Détection de mouvement entre deux images

Gwenael Mercier*

Ce projet utilise des images. Malheureusement, Scilab ne dispose pas de bibliothèque native pour leur traitement. On trouvera en annexe une façon de contourner cette absence.

On n'hésitera pas à contacter l'auteur du sujet pour toute difficulté technique, ou si une question paraît trop vague, n'est pas comprise, semble fausse ou trop difficile.

Dans toute l'implémentation Scilab, on évitera au maximum d'utiliser les boucles `for`, on préfèrera des calculs purement matriciels (on les utilisera tout de même dans les méthodes itératives). Leur utilisation superflue sera (légèrement) pénalisée.

De plus, un certain nombre de fonctions seront à tester sur de vraies images. On pourra choisir des images librement (deux images consécutives d'une séquence vidéo qui ne bouge pas trop peuvent être de bonnes candidates). On trouvera néanmoins des exemples ici : `http://vision.middlebury.edu/flow/data/comp/zip/eval-gray-twoframes.zip`.

Enfin, le correcteur appréciera d'avoir un code propre, commenté et correctement indenté.

1 Position du problème

Soit $I(x, y, t)$ une séquence video (en noir et blanc, dans toute la suite, c'est-à-dire une application de $[0, 1]^2 \times [0, T]$ dans $\mathbb{R}$). Le but de ce projet est de calculer la trajectoire $(x(t), y(t)$ d'un point matériel tout au long de la séquence. Plus précisément, on souhaite connaître le déplacement des points à chaque instant de la séquence :

$$h = \left(\frac{dx}{dt}, \frac{dy}{dt}\right) = (u, v).$$

On définit en outre $\tilde{I}(t) = I(x(t), y(t), t)$. Commençons par quelques hypothèses.

On suppose dans la suite que la luminosité des points est constante au cours du temps, c'est-à-dire que

$$\frac{d\tilde{I}}{dt} = 0.$$

Q1. Montrer que

$$\nabla I \cdot h + \partial_t I = 0.$$

Peut-on résoudre cette équation directement en chaque point sans hypothèse supplémentaire ?

2 Méthode de Horn et Schunk

Ces deux auteurs proposent d'ajouter l'hypothèse de régularité du flot pour obtenir une solution à l'équation. Plus précisément, ils choisissent de déterminer h par une méthode variationnelle : ils proposent de minimiser l'énergie suivante sur Ω, domaine de définition de l'image.

$$J(u, v) = \int_\Omega (I_x u + I_y v + I_t)^2 + \alpha^2(|\nabla u|^2 + |\nabla v|^2), \qquad u, v \in H^1(\Omega).$$

Que se passe-t-il si $\alpha = 0$?

Q2. Ce minimum existe-t-il ? Est-il unique ?

*gwenael.mercier@cmap.polytechnique.fr

Q3. Montrer qu'au point de minimum, on a

$$I_x^2 u + I_x I_y v = \alpha^2 \operatorname{div}(\nabla u) - I_x I_t$$

$$I_x I_y u + I_y^2 v = \alpha^2 \operatorname{div}(\nabla v) - I_y I_t.$$

Ce sont ces deux équations que l'on va chercher à résoudre dans la suite.

Implémentation On va résoudre ces équations par une méthode aux différences finies. On suppose avoir deux images carrées (c'est-à-dire deux applications de $[0,1]^2 \to [0,1]$, où 1 correspond à une image blanche et 0 à une image noire) I_1 et I_2 qui se succèdent dans la séquence. Informatiquement, ces deux images ne sont rien d'autre que des matrices carrées dont on notera les coefficients $I_{1,i,j}$ et $I_{2,i,j}$. Attention néanmoins : le coefficient $I_{1,1,1}$ correspond au coin en haut à gauche de la première image, $I_{1,n,1}$ le coin en bas à gauche, etc.

Q4. Écrire une fonction Scilab qui calcule les quantités I_x, I_y et I_t en utilisant le schéma suivant

$$I_x = \frac{1}{4}(I_{1,i,j+1} - I_{1,i,j} + I_{1,i+1,j+1} - I_{1,i+1,j} + I_{2,i,j+1} - I_{2,i,j} + I_{2,i+1,j+1} - I_{2,i+1,j})$$

$$I_x = \frac{1}{4}(I_{1,i+1,j} - I_{1,i,j} + I_{1,i+1,j+1} - I_{1,i,j+1} + I_{2,i+1,j} - I_{2,i,j} + I_{2,i+1,j+1} - I_{2,i,j+1})$$

$$I_t = \frac{1}{4}(I_{2,i,j} - I_{1,i,j} + I_{2,i+1,j} - I_{1,i+1,j} + I_{2,i+1,j+1} - I_{1,i+1,j+1} + I_{2,i,j+1} - I_{1,i,j+1}$$

Avec condition de Neumann à la frontière (c'est-à-dire qu'on prolonge I a tout $\mathbb{Z}^2$ en posant que $I_{i,j} = I_{1,j}$ pour $i \leqslant 1$, etc., de sorte que la dérivée verticale discrète $I_{i+1,j} - I_{i,j}$ soit nulle à la frontière $i = 0$.).

Q5. Écrire une fonction qui, étant donnée une matrice U, retourne la matrice $\overline{U}$ des coefficients de U moyennés :

$$\overline{U}_{i,j} = \frac{1}{6}(U_{i-1,j} + U_{i+1,j} + U_{i,j+1} + U_{i,j-1}) + \frac{1}{12}(U_{i-1,j-1} + U_{i-1,j+1} + U_{i+1,j-1} + U_{i+1,j+1})$$

Q6. Avec ces notations, le calcul de la solution approchée se résumé à la résolution d'un système linéaire. Lequel ? Montrer que le résoudre de manière approchée grâce à la méthode de Jacobi [1] conduit au schéma itératif suivant (u, v désignent un même coefficient (quelconque) de U et V).

$$u^{n+1} = \overline{u}^n - I_x \frac{I_x \overline{u}^n + I_y \overline{v}^n + I_t}{\alpha^2 + I_x^2 + I_y^2},$$

$$v^{n+1} = \overline{v}^n - I_y \frac{I_x \overline{u}^n + I_y \overline{v}^n + I_t}{\alpha^2 + I_x^2 + I_y^2}.$$

On définira l'erreur par $err = \operatorname{Tr}({}^t(U^{n+1} - U^n)(U^{n+1} - U^n)) + \operatorname{Tr}({}^t(V^{n+1} - V^n)(V^{n+1} - V^n))$

Q7. Implémenter le programme (on arrêtera le schéma itératif lorsque l'erreur sera inférieure à un paramètre d'entrée ε ou lorsque le nombre d'itérations dépassera un paramètre d'entrée `niter`), et le tester avec deux images. Commenter.

1. Si l'on n'a pas entendu parler de cette méthode, on n'hésitera pas à consulter le livre de Grégoire Allaire, *Algèbre linéaire numérique*.

3 Une autre minimisation

Q8. Il n'y a en fait pas vraiment de raison que le champ de mouvement soit lisse dans les directions de fort gradient (ils correspondent typiquement au passage d'un objet physique à un autre, qui n'ont pas de raison de se déplacer de la même façon). On va donc remplacer le terme régularisant $\int |\nabla u|^2 + |\nabla v|^2$ (norme L^2 des gradients des déplacements) par la norme L^2 de leur projection sur $\nabla g^\perp$. Montrer qu'on est alors amené à minimiser

$$J(u,v) = \int_\Omega (I_x u + I_y v + I_t)^2 + \alpha^2 \operatorname{Tr}((\nabla h)^T \tilde{W} \nabla h)$$

où

$$\tilde{W} = \frac{1}{I_x^2 + I_y^2} \begin{bmatrix} I_y^2 & -I_x I_y \\ -I_x I_y & I_x^2 \end{bmatrix}.$$

Il y a néanmoins un dernier problème : la matrice W n'est pas définie lorsque l'intensité I a un gradient nul : pour pallier ce problème, on la remplace donc par

$$W = \frac{1}{I_x^2 + I_y^2 + 2\gamma} \begin{bmatrix} I_y^2 + \gamma & -I_x I_y \\ -I_x I_y & I_x^2 + \gamma \end{bmatrix}.$$

Lorsque $DI = 0$, on retrouve donc le facteur régularisant de la méthode de Horn et Schunk.

Q9. Calculer les équations d'Euler Lagrange pour u et v associées à ce problème de minimisation, et déterminer les formulations variationnelles associées (avec conditions de Neumann au bord de Ω).

Q10. Implémenter ce problème avec FreeFem++. On utilisera des éléments finis $P1$ bidimensionnels. Le terme I_t sera estimé simplement par $\delta I := I_2 - I_1$. On choisira un maillage adapté au problème : on commencera par introduire un maillage structuré qu'on raffinera grâce aux quantités I_1 et I_2, en utilisant la commande `adaptmesh`[2]. Faire varier α (on pourra prendre 15 comme valeur de référence). Commenter.

Pour améliorer les résultats, on doit s'assurer que δI est bien constitué de points qui représentent une différence entre les deux images, et pas juste un bruit. Aussi, il paraît judicieux de lisser légèrement les images avant de leur appliquer l'algorithme.

3.1 Lissons une image

Dans cette partie, les conditions au bord seront des conditions de réflexion, c'est-à-dire que si $I \in \mathbb{R}^{n\times n}$, on prolonge I en posant, pour $i \in [\![n+1, 2n]\!]$, $j \in [\![1, n]\!]$, $I(i,j) = I(-(i-n), j)$, etc. On va effectuer un lissage gaussien, c'est-à-dire que chaque point (x_0, y_0) de I est remplacé par une moyenne sur ses voisins, moyenne pondérée par les valeurs d'une gaussienne[3] centrée en (x_0, y_0). Mathématiquement, il s'agit évidemment de convoler I avec G.

Q11. Implémenter ce lissage sous Scilab et le tester sur différentes images, avec différentes valeurs de σ.

Q12. (facultative). Discrétiser l'équation de la chaleur $u_t = \Delta u$ avec un schéma aux différences finies de votre choix (mais qui converge !) et l'appliquer aux images utilisées dans la question précédente (pour différents temps pas trop grands). Le résultat vous étonne-t-il ? Pourquoi ?

Q13. Faire tourner un des deux alorithmes précédents avec une paire d'images bruitées, puis avec la même paire d'images préalablement lissées. Comparer les résultats et commenter.

2. Cette commande prend un maillage en entrée, ainsi qu'une ou plusieurs fonctions auxquelles le maillage de sortie doit être adapté. En option, elle peut de plus prendre comme paramètre l'erreur d'interpolation en ajoutant comme argument `err=...` (voir annexe). La valeur par défaut de cette erreur est 0.01.

3. Courbe de la forme $G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}}$

4 Prise en charge de grands déplacements

Cette partie est plus compliquée. Il est préférable de ne l'aborder qu'après avoir traité les questions précédentes.

Plus haut, on a supposé que les déplacements étaient petits par rapport aux variations de l'image. On va essayer ici de prendre en compte des déplacements plus importants comparés à la résolution de l'image. Pour ce faire, on va réduire cette résolution et calculer le flot optique. On va ensuite remettre le flot à l'échelle originale, déformer I_2 grâce à ce flot et relancer un calcul pour affiner le résultat.

4.1 Déformons une image avec un flot : un peu d'interpolation

On dispose d'une image I et d'un flot h. On souhaite calculer $I(x-u, y-v)$ (avec conditions de Neumann à la frontière). Pour Scilab, on dispose donc d'une matrice I_{ij} et de deux matrices du flot U_{ij} (qui agit sur les lignes) et V_{ij} (qui agit sur les colonnes). On a naturellement envie de définir, si $\tilde{I}$ est l'image modifiée : $\tilde{I}_{ij} = I_{u_{ij}+i, v_{ij}+j}$.

Problème : les coefficients de U et V n'ont aucune raison d'être entiers. On va donc interpoler à partir des coefficients entiers les plus proches, de la façon suivante.

Cas 1D. Soit $H : [\![1, n]\!] \to \mathbb{R}$ un vecteur ligne quelconque. On veut calculer la translation $\tilde{H}(x) = H(x + u(x))$, avec $u : [\![1, n]\!] \to \mathbb{R}$ quelconque (schématiquement, on veut $\tilde{H}_i = H_{i+u_i}$). On calcule alors le i^{e} coefficient de $\tilde{H}$ (noter que si $i \leqslant 0$ et si $j \geqslant n$, les conditions de Neumann permettent d'écrire $H(i) = H(1)$ et $H(j) = H(n)$) en définissant

$$\begin{gathered} sx = \operatorname{sgn}(x)\ ^a \\ ui = \text{l'entier le plus proche de } u_i + i \\ mx = ui - sx \\ x = ui \\ dx = ui + sx \\ ddx = ui + 2sx. \end{gathered}$$

a. On souhaite ici que le signe de zéro soit 1. On se méfiera donc de la fonction `sign` de Scilab...

On pose alors

$$\tilde{H}_i = H_x + 0.5x \cdot (H_{dx} - H_{mx} + x(2H_{mx} - 5H_x + 4H_{dx} - H_{ddx} + x(3(H_x - H_{dx}) + H_{ddx} - H_{mx}))).$$

Q14. On appelle cette interpolation *cubique*. Justifier cette appellation. Écrire une fonction qui prend en entrée u et H et qui retourne $H(x + u)$.

Cas 2D. En deux dimensions, on va simplement appliquer la procédure précédente dans les deux directions : on calcule $I(x + u, y)$ par la méthode précédente (on obtient une matrice $\hat{I}$) puis on calcule $\hat{I}(x, y + v)$ de la même façon.

Q15. Implémenter cette méthode et la tester sur une image de votre choix (essayer notamment des flots non constants).

4.2 Redimensionnons une image

Q16. Lors de la réduction d'une image, le même type de problème se pose. Écrire, en utilisant la même interpolation, une procédure qui prend en entrée une image de taille $n \times n$ et un facteur de réduction $\eta \in]0, 1[$ et qui retourne la même image, mais réduite du facteur η (de taille $E(\eta n) \times E(\eta n)$, où E désigne la partie entière).

Q17. Adapter la procédure précédente à l'agrandissement d'images ($\eta > 1$). Comparer le résultat à celui proposé par la commande `convert` d'Imagemagick.

4.3 Un deuxième calcul du flot optique

On souhaite donc conduire un calcul multiéchelles du flot optique de la façon suivante.

Q18. Écrire une fonction qui prend en entrée deux images I_1 et I_2, un facteur de réduction η et qui
- lisse les images I_1 et I_2 avec un filtre gaussien d'écart type $\sigma = 0.8$,
- réduit les images lissées d'un facteur η,
- fait tourner un des programmes précedents (Q7/Q10) sur les images réduites (on obtient un flot h_η),
- remet le flot obtenu à l'échelle originale (on obtient un flot h),
- refait tourner le programme Q7/Q10 sur les images de taille originale I_1 et $I_2(\cdot - h)$ (on obtient un flot k),
- retourne le flot affiné $h + k$.

Tester ce programme [4].

N.B. Si vous trouvez ce projet trop court ou trop facile, ou si vous avez tout simplement envie d'en savoir un peu plus, sachez qu'il est possible de programmer un calcul de flot optique pyramidal, c'est-à-dire où la procédure de la question 15 est appliquée à beaucoup d'échelles différentes. Si vous voulez essayer, n'hésitez pas ! Vous pouvez bien sûr me contacter si vous voulez des détails.

A Quelques commandes utiles pour utiliser les images

A.1 Importer une image sous Scilab

On suppose disposer d'une image en niveaux de gris [5] carrée `image.jpg` comprenant $n \times n$ pixels. On souhaite, pour que Scilab puisse jouer avec, la transformer en une matrice carrée I de taille n dont le coefficient I_{ij} vaut la luminosité du pixel (i, j) (origine en haut à gauche, coefficient (n, n) en bas à droite). Ceci peut-être fait de deux façons.
- En utilisant Matlab, il est possible de lire directement l'image avec la commande `imread`, puis d'exporter la matrice obtenue par `dlmwrite`.
- En utilisant Octave (logiciel librement téléchargeable), on peut faire la même chose : commande `imread` pour lire l'image puis commande `dlmwrite` pour écrire dans un fichier.

On peut ensuite importer ce fichier texte sous Scilab grâce à la commande `fscanfMat`.

A.2 Importer une image sous FreeFem

On va utiliser Scilab pour transformer la matrice générée à la question précédente en un fichier texte de la forme xyf, lisible par FreeFem. Pour ce faire, il faut transformer la matrice $n \times n$ $[I_{ij}]$ en une matrice J de taille $n^2 \times 3$ telle que la ligne générique de J soit composée de $[i, j, I_{ij}]$.

Pour construire J, on peut simplement faire :

```
[X,Y] = meshgrid(1:size(I));
J=[matrix(X,size(I,1)^2,1),matrix(Y,size(I,1)^2,1),matrix(I,size(A,1)^2,1)];
```

On exporte ensuite la matrice J dans un fichier texte grâce à la commande `printfMat`.

Il faut alors importer ce fichier dans FreeFem : voir le fichier joint au projet qui explique comment faire à travers un exemple simple. Si vous avez des difficultés dans cette étape, contactez-moi (inutile de perdre du temps...).

4. Cette question est un peu plus difficile. On n'hésitera pas à présenter du code, même s'il ne fonctionne pas ou n'est pas fini. Toute tentative à peu près pertinente sera valorisée.

5. Pour transformer une image couleur en niveaux de gris, on peut taper en ligne de commande `convert imagecouleur colorspace gray imageniveauxgris`.

A.3 Afficher une image avec Scilab

Pour afficher une matrice carrée sous forme d'image, on va procéder comme dans le premier point, mais à l'envers. On commence par écrire la matrice dans un fichier à l'aide de la commande `printfMat`. On importe alors la matrice sous Matlab ou Octave grâce à `dlmread`, et on la visualise avec `imshow`.