Fluxo Ótico para o Monitoramento Inteligente de Plataformas de Trens

Luiz Carlos Maia Junior [1], **Anna Helena Reali Costa** [1]

[1]Laboratório de Técnicas Inteligentes
Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, 158, tv. 3 – 05508-900 São Paulo, SP
{luiz.maia,anna.reali}@poli.usp.br

Abstract. *This paper describes a real time train platform intelligent monitoring system. The system counts with optical flow field/movimentation field extraction from video frame sequences through two different algorithms, and with a man-machine interface that permits the configuration of parameters, masks, etc that reduce the processed area and raise the frames per second performance.*

Resumo. *Este artigo descreve um sistema de monitoramento inteligente de plataformas de trens em tempo real. O sistema conta com extração do campo de fluxo ótico/campo de movimento de sequências de quadros de vídeo através de dois algoritmos diferentes, e com uma interface homem-máquina que permite configurar parâmetros, máscaras, etc de modo a reduzir a área de processamento e aumentar o desempenho de quadros processados por segundo.*

1. Introdução

Este trabalho descreve um sistema de monitoramento inteligente em tempo real que deve atender às seguintes funcionalidades: detecção de situações de risco para o usuário ou impedimento de circulação dos trens por haver algum objeto ou pessoa na via; distinção entre pessoas e objetos na área de risco (próximo aos trilhos); emissão de alarmes correspondentes aos riscos detectados (níveis de alarmes); direcionamento da atenção do operador ao risco.

O monitoramento inteligente deverá tornar a tarefa dos operadores mais rápida e fácil embora não pretenda substituí-los. É importante ressaltar que decisões finais de proteção, como uma ação direta na parada da circulação dos trens, não serão tomadas pelo sistema mas sim passadas para o operador que será o julgador final. Toda a interação entre o sistema de monitoramento e o operador deverá ser armazenada com data, hora e local.

Situações consideradas de risco para os usuários e bloqueio da circulação de trens são: pessoas que acidentalmente caiam na via; pessoas tentando invadir a área de circulação de trens; e objetos que venham a cair na via.

Neste trabalho avaliou-se o uso de dois algoritmos de Fluxo Ótico: Camus (1997) e Proesman (et al. 1994), assim como sua combinação com outros mais simples para tratamento de imagem relativos à extração de bordas, convolução e operadores morfológicos

open e *close* [Gonzalez 1993]. Experimentos foram realizados utilizando dados obtidos a partir do Circuito Fechado de TV de estações de metrô da cidade de São Paulo.

Como o objetivo é a aplicação em situações reais, onde o tempo de resposta deve ser reduzido ou mesmo ser comparado às respostas obtidas em sistemas de tempo real, todas as etapas de processamento das imagens devem ser muito bem definidas e simplificadas a fim de prover funcionalidade e velocidade ao sistema.

Na seção 2 é feita uma apresentação simplificada dos tipos de algoritmo de extração de campo de movimento e de campo de fluxo ótico. Nas seções 3 e 4 (Fluxo Ótico de Proesman, Campo de Movimento de Camus) são mostrados detalhes dos algoritmos implementados assim como parâmetros de calibragem. Na seção 5 é apresentada a interface homem máquina e alguns dos resultados obtidos. Finalmente, a seção 6 tem uma conclusão do estudo assim como passos para o produto final e o que hoje é considerado o estado da arte do sistema.

2. Fluxo Ótico

A função de fluxo ótico (ou campo de movimento, dependendo do tipo de algoritmo) pode ser calculada a partir de uma seqüência de imagens (no mínimo duas) para cada pixel. Um vetor de velocidade $\mathbf{v} = (u, v)$ é encontrado, o qual diz o quão rápido o objeto que contém o pixel monitorado está se movendo através das imagens (módulo do vetor); e a direção do movimento (direção do vetor).

Com estes dados da direção do movimento de cada pixel da imagem, pode-se então utilizar subtração do fundo, ou expansão de regiões para isolar certas partes e daí determinar a velocidade de diferentes objetos. A forma mais comum de se representar um campo de movimento ou um campo de fluxo ótico é o *needle map*. O *needle map* é uma figura com a representação dos vetores de todos os pontos ou apenas algumas áreas interessantes da imagem.

O problema principal na análise do movimento é a estimação das componentes tridimensionais do movimento dos objetos sendo observados. Isto é de grande relevância em muitos problemas como reconstrução 3D, rastreamento de objetos e navegação de robôs. A informação disponível em um sistema baseado em visão baseado em análise de movimento é relacionada com a projeção da velocidade 3D real no plano da imagem.

Três principais dificuldades afetam o problema de estimação do campo de movimento: (1) Descontinuidades no campo de movimento que são originados pela presença de ruído no brilho da imagem; (2) A presença de oclusões entre diferentes objetos em movimento (que usualmente têm diferentes velocidades) e entre os objetos em movimento e o fundo estático; (3) O problema de *aperture* que é relacionado com a impossibilidade de recuperar a direção do movimento se o objeto é observado através de uma abertura que é menor que o tamanho do objeto, e as referências ao objeto sob observação (como textura) não são suficientes para perceber a componente transversal do movimento do objeto.

2.3 Tipos de Algoritmos de Análise do Movimento

Na literatura, duas abordagens principais para estimação de campo de movimento em tempo real podem ser identificadas: baseado em correspondência (*matching-based*), e baseado em gradiente (*gradient-based*) [Laplante e Stoyenko 1997]. Por sua vez, as abordagens baseadas em gradiente podem ser classificadas em baseadas em regularização (*regularization-based*) e baseadas em multi-restrição (*multiconstraint-based*).

Abordagens baseadas em correspondência utilizam-se da identificação de um conjunto de características esparsas e facilmente identificáveis dos objetos em movimento. Pelo rastreamento destas características, uma correspondência entre os quadros é buscada para estimar o movimento das características no plano da imagem. Essas características facilmente identificáveis podem ser classificadas em alto nível (linhas e formas) e baixo nível (padrões, curvaturas). Estas técnicas são muito úteis para rastrear objetos em movimento no espaço 3D utilizando-se características de alto nível, porque as características rastreadas podem ser partes relevantes do objeto. Mas o rastreamento destas características requer uma fase de pré-processamento (como filtragem, extração de bordas) nas imagens, com os correspondentes custos computacionais. Além disso, as técnicas de correspondência devem ser independentes de rotação e escala e deste ponto de vista, melhores soluções são obtidas utilizando-se características de baixo nível. Entre as técnicas de correspondência, a abordagem mais difundida adota como característica o padrão do brilho da imagem. Esta técnica é chamada adequação por modelo (*block matching*). Com esta técnica, a estimação do campo de movimento para seqüências de imagens de longa duração é afetada pelo acúmulo de erro (principalmente devido à discretização da imagem).

Abordagens baseadas em gradiente provêm uma solução para estimação de movimento da observação no tempo das mudanças no brilho da imagem. Essas mudanças são modeladas por equações diferenciais parciais, que são normalmente chamadas de equações de restrição (*constraint equations*). O campo de vetores de velocidade obtido pela solução dessas equações é normalmente chamado de fluxo ótico (*optical flow*) ou fluxo da imagem (*image flow*). Genericamente, o conceito de campo de fluxo ótico difere do conceito de campo de movimento porque o campo de movimento é um conceito puramente geométrico, enquanto que o conceito de fluxo ótico é baseado na observação de mudanças no brilho da imagem. A equação diferencial parcial mais importante para a modelagem do campo de fluxo ótico é obtida pela consideração das mudanças no brilho da imagem E(x(t),y(t),t) como estacionário em respeito ao tempo (dE/dt = 0): $E_x u + E_y v + E_t = 0$, na qual u e v correspondem à dx/dt, dy/dt, representando os componentes do vetor de velocidade local **v** em toda extensão das direções **x** e **y**, respectivamente. Esta equação é normalmente chamada de Restrição de Fluxo Ótico (OFC - *Optical Flow Constraint*). Além disso, a OFC pode também ser considerada como a equação de uma reta no plano (u,v): $\mathbf{v} = mu + c$, na qual $m = -(E_x/E_y)$ é a inclinação e $c = -(E_t/E_y)$ é a raiz. Qualquer ponto ao longo dessa linha é uma possível solução para o problema de estimação do fluxo ótico.

3 Fluxo Ótico de Proesman

O algoritmo de Fluxo Ótico de Proesman [Proesman et al. 1994] consiste em extrair um campo denso de velocidade de uma seqüência de imagens. Utilizando um pré-processamento

de convolução com máscaras de Sobel [Gonzalez 1993], são extraídas as bordas verticais e horizontais da imagem, o que gera respectivamente os gradientes E_x e E_y. O gradiente no tempo E_t é obtido subtraindo-se a intensidade dos pixels das imagens consecutivas sendo processadas. Após a obtenção dos gradientes alguns parâmetros são usados durante o cálculo. Esses parâmetros definem a calibragem do algoritmo. São eles:

- Níveis – Define a escala de redução da imagem para resoluções menores. A menor figura obtida utilizando-se essa escala será utilizada para fazer a primeira estimação do fluxo ótico. A escala de redução é dada por $1/4^{níveis}$.
- Iterações – O número de iterações de refinamento do fluxo ótico em cada nível de resolução;
- Lambda – Define relação entre gradientes do espaço e o gradiente do tempo. Para imagens com ruído o lambda deve ser próximo de zero e para imagens sintéticas ou sem ruído o lambda deve ter um valor alto (acima de 1000).
- Estimação – Variável booleana que define se resultados dos quadros anteriores devem ser usados nos cálculos atuais.

A análise da imagem para extração do fluxo ótico mencionada anteriormente é feita em diferentes níveis de resolução, do nível mais baixo de resolução para o mais alto, utilizando os dados extraídos em resoluções inferiores como base para extrair o fluxo ótico em resoluções melhores, com mais detalhes. No nível de resolução mais baixa o algoritmo faz a primeira tentativa em obter o fluxo ótico. Nesta tentativa é utilizado o algoritmo Lucas & Kanade [Lucas e Kanade 1981] que utiliza a máscara definida na Figura 1.

1	2	1
2	4	2
1	2	1

Figura 1 – Máscara de utilizada por Lucas & Kanade

Esta máscara é convoluída com todos os gradientes das duas imagens. O resultado da primeira imagem (que é a imagem atual da seqüência) é chamado de *forward flow* e o da segunda de *reverse flow*. Seus resultados (e.g.: $E_{xconv} = conv(E_x, máscara)$) são usados da seguinte maneira:

Tendo:

$A = E_{xconv} * E_{xconv,}$ $B = E_{xconv} * E_{yconv,}$ $C = -E_{xconv} * E_{tconv,}$

$D = E_{yconv} * E_{yconv,}$ $E = -E_{yconv} * E_{tconv}$ e $F = E_{yconv} * E_{xconv}$.

$$\frac{\partial E}{\partial v_x} = \frac{E * C - B * F}{E * A - B * D} \qquad \frac{\partial E}{\partial v_Y} = \frac{D * C - A * F}{D * B - A * E}$$

Através das equações diferenciais parciais anteriores é obtido o vetor da velocidade do ponto central da máscara para todos os pontos da imagem. Depois dessa "primeira tentativa" esse resultado é usado nas outras resoluções e sempre sendo refinado de acordo com o número de iterações definido. O próximo passo do algoritmo é comparar o *forward*

flow com o *reverse flow* para eliminar resultados discordantes. Neste caso, resultados discordantes são vetores com a mesma direção nos dois mapas. O última passo é aplicar um refinamento que depende da constante Lambda e leva em consideração o ruído e os gradientes no tempo e espaço para definir a direção do vetor de velocidade.

Os dois últimos passos anteriormente descritos são os que podem ser aplicados repetidamente (definido na constante "Iterações") para que melhores resultados sejam obtidos.

4 Campo de Movimento de Camus

O algoritmo de Camus [Camus 1997] difere do algoritmo anterior porque extrai o campo de movimento e não o campo do fluxo ótico. Para isso o algoritmo de Camus utiliza um entre seis métodos diferentes, são eles: SAD: Sum of Absolute Differences; ZSAD: Zero-mean SAD; SSD: Sum of Squared Differences; ZSSD: Zero-mean SSD; NCC: Normalized Cross Correlation; ZNCC: Zero-mean NCC.

Uma máscara 3x3 é convoluída com todas as imagens duas a duas (podem ser mais que duas no total) efetuando um dos cálculos selecionados. Na convolução são obtidas as melhores direções e as melhores distâncias da velocidade de cada ponto. É um processamento bem mais simples que o fluxo ótico de Proesman, por isso é muito mais rápido.

5 Resultados

5.1 Seqüência de Entrada e Máscara

O programa desenvolvido utiliza os algoritmos de Proesman e de Camus para extrair o fluxo ótico de uma seqüência de imagens. O programa carrega seqüências de figuras nos formatos "BMP", "PPM", "PGM" ou "PBM" e salva resultados intermediários e finais tanto no formato "PPM" quanto "PGM" e "PBM".

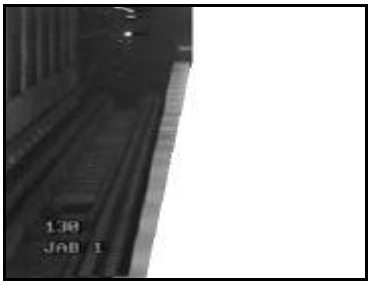
Em alguns dos testes foi utilizada uma máscara que tem o objetivo de remover o fundo estático. A máscara utilizada foi a mesma em todos os testes e é mostrada na Figura 2a.

Para os *needle maps* que foram obtidos nos dois algoritmos o limiar de subtração utilizado foi igual a 20 e a máscara obtida é mostrada na Figura 2b. Nessa mesma máscara foram utilizados os operadores morfológicos *open* e *close* [Gonzalez 1993] para que eliminar pontos isolados e depois aumentar um pouco a área de processamento. Existe a opção de se usar ou não esses operadores na interface homem máquina.

Para cada novo *needle map* que precisava ser obtido (de um conjunto de dois ou mais quadros) uma nova máscara era obtida utilizando-se a subtração do fundo. A máscara da Figura 2b é apenas um exemplo, não é fixa.

Da seqüência de entrada são mostrados os seis primeiros quadros (veja Figura 3). A taxa de captura do vídeo para essa seqüência é de 5 quadros/segundo, daí pode-se concluir que o tempo entre as capturas é de 200 ms. Esse dado será útil quando houver necessidade de calcular a velocidade relativa de um objeto na cena.

A seguir são mostrados exemplos tanto da interface quanto dos resultados obtidos com os dois algoritmos. É importante notar que as figuras foram colocadas com uma



(a)


(b)

Figura 2 – (a) Máscara de subtração do fundo. A área em branco não será processada. (b) Máscara obtida para os dois primeiros quadros com limiar de subtração igual a 20.

Figura 3 – Parte da seqüência de quadros utilizados nos testes.

resolução muito inferior àquela em que são obtidas; por isso é difícil que os vetores sejam observados. Para tornar a visualização mais fácil foram colocados recortes dos *needle maps* e são mostradas apenas figuras nas quais foi usada a máscara durante o processamento. Dados de desempenho também são mostrados, já que o objetivo em trabalhos futuros consiste em processar vídeo em tempo real.

5.2 Interface homem máquina

Na Figura 4a é mostrada a aparência da interface e logo após são explicados suas principais funções.

A interface permite escolher o algoritmo, as figuras que serão utilizadas como entrada e a máscara. Podem ainda ser configurados os parâmetros da execução de cada algoritmo, o limiar do gradiente e o limiar da subtração. O usuário pode ainda escolher quais figuras serão salvas entre *needle map*, máscara e gradientes.

Existem ainda duas outras janelas para auxiliar a visualizar dados durante a execução. Uma delas mostra a ocupação do processador (veja Figura 4b) e a outra, dados como o tempo de execução de um comando (veja Figura 4c).

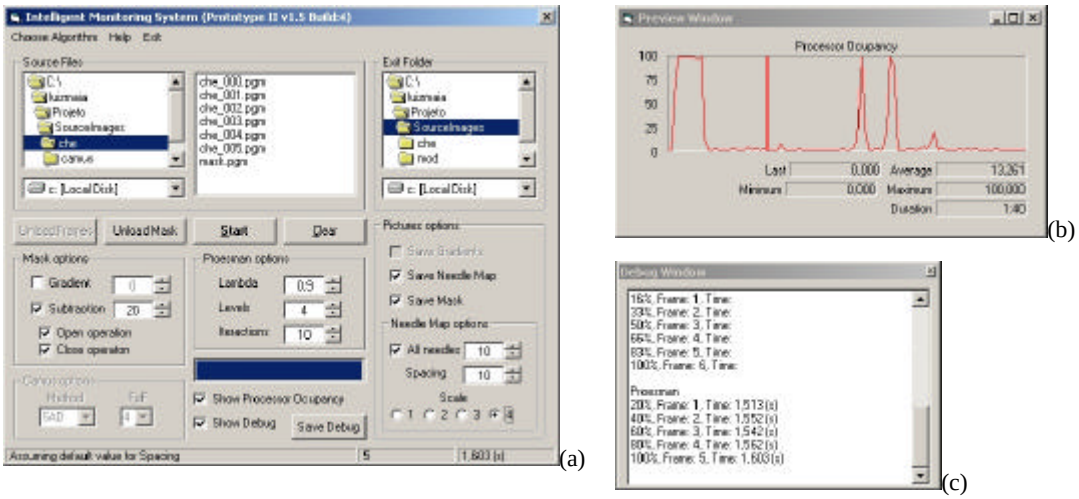


Figura 4 – (a) Interface Homem Máquina desenvolvida, área principal de comandos. (b) Janela que informa a ocupação do processador. (c) Janela que informa as operações sendo executadas e o tempo que demoram.

5.2 Resultados do algoritmo de Proesman

Como já foi dito anteriormente o algoritmo de Proesman utiliza quatro parâmetros de entrada. Destes quatro, três podem ser variados livremente (Lambda, Níveis e Iterações) e um deles (Estimação) é uma variável booleana.

Em seguida é mostrado o intervalo para cada parâmetro. Estas são as faixas nas quais foram obtidos os melhores resultados considerando-se o desempenho de quadros processados por segundo: Lambda: 0,7 a 0,9; Níveis: 2 a 4; Iterações: 4 à 16; Estimação: sempre 0.

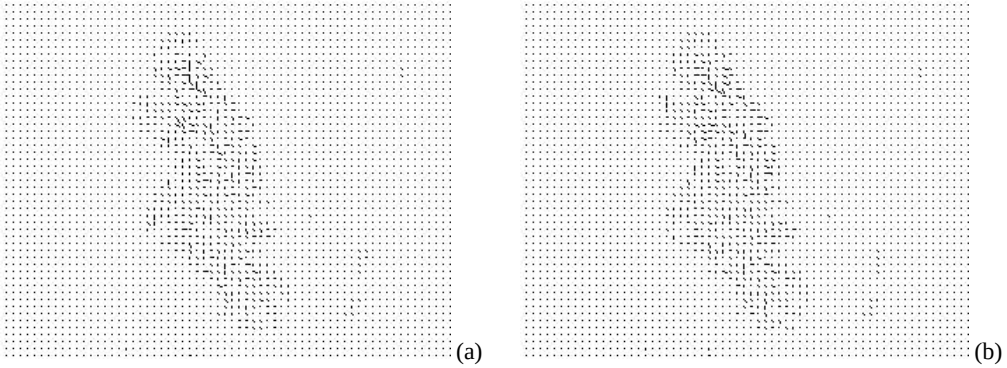


Figura 5 – (a) Recorte de um Needle Map com Lambda = 0.9, Níveis = 2, Iterações = 16 e Limiar de subtração = 20. (b) Recorte de um Needle Map com Lambda = 0.9, Níveis = 2, Iterações = 8 e Limiar de subtração = 20

As Figuras 5a e 5b são exemplos de fluxo ótico extraídos da seqüência de quadros mostrada na Figura 3. São pequenas as diferenças entre as Figuras 5a e 5b, mas o que acontece é o seguinte: com 8 iterações tem-se vetores com maior módulo, enquanto que com 16 iterações o módulo dos vetores é menor e podem-se definir áreas maiores com um conjunto de vetores na mesma direção. Ou seja, a suavização do resultado é melhor.

Como tempo real é um dos objetivos deste trabalho, o desempenho dos algoritmos é algo importante a ser analisado. Em seguida, na Tabela 1, são apresentados dados para uma análise comparativa do desempenho de algumas configurações de parâmetros.

Tabela 1 – Desempenho de Proesman com variação de alguns parâmetros.

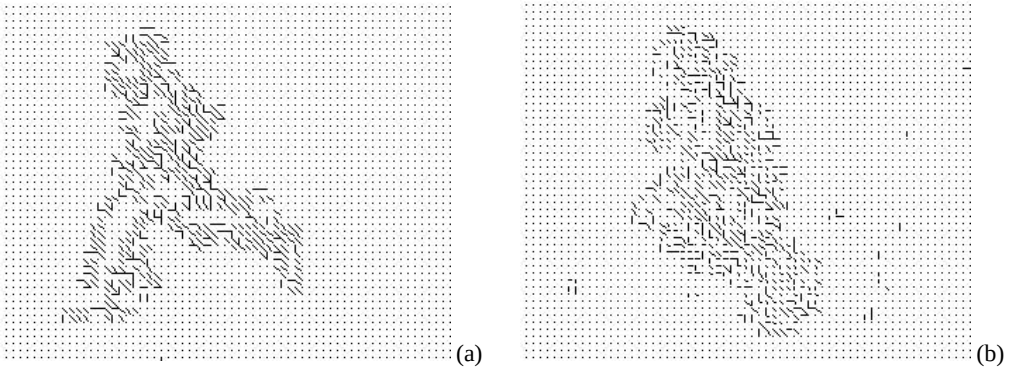
Lambda	Níveis	Iterações	ms/fluxo* qps**	máscara ms/fluxo*** qps
0,9	2	4	1830 / 0,54	230 / 4,35
		8	3600 / 0,27	380 / 2,63
		16	7150 / 0,14	700 / 1,43
	4	4	1850 / 0,54	230 / 4,35
		8	3600 / 0,27	380 / 2,63
		16	7150 / 0,14	710 / 1,41

*ms/fluxo = milisegundos por resultado intermediário; **qps = quadros por segundo;
***limiar de subtração da máscara utilizado é igual a 20.

5.3 Resultados do algoritmo de Camus

O algoritmo de Camus utiliza dois parâmetros de entrada, o método para obtenção do campo de movimento e o número de frames processados em conjunto. Os melhores resultados foram obtidos utilizando-se o método NCC para dois ou três quadros processados em conjunto. Esse método foi escolhido porque apresentou melhores resultados com texturas e porque é o mais rápido. Os resultados são baseados nesse método.

As Figuras 6a e 6b a seguir são exemplos de campos de movimento extraídos da seqüência de quadros mostrada na Figura 3.



(a) (b)

Figura 6 – (a) Campo de Movimento com Método NCC e dois quadros de entrada. (b) Campo de Movimento com Método NCC e três quadros de entrada

Na Figura 6a os vetores são nulos ou têm tamanho máximo, não existem vetores com módulo intermediário. Isso ocorre porque esta estimação do campo de movimento foi feita apenas com dois quadros. A vantagem que ela apresenta é que foi estimada muito rapidamente e é coerente.

Diferentemente da Figura 6a, na Figura 6b encontram-se vetores com tamanhos variados, mas ainda não se vê vetores com ângulos diferentes de 0, 45 ou 90 graus. Seriam necessários mais quadros no processamento para se conseguir a correção da direção. Em

termos de tempo de processamento, o algoritmo de Camus é muito mais rápido que o de Proesman, mas ainda deve-se considerar a qualidade do resultado. A Tabela 2 mostra o desempenho do algoritmo de Camus em função da variação de alguns parâmetros.

Tabela 2 - Desempenho de Camus com variação de alguns parâmetros.

Método	Quadros	ms/fluxo* qps**	máscara ms/fluxo*** qps
SAD	2	690 / 1,45	85 / 11,76
	3	1360 / 0,73	165 / 6,06
	4	2025 / 0,44	200 / 5
	5	2700 / 0,37	295 / 3,39
NCC	2	575 / 1,74	80 / 12,5
	3	1125 / 0,89	140 / 7,14
	4	1690 / 0,59	180 / 5,55
	5	2230 / 0,45	230 / 4,35

*ms/fluxo = milisegundos por resultado intermediário; **qps = quadros por segundo;
***limiar de subtração da máscara utilizado é igual a 20.

6 Conclusão

Para que bons resultados sejam obtidos com o algoritmo de Proesman é necessária uma calibragem, do sistema principalmente no que diz respeito ao lambda e o número de iterações e, secundariamente, a quantidade de níveis de resolução:

O número de iterações influi diretamente no tempo de processamento da sequência. Quanto ao número de iterações, que implica em um refinamento maior do fluxo ótico, não se pode esquecer que é muito aumentado o custo computacional a cada iteração que for adicionada. O lambda influi na qualidade desse processamento e na proximidade de busca dos pixels. Em imagens sem ruído o valor do lambda praticamente não influencia o resultado. A quantidade de níveis de resolução influi um pouco no desempenho e um pouco na qualidade do resultado. Por isso ao escolher a quantidade de níveis, deve-se preocupar mais com o fato que na primeira tentativa será utilizada a menor resolução onde serão extraídas bordas e etc, uma perda muito grande de resolução causa também a perda de bordas que podem ser cruciais para o desenvolvimento do resultado. Essa calibragem foi feita empiricamente através de várias tentativas e análise dos resultados.

Em relação ao algoritmo de Camus o que foi observado é que ele apresenta resultados coerentes rapidamente mas que, para refirnar esse resultado, ter-se-ia o mesmo custo computacional do algoritmo de Proesman. Mesmo o método NCC sendo bom para imagens com textura, o que foi observado é que deve-se utilizar mais que três quadros em conjunto por processamento, para que os vetores do campo comecem a ficar mais precisos.

É importante observar que quando o limiar de subtração é aumentado, a área que será processada está sendo diminuida e informações muito importantes podem estar sendo perdidas. Por isso foi escolhido o limiar igual a 20 porque apesar de restar algum ruído na imagem pelo menos tem-se a certeza de não perder variações na imagem que podem indicar movimento. Diferentes limiares foram testados, entre eles um limiar de 120 que fazia com que no as pernas do corpo (por terem cor mais próxima ao fundo do que o resto) em movimento

sumissem, isso fez com que ainda se pude-se estimar a direção e a velocidade do corpo mas não seria possível obter o tamanho do objeto que é uma informação que será usada em trabalhos futuros.

O importante em todos os parâmetros além de serem obtidos bons resultados é que seja observado o custo computacional. Um sistema pode ter resultados excelentes com horas de processamento, mas talvez não seja necessária tanta precisão mas sim respostas mais rápidas, um sistema bem calibrado fornece a resposta no tempo certo e não extrai informação que depois será jogada fora. Cabe a cada um analisar sua aplicação e determinar o que é ou não importante, que áreas da imagem são importantes e quais as características que se quer rastrear.

Quanto à aplicabilidade dos algoritmos estudados o que faria com que o algoritmo de Camus fosse escolhido é o fato que sua estimação pode não ser a mais precisa mas pode ser feita muito rapidamente. Outro ponto importante é que o algoritmo de Camus não precisa de calibragem.

Uma possibilidade de trabalho futuro seria reunir os dois algoritmos para formar um terceiro que teria o seguinte funcionamento: a primeira estimação de movimento do algoritmo de Proesman seria feita pelo algoritmo de Camus e daí por diante seguiria o algoritmo de Proesman. Pois o algoritmo de Camus é bom para determinar direção inicial, e apresenta o campo de movimentação com conjuntos de vetores que indicam melhor a direção do movimento de uma região à primeira vista. E o algoritmo de Proesman é bom para determinar velocidades e direções mais precisas dos vetores.

7. REFERÊNCIAS

Camus, T. (1997) “Real-Time Quantized Optical Flow”, Journal of Real-Time Imaging, Vol. 3, pg. 71-86.

Gonzalez, R.C. e Woods, R.E. (1993) “Digital Image Processing”, Addison Wesley.

Laplante, Philip A. e Stoyenko, Alexander D., (1997) “Real Time Imaging - Theory, Techniques, and Applications”, IEEE Press.

Lucas, B. and Kanade, T. (1981) “An Iterative Image Registration Technique with an Application to Stereo Vision”, Proc. Of 7th International Joint Conference on Artificial Intelligence (IJCAI), pg. 674-679.

Proesman, M., Van Gool, L., Pawels, E. e Oosterlinck, A. (1994) “Determination of optical flow and its Discontinuities using non-linear diffusion”, In: 3rd European Conference on Computer Vision, ECCV’94, Vol. 2, pg. 295-304.