Rastreamento e Identificação de Veículos através das técnicas de Redes Neurais e de Subtração e Reconstrução Dinâmica de Fundo

Bruno Clemente Guingo[1]
FESO – Fundação Educacional Serra dos Órgãos
Graduação em Tecnologia em Processamento de Dados
Av Alberto Torres, 111, 25964000 Teresópolis, RJ
BRASIL

Thiago da Silva Pontes[2]
UFRJ –Universidade Federal do Rio de Janeiro
Departamento de Ciência da Computação
Caixa Postal 2324, 20001970 Rio de Janeiro, RJ
BRASIL

Antonio Carlos Gay Thomé[3]
UFRJ –Universidade Federal do Rio de Janeiro
Área de Ensino e Pequisa, NCE/IM
Caixa Postal 2324, 20001970 Rio de Janeiro, RJ
BRASIL
bruno.guingo@ufrj.br[1], thiago.pontes@ufrj.br[2], thome@nce.ufrj.br[3]

Abstract

Identifying moving objects in a video sequence is fundamental in many computer-vision applications, such as traffic automatic monitoring. In this paper we describe a research performed on a real time application on tracking and identification a flow of moving vehicles in a urban traffic. The objective is to identify when a vehicle emerges on the monitored scenario and then tracks its movement along the visual camp of the video camera and to make its identifications. The proposed approach is based on background subtraction and reconstruction and neural networks.

1. Introdução

Existem vários desafios na concepção de um bom algoritmo de subtração de fundo. Este deve ser robusto contra mudanças na iluminação; deve evitar a detecção de objetos não estacionários que pertencem ao fundo; e precisa reagir rápido a mudanças no fundo, como veículos parando ou iniciando movimento [1].

No caso de aplicações de monitoramento de tráfego, estes desafios incluem, por exemplo, condições de tempo como chuva, nevoeiro, neve, reflexos, sombras e a velocidade do veículo. A variação da velocidade é pequena quando o semáforo está verde, e aumenta muito quando o semáforo fica em vermelho.

Monitoramento de tráfego e reconhecimento automático de placas está se tornando uma tarefa muito importante para a engenharia de controle de tráfego. Apesar de sua importância, a maioria dos sistemas ainda faz uso de um sensor de presença para ativar a captura da imagem e iniciar a procura e reconhecimento automático da placa [2, 3]. Neste artigo é apresentada uma abordagem que não necessita do sensor de presença.

No método proposto, a imagem é continuamente capturada e enviada ao algoritmo de rastreamento que executa uma procura por objetos em movimento e automaticamente verifica se o objeto detectado pode ou não ser classificado como a silhueta de um veículo.

Este texto cobre alguns passos básicos da captura da imagem; a subtração de cada quadro capturado aplicada ao fundo; a escolha e reconstrução dinâmica do quadro de referência; a detecção do objeto em movimento, a sua classificação como veículo ou não e o reconhecimento de sua placa.

2. Captura da Imagem

A imagem é capturada de forma contínua por uma câmera de vídeo. A taxa de captura deve ser a mais alta possível, pelo menos igual ou superior a 15 quadros por segundo. Deve-se levar em consideração a velocidade

mínima e máxima esperada dos veículos em movimento por aquele trecho da via.

As imagens utilizadas neste estudo foram capturadas na Rua Bartolomeu Mitre, próximo ao 23º Batalhão de Polícia Militar, no Bairro do Leblon, na cidade do Rio de Janeiro. Foi utilizada uma câmera em cores, de alta resolução, instalada em um poste próximo ao meio fio a uma altura de aproximadamente 3 metros. As imagens foram capturadas a uma taxa de 15 quadros por segundo, com uma resolução de 320x240 pixels, 24 bits de cor e formato BMP.

Os veículos utilizados neste estudo são carros de passeio, ônibus, caminhões e utilitários de diversas marcas, cores e modelos, que passavam pelo local em velocidade variável entre 10 km/h e 80 km/h.

3. Algoritmo de Rastreamento

O algoritmo de rastreamento é composto por um conjunto de filtros e funções, divididos em módulos, onde cada um tem sua funcionalidade específica. As imagens, à medida que vão sendo adquiridas pelo algoritmo de captura, vão sendo colocadas em uma fila FIFO que serve de entrada para o algoritmo de rastreamento.

O primeiro quadro, a contar do instante inicial de funcionamento do programa, é tomado como quadro de referência ou de fundo. A partir daí, em um processamento cíclico, cada novo quadro contido na fila FIFO é comparado com o quadro de referência. Os objetos em movimento no cenário são detectados e aqueles que foram considerados similares com a silhueta de um veículo são selecionados.

A fim evitar o excesso da fila do FIFO, o algoritmo rejeita alguns quadros capturados até o ponto onde ele identifica uma silhueta de um veículo. No fim de cada ciclo de processando, o quadro de referência é atualizado. Sempre que for detectado um veículo em movimento pela primeira vez, o algoritmo marca a borda anteriror e entra na fase de acompanhamento e busca da borda posterior.

3.1. Comparação entre os quadros

A imagem rastreada é subtraída continuamente do frame atual com o fundo. Para a realização desta tarefa foi desenvolvida a função dif (x, y).

$$imdif_{i,j} = \begin{cases} 1, \textit{ se } dif(fa_{i,j}, fd_{i,j}) \geq \lambda \\ 0, \textit{ caso contrário} \end{cases}$$

onde,

$imdif_{i,j}$ = a cada ponto da matriz diferença;

$fa_{i,j}$ = representa a posição *i,j* do quadro atual;

$fd_{i,j}$ = representa a posição *i,j* no quadro de fundo;

O limiar λ é o limite mínimo que deve ser alcançado para que a diferença seja considerada. A função *dif(x,y)* depende de um outro limiar α, usado para indicar a luminosidade relativa, ou seja, se o pixel tem luminosidade alta ou baixa em relação a luminosidade média do fundo.

Dependendo do valor de α, classifica-se o pixel como sendo de luminosidade baixa ou alta em relação ao fundo e aplica-se uma estratégia diferenciada para calcular a função *dif(x,y)*. Esta classificação visa tratar adequadamente imagens claras e escuras.

Para luminosidades consideradas baixas, a diferença é calculada com base na seguinte equação:

$$dif(C_{i,j}, B_{i,j}) = \max(r_{i,j}, g_{i,j}, b_{i,j})$$

onde,

$$r_{i,j} = \left|(B_{i,j})_R - (C_{i,j})_R\right|$$

$$g_{i,j} = \left|(C_{i,j})_G - (B_{i,j})_G\right|$$

$$b_{i,j} = \left|(C_{i,j})_B - (B_{i,j})_B\right|$$

Para luminosidades consideradas altas, a diferença é calculada com base no seguinte procedimento:

$$d1_{i,j} = \left|r_{i,j} - g_{i,j}\right|$$

$$d2_{i,j} = \left|r_{i,j} - b_{i,j}\right|$$

$$d3_{i,j} = \left|b_{i,j} - g_{i,j}\right|$$

$$\sigma = \frac{lum(fa_{i,j})}{\alpha}$$

E, finalmente:

$$dif(C_{i,j}, B_{i,j}) = (d1_{i,j} + d2_{i,j} + d3_{i,j}).\sigma$$

Na comparação de quadros para detecção de objetos em movimento geralmente se depara com a incorporação indesejada de inúmeros pixels que não pertencem ao objeto em movimento e que deveriam ser classificados como ruído. Na figura 1 a seguir pode-se comparar o resultado obtido com uma estratégia convencional e o obtido com a estratégia acima exposta.

(a)

(b)

Fig.1. Detecção de movimento, (a) algoritmo convencional; (b) algoritmo proposto.

3.2. Coeficiente de luminosidade relativa

A finalidade da luminosidade relativa é garantir robustez ao algoritmo, permitindo que ele funcione sob diferentes condições de iluminação. O coeficiente é dado pela expressão abaixo, onde a luminosidade de cada pixel é normalizado pela média da luminosidade do fundo.

$$\alpha = \frac{lum(C_{i,j}).m.n}{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(lum(B_{i,j}))}$$

onde,

m e n = são as dimensões do quadro;

lum(x) = representa a luminosidade (nível de cinza) de um determinado pixel.

Existem também ruídos microscópicos (figura 2) caracterizados por pontos isolados que não correspondem a movimento, mas que podem ser considerados como tal. Para eliminá-los aplica-se um filtro de erosão seguido de uma dilatação [4, 5].

Fig.2. Ruído microscópico no canto inferior direito do quadro diferença.

3.3. Filtro de erosão

O filtro da erosão [4, 5] considera os pixels vizinhos pixels com um pixel atual. A vizinhança é computada baseada num raio r previamente estabelecido. Quanto maior for o valor de ***r***, maior será o número de pixels eliminados. A escolha do valor de ***r*** deve ser feita com cuidado, pois se ele for pequeno demais, pode-se deixar de eliminar muito ruído, e se for demasiadamente grande corre-se o risco de eliminar partes do objeto alvo.

Em termos práticos, tem-se uma matriz binária de tamanho 320x240 e, no pior caso, tem-se que verificar a vizinhança de todos os 76800 pontos na matriz diferença. A figura 3 representa o pixel corrente (em preto) e sua vizinhança para um raio ***r = 3***.

1	1	1	0	0	1	1
1	1	0	0	1	1	1
0	0	0	0	0	0	1
0	0	1	1	1	0	0
0	0	1	1	1	1	0
0	1	1	1	1	1	1
0	0	1	0	0	1	0

Fig.3. Matriz Binária com vizinhança de *r = 3*.

O procedimento da erosão neste caso faz o seguinte: se um ponto possuir uma densidade suficientemente grande (acima de um ponto específico), ele permanecerá marcado na matriz diferença, senão, será desmarcado, isto é, recebe valor zero. Todos os pontos na matriz da diferença devem passar por esta operação, o que significa que para cada pixel no frame, o algoritmo deve computar $(2*r+1)^2-1$ verificações. Assim, para um ***r = 3***, poderá ser feito um total de 3.686.400 operações de verificação $(48*320*240)$. Para uma taxa de captura de 15fps, tem-se um total de 55.296.000 verificações por segundo, um grau de processamento muito elevado para uma aplicação que de tempo real.

Se a taxa da captura for igual a 15fps, então a quantidade total de operações a serem feitas pelo algoritmo é de 55.296.000 verificações por segundo, que é demasiadamente elevado para uma aplicação real-time.

Um raio de três unidades proporciona um resultado final considerado bom, porém é impraticável considerando-se o custo computacional. Para evitar esse processamento excessivo foram escolhidos alguns pontos estratégicos da vizinhança, diminuindo-se o número de verificações em mais de 10 vezes. A perda em termos de qualidade foi desprezível.

Primeiro calcula-se a densidade da vizinhança do pixel corrente representada por $\rho v_{i,j}$.

$$\rho v_{i,j} = \frac{\sum_{i=x-r}^{x+r}\sum_{j=y-r}^{y+r} imdif_{i,j} - 1}{(2.r+1)^2 - 1}$$

A seguir seleciona-se os pixels que permanecerão marcados com base na fórmula:

$$imdif_{i,j} = \begin{cases} 1, \, se \; \rho v_{i,j} \geq \beta \\ 0, \, caso \; contrário \end{cases}$$

onde,

β = representa a densidade mínima aceitável.

A figura 4 mostra a execução do filtro de erosão, observe que além de eliminar o ruído indesejável ele também elimina parte do objeto alvo. Este fato torna necessário o resgate de pontos cuja falta pode causar falha na detecção e, para resolver este problema, utilizou-se uma dilatação aplicada sobre o resultado da erosão.

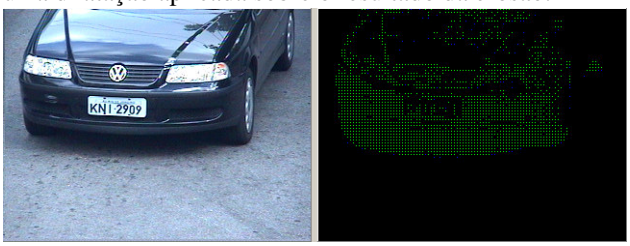


Fig.4. Filtro de erosão aplicado na matriz diferença.

Após a aplicação dos filtros de erosão e dilatação, espera-se que a matriz diferença destaque somente os pontos pertencentes ao objeto alvo em movimento. Assim, pode-se começar a busca pelas bordas do objeto.

3.4. Demarcação das bordas

Para identificar onde se dá o início e o fim de cada veículo, faz-se necessário demarcar as bordas anterior e posterior do veículo na imagem diferença. Essas bordas ou limites são encontrados através da projeção vertical da matriz diferença. A figura 5 mostra um exemplo dessa projeção.

0	0	0	0	0	0	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1	2
0	0	0	0	1	1	1	1	1	1	0	0	6
0	0	0	1	1	1	1	1	1	1	0	0	7
0	0	1	1	1	1	1	1	1	1	0	0	8
0	0	1	1	1	1	1	1	1	1	0	0	8
0	1	1	1	1	1	1	1	1	0	0	0	8
1	1	1	1	1	1	1	1	0	0	0	0	8
0	1	1	1	1	1	1	0	0	0	0	0	6
0	0	0	0	0	0	0	0	0	0	0	1	1

Fig.5. Matriz diferença e a sua projeção horizontal.

Fixando um limiar pode-se a partir desta projeção identificar as bordas anterior e posterior do veículo. A projeção não é feita sobre o eixo ortogonal e sim sobre uma linha paralela ao sentido do fluxo dos veículos no cenário considerado.

Na figura 6 as linhas verde e vermelha representam respectivamente as bordas anterior e posterior do primeiro veículo e as linhas azul e amarela representam o mesmo para o segundo veículo.

A imagem direita mostra a linha da projeção que está paralela ao sentido do fluxo dos veículos.

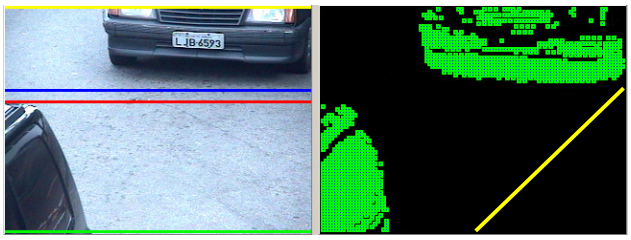


Fig.6. Delimitação das bordas anterior e posterior de um veículo.

3.5. Reconstrução dinâmica de fundo

Este passo é importante pelo fato de que a primeira parte do algoritmo pode não funcionar corretamente, em função da falta de uma atualização adequada do quadro de referência (fundo). O algoritmo de reconstrução do fundo se baseia na premissa de que diferenças significativas entre o quadro atual e o quadro de fundo representam movimento.

O quadro de referência pode ser um cenário específico, tomado em situação especial, ou pode ser adotado como o primeiro quadro adquirido pelo sistema toda vez que este é re-inicializado. De qualquer forma este quadro precisa ser constantemente atualizado para refletir, de forma adequada, as mudanças que por ventura ocorram no cenário no decorrer do tempo. Esta atualização, feita em cada componente de cor de cada pixel do fundo, é dada por:

$$(B_{i,j})_r = (B_{i,j})_r + [(C_{i,j})_r - (B_{i,j})_r] * \partial$$

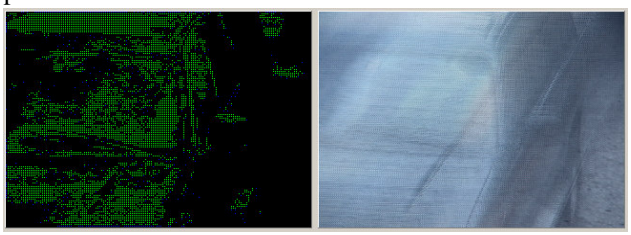
$$(B_{i,j})_g = (B_{i,j})_g + [(C_{i,j})_g - (B_{i,j})_g] * \partial$$

$$(B_{i,j})_b = (B_{i,j})_b + [(C_{i,j})_b - (B_{i,j})_b] * \partial$$

Onde,

δ - taxa de atualização.

Quanto maior o valor de δ, mais rápida será a s mudanças incorporadas ao fundo. A figura 7 apresenta um exemplo da reconstrução do fundo enquanto o veículo diminui sua velocidade e um exemplo do fundo reconstruído após alguns segundos após o veículo já estar parado.



(a)

(b)

Fig.7. Reconstrução do fundo, (a) veículo diminuindo a velocidade, (b) veículo parado.

4. Reconhecimento das Placas dos Veículos

Todo processo de reconhecimento está pautado nos algoritmos implementados no sistema *Kapta* [4], que entra em ação logo após a identificação da silhueta do veículo seja feito pelo algoritmo de rastreamento. Os quadros que contem a imagem dos veículos são passados para este módulo, e então é feito o reconhecimento e validação dos caracteres que compõem a placa do veículo. Este módulo é subdivido em quatro submódulos.

4.1. Localização da Placa

É o submódulo responsável por encontrar a região da placa. Para a execução desta tarefa, são aplicados algoritmos de processamento de imagens [2]. Completada a localização, a região definida como a placa é extraída e passada para a fase seguinte, a segmentação.

4.2. Segmentação

É responsável por separar um caractere do outro e, idealmente, criar sete subimagens cada uma contendo apenas um dos caracteres que compõem a placa [2].

4.3. Extração de Características

É responsável por extrair de cada caractere segmentado, informações que permitam que o mesmo seja mais facilmente classificado pelo submódulo de reconhecimento [2].

4.4. Reconhecimento

É responsável por reconhecer cada caractere que compõe a placa, através das informações disponibilizadas pelo módulo de extração das características.

Os processos de localização e de segmentação têm forte impacto sobre o desempenho do processo de reconhecimento. Tanto o erro como pequenas falhas ocorridas na localização e na segmentação são propagados para o reconhecimento, causando com isso uma queda no desempenho do sistema.

Para o reconhecimento faz-se uso do paradigma de redes neurais. Duas configurações distintas de redes neurais são utilizadas, uma para as letras e outra para os dígitos. Os três primeiros vetores de características são encaminhados para a rede de reconhecimento de letras, e os quatro vetores restantes para a de reconhecimento de dígitos.

O modelo de rede neural adotado para o reconhecimento dos caracteres foi o *Multi-Layer Perceptron - MLP* [9, 10], treinado com o algoritmo "*backpropagation*". O software utilizado para a simulação foi o *MatLab* versão 5.2 [7] e versão 6.0 [8]. A função de ativação faz uso de um somatório simples das entradas ajustadas pelos seus respectivos pesos. Como função de propagação, utilizou-se a função logística *sigmoidal* ("*logsig*" no Matlab), tanto na camada escondida quanto na camada de saída.

O treinamento da rede foi realizado usando a opção "*traingdx*" com momento e taxa de aprendizado adaptativa. A entrada da rede para cada placa é composta por uma matriz de tamanho λx7, onde o λ depende do número de características extraídas. Os três primeiros vetores desta matriz são dirigidos para uma rede especializada em letras e os quatros restantes para uma especializada em dígitos.

A saída para cada caractere foi configurada como um vetor ortogonal de dimensão 10 para os dígitos e 26 para as letras.

Para normalização dos dados de entrada foi adotada a técnica "Z-score" [11]. Esta técnica apresenta como resultado o número de desvios-padrão que cada elemento do conjunto de dados está distante da média do conjunto. A equação abaixo mostra como é feito este cálculo.

$$p_n = \frac{p - \overline{p}}{\sigma_p}$$

onde,

p_n = *vetor de entrada normalizad o*

p = *vetor de entrada*

$\overline{p}$ = *média dos vetores de entrada*

σ_p = *desvio − padrão dos vetores e entrada*

Também é feito o uso da técnica de PCA com fator de corte de 0,001, com o objetivo de reduzir a dimensionalidade dos vetores de características.

5. Resultados Obtidos

Os testes foram realizados com uma base de imagens com 130 conjuntos distintos em formato "AVI". Os vídeos têm durações diferentes: pequena (aproximadamente 1 minuto de vídeo), média (aproximadamente 3 minutos de vídeo) e alta (aproximadamente 5 minutos de vídeo).

O algoritmo trabalhou bem em todos os casos, isto é, os veículos foram corretamente rastreados e identificados. Tanto a borda dianteira quanto a posterior, foram encontrados com correção pelo algoritmo.

6. Conclusões

Toda a pesquisa foi realizada no âmbito do Laboratório de Inteligência Computacional (LABIC) da Universidade Federal do Rio de Janeiro (UFRJ). As estratégias descritas neste artigo vêm sendo constantemente aperfeiçoadas à medida que novos desafios são enfrentados.

As aplicações em que esta estratégia de rastreamento pode ser inserida são inúmeras, não só em função da sua rapidez de processamento, mas também, pela flexibilidade fornecida pela eliminação da necessidade de uso de um sensor de presença nos sistemas de monitoração de tráfego.

7. Referências

[1] S. C. Cheung and C. Kamath. "*Robust techniques for background subtraction in urban traffic video*". Proceedings of Electronic Imaging: Visual Communications and Image Processing 2004 (Part One), January 20-22 2004, San Jose, California. Bellingham, WA:SPIE. (5308):881-892.

[2] B. C. Guingo. "*Reconhecimento Automático de Placas de Veículos Automotores*". Dissertação de mestrado. Universidade Federal do Rio de Janeiro. Rio de Janeiro-RJ, 2003.

[3] Empresa Automatisa Ltda. Disponível na internet via http://www.automatisa.com.br/siav2.htm

[4] B. C. Guingo, G. M. Stiebler and A. C. G. Thomé. "*Kapta – Um Sistema de Reconhecimento Automático de Placas de Veículos baseado nas Técnicas de Redes Neurais e Processamento de Imagens*". Congresso Brasileiro de Tecnologia da Informação e Comunicação – SUCESU2004, Florianópolis-SC, 2004.

[5] R. C. Gonzalez and R. E. Woods, *Processamento de Imagens Digitais.* Editora Edgard Blucher Ltda., 2000.

[6] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice Hall Information and System Sciences Series, 1989.

[7] H. Demuth and M. Beale, *Neural Network Toolbox for Use with Matlab: User's Guide Version 3.0*. Copyright 1992 - 1997 by The MathWorks, Inc.

[8] Guide is MATLAB6.0, Neural Network Toolbox 4.0 Release Notes. Chapter 32.

[9] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[11] R. A. Johnson and WICHERN, D. W. Wichern, *Applied Multivariate Statistical Analysis*. 3 ed. Prentice-Hall International, Inc., 1992.