

Comparação de métodos para localização de fluxo óptico em sequências de imagens

Vantuil José de Oliveira Neto, David Menotti Gomes
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email: vantuiljose@gmail.com, menottid@gmail.com

Resumo—O fluxo óptico descreve o movimento aparente em uma sequência de imagens digitais. O Movimento nessa sequência pode ser utilizado em diversas aplicações, como acompanhamento de objetos em vídeos e estimação de tempo de colisão entre elementos de uma cena. Neste trabalho foi realizado o estudo, implementação e análise de dois dos principais métodos para computação do fluxo óptico, os métodos Lucas & Kanade e Horn & Schunck. Será feita a análise de complexidade dos algoritmos e de seus tempos de execução, bem como avaliação dos resultados sobre uma sequência de imagens artificiais. Ao final foi traçado um paralelo entre os principais métodos, a complexidade dos algoritmos, tempo de execução e sua aplicabilidade na solução de problemas reais em que possa ser utilizado o fluxo óptico.

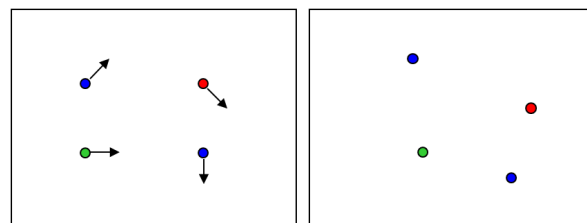
Keywords—fluxo óptico, análise de complexidade, Lucas & Kanade, Horn & Schunck

I. INTRODUÇÃO

Uma importante técnica para estimar movimento em sequência de imagens é o chamado *Optical Flow* (Fluxo Óptico). Fluxo óptico é a distribuição da velocidade aparente do movimento dos padrões de intensidade em uma imagem. Pode surgir de um movimento relativo de objetos e vistas. Consequentemente, fluxo óptico pode dar uma informação importante sobre o arranjo dos objetos vistos e a taxa de mudança destes mesmos arranjos. Assim como a descontinuidade no Fluxo Óptico pode ajudar em segmentação de imagens em regiões que correspondem a diferentes objetos [1].

A Figura 1 mostra o fluxo óptico. Na Figura 1a a imagem não possui movimento, ela seria a primeira imagem de uma sequência. As setas mostram o movimento entre a Figura 1a e a Figura 1b, na última temos a imagem após o movimento, ou seja uma imagem depois da primeira onde foi captado algum tipo de movimento.

O fluxo óptico será sempre interessante quando o movimento dos objetos contidos em uma cena tiver relevância. O fluxo óptico pode ser utilizado com diferentes aplicações no processamento e análise de imagens digitais. Entre as aplicações podemos destacar algumas: interpretação de cena, navegação exploratória, acompanhamento de objetos, avaliação de tempo de um corpo contra o outro e segmentação de objetos [2]. Mas sua aplicabilidade é ainda mais extensa e também pode ser utilizado em diversos outros campos, tais como visão robótica e aplicações de vigilância.



(a) Imagem original (as setas re-(b) Imagem após movimento em
presentam o movimento) relação a Figura 1a

Figura 1: Sequência de imagens artificiais utilizadas nos testes

Este trabalho consiste em pesquisar, implementar e testar alguns dos principais métodos para fluxo óptico. Inicialmente pretende-se implementar os métodos Lucas & Kanade proposto em [3] e Horn & Schunck proposto em [4]. Serão analisados a ordem de complexidade e o tempo de execução desses métodos. Eles serão implementados e testados, dessa forma será possível mostrar o desempenho e o resultado dos métodos estudados. Serão levadas em consideração apenas sequências de imagens que possuam apenas um movimento entre a câmera e o objeto observado, e que as condições de iluminação não mudem, assim como proposto em [5].

Primeiramente foi feito um levantamento bibliográfico sobre o tema, a fim de refinar melhor os métodos para cálculo do fluxo óptico, e de conhecer as utilidades e as restrições dos métodos clássicos, bem como conhecer novas técnicas e adaptações dos métodos originais.

A partir desses métodos foram feitas algumas análises, como a análise teórica do algoritmo, ou seja sua ordem de complexidade, tanto de tempo quanto de espaço. Os métodos foram implementados em MatLab¹, visto a facilidade de tratar imagens nesta linguagem.

O artigo está organizado da seguinte forma: A Seção II acrescentará a esta seção a ideia de fluxo óptico, dando a ele um significado mais didático. Já a Seções seguintes mos-

¹MATLAB (MATrix LABoratory) é um software interativo de alto desempenho voltado para o cálculo numérico. O MATLAB integra análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos em ambiente fácil de usar onde problemas e soluções são expressos somente como eles são escritos matematicamente, ao contrário da programação tradicional [6].

traram algumas das principais operações realizadas sobre as imagens em técnicas diferenciais para a computação do fluxo óptico. Dessa forma a Seção IV mostra como calcular a convolução de uma imagem digital, uma técnica muito importante para encontrar a derivada de uma imagem que é abordada na Seção IV. As Seções V e VI mostram os dois métodos e a implementação dos mesmos, bem como comentários sobre o código e a ordem de complexidade deles. A Seção VII trás os principais resultados obtidos com a execução e os testes dos algoritmos estudados, nesta seção é apresentada de forma resumida a complexidade dos métodos assim como os tempos de execução para imagens de diferentes tamanhos. A Seção VIII conclui este trabalho, mostrando os objetivos alcançados, destacando os problemas encontrados e apresentando trabalhos futuros que podem ser derivados deste artigo.

II. FLUXO ÓPTICO

A ideia por trás do fluxo óptico é encontrar para cada pixel um vetor $\vec{u} = (u, v)$ que diga quão rápido o pixel se move pela imagem e em que direção o mesmo se move.

A Figura 2 exemplifica o fluxo óptico. Um ponto P na cena projeta um ponto $p[x, y]$ no sistema de coordenadas do plano de imagem da câmera centrada na origem do sistema de coordenadas da câmera $[X, Y, Z]$, que é o eixo óptico apontando da direção de Z . O movimento da câmera é descrito por sua translação $[Tx, Ty, Tz]$ e rotação $[\Omega_x, \Omega_y, \Omega_z]$ [1] [7].

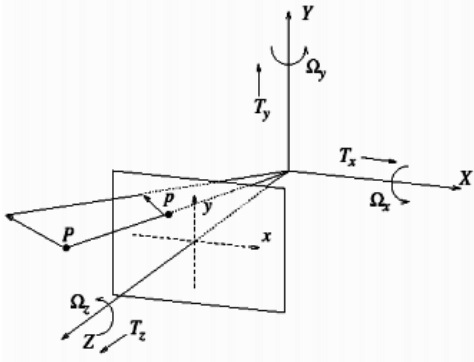


Figura 2: Fluxo Óptico [7]

Fluxo Óptico ainda pode ser definido como a representação do movimento aparente do mundo projetado no plano de imagem. Fluxo Óptico é o campo de velocidade 2D, descrevendo o movimento aparente na imagem, que resulta de movimentos independentes de objetos na cena ou de um observador em movimento. Considere a Figura 2 que ilustra como a translação e rotação da câmera causa a localização projetada p de um ponto P na cena em movimento [1]. Da mesma forma, se o ponto P esta se movendo independentemente, sua projeção no plano de imagem irá mudar, mesmo quando a câmara estiver estacionária. É o

vetor de campo, $u(x, y) = [u(x, y), v(x, y)]$, descrevendo o movimento horizontal e vertical, para todos os pontos da cena [8].

Ambos os métodos estudados neste trabalho, Lucas & Kanade e Horn & Schunk são técnicas diferenciais, onde a hipótese inicial para a computação do fluxo óptico é a de que a intensidade entre quadros diferentes em uma sequência de imagens é aproximadamente constante [1]. A partir dessa premissa, usando equações diferenciais é possível calcular o fluxo óptico. Entretanto esse cálculo não é suficiente para determinar o fluxo óptico, dessa forma são necessários os diferentes métodos. O Lucas & Kanade propõe um método onde, para encontrar o fluxo óptico, outro conjunto de equações é necessário. A solução dada por Lucas & Kanade é um método não iterativo que assume um fluxo óptico constante local [3]. Já no método Horn & Schunk, a velocidade da imagem é computada a partir das derivadas espaço-temporal das intensidades na imagem [4].

III. AVALIANDO A COMPLEXIDADE DA CONVOLUÇÃO

A convolução é uma operação matemática comum em processamento de imagens [9]. Consiste em, dada uma máscara M e uma imagem G , calcular G' , tal que para todo $m, n \in G$:

$$G'(m, n) = \sum_{i=-k}^k \sum_{j=-k}^k G(m+i, n+j) \times M(i, j)$$

onde k é o tamanho da máscara M , e n o tamanho da imagem G [10].

A Figura 3 ilustra o processo de convolução. O *kernel* é a máscara, *input* a imagem de entrada, e *output* a imagem gerada pela convolução.

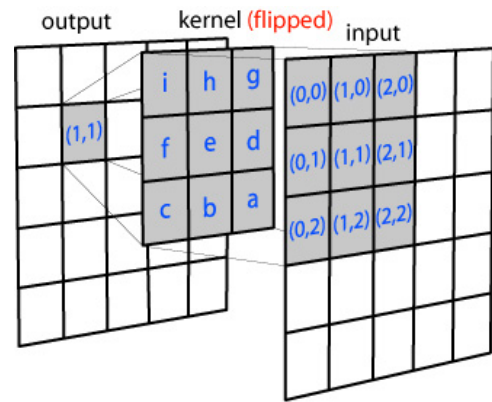


Figura 3: Convolução [7]

Para cada *pixel* na imagem, é calculado um novo valor, baseado no valor que se encontra na máscara. A função de complexidade desse algoritmo, considerando o número de cálculos, é:

$$f(n) = n^2 * k^2$$

onde n é o tamanho da imagem G , e k o tamanho da máscara M .

IV. CALCULANDO A DERIVADA DE UMA IMAGEM

O Programa 1 é uma possível implementação para calcular a derivada entre duas imagens. Este método consiste na convolução da imagem com diferentes máscaras, a fim de obter a derivada horizontal, vertical e temporal.

Programa 1: Cálculo da derivada entre duas matrizes quaisquer

```
1 function [fx, fy, ft] = ...
   ComputeDerivatives(im1, im2);
2 %Calcula derivada horizontal, vertical e ...
   temporal
3 %entre duas imagens em escala de cinza
4 fx = conv2(im1, 0.25* [-1 1; -1 1]) + ...
   conv2(im2, 0.25*[-1 1; -1 1]);
5 fy = conv2(im1, 0.25*[-1 -1; 1 1]) + ...
   conv2(im2, 0.25*[-1 -1; 1 1]);
6 ft = conv2(im1, 0.25*ones(2)) + conv2(im2, ...
   -0.25*ones(2));
```

Como k é um valor fixo, podemos definir a complexidade da função *computeDerivates*: Para cada um das derivadas $f(x)$, $f(y)$, e $f(t)$, é necessário convoluir as duas imagens de tamanho n com uma máscara de tamanho 2, e ainda somá-las duas a duas para cada uma das derivadas.

A complexidade para calcular $f(x)$ seria

$$f(n) = 2 \times \underbrace{(n^2 \times 2^2)}_{\text{Custo da convolução}} + \underbrace{n}_{\text{Custo de somar duas convoluções}}$$

Desenvolvendo a equação, vem que:

$$f(n) = 2 \times 4n + n = 8n + n = 9n$$

Esta é a função de complexidade para calcular um derivada, mas é necessário calcular 3 delas, $f(x)$, $f(y)$, e $f(t)$, portanto o custo total da função *calculeDerivates* é $3 \times 9n = 27n$.

Ou seja, como é usada uma máscara de tamanho 2×2 para calcular a derivada entre duas imagens, o custo de calculá-la é de ordem linear, $O(n) = n$, já que o maior grau da função $f(n) = 27n$ é igual a 1.

O cálculo da complexidade para o cálculo das derivadas será de grande importância para o desenvolvimento deste trabalho, uma vez que os dois métodos estudados utilizam este cálculo para encontrar o fluxo óptico. Portanto o valor $27n$, ainda será muito utilizado nas seções seguintes.

V. LUCAS & KANADE

Esse método simplesmente consiste no deformamento da imagem, diferenciação, produção de pontos na imagem, multiplicação com o inverso de Hessian e atualização da deformação. [11] Sua abordagem consiste em dividir a

imagem em janelas, e calcular se os *pixels* mudaram de lugar entre as janelas vizinhas.

O Programa 2 mostra uma implementação deste método para localização de fluxo óptico.

Programa 2: Método Lucas & Kanade

```
1 function [u, v] = LucasKanade(im1, im2, ...
   windowSize);
2
3 [fx, fy, ft] = ComputeDerivatives(im1, im2);
4
5 u = zeros(size(im1));
6 v = zeros(size(im2));
7
8 halfWindow = floor(windowSize/2);
9 for i = halfWindow+1:size(fx,1)-halfWindow
10     for j = halfWindow+1:size(fx,2)-halfWindow
11         curFx = fx(i-halfWindow:i+halfWindow, ...
12             j-halfWindow:j+halfWindow);
13         curFy = fy(i-halfWindow:i+halfWindow, ...
14             j-halfWindow:j+halfWindow);
15         curFt = ft(i-halfWindow:i+halfWindow, ...
16             j-halfWindow:j+halfWindow);
17
18         curFx = curFx';
19         curFy = curFy';
20         curFt = curFt';
21
22         curFx = curFx(:);
23         curFy = curFy(:);
24         curFt = -curFt(:);
25
26         A = [curFx curFy];
27
28         U = pinv(A'*A)*A'*curFt;
29
30         u(i, j)=U(1);
31         v(i, j)=U(2);
32     end;
33 end;
34
35 u(isnan(u))=0;
36 v(isnan(v))=0;
```

O custo do método Lucas & Kanade será calculado a partir do código mostrado no Programa 2. O laço na linha 9, é onde tem-se o maior custo de processamento para o método, já que as derivadas já foram calculadas na linha 3. Note que o custo deste laço depende do parâmetro *windowSize*, que será chamado de n , quanto maior o valor deste parâmetro, menos vezes os laços executarão, por outro lado, as operações internas serão feitas sobre imagens de tamanho de w , fazendo que, com um valor do parâmetro menor as operações internas sejam menos custosas. Para facilitar os cálculos foram consideradas imagens e máscaras com tamanhos que sejam potências de 2, assim não houve resto na divisão das janelas.

A função divide a imagem em $n/\text{windowSize}$ janelas. Portanto este é o número de vezes que ele executou os dois laços, nas linhas 9 e 10. Não é possível determinar qual o tamanho ideal para janela, já que o tamanho pode e

deve variar de acordo com a aplicação. Um dos parâmetros que podem ser utilizados para avaliar qual o tamanho da janela é a magnitude do movimento. Movimentos maiores seriam melhor encontrados com janelas maiores, enquanto os menores seriam mais recomendados para deslocamentos também menores.

Da linha 11 até a 24, não tem-se efetivamente nenhum cálculo, mas apenas atribuições. As operações de alta ordem aparecem na linha 26, onde há a multiplicação de matrizes. A multiplicação de matrizes tem complexidade algorítmica de n^3 . Mas ainda é necessário notar que na linha 24 a variável A recebe a concatenação de fx e fy , portanto passa a ter tamanho $2n$. Nessa linha são calculadas três multiplicações, portanto a função é $f(n) = (2w)^3 + (2w)^3 + w^3 = 17w^3$.

As linhas 33 e 34 apenas completam os valores dos vetores de velocidade, u e v que não foram calculado com o valor 0, então ele faz $O(n^2)$ comparações, e no pior caso $O(n^2)$ atribuições, pois podem existir no máximo n^2 valores que não foram calculados.

Desprezando-se os termos de baixa ordem, pode-se dizer q complexidade deste algoritmo é

$$\begin{aligned} w^3 \times (n - w) \times (n - w) = \\ w^3 \times (n^2 - 2nw + w^2) = \\ w^5 + n^2w^3 - 2nw^4 \end{aligned}$$

O método Lucas & Kanade é de complexidade quadrática, ou seja $\Omega(n^2)$. Entretanto existe o custo de multiplicar e calcular as inversas de w e embora $w < n$, w possui graus altos e ainda é multiplicado por n^2 , o que significa que, em prática, ele terá uma grande impacto no tempo de execução, por isso é preferível dizer que este método possui complexidade $O(n^2 \times w^5)$. Vale ressaltar que os calculos efetuados sobre w são de ordem $O(n^3)$, que é o grau da complexidade para cálculo da inversa e da multiplicação das matrizes (linha 26).

VI. HORN & SCHUNCK

O Programa 3 mostra a implementação do método Horn & Schunck.

Programa 3: Método Horn & Schunck

```
1 function [u, v] = HS(im1, im2, alpha, ite, ...
   displayFlow, displayImg)
2
3 %% Converte as imagens para escala de cinza
4 if size(size(im1), 2) == 3
5     im1=rgb2gray(im1);
6 end
7 if size(size(im2), 2) == 3
8     im2=rgb2gray(im2);
9 end
10 im1=double(im1);
11 im2=double(im2);
12
13 im1=smoothImg(im1, 1);
```

```
14 im2=smoothImg(im2, 1);
15
16
17 % Definindo os valores iniciais para os ...
   vetores de velocidade
18 u = zeros(size(im1(:, :, 1)));
19 v = zeros(size(im2(:, :, 1)));
20
21 % Estimando as derivada espaço-temporais
22 [fx, fy, ft] = computeDerivatives(im1, im2);
23
24 % Criando Máscara
25 kernel_1=[1/12 1/6 1/12; 1/6 0 1/6; 1/12 1/6 ...
   1/12];
26
27 % Iterações
28 for i=1:ite
29     %Calculando as médias locais dos ...
   vetores de fluxo
30     uAvg=conv2(u, kernel_1, 'same');
31     vAvg=conv2(v, kernel_1, 'same');
32     % Computando vetores de fluxo limitado ...
   por sua média local e as restrições ...
   de fluxo óptico
33     u= uAvg - ( fx .* ( ( fx .* uAvg ) + ( ...
   fy .* vAvg ) + ft ) ) ./ ( alpha^2 ...
   + fx.^2 + fy.^2);
34     v= vAvg - ( fy .* ( ( fx .* uAvg ) + ( ...
   fy .* vAvg ) + ft ) ) ./ ( alpha^2 ...
   + fx.^2 + fy.^2);
35 end
36
37 u(isnan(u))=0;
38 v(isnan(v))=0;
39
40 % Plotando o fluxo
41 if displayFlow==1
42     plotFlow(u, v, displayImg, 5, 5);
43 end
```

Da linha 2 até a linha 12 existem apenas algumas verificações, para saber se a imagem está em escala de cinza, e a conversão de tipos para permitir o cálculo. Todas as operações nessas linhas tem no ordem $O(n^2)$.

Na linha 13 são calculadas as derivadas entre as duas imagens, e como visto na seção IV tem função de complexidade de $27n$

Nas linhas e temos a função smoothIm que é usada para suavizar uma imagem utilizando um filtro Gaussiano. Um filtro Gaussiano, com uma máscara já definida, tem complexidade $O(n^2)$ [12]. Nas linhas 13 e 14 são criadas os vetores de velocidades com valores 0, e na linha 22 calculamos as derivadas entre as duas imagens, complexidade $O(n^2)$.

Na linha 29 há um laço, que vai até a um valor pré-determinado. Este valor é um parâmetro da função, e também está relacionado a distância em que os objetos se moveram. Portanto ele pode ser considerado uma constante, pois não cresce quando o tamanho n da imagem cresce. Para os testes foram utilizadas 10 iterações.

Dentro do laço, nas linhas 30 e 31 são feitas duas convoluções com uma máscara de tamanho fixo, que, como visto, possui complexidade quadrática. Nas linhas 32 e 34 são realizadas apenas operações pontuais, como soma e

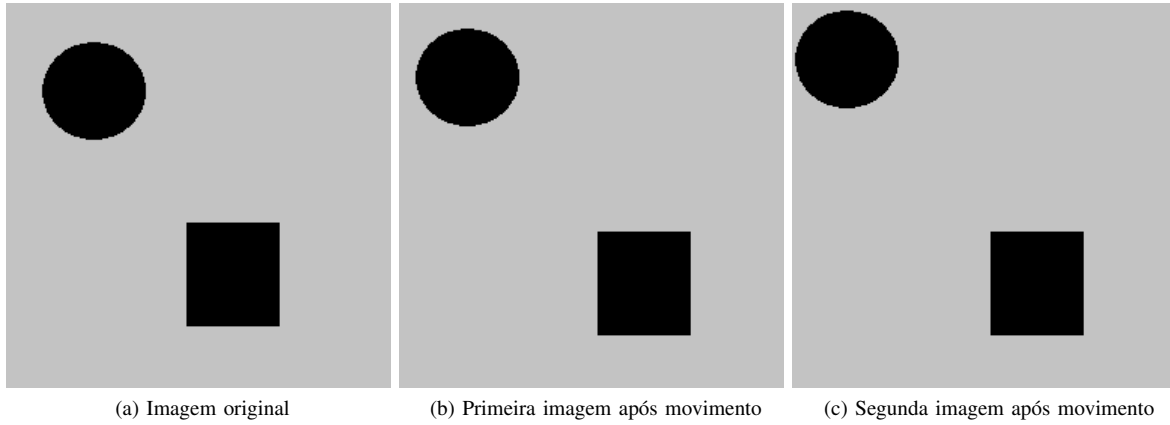


Figura 4: Exemplo de Fluxo Óptico

multiplicação, essas operações pontuais tem complexidade $O(n^2)$. Assim como no método Lucas & Kanade, as últimas linhas apenas substituem os valores que não foram calculados pelo valor 0.

Com isso, pode-se dizer que a ordem de complexidade deste método é $O(n^2)$, ou seja o método Horn & Schunck tem complexidade quadráticas. Isto pode ser notado atentando-se ao fato de que os termos de maior ordem são quadráticos, e todos os outros são constantes, não variando de acordo com o tamanho da imagem.

VII. RESULTADOS

Com base nas análises feitas nas Seções VI e V, e com testes de desempenho realizados no MatLab, serão apresentados alguns resultados. O fluxo óptico que é calculado para os dois métodos também será apresentado nessa seção. Para os testes foi usada janela de tamanho 8 para o Lucas & Kanade e foram feitas 5 iterações com o método Horn & Schunck. Para medir o tempo de execução, foram feitos 10 testes com cada tamanho de imagem, e é apresentado a média do tempo de execução para cada um dos tamanhos de imagens testadas.

A. Ordem de Complexidade e Tempo de execução

A tabela da Figura II mostra um resumo sobre os tempos de execução das duas técnicas de computação de fluxo óptico.

A ordem de complexidade do método Lucas & Kanade apesar de não ser a do Horn & Schunck, possui outros valores que podem aumentar muito o tempo de execução, como o leitor pode notar na Tabela I. Para imagens de tamanho grande, deve-se tomar cuidado ao optar pelo primeiro método. O comportamento assintótico des algoritmo, dependendo da escolha do tamanho da janela, é muito alto para aplicações que necessitem de resposta em pouco tempo.

Veja a Figura I mostra o tempo de execução entre os dois métodos. O leitor pode notar o quão grande pode ser o

tempo com uma solução com complexidade n^2 , porém com outros valores que tem grande impacto sobre essa a função de complexidade, quando a imagem cresce de tamanho, o gráfico na Figura 6 também mostra este aspecto. A Figura 7 também mostra o fluxo óptico, porém utilizando o método Horn & Schunck.



Figura 6: Gráfico comparativo do tempo de execução dos métodos

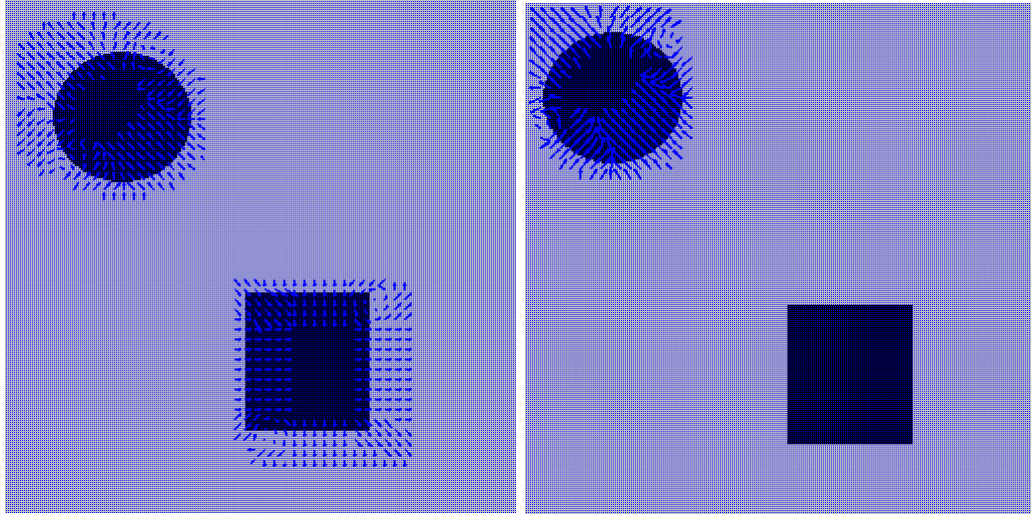
Tabela I: Tempo de execução dos métodos de fluxo óptico (em ms)

Tamanho da imagem	Lukas-Kanade	Horn-Schunck
16	0,029407	0,00817
32	0,191916	0,024754
64	0,740171	0,060697
128	2,536055	0,177277
256	9,603240	1,066800
512	35,715323	2,996775
1024	133,563754	12,640983

A Tabela II resume a complexidade dos métodos estudados neste trabalho.

B. Experimentos com sequência de imagens

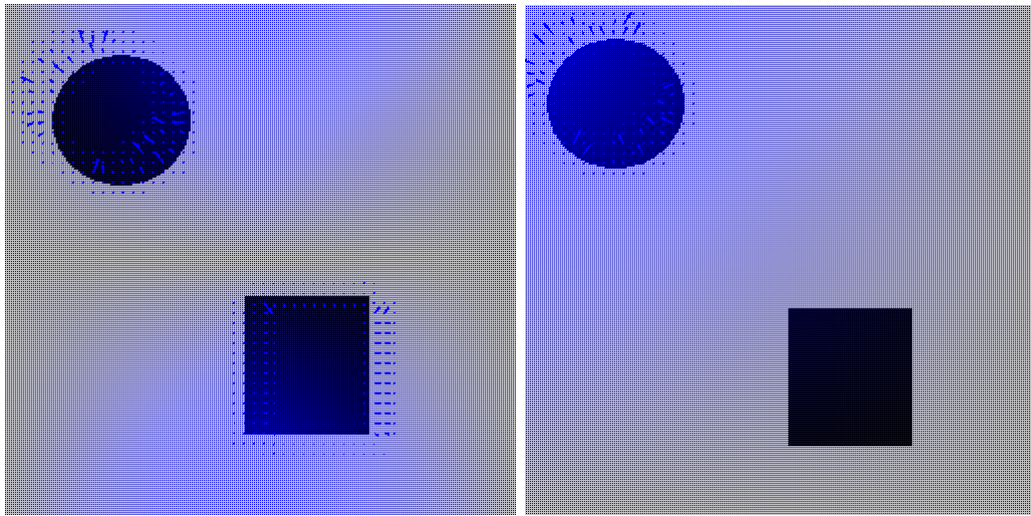
Para os experimentos foram criadas três imagens artificiais, contendo um círculo e um quadrado, com leves



(a) entre Figuras 4a e 4b

(b) entre Figuras 4b e 4c

Figura 5: Resultado com método Lucas & Kanade - Fluxo óptico



(a) entre Figuras 4a e 4b

(b) entre Figuras 4b e 4c

Figura 7: Resultado com método Horn & Schunck - Fluxo óptico

Tabela II: Resumo da ordem de complexidade dos métodos de detecção de fluxo óptico

Método	Complexidade
Lukas-Kanade	$O(n^2 \times w^5)$
Horn-Schunck	$O(n^2)$

movimentos entre as imagens. E a partir desta primeira, foram criadas mais duas imagens, a primeira delas com um pequeno movimento tanto do círculo quanto do quadrado e outra foi construída a partir da segunda imagem, mas apenas o círculo foi movido. As imagens geradas para os testes

aparecem na Figura 4.

Na Figura 5 são mostrados dois fluxos ópticos plotados sobre a imagem de origem, cada um sobre um das figuras e ambos gerados pelo método Lucas & Kanade. Eles foram plotados utilizando a função *plotflow* do MatLab, que dados os vetores de movimento horizontal e vertical e uma imagem, é capaz de gerar um diagrama de agulhas do fluxo óptico sobre a imagem dada.

Alguns teste também foram feitos com imagens reais, onde os dois métodos mostraram-se eficientes para encontrar o fluxo óptico. As imagens utilizadas para teste são as imagens da Figura 8, e o resultado é apresentado na Figura

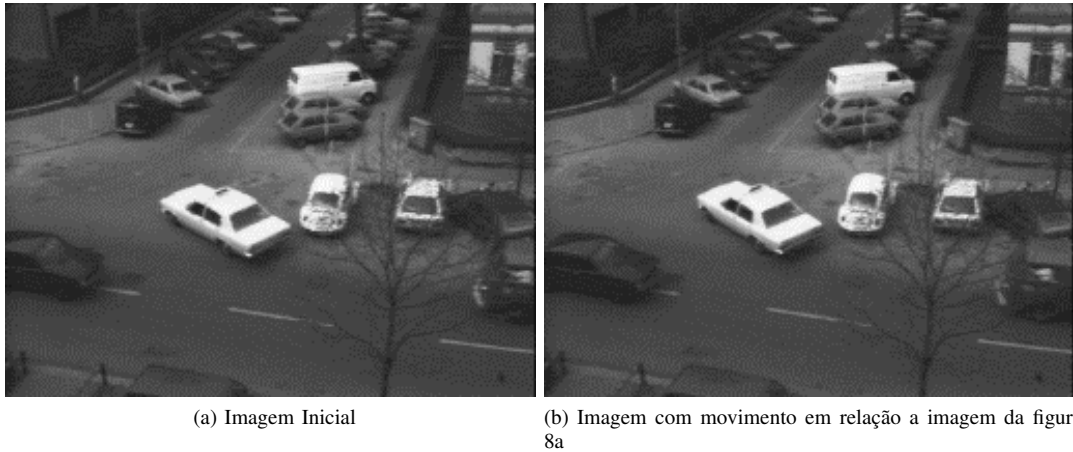


Figura 8: Imagens reais utilizadas para experimento

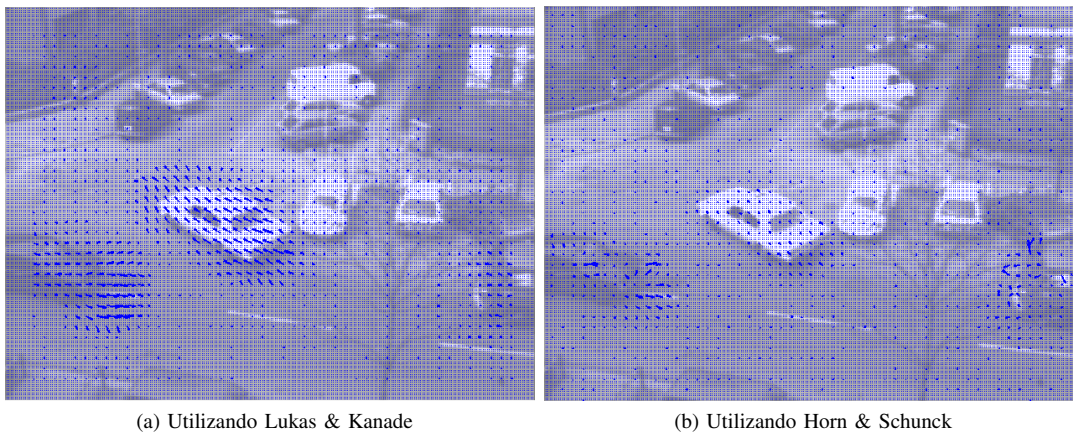


Figura 9: Resultado do cálculo do fluxo óptico

9. As implementações foram simples, e por esse motivo os métodos mostraram-se sensíveis a ruídos nas imagens reais.

VIII. CONCLUSÃO

O fluxo óptico é capaz de determinar a velocidade e a direção em que os objetos se movem em uma sequência de imagens. Dentre alguns dos principais métodos estão Lucas & Kanade e Horn & Schunck, que foram estudados, implementados e testados neste trabalho.

Ambos os métodos mostraram-se eficazes no cálculo do fluxo óptico para as imagens testadas. A partir do diagrama de agulhas foi possível notar que a velocidade e a direção em que os objetos se moviam, detectada pelos diferentes métodos, eram as velocidades reais e direções reais nas quais a imagem foi criada. Os métodos funcionaram bem tanto com as imagens artificiais quanto com as imagens reais, com uma pequena sensibilidade a ruídos.

Entretanto constatou-se uma grande limitação do método Lucas & Kanade, a sua ordem de complexidade. Com

um custo computacional tão alto, o método não mostrou nenhuma vantagem sobre o outro estudado. Problemas com custo como o do método proposto por Lucas & Kanade podem tornar-se intratáveis quando o tamanho da entrada de dados for muito grande.

Já o método Horn & Schunck apresentou resultados bem mais satisfatórios. Mesmo com imagens grandes foi possível ter um tempo razoavelmente pequeno.

Este trabalho abre o problema de comparar os métodos disponíveis para computação do fluxo óptico, comparando não apenas a complexidade, mas também a utilidade e aplicabilidade dos métodos.

REFERÊNCIAS

- [1] A. W. C. Faria, "Fluxo optico," 2007. [Online]. Available: http://www.verlab.dcc.ufmg.br/_media/cursos/visao/2007-1/alunos/alexandrewagner/optical_flow_article.pdf
- [2] A. Mitiche and A. reza Mansouri, "On convergence of the horn and schunck optical-flow estimation method," *IEEE*

Transactions on Image Processing, vol. 13, pp. 848–852, 2004.

- [3] B. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision (ijcai),” in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, April 1981, pp. 674–679.
- [4] B. K. P. Horn and B. G. Schunk, “Determining Optical Flow,” *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.
- [5] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.
- [6] MathWorks®, “Matlab - the language of technical computing,” Julho 2011. [Online]. Available: <http://www.mathworks.com/products/matlab/>
- [7] J. L. Barron and N. A. Thacker, “Tutorial: Computing 2d and 3d optical flow,” 2004. [Online]. Available: <http://www.tina-vision.net/docs/memos/2004-012.pdf>
- [8] M. J. Black, “Robust incremental optical flow,” Ph.D. dissertation, Yale University, 1992. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.6628&rep=rep1&type=pdf>
- [9] Z. Fang, X. Li, and L. M. Ni, “On the communication complexity of generalized 2-d convolution on array processors,” *IEEE Transactions on Computers*, vol. 38, pp. 184–194, 1989.
- [10] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.
- [11] S. Baker and I. Matthews, “Lucas-kanade 20 years on: A unifying framework: Part 1,” Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-02-16, July 2002.
- [12] L. J. Van Vliet, I. T. Young, and P. W. Verbeek, “Recursive gaussian derivative filters,” in *IEEE International Conference on Pattern Recognition*, vol. 1, no. August. IEEE Comput. Soc, 1998, pp. 509–514. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=711192>