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 `DeltaI,I`, deux images numériques réelles de même taille `[N1,N2]`. L'image `DeltaI` est une image obtenue comme `DeltaI = I - Ideplacee`, où `Ideplacee` est une image déduite de I par un déplacement infinitésimal (par exemple une rotation de petit angle `theta`). Soient `NablaI1, NablaI2` les composantes du gradient de l'image `I` :

```
>> [NablaI1,NablaI2]= 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 $E_{\texttt{N1,N2}}$ le $\mathbb{R}$-espace vectoriel des images numériques réelles de taille `N1,N2` (pixelisées au dessus de la grille `[1:N1] x [1:N2]`) et $\mathbf{\Phi}$ la fonctionnelle :

$$\begin{aligned}\mathbf{\Phi} \ : \ &= \texttt{(V1,V2)} \in E^2_{\texttt{N1,N2}} \longmapsto \\ &\texttt{norm}^2\Big(\texttt{DeltaI+ V1.*NablaI1+ V2.*NablaI2}\Big) + \\ &+\lambda\Big(\texttt{norm}^2(\texttt{Gradient(V1)}) + \texttt{norm}^2(\texttt{Gradient(V2)})\Big)\end{aligned}$$

(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 images différentiables et à support compact) :

$$\begin{aligned}\Phi \ : (v_1, v_2) \longmapsto \iint \Big[\big(\texttt{deltaI} + \texttt{nablaI1} \times v_1 + \texttt{nablaI2} \times v_2\big)^2 + \\ +\lambda\left(\|\vec{\nabla} v_1\|^2 + \|\vec{\nabla} v_2\|^2\right)\Big]\, dxdy\end{aligned}$$

(où `deltaI, nablaI1, nablaI2`, dénotent ici les versions analogiques des images digitales `DeltaI, NablaI1, NablaI2`). Montrer, en utilisant la

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

$$\begin{aligned}\Phi(v_1+h_1,v_2+h_2) &= \Phi(v_1,v_2)+\\ &+2\iint \big(\texttt{deltaI}+\texttt{nablaI1}\times v_1+\texttt{nablaI2}\times v_2\big)\\ &\quad\times\big(\texttt{nablaI1}\times h_1+\texttt{nablaI2}\times h_2\big)\,dxdy\\ &+2\iint \big(\Delta[v_1]\times h_1+\Delta[v_2]\times h_2\big)\,dxdy+o(|(h_1,h_2)|).\end{aligned}$$

En déduire que le gradient `GradientPhiVV` au vecteur courant `VV` de la fonctionnelle $\mathbf{\Phi}$, 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 $\mathbf{\Phi}$ (sur un $\mathbb{R}$-espace vectoriel 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.

(3) Rédiger une routine

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

réalisant le résultat obtenu (`OptFlot`) au terme de `iter` itérations de l'algorithme précédent (avec un pas fixé à `tau`) pour la fonctionnelle $\mathbf{\Phi}$ attachée aux deux images `DeltaI,I` (le paramètre de relaxation valant `lambda`), ainsi que la valeur (`EnergiePhi`) de la fonctionnelle $\mathbf{\Phi}$ en toutes les étapes intermédiaires `VV` 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

$$\texttt{VV} \longmapsto \texttt{GradientPhiVV}$$

est l'application qui à `(V1,V2)` associe le gradient `GradientPhiVV` de $\mathbf{\Phi}_{\texttt{DeltaI,I},\lambda}$ au point `(V1,V2)` (voir l'exercice **1**), vérifier que cette application s'exprime sous la forme d'une application affine :

$$\texttt{VV} \longmapsto 2*\big(\texttt{reshape }(\mathbf{A}_\lambda\texttt{(VV(:)),[N1,N2,2])+NablaI.*cat(3,DeltaI,DeltaI)}\big),$$

où l'application linéaire $\mathtt{A}_\lambda$ 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

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

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 = FonctionEnergiePhi(VV,DeltaI,I,lambda);
```

qui calcule la valeur de la fonctionnelle $\mathbf{\Phi}$ 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 la valeur de $\mathbf{\Phi}$ au point `VV` obtenu via l'algorithme du gradient conjugué élaboré à la question **1**. Comparer avec la valeur

prise par $\mathbf{\Phi}$ au point `VV=zeros(N1,N2,2)` ainsi qu'avec la valeur finale de la variable `EnergiePhi` obtenue avec la routine `FlotOptimal1` réalisée à l'exercice **1**, question **3**.

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

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

La commande

```
>> VisualisationFlotOptimal(VV,I,pas1,pas2);
```

fournit d'une part une image « volumétrique » (en couleurs) du tableau de taille `[N1,N2,2]` correspondant au point minimisant `VV` (image de gauche sur la figure produite) ; sur l'image de droite de la même figure, le flot optique obtenu est indiqué par des flèches (outil graphique `quiver`) aux pixels `[1:pas1:N1] x [1:pas2:N2]`, ce en superposition à l'affichage de l'image `I` (le choix de `pas1` et de `pas2` de l'ordre de 10 conviendra pour un affichage suffisamment clair).
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 θ et de même taille, ce par simple troncature du résultat `Irot` obtenu avec `rotationdirect.m` ; lorsque l'angle θ est, comme ici, petit (de l'ordre d'un ou deux degrés), la troncature d'image n'engendre en effet quasiment pas de perte d'information.