna-se mais presente. A construção de veículos autônomos já não é mais ficção científica, fazendo com que a Visão Computacional seja indispensável para tais veículos. Com um método para detectar placas em um tempo aceitável, o processo para reconhecimento não despenderia processamento com regiões irrelevantes, seria aplicado apenas na região onde foi detectada a placa.

Assim a adaptação do método utilizado por Viola e Jones (2001), para detecção de placas, mostrou-se totalmente possível, e os resultados obtidos com a metodologia proposta neste trabalho são expressivos, além de apresentar uma nova maneira de realizar o treinamento, para eliminação de várias classes de sinais. Comparando com os trabalhos de Brkić (2010), Matinović (2010), Landeza-Vázquez, Parada-Loira e Alba-Castro (2010) e Timofte, Zimmermann e Van Gool (2011), que realizam a detecção de placas de um ou dois formatos específicos com resultados em tempo real, o presente trabalho realizou a detecção de quatro formatos de placas em um tempo relativamente aceitável, com altas taxas de acertos e com o diferencial em como foi realizada a etapa de treinamento.

Observou-se, após inúmeras tentativas, que o treinamento é o ponto vital para um bom resultado no processo de detecção. Com isso, destaca-se que todo o processo deve ter atenção redobrada, desde a captura de imagens para o treinamento até a gravação do vídeo para testes, pois o algoritmo é altamente sensível à iluminação e diferentes pontos de vista das placas.

Os experimentos realizados evidenciam a grande influência do fator de escala no processo de detecção, portanto, experimentos com outros fatores de escala são necessários para uma avaliação mais ampla de qual seria o fator de escala adequado para aplicar no processo de detecção de placas de trânsito.

O filtro HSV utilizado no algoritmo de detecção mostrou-se eficiente para algumas cores, mas para abranger toda a variação de iluminação que uma cor pode apresentar, torna-se difícil definir o espaço correto. Um método mais eficaz diminuiria a taxa de falsos positivos.

Por fim, acredita-se que, com os resultados obtidos no âmbito deste trabalho, que o método desenvolvido é viável para trabalhar com algum algoritmo de reconhecimento, em uma segunda etapa, podendo atuar em tempo real.

REFERÊNCIAS

ACHARYA, T.; RAY, A. K. **Image processing:** principles and applications. Hoboken: John Wiley & Sons, 2005. 428 p.

BARÓ, X. et al. Traffic sign recognition using evolutlionary adaboost detection and forest-ECOC classification. In: IEEE TRANS. ON INTELLIGENT TRANSPORTATION SYSTEMS. **Proceedings…** 2009. p. 113-126.

BRKIĆ, K. **An overview of traffic sign detection methods.** 2010. Department of Electronics, Microeletronics, Computer and Intelligent Systems - Faculty of Electrical Engineering and Computing Unska 3, Zagreb, Croatia.

BRKIĆ, K.; PINZ, A.; SEGVIĆ, S. Traffic sign detection as a component of a an automated traffic infrastructure inventory System. In: IEEE INTERNATIONAL CONFERENCE INTELLIGENT TRANSPORTATION SYSTEM (ITSC), 13. **Proceedings...** 2010.

CHAVES, B. B. **Estudo do algoritmo AdaBoost de aprendizagem de máquina aplicado a sensores e sistemas embarcados**. 2012. 119f. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, São Paulo, SP.

CHEN, S.; HSIEH, J. Boosted road sign detection and recognition. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND CYBERNETICS. **Proceedings…** July 2008. v. 7, p. 3823–3826.

FREUND, Y.; SCHAPIRE, R. E. A Short introduction to booting. **Journal of Japanese Society for Artificial Intelligence**, p. 771-780, 1999.

FREUND, Y.; SCHAPIRE, R. E. Experiments with a new boosting algorithm. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 13. **Proceedings...** 1996. p. 148-156.

LANDEZA-VÁZQUEZ, I.; PARADA-LOIRA, F.; ALBA-CASTRO, J. Fast real-time multiclass traffic sign detection based on novel shape and texture descriptors. In: ANNUAL CONFERENCE ON INTELLIGENT TRANSPORTATION SYSTEMS. **Proceedings…** Madeira Island, Portugal, September 19-22, 2010.

LIENHART, R.; MAYDT, J. An extended set of Haar-like features for rapid object detection. **IEEE ICIP**, p. 900–903, 2002.

MARTINOVIĆ, A. et al. Real-time detection and recognition of traffic signs. In: INTERNATION CONVENTION ON INFORMATION AND COMMUNICATION TECHNOLOGY, ELECTRONICS AND MICROELETRONICS – MIPRO 2010, 33. **Proceedings...** 2010. v. III, p. 247-252.

RAMOS, G. A. **Detecção e rastreamento de lábios em dispositivos móveis**. 2012. 60 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, SP.

TIMOFTE, R.; ZIMMERMANN, K.; VAN GOOL, L. Multi-view traffic sign detection, recognition and 3D localization. **Machine Vision and Aplications Manuscript**, Sep., 2011.

VIOLA, P.; JONES, M. Robust real-time object detection. In: INTERNATIONAL WORKSHOP ON STATISTICAL AND COMPUTATIONAL THEORIES OF VISION-MODELING, LEARNING, COMPUTING AND SAMPLING, 2. **Proceedings…** Vancouver, Canada, 2001.