

Analyse d'image

Estimation de mouvement

Axel Bruyere, Dorian Fabregue
4ETI-IMI

26 mai 2021

- Objectifs** –
- Implémentation de la méthode de *Horn* et *Schunck* pour la détection de flux optique
 - Implémentation de la méthode de *Lucas* et *Kanade* pour la détection de flux optique
 - Comparatif des deux méthodes

Table des matières

1	Contexte	2
2	Algorithme de <i>Horn</i> et <i>Schunck</i>	3
2.1	Introduction	3
2.2	Méthode	3
2.3	Résultats	3
2.4	Conclusion	5
3	Algorithme de <i>Lucas</i> et <i>Kanade</i>	6
3.1	Introduction	6
3.2	Méthode	6
3.3	Résultats	7
3.4	Conclusion	8
4	Conclusion générale	8
5	Annexes	9
5.1	Méthode de <i>Horn</i> et <i>Schunck</i>	9
5.2	Méthode de <i>Lucas</i> et <i>Kanade</i>	11
5.3	Script main	13

1 Contexte

Pour beaucoup de situations, un mouvement entre deux images est aisément interprétable pour l'oeil humain, qui saura détecter la présence et le sens de celui-ci. Dans le cas d'une machine, il s'agit de s'appuyer sur des données tangibles et quantifiables afin d'effectuer cette analyse. L'objectif de ce TP sera alors d'implémenter deux approches de cette problématique : celle proposée par Berthold Horn et Brian Schunck en 1981, et celle publiée par Bruce Lucas et Takeo Kanade en 1981 également. Le but de ces algorithmes est de déterminer le flux optique entre deux images successives. Nous étudierons alors les avantages et inconvénients présentés par ces méthodes. Pour cela, nous travaillerons avec les séquences d'images ci-dessous.

FIGURE 1: Séquences sur lesquelles nous travaillerons

Pour implémenter ces deux méthodes, nous avons besoin de faire deux hypothèses : la première est l'invariance de l'intensité lumineuse d'une image à l'autre.

$$\nabla_x I \cdot u + \nabla_y I \cdot v + \nabla_t I = 0 \quad (1)$$

Avec (u, v) le flux optique

Notre deuxième hypothèse porte sur la faible vitesse de variation locale du champ de vitesse.

$$||\nabla u|| \approx 0, ||\nabla v|| \approx 0 \quad (2)$$

Nous aurons l'occasion par la suite de nous intéresser au cas où ces hypothèses ne sont pas respectées.

2 Algorithme de *Horn et Schunck*

2.1 Introduction

L'algorithme de *Horn et Schunck* est une méthode différentielle globale d'estimation du flux optique entre deux images. Nous allons donc chercher le flux (u, v) minimisant la quantité suivante :

$$\int (\nabla_x I \cdot u + \nabla_y I \cdot v + \nabla_t \cdot I)^2 + \lambda [(\nabla_x u)^2 + (\nabla_y u)^2 + (\nabla_x v)^2 + (\nabla_y v)^2] dx dy$$

Avec $\lambda > 0$ un paramètre jouant sur l'influence de la cohérence spatiale

2.2 Méthode

A cette fin, l'algorithme proposé par *Horn et Schunck* est le suivant :

- On calcule $[I_x, I_y, I_t] = \text{gradients}(I_1, I_2)$
- Pour chaque pixel, on calcule \bar{u}, \bar{v} les moyennes de u et v au voisinage de (i, j)
- On calcule ensuite le flux optique lié au pixel courant :

$$u = \bar{u} - I_x \cdot \frac{I_x \cdot \bar{u} + I_y \cdot \bar{v} + I_t}{\lambda^2 + I_x^2 + I_y^2}$$

- On réitère ces opérations jusqu'à convergence

Les moyennes \bar{u} et \bar{v} au voisinage du pixel courant sont calculées à l'aide d'un filtre moyenneur. Nous avons donc implémenté des filtres 4-voisinage et 9-voisinage non pondérés, mais nous pourrions utiliser d'autres filtres moyenneurs, par exemple en introduisant un poids pour les voisins. Par ailleurs, nous définissons un critère d'arrêt portant sur la différence entre $(u, v)_{it}$ et $(u, v)_{it+1}$.

Vous pouvez retrouver le code en annexes.

2.3 Résultats

Lorsque nous exécutons notre programme sur les images de la Mini Cooper, nous obtenons la séquence suivante :

Nous voyons que les mouvements sont assez bien perçus. L'algorithme s'avère très efficace au niveau du bord du coffre de la voiture. En effet, le mouvement est assez stable, et suit une trajectoire quasiment rectiligne. De plus, l'absence de bruit et de variation dans l'éclairage conditionne idéalement l'algorithme. En revanche, les mouvements de l'homme et de ses vêtements sont bien plus erratiques. Par conséquent, la détection de ceux-ci est bien plus "brouillon".

Lorsque nous appliquons cet algorithme à l'image du jongleur, nous obtenons les résultats suivants :

Sur cette séquence, nous voyons clairement une limite de notre programme : la détection du mouvement des mains est très parasitée par le mouvement du t-shirt. En effet, celui-ci se trouvant en fond derrière les mains, son mouvement vient se superposer à celui que l'on cherche à détecter. Nous arrivons tout de même à détecter qu'il y a un mouvement franc, mais nous ne pouvons pas en retirer un véritable sens ni une vitesse.

2.4 Conclusion

Ces deux séquences d'images nous permettent de mettre en évidence les principales forces et faiblesses de l'algorithme de *Horn* et *Schunck*. En effet, celui-ci permet de très bien détecter les mouvements quasi-rectilignes, mais n'est pas très robuste aux petits mouvements erratiques comme ceux d'un tissu.

Il faut également noter le faible impact que semble avoir le choix du paramètre λ . En faisant varier sa valeur entre 1 et 100, nous ne détectons pas de différence majeure. Ceci pourrait *a priori* s'expliquer par l'absence de problème d'ouverture sur ces deux séquences. La perte de précision sur l'amplitude des vecteurs n'étant pas significative pour ces valeurs, nous n'observons pas de différence flagrante.

3 Algorithme de *Lucas et Kanade*

3.1 Introduction

L'algorithme de *Lucas et Kanade* est une méthode différentielle locale d'estimation du flux optique entre deux images. Nous allons donc chercher le flux (u, v) minimisant la quantité suivante :

$$\sum_{(i,j) \in \mathcal{V}} \|\nabla_x I \cdot u + \nabla_y I \cdot v + \nabla_t I\|^2$$

L'idée est proche de celle d'*Horn et Schunck*, avec une cohérence spatiale introduite par la définition d'un voisinage \mathcal{V} . Cette méthode suppose que le flot est essentiellement constant dans un voisinage local du pixel considéré, et résout l'équation du flot optique pour tous les pixels dans ce voisinage par la méthode des moindres carrés.

3.2 Méthode

Dans cette optique de minimisation, l'algorithme proposé par *Lucas et Kanade* est le suivant :

- on définit Ω le voisinage $n \times n$ de (i, j)

Ensuite, pour chaque pixel (i, j) :

- On définit $[g_x, g_y, g_t] \in \mathbb{R}^{n^2}$ les vectorisés de $\nabla_x I_\Omega, \nabla_y I_\Omega, \nabla_t I_\Omega$
- On calcule $w \in \mathbb{R}^{n^2}$ le poids de chaque voisin, et $W = \text{diag}(w) \in \mathbb{R}^{n^2 \times n^2}$
- On définit $A = [g_x, g_y] \in \mathbb{R}^{n^2 \times n^2}$ et $b = [-g_t] \in \mathbb{R}^{n^2}$
- On calcule la matrice $M = \begin{pmatrix} \sum_{\Omega} w^2(i, j) \cdot (\nabla_x I(i, j))^2 & \sum_{\Omega} w^2(i, j) \cdot (\nabla_x I(i, j) \nabla_y I(i, j)) \\ \sum_{\Omega} w^2(i, j) \cdot (\nabla_y I(i, j) \nabla_x I(i, j)) & \sum_{\Omega} w^2(i, j) \cdot (\nabla_y I(i, j))^2 \end{pmatrix}$

Ensuite pour les images I_1 et I_2 :

- nous calculons les gradients $[I_x, I_y, I_t]$

Ensuite, pour tous les pixels (i, j) :

- nous trouvons (u, v) tels que $M \begin{pmatrix} u \\ v \end{pmatrix} = A^T W^2 b$

Le choix du paramètre n , dimension du voisinage, occupe une place centrale de cet algorithme. Celui-ci doit être impair et pour les figures ci-dessous, nous avons choisi $n = 21$. Cette valeur repose sur un compromis que nous avons fait entre la qualité du résultat et le temps d'exécution.

Vous pouvez retrouver le code en annexes.

3.3 Résultats

Lorsque nous appliquons l'algorithme de *Lucas* et *Kanade* à la séquence de la Mini Cooper, nous obtenons les résultats suivants :

Nous pouvons voir sur cette séquence que la localisation des mouvements est bien capturée. En revanche, l'estimation de la direction de celui-ci ainsi que de son amplitude est moins satisfaisante. Même si les résultats de notre algorithme ne sont pas aberrants, nous ne retrouvons pas la même précision que pour la méthode de *Horn* et *Schunk*. Lorsque nous essayons d'appliquer la méthode de *Lucas* et *Kanade* à la séquence du jongleur, voici ce que nous obtenons :

Nos résultats pour cette séquence s'avèrent être meilleurs que pour la méthode de *Horn* et *Schunk*. En effet, la localisation, le sens et l'amplitude des mouvements sont plutôt bien retranscrits. Cette qualité s'observe très bien au niveau de la main gauche du jongleur.

3.4 Conclusion

Nos résultats diffèrent de ceux obtenus pour la méthode de *Horn et Schunk*. En effet, là où celle-ci performait sur la séquence de la Mini Cooper, la méthode de *Lucas et Kanade* propose de bien meilleurs résultats sur la séquence du jongleur. En effet, en supposant le voisinage suffisamment bien dimensionné, les mouvements de notre jongleur se trouvent être plus ou moins constants au sein de ce voisinage. En revanche, il faudrait choisir un n beaucoup plus grand pour ne pas rencontrer de problème de dimensionnement sur la séquence de la Mini Cooper. Par conséquent, nos résultats sont moins bons pour celle-ci.

4 Conclusion générale

Chacune de ces méthodes présente des forces et des faiblesses. La méthode de *Horn et Schunk* s'avère très efficace sur la séquence de la Mini Cooper. Pour cause, peu de petits mouvements viennent parasiter les mouvements les plus significatifs. Par conséquent, nous obtenons une estimation satisfaisante du flux optique. La méthode de *Lucas et Kanade* quant à elle propose des très bons résultats pour la séquence du jongleur. En effet, les petits mouvements ne sont pas gênants dans l'analyse, de par la nature locale de celle-ci. Le choix de la méthode d'estimation dépend donc beaucoup, sans surprise, de la nature de la séquence à traiter. Par ailleurs, une différence majeure est également à noter : une exécution considérablement plus longue pour l'algorithme de *Lucas et Kanade*. En effet, le nombre très important d'itérations de celui-ci implique une contrainte temporelle conséquente. Celle-ci ne l'est que plus si la résolution de la séquence est haute, ou si le nombre d'images à traiter est important.

5 Annexes

5.1 Méthode de *Horn et Schunck*

```
1 function [u,v,H,W,I1 , it ] = motion_hs(image1,image2,lambda,mean_filter,
2 %Parametres d'entree :
3 % image1 -> adresse de la premiere image
4 % image2 -> adresse de la seconde image
5 % lambda
6 % mean-filter -> filtre moyenneur => '4-voisinage' ou '9-voisinage'
7 % threshold -> critere d'arret
8
9 %Parametres de sortie :
10 % -(u,v) -> le flux optique
11 % -(H,W) les dimensions des images
12 % -it -> nombre d'iterations faites
13
14 %Recuperation des images
15 I1 = im2double(imread(image1));
16 I2 = im2double(imread(image2));
17 [H,W] = size(I1);
18 %Calcul des gradients
19 It = (I2 - I1);
20 [Ix,Iy] = imgradientxy(I1);
21
22 %Initialisation de (u,v) et (u\,v\)
23 u = zeros(H,W);
24 v = zeros(H,W);
25 u_barre = u;
26 v_barre = v;
27
28 %Creation du filtre moyenneur
29 switch mean_filter
30     case '4-voisinage'
31         filtre = 1/4*[0,1,0;
32                         1,0,1;
33                         0,1,0];
34     case '9-voisinage'
35         filtre = 1/9*ones(3,3);
36         filtre(2,2) = 0;
37 end
38
39 %Initialisation du critere d'arret
40 mean_ctrl_u = 10;
41 mean_ctrl_v = 10;
42 mean_u = 0;
43 mean_v = 0;
44
```

```

45 it = 0;%Compteur d'iterations
46
47 %Calcul du flux optique
48 while (abs(mean_ctrl_u - mean_u) > threshold || abs(mean_ctrl_v -
    mean_v) > threshold )
49     it = it + 1;
50     mean_ctrl_u = mean_u;
51     mean_ctrl_v = mean_v;
52     u_barre = imfilter(u, filtre);
53     v_barre = imfilter(v, filtre);
54
55     u = u_barre - Ix.* (Ix.*u_barre + Iy.*v_barre + It) ./ ...
56         (lambda^2+Ix.^2+Iy.^2);
57     v = v_barre - Iy.* (Ix.*u_barre + Iy.*v_barre + It) ./ ...
58         (lambda^2+Ix.^2+Iy.^2);
59
60     mean_u = mean(mean(u));
61     mean_v = mean(mean(v));
62 end

```

5.2 Méthode de Lucas et Kadane

```

1 function [u,v,H,L,I1] = motion_lk(image1,image2,n,mu,sigma,img,
2 nbr_images)
3 %Parametres d'entree
4 % - image1 -> adresse de la premiere image
5 % - image2 -> adresse de la seconde image
6 % - n -> taille du voisinage considere
7 % - mu,sigma -> parametres de la gaussienne pour le poids des
8 % voisins
9 % - img -> numero de la premiere image courante
10 % - nbr_images -> nombre d'images a traiter
11 %Parametres de sortie
12 % - (u,v) -> le flux optique
13 % - (H,W) -> dimensions des images
14 %Recuperation des images
15 I1 = im2double(imread(image1));
16 I2 = im2double(imread(image2));
17 [H,L] = size(I1);
18 %Calcul des gradients
19 It = I2-I1;
20 [Ix,Iy] = imgradientxy(I1);
21 %Poids des voisins
22 wt = linspace(-1,1,n*n);
23 w = 1/(sqrt(2*pi)*sigma)*exp(-(wt-mu).^2/(2*sigma.^2));
24 W = diag(w);
25 W2 = W.^2; %Calcul de W^2 hors de la boucle (temps d'execution)
26 %Initialisation de (u,v)
27 u = zeros(H,L);
28 v = zeros(H,L);
29 %Variable pour le dimensionnement
30 k = n-fix(n/2)-1;
31 iter = 0;
32 for i = 1+k:H-k
33     for j = 1+k:L-k
34         iter = iter + 1;
35         %Vectorisation des gradients (1*n^2)
36         gt = reshape(It(i-k:i+k,j-k:j+k),[],1);
37         gx = reshape(Ix(i-k:i+k,j-k:j+k),[],1);
38         gy = reshape(Iy(i-k:i+k,j-k:j+k),[],1);
39     end
40 end
41
42
43
44
45

```

```

46 %Matrices A et b
47 A = [gx,gy];
48 b = -gt;
49
50 %Matrice M
51 M = [sum((w(:).^2).*gx(:).^2),sum((w(:).^2).*gx(:).*gy(:));
52 sum((w(:).^2).*gx(:).*gy(:)),sum((w(:).^2).*(gy(:).^2))];
53
54 uv = pinv(M)*A'*W2*b;
55 u(i,j) = uv(1);
56 v(i,j) = uv(2);
57 if rem(iter,5000) == 0
58     it_disp = [ 'Iteration ',int2str(iter), '/', int2str((H-2*k)*(L
59             -2*k)+1),...
59         ' -> Image ',int2str(img), '/', int2str(nbr_images - 1)]
60     end
61 end
62 end
63 it_disp = [ 'Iteration ',int2str((H-2*k)*(L-2*k)+1), '/', int2str((H-2*k)*(L
63             -2*k)+1),...
64         ' -> Images ',int2str(img), ' - ', int2str(img+1)]

```

5.3 Script main

```

1 clc; close all; clear variables;
2
3 %%Parametres
4 nbr_images = 8; %Nombre d'images a traiter
5 dir = 'MiniCooper'; %Choix des images a traiter
6 method = 'LK'; %Choix de l'algorithme ('HS' pour Horn et Schunk,
7 % % pour Lucas et Kanade)
8
9 %%Parametres de l'algorithme de Horn et Schunk
10 lambda = 1;
11 threshold = 1e-07; %Seuil pour le critere d'arret
12 mean_filter = '25-voisinage'; %Choix des dimensions du moyennage
13
14 %%Parametres de l'algorithme de Lucas et Kanade
15 n = 21; %Dimensions du voisinage (matrice n * n, n impair pour
16 %centrer le pixel courant)
16 mu = 0; sigma = 1; %Parametre de la gaussienne (poids des voisins)
17
18 %%Parametres d'affichage
19 display_step = 6; %Pas d'affichage du champ de vecteurs
20 im2display = 2;%Choix de l'image a afficher seule (affiche le mouvement
21 %entre les images i et i+1)
22
23
24 for i = 1:nbr_images-1
25     image1_f = [dir , '/i000' , int2str(i) , '.png'];
26     image2_f = [dir , '/i000' , int2str(i+1) , '.png'];
27
28 %%Estimation du flux
29     switch method
30         case 'HS'
31             [u,v,H,W,I1,it] = motion_hs(image1_f,image2_f,lambda,
32                                         mean_filter,threshold);
32         case 'LK'
33             [u,v,H,W,I1] = motion_lk(image1_f,image2_f,n,mu,sigma,i,
34                                         nbr_images);
34     end
35
36
37 %%Affichage
38 [X,Y] = meshgrid(1:W,1:H);
39 figure(i)
40 % subplot(2,fix(nbr_images/2),i)
41 imshow(I1,[])
42 hold on
43 quiver(X(1:display_step:H,1:display_step:W),Y(1:display_step:H,1:
44 display_step:W),...

```

```

44      u(1:display_step:H,1:display_step:W), v(1:display_step:H,1:
        display_step:W),5,'linewidth',0.5)
45      title(sprintf('Flux optique entre les images %d et %d',i,i+1))
46
47      %Enregistrement des figures
48      % newdir = [ dir ,method ];
49      % mkdir(newdir)
50      % filename = [ 'fig ',int2str(i)];
51      % saveas(gca, fullfile(newdir, filename), 'png');
52
53      %Affichage d'une seule figure : decommenter au besoin et choisir
      % dans
54      %les parametres la figure a afficher (flux optique entre l'image
      % im2display et
55      %l'image im2display+1)
56      % if i == im2display
57      %     figure()
58      %     imshow(I1,[])
59      %     hold on
60      %     quiver(X(1:display_step:H,1:display_step:W),Y(1:display_step:
      H,1:display_step:W),...
61      %             u(1:display_step:H,1:display_step:W), v(1:display_step:H,1:
      display_step:W),5,'linewidth',0.5)
62      %             title(sprintf('Flux optique entre les images %d et %d',i,i+1))
63      %         )
64      %     end
64 end

```