

TP8-2013. Flot optique d'une image, méthodes du gradient

Les deux exercices de ce TP sont adaptés d'un TP proposé par Gabriel Peyré et que l'on trouvera en ligne sous ce lien :

```
>> https://www.ceremade.dauphine.fr  
/~peyre/numerical-tour/tours/multidim\_5\_opticalflow/
```

Je l'ai étoffé mais m'en suis fortement inspiré.

EXERCICE 1 (Calcul du flot optique : méthode du gradient ordinaire suivant Horn-Schunck (1981)).

- (1) Soient I , ΔI , deux images numériques réelles de même taille $[N1, N2]$. L'image ΔI est une image obtenue comme $\Delta I = I - I_{\text{deplacee}}$, où I_{deplacee} est une image déduite de I par un déplacement infinitésimal (par exemple une rotation de petit angle θ). Soient $\nabla I1$, $\nabla I2$ les composantes du gradient de l'image I :

```
>> [ $\nabla I1, \nabla I2$ ] = Gradient( $I$ );
```

(voir la routine réalisée au TP3, on pourrait aussi utiliser la routine pré-implémentée `gradient`). Soit aussi λ un paramètre réel positif. On désigne par $E_{N1, N2}$ le \mathbb{R} -espace vectoriel des images numériques réelles de taille $N1, N2$ (pixelisées au dessus de la grille $[1:N1] \times [1:N2]$) et Φ la fonctionnelle :

$$\begin{aligned} \Phi : (v_1, v_2) \in E_{N1, N2}^2 \mapsto & \\ & \text{norm}^2(\Delta I + v_1 * \nabla I1 + v_2 * \nabla I2) + \\ & + \lambda (\text{norm}^2(\text{Gradient}(v_1)) + \text{norm}^2(\text{Gradient}(v_2))) \end{aligned}$$

(la norme des tableaux de taille $[N1, N2]$ étant ici la norme euclidienne). Il s'agit donc ici de la discrétisation de la fonctionnelle (agissant cette fois sur les couples (v_1, v_2) d'images analogiques différentiables et à support compact) :

$$\begin{aligned} \Phi : (v_1, v_2) \mapsto & \iint \left[(\Delta I + \nabla I1 \times v_1 + \nabla I2 \times v_2)^2 + \right. \\ & \left. + \lambda (\|\vec{\nabla} v_1\|^2 + \|\vec{\nabla} v_2\|^2) \right] dx dy \end{aligned}$$

(où ΔI , $\nabla I1$, $\nabla I2$, dénotent ici les versions analogiques des images digitales ΔI , $\nabla I1$, $\nabla I2$). Montrer, en utilisant la

formule de Pythagore, puis la formule d'intégration par parties, que

$$\begin{aligned} \Phi(v_1 + h_1, v_2 + h_2) &= \Phi(v_1, v_2) + \\ &+ 2 \iint (\text{deltaI} + \text{nablaI1} \times v_1 + \text{nablaI2} \times v_2) \\ &\quad \times (\text{nablaI1} \times h_1 + \text{nablaI2} \times h_2) \, dx dy \\ &- 2\lambda \iint (\Delta[v_1] \times h_1 + \Delta[v_2] \times h_2) \, dx dy + o(|(h_1, h_2)|), \end{aligned}$$

où Δ désigne le laplacien. En déduire que le gradient `GradientPhiVV` au vecteur courant `VV` de la fonctionnelle Φ , représenté par un tableau `VV` de taille `[N1,N2,2]` (deux images de taille `[N1,N2]`), est donné par la suite d'instructions décrites ci-dessous :

```
>> NablaI = cat(3,NablaI1,NablaI2);
>> UU = DeltaI + sum(VV.*NablaI,3);
>> [nabla11,nabla12] = Gradient(VV(:, :, 1));
>> [nabla21,nabla22] = Gradient(VV(:, :, 2));
>> LL = cat(3, Divergence(nabla11,nabla12),
            Divergence(nabla21,nabla22));
>> GradientPhiVV = 2*(NablaI.*repmat(UU,[1 1 2])
                    - lambda *LL);
```

(on examinera pour cela le `help` des routines `cat` et `repmat` utilisées ci-dessus).

- (2) La méthode classique du gradient (ordinaire) consiste, étant donnée une fonctionnelle convexe Φ (sur un \mathbb{R} -espace vectoriel E de dimension finie, ici l'espace des couples de matrices de taille `[N1,N2]`, soit l'espace des tableaux de taille `[N1,N2,2]`) que l'on entend minimiser, consiste, étant donné un pas `tau`, à itérer l'algorithme suivant :

```
>> VV = zeros (dimensions (E));
>> GV = GradientPhiVV;
>> VV = VV - tau*GV;
```

L'itération de cet algorithme conduit à l'approximation d'un minimum de la fonctionnelle Φ lorsque celle-ci (comme c'est le cas ici) est strictement convexe ainsi qu'une approximation `VV` du point où ce minimum se trouve réalisé. Rédiger une routine

```
>> [OptFlot,EnergiePhi] =
    FlotOptimal1(DeltaI,I,lambda,tau,iter);
```

réalisant le résultat obtenu (`Optflot`) au bout de `iter` itérations de l'algorithme précédent (avec un pas fixé à `tau`) pour la fonctionnelle Φ attachée aux images `DeltaI,I` (le paramètre de relaxation valant `lambda`), ainsi que la valeur de la fonctionnelle Φ en tous les résultats intermédiaires de l'itération. Constater la décroissance de la variable `EnergiePhi` au fil des itérations.

EXERCICE 2 (Calcul du flot optique : méthode du gradient conjugué).

- (1) Si VV désigne un tableau numérique de taille $[N1, N2, 2]$ où l'on note $V1=VV(:, :, 1)$ et $V2 = VV(:, :, 2)$, et que

$$VV \mapsto \text{GradientPhi}VV$$

désigne l'application qui à $(V1, V2)$ associe le gradient $\text{GradientPhi}VV$ de $\Phi_{\text{DeltaI}, I}$ au point $(V1, V2)$ (voir l'exercice 1), vérifier que cette application s'exprime sous la forme d'une application affine :

$$VV \mapsto 2 * (\text{reshape}(\mathbf{A}(VV(:)), [N1, N2, 2]) + \text{NablaI} .* \text{cat}(3, \text{DeltaI}, \text{DeltaI})),$$

où l'application linéaire A agit ainsi sur un vecteur X :

```
function Y = FonctionLineaireA(X, tmp);
global lambda ; global NablaI ;
XX = reshape(X, [N1, N2, 2]);
UU0 = sum(XX .* NablaI, 3);
[nabla11, nabla12] = gradient(XX(:, :, 1));
[nabla21, nabla22] = gradient(XX(:, :, 2));
LL = cat(3, divergence(nabla11, nabla12),
         divergence(nabla21, nabla22));
YY = NablaI .* repmat(UU0, [1, 1, 2]) - lambda * LL;
Y = YY(:);
```

(NablaI étant le tableau de taille $[N1, N2, 2]$ défini à partir des deux images NablaI1 et NablaI2 dans l'exercice 1). Implémenter l'algorithme du gradient conjugué pour la résolution approchée du système linéaire

$$\mathbf{A}(VV(:)) = -\text{NablaI} .* \text{cat}(3, \text{DeltaI}, \text{DeltaI})$$

en utilisant la routine `cgs` (Conjugate Gradient Squares) :

```
>> B = - NablaI .* cat(3, DeltaI, DeltaI);
>> [Y, FLAG, RELRES, ITER, RESVEC] =
    cgs(@FonctionLineaireA, B(:), tol, maxITER);
>> VV = reshape(Y, [N1, N2, 2]);
```

(se reporter au `help` de `cgs` pour l'analyse des sorties). Pour la description du synopsis de l'algorithme de descente dit *du gradient conjugué* sur lequel se trouve bâtie la routine `MATLAB cgs`, on pourra par exemple se reporter au polycopié (en ligne) du cours suivant :

<http://www.math.u-bordeaux1.fr/~yger/mht632.pdf>

(pages 37-38). Ici `RELRES` dénote la norme résiduelle relative, `RESVEC` le vecteur des normes résiduelles à chaque itération, `ITER` désignant le nombre d'itérations (au plus égal au nombre `maxITER` imposé *a priori*). La variable `tol` précise la tolérance pour l'erreur.

- (2) Rédiger une routine

```
>> E = EnergiePhi(VV, DeltaI, I, lambda);
```

qui calcule la valeur de la fonctionnelle Φ de l'exercice 1 au point VV (donné comme un tableau de taille $[N1, N2, 2]$, où $[N1, N2]$ est la taille de I et de DeltaI). Calculer le résultat obtenu lorsque VV figure le résultat obtenu *via* l'algorithme du gradient conjugué au terme de la question 1.

Comparer avec la valeur de Φ lorsque VV est le tableau identiquement nul ou lorsque VV est le résultat obtenu comme valeur finale de la variable `EnergiePhi` dans la routine `FlotOptimal1` réalisée à l'exercice 1.

- (3) Pour visualiser le résultat obtenu dans les questions précédentes, on utilisera la routine `visualisationFlotOptimal` réalisée avec l'aide des routines proposées par Gabriel Peyré (`imageplot.m` et `rescale.m`). On téléchargera ces routines au préalable sur :

<http://www.math.u-bordeaux1.fr/~ayger/MATLABSignal/RoutinesTP/RoutinesPeyre>

L'image `I` que l'on pourra exploiter sera `lena.jpg`, le déplacement étant une rotation d'angle petit (par exemple $3 * \pi / 200$).

La routine `rotationdirectcrop.m` ramènera dans ce cas l'image à une image tournée de même taille par tronquature du résultat `Irot` obtenu avec `rotationdirect.m`; lorsque l'angle est petit, la tronquature d'image n'engendre en effet quasiment pas de perte d'information.