ESIR3-IN : I.M. TP 2

Recalage iconique 2D : descente de gradient

1 Histogramme joint, critères de similarités

1. La fonction `hist2(I,J)` calcule l'histogramme conjoint de deux images `I` et `J`[1]. Vérifiez que $\sum_{i,j} H_{I,J}(i,j) = n * p$ pour des images de taille $n \times p$.
2. Écrivez deux fonctions, `ssd(I,J)` et `correlation(I,J)`, qui calculent respectivement la somme des différences au carré et le coefficient de corrélation entre les deux images. Pour `correlation(I,J)` vous utiliserez la fonction `corr2`.
3. La fonction `mutual_information(H)` calcule l'information mutuelle des deux images, à partir de l'histogramme conjoint $H_{I,J}$. Regardez son code.
4. Exécutez le script `demo_TP2` qui, pour chaque couple (I_k, J_k) fourni, affiche l'histogramme conjoint (échelle logarithmique), calcule et affiche les valeurs des trois critères de similarités précédents. Analysez les résultats.
5. Le fichier `RIRE_database.mat` contient quelques données extraites de la base de données du projet Retrospective Image Registration Evaluation (RIRE)[2]. Ces images correspondent à différentes modalités d'imagerie et concerne le même sujet (`CT`=CT-scan/scanner, `PD`=IRM-proton-density, `T1`=IRM-T1-contrast, `PET`=TEP /tomographie par émission de positons).

   Exécutez le fichier `demo_RIRE` qui simule un recalage en cours et affiche différentes mesures. Analysez les résultats.

Dans la suite, vous appliquerez vos fonctions pour recaler les images présentes dans `TP2_donnees.mat` et afficherez les histogrammes joints correspondants.

2 SSD+Translations

1. La fonction `translation(I,p,q)` produit une image `I_t` correspondant à l'image `I` translatée du vecteur $\vec{t} = (p, q)$. Testez cette fonction sur une image I en créant une version translatée J de cette image en prenant $p = 20.5, q = 10.2$. L'image résultat est-elle définie par $J(x, y) = I(x + p, y + q)$ ou $J(x, y) = I(x - p, y - q)$ ou $J(x, y) = I(x + q, y + p)$ ou autre ?
2. La fonction `grad_centre(M)` calcule le gradient d'une image `M` (2D ou 3D) en utilisant les différences centrées comme schéma de discrétisation. Implémentez le recalage 2D vu en cours en minimisant la SSD et en considérant uniquement les translations. Au cours de l'évolution sauvegardez et affichez l'énergie `SSD` correspondante à chaque état.
3. Avec un pas de temps suffisamment faible, vous devriez observer des remontées locales d'énergie. Qu'en pensez-vous ? Cela vous semble-t-il cohérent avec ce qu'on fait ? Que faudrait-il faire pour être plus correct ?

1. `I` et `J` sont de même taille, 2D ou 3D, et on suppose que les intensités $\in [0, 255]$.
2. `http://www.insight-journal.org/rire/`

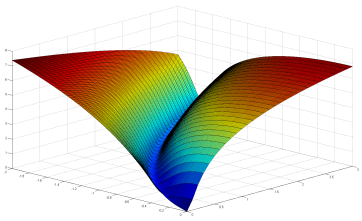
3 SSD+Rotations

1. La fonction `rotation(I,theta)` produit une image `I_theta` correspondant à l'image `I` à laquelle on a appliqué une rotation d'angle `theta` et de centre $(0, 0)$ (le coin en haut à gauche de l'image). Complétez le script `registration_rotation_2D` pour implémenter le recalage 2D vu en cours, en minimisant la SSD et en considérant uniquement les rotations de centre $(0, 0)$.

   Testez et, au cours de l'évolution, sauvegardez l'énergie `SSD` correspondante à chaque état (même remarque qu'au dessus).

4 SSD+transformations rigides 2D

1. La fonction `rigid_transformation(I,theta,p,q)` correspond à la transformation rigide 2D vue en cours. Implémentez la descente de gradient pour minimiser la SSD en considérant cette fois les transformations rigides (rotations+translations).
2. Testez avec $\theta = \frac{\pi}{10}, a = 1, b = 1$. Que se passe-t-il à votre avis ? En quoi l'image ci-dessous peut probablement "illustrer" ce qu'il se passe ?



3. Pour améliorer les performances, il faut donc utiliser une méthode d'optimisation un peu plus évolué que la descente de gradient **à pas fixe**. La toolbox "optimization" de matlab fourni la fonction `fminunc` pour minimiser (sans contraintes) une fonction à plusieurs variables. Avec les bonnes options, cette fonction implémente une variante de la descente de gradient (voir la doc pour plus d'info) plus performante que la descente à pas fixe implémentée plus haut. Regardez les pages suivantes, ainsi que l'exemple ci-dessous (tiré de la première page, et fourni dans le fichier `demo_fminunc.m`).

   `http://www.mathworks.fr/help/toolbox/optim/ug/fminunc.html`

   `http://fr.mathworks.com/help/optim/ug/passing-extra-parameters.html`

```
function [f,g] = myfun(x)
f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;
if nargout > 1
    g(1) = 6*x(1)+2*x(2);
    g(2) = 2*x(1)+2*x(2);
end
```

```
options = optimset('GradObj','on');
options.Display = 'iter';
x0 = [1,1];
[x,fval] = fminunc(@myfun,x0,options);
```

4. Servez-vous de cette fonction pour implémenter le recalage rigide 2D (un squelette du programme principal est fourni).