Estudo e implementação de algoritmos para contagem de veículos

Henrique Domiciano Osinski, Sibilla B. da Luz França
Departamento de Engenharia Elétrica UFPR, Curitiba, Brasil
henriquedomiciano@yahoo.com

Resumo*—O estudo e desenvolvimento de métodos de reconhecimento e contagem de objetos possibilitam a obtenção de dados de maneira automatizada para utilização em uma série de aplicações. Este artigo apresenta um estudo sobre diferentes métodos de processamento de imagem com o objetivo de implementar, em software e posteriormente em hardware, algoritmos capazes de identificar e contar veículos a partir de imagens e vídeos. Os resultados obtidos foram a implementação de três algoritmos de contagem de veículos, em software, obtendo uma precisão máxima de 88%. Foi construído, em hardware, um procedimento para leitura de imagens e alguns métodos de processamento de imagem. Tendo como objetivo, futuramente, sua utilização na implementação dos algoritmos em hardware.

I. INTRODUÇÃO

O controle inteligente de trânsito tem sido um tema de grande interesse nos últimos anos. Algoritmos baseados em aprendizado de máquina estão sendo utilizados com o objetivo de melhorar o fluxo de veículos nas vias, reduzindo o tempo gasto nos deslocamentos. Para realização do controle trânsito é necessário a obtenção de informações sobre a quantidade de veículos em determinadas vias [1]. A automação na aquisição destes dados possibilita maior acurácia e qualidade nos modelos de previsão e controle do fluxo de veículos. Através de métodos de visão computacional é possível realizar a contagem de veículos utilizando vídeos registrados da via. A implementação dos métodos de visão computacional em FPGA (*Field Programable Gate Array*) é vantajosa devido sua característica de paralelismo, permitindo que diversas tarefas sejam realizadas simultaneamente.

II. PROCESSAMENTO DE IMAGENS

Imagens digitais são constituídas por elementos de imagem, mais conhecidos como pixel. Existem diferentes sistemas para a apresentação de imagens com cor. O mais utilizado é o sistema RGB, formado por três canais de cor, Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*). Imagens são formadas a partir da sobreposição destes canais. Uma alternativa ao uso de sistemas de cores, é o de escala de cinza, que apresenta imagens em cinza, utilizando somente um canal. As diferenças entre o uso de imagens em escala de cinza e de cor é a necessidade de obtenção e tratamento de dados, as imagens em escala de cinza possibilitam o uso de três vezes menos informação.

De acordo com [2], o processamento de imagem pode ser dividido em três níveis diferentes. O nível baixo que faz transformações em um pixel único, como transformações em escala de cores e de contraste. O nível médio de processamento que utiliza regiões de pixels para fazer a transformação de um pixel e salientar ou obter regiões da imagem. E, finalmente, o alto nível que contempla a visão computacional, fazendo a contagem e classificação de imagens.

Para construir os algoritmos capazes de contar o número de veículos é necessário aplicar filtros de baixo e médio nível em imagens, para então poder estruturar métodos de alto nível.

A. Metodos de processamento de baixo nivel

O filtro de *trheshold*, ou de limiarização [3] é um filtro muito utilizado. Este filtro é aplicado a partir da equação 1

$$c_2(x,y) = \begin{cases} 255 \; se \; c(x,y) > threshold \\ 0 \text{ se } c(x,y) \leq threshold \end{cases} \quad (1)$$

sendo $c_2(x,y)$ o pixel da imagem de saída no ponto (x,y), o *threshold* o valor determinado para a binarização da imagem e $c(x,y)$ a imagem original. Existem diferentes métodos para a obtenção do valor ótimo para o *threshold* um dos métodos mais utilizados é o de Otsu [4].

Outro filtro importante de baixo nível é o de inversão que é utilizado para facilitar a retirada de informações de imagens com fundo escuro [2]. O funcionamento deste filtro é dado pela equação 2

$$c_2(x,y) = 255 - c(x,y) \quad (2)$$

sendo $c_2(x,y)$ o pixel da imagem de saída no ponto (x,y) e $c(x,y)$ a imagem original.

Um método de processamento de baixo nível que é utilizado no processamento em vídeos é a subtração de

fundo, ou a subtração de imagens em diferentes frames, para a obtenção dos objetos em movimento.

B.Metodos de processamento de médio nível

Todos os métodos de processamento de imagem de médio nível utilizam uma operação denominada convolução. A convolução, no contexto do processamento de imagens é dada como resultado da equação 3 e sendo representada pela equação 4

$$r_{\frac{u+1}{2},\frac{l+1}{2}} = \sum_{\substack{0<i\leq n \\ 0<j\leq n}} k_{i,j} \sum_{\substack{u<f\leq u+n \\ l<g\leq l+n}} v_{f,g} \tag{3}$$

$$R = K * V \tag{4}$$

Na equação 3, *r* representa o elemento da matriz resultado, *k* o elemento da máscara de convolução matricial, *v* o pixel da imagem, *u* e *l* representam, respectivamente, a linha e coluna da posição inicial do processo de convolução e *n* simboliza a ordem da máscara matricial de convolução. Na equação 4, *R* simboliza a imagem de saída, *K* a máscara de convolução e *V* a imagem original.

As máscaras, que são matriciais, determinam quais são as transformações ocorridas em uma imagem de saída. Um destes processos de tratamento de imagens é o uso do filtro de média (equação 5), que calcula a média dos pixels ao redor do pixel central

$$K = \frac{1}{N^2}\begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{N^2} & \cdots & \frac{1}{N^2} \\ \vdots & \ddots & \vdots \\ \frac{1}{N^2} & \cdots & \frac{1}{N^2} \end{pmatrix} \tag{5}$$

sendo *N* a ordem da matriz e *K* a máscara do filtro de convolução.

Outro filtro de grande importância é o filtro Sobel, para obtenção de bordas, na direção *x* (equação 6) e na direção *y* (equação 7)

$$M_x = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{6}$$

$$M_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{7}$$

onde M_x é a máscara de convolução do filtro Sobel na direção x e M_y a máscara de convolução do filtro Sobel na direção *y*. Para a apresentação de uma imagem com ambas as bordas, na direção horizontal e vertical, aplica-se a equação 8

$$C(x,y) = \sqrt{(S_x(x,y))^2 + (S_y(x,y))^2} \tag{8}$$

sendo $C(x,y)$ o pixel da imagem de saída e $S_x(x,y)$ o pixel da imagem aplicado o filtro Sobel na direção x, $S_y(x,y)$ o pixel da imagem aplicado o filtro Sobel na direção y, todos nos pontos (x,y).

C.Metodos de processamento de Alto nivel

Um dos algoritmos de processamento de alto nível utilizados no desenvolvimento deste trabalho é o de Haar Cascades, que consiste na construção de máscaras de comparação entre regiões da imagem. Este algoritmo faz subtrações das regiões de máscara, e caso o resultado desta subtração seja positivo o algoritmo passa para a próxima máscara, caso contrário, passa para a próxima região da imagem. As máscaras de comparação são construídas a partir de aprendizado de máquina, utilizando uma grande quantidade de imagens positivas e negativas dos objetos [5].

Outro algoritmo de processamento de alto nível também utilizado neste trabalho é o Blob Detection. Este algoritmo robusto classifica e encontra grupos de pixels em uma imagem e constrói grupos de pixels, denominados *Blobs*, a partir de um determinado valor de distância pré-determinado. O algoritmo encontra o primeiro pixel em nível alto e em seguida encontra a distância entre esse pixel com o próximo de valor alto, ou com os outros *Blobs*. Se o pixel está dentro daquela distância, realoca a nova posição daquele *Blob* para a posição do pixel, caso contrário é construído mais um *Blob* e o número de objetos é aumentado. [6]. Quando o algoritmo de Blob Detection é aplicado é necessária a utilização de métodos de baixo e médio processamento de imagens. No uso de vídeos o melhor resultado é obtido com a subtração de imagens.

O algoritmo de Feitosa [7], também usado neste trabalho, utiliza o acompanhamento da região de movimentação do veículo. Este algoritmo utiliza filtros de média e baixa intensidade para construir uma imagem de fundo. Inicialmente, o método utiliza um histórico das imagens de um vídeo e faz a média dos valores de pixels. A partir destas médias obtém uma imagem de fundo para ser subtraída da imagem atual do vídeo. Então, um *threshold* é aplicado nas imagens. Estas imagens são somadas, formando assim uma região de interesse. Logo que esta região de interesse é formada os objetos dentro desta região são segmentados, utilizando o filtro de detecção de bordas. Em seguida, ocorre a localização dos veículos e a contagem a partir da apresentação do número de objetos segmentados.

III. DESENVOLVIMENTO E RESULTADOS

O trabalho foi inicialmente desenvolvido em software, utilizando na linguagem Python. Primeiramente realizou-se a implementação de filtros de baixo nível. Em seguida, foram implementados filtros de médio nível para então, iniciar o processamento de alto nível. Após o desenvolvimento em software, iniciou-se a implementação dos métodos de baixo e médio nível em hardware, utilizando a linguagem de descrição de hardware VHDL.

Existem diversos métodos para a contagem de objetos utilizando imagens. No escopo deste trabalho foram selecionados três métodos. O método de Haar Cascades, foi selecionado pelos resultados sólidos apresentados em [5]. O algoritmo de *Blob Detection* foi escolhido pela grande capacidade de adaptação do algoritmo, que facilmente pode ser alterado para a detecção de diferentes objetos em movimento, e da ampla documentação disponível. O algoritmo de Feitosa [7] foi selecionado por já ter sido aplicado à contagem de veículos.

Os resultados obtidos em software, inicialmente, abrangem, os processamentos de baixo e médio nível, que são apresentados na Figura 1. No quadro superior à esquerda, é aplicado o filtro de média de ordem 3, no quadro superior à direita é aplicado um filtro Sobel vertical, no quadro inferior à esquerda, o filtro laplaciano, que é um filtro convolucional semelhante ao do filtro Sobel, e a no quadro inferior à direita está a imagem original.

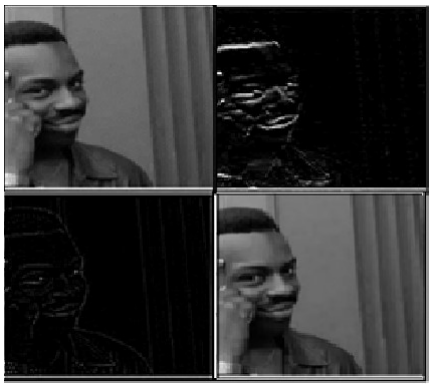

Fig 1. Resultados obtidos em software para os métodos de baixo e médio nível de processamento.

Para a implementação dos métodos de alto nível foi necessário, previamente, coletar uma serie de imagens de carros e de outros objetos, para então construir as máscaras de detecção.

A figura 2 apresenta os resultados obtidos utilizando Haar Cascades em imagens com carros, sendo os quadrados verdes as posições dos carros encontrados. O algoritmo obteve uma precisão média de 74%, quando comparado a uma contagem manual.

Fig 2. Resultados obtidos em software utilizando o algoritmo Haar Cascades.

Para a implementação do algoritmo Blob Detection, é necessário, primeiramente, realizar a subtração de imagens. Para isso, foi utilizado o filtro de média e em seguida, o filtro de limiarização. Mesmo após a utilização destes filtros ainda foram obtidos resultados insatisfatórios, sendo necessária a aplicação de outro filtro de média e de *threshold* sobre a imagem. O resultado obtido é apresentado na Figura 3. A imagem à esquerda é resultado da subtração, aplicando estes filtros, entre a imagem central e a imagem à direita.

Fig 3. Resultados da subtração de dois frames com a aplicação de filtros de baixa e média complexidade.

Após a aplicação do método de subtração de imagens foi construído o algoritmo de Blob Detection. O resultado a mesma imagem utilizada na figura 2 é apresentado na figura 4, sendo as localizações dos carros destacadas pelos círculos. O algoritmo obteve uma precisão média de 81%, quando comparado a contagem manual

Fig 4. Resultados obtidos em software utilizando o algoritmo Blob Detection.

Implementou-se também, em software, o algoritmo de Feitosa [7]. A precisão média na contagem de veículos foi de 88%. Os resultados obtidos nesta implementação, para o mesmo vídeo utilizado no Haar Cascades e Blob Detection , são apresentados na Tabela 1:

TABELA 1. RESULTADOS OBTIDOS EM SOFTWARE UTILIZANDO O ALGORITMO DE FEITOSA.

Contagem manual	Contagem por software
14	12
26	22
15	14
28	26
41	36

Para iniciar o processamento de imagens em FPGA foram realizados alguns passos adicionais. Para poder

utilizar as imagens em VHDL é necessário armazenar a imagem em um arquivo do tipo .MIF, para então, realizar o processamento [8]. A construção do arquivo .MIF é feita em software. Uma vez construído o arquivo é necessário fazer a leitura deste utilizando a biblioteca textio. O método de construção do arquivo .MIF segue o fluxograma da figura 5.

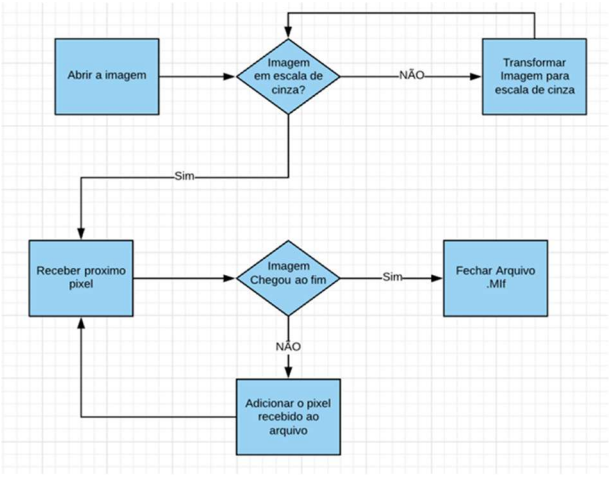

Fig 5. Fluxograma para construção do arquivo .MIF.

Utilizando a linguagem de descrição de hardware VHDL foram construídos métodos de baixo e médio processamento. Alguns resultados são expostos na figura 6. No quadro superior a esquerda o filtro de média de ordem 3 foi aplicado a imagem. No quadro superior a direita aplicou-se o filtro Sobel na direção vertical. No quadro inferior à esquerda aplicou-se o filtro Laplaciano. O inferior à direita apresenta a imagem original.

Fig 6. Resultados obtidos em hardware aplicando os métodos de processamento de médio e baixo nível.

IV. CONCLUSÃO

A utilização de métodos de detecção de veículos é de grande importância para se possa obter dados relevantes para a implementação de sistemas controle inteligente de trânsito.

Dado os diferentes algoritmos, podemos concluir que o método de Haar Cascades e Blob Detection obtiveram precisão média de 74% e 81%, respectivamente, quando comparados a contagem manual.

O método de Haar Cascades possui alguns falsos positivos, principalmente em regiões com sombra, como também deixam de apontar alguns carros na imagem. Já o algoritmo de Blob Detection tem o problema de realizar a contagem de qualquer objeto que se mova dentro do vídeo. Esta questão pode ser minimizada com o posicionamento mais adequado da câmera para cada local. Outro problema do método é a apresentação de falsos positivos a partir de grandes mudanças de luminosidade na imagem, o que ocorre com imagens em ambientes não controlados. Esta situação ocorre dentro do escopo deste trabalho, contudo, a utilização de alguns métodos de média e baixa complexidade podem diminuir, ou até mesmo anular, sua ocorrência. Já em relação ao algoritmo de Feitosa a precisão média foi de 88%.

A aplicação de processamento de imagens de baixo e médio nível em hardware obtiveram resultados de precisão iguais aos obtidos em software.

Como trabalhos futuros pretende-se concluir as implementações em hardware dos algoritmos Haar Cascades, Blob Detection e de Feitosa.

AGRADECIMENTOS

Os autores agradecem o apoio financeiro provido através do programa de iniciação científica UFPR-TN.

REFERÊNCIAS

[1] FREITAS, A J. M. ; YAMASAKI, M.Jr . *Desenvolvimento e implementação em hardware de um projeto de controle de trânsito baseado em aprendizagem por reforço.* Monografia de graduação. Departamento de Engenharia Elétrica. UFPR Curitiba-PR.. 2017. Disponível em: http://eletrica.ufpr.br/p/arquivostccs/439.pdf.

[2] GONZALEZ, Rafael C. *Digital image processing.* 3rd ed Upper Saddle River: Pearson Prentice Hall, c2008

[3] QUEIROZ, José & GOMES, Herman. (2006). *Introdução ao Processamento Digital de Imagens*. RITA. 13. 11-42

[4] N. OTSU, *A Threshold Selection Method from Gray-Level Histograms,* in IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, jan. 1979

[5] ONES, P.; VIOLA, Paul; JONES, MICHAEL.*Rapid object detection using a boosted cascade of simple features*. In: University of Rochester. Charles Rich. 2001. Diponivel em: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf.

[6] LEARN OPENCV (2020)**.** Blob Detection Using OpenCV (Python,C++). Disponível em: https://www.learnopencv.com/blob-detection-using-pencv-python-c/.

[7] FEITOSA, F.C.C. *Um estudo prático para contagem volumétrica automática de veículos usando Visão Computacional*. Goiânia-GO, 2012. 138p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás. Disponivel em : https://docplayer.com.br/5322841-Modelo-um-estudo-pratico-para-contagem-volumetrica-automatica-de-veiculos-usando-visao-computacional.html

[8] FPGA4STUDENT (2020)**.** How to Read Image in VHDL. Disponivel em: https://www.fpga4student.com/2018/08/how-to-read-imagein vhdl.html.