

TP2 : Restauration d'images

pierre.maurel@irisa.fr

https://perso.univ-rennes1.fr/pierre.maurel/Prepa_Agreg/

Pour l'ensemble du TP vous **limitez** l'usage des boucles (`for` ou `while`) uniquement aux itérations correspondant aux évolutions. Les manipulations sur les matrices seront donc faites directement grâce aux fonctions et opérateurs Matlab (donc sans boucle `for`).

1 Gradient, Divergence, Laplacien

1.1 Gradient

On note $u \in \mathbb{R}^N$ une image discrète de $N = n \times n$ pixels. On peut définir une version discrétisée du gradient de cette image par

$$\nabla u[i, j] = \left(\frac{\partial u[i, j]}{\partial x}, \frac{\partial u[i, j]}{\partial y} \right)$$

où les dérivées selon x et y sont approximées par des schémas avant :

$$\frac{\partial u[i, j]}{\partial x} = \begin{cases} u[i + 1, j] - u[i, j] & \text{si } 1 \leq i < n, \\ 0 & \text{si } i = n, \end{cases}$$
$$\frac{\partial u[i, j]}{\partial y} = \begin{cases} u[i, j + 1] - u[i, j] & \text{si } 1 \leq j < n, \\ 0 & \text{si } j = n. \end{cases}$$

Le gradient est donc un champ de vecteurs $\nabla u \in \mathbb{R}^N \times \mathbb{R}^N$.

Exercice 1 La fonction `grad`, fournie, prend en argument une image et renvoie le gradient de cette image utilisant la discrétisation décrite ci-dessus. Écrivez une fonction qui prend une image u en argument et renvoie une image N , telle que $N(i, j) = \|\nabla u[i, j]\|$. Chargez une image et testez cette fonction en affichant la norme du gradient de cette image.

1.2 Divergence

La divergence d'un champ de vecteurs $p = (p_1, p_2) \in \mathbb{R}^N \times \mathbb{R}^N$ peut être calculée ainsi :

$$\text{div}(p) = \partial_x^* p_1 + \partial_y^* p_2$$

avec

$$\partial_x^* f[i, j] = \begin{cases} f[i, j] & \text{si } i = 1, \\ f[i, j] - f[i - 1, j] & \text{si } 1 < i < n, \\ -f[i - 1, j] & \text{si } i = n, \end{cases}$$
$$\partial_y^* f[i, j] = \begin{cases} f[i, j] & \text{si } j = 1, \\ f[i, j] - f[i, j - 1] & \text{si } 1 < j < n, \\ -f[i, j - 1] & \text{si } j = n. \end{cases}$$

Exercice 2 La fonction `div`, fournie, prend en argument un champ de vecteurs et renvoie sa divergence utilisant la discrétisation décrite ci-dessus. Chargez une image et testez ces 2 fonctions (`∇` et `div`) en affichant le laplacien de cette image, vous pourrez vérifier (visuellement) que le laplacien d'une image passe par zéro au niveau des contours de cette image. On rappelle que $\Delta u = \text{div}(\nabla u)$.

2 Restauration de Tikhonov

On dispose d'une image bruitée `gatlin_noisy.png`. Pour débruiter cette image, on décide de minimiser l'énergie de Tikhonov vue en cours (f est la donnée, une image bruitée) :

$$\inf_u J(u) = \inf_u \underbrace{\int_{\Omega} (u(x) - f(x))^2 d\Omega}_{\text{terme de fidélité, attache aux données}} + \lambda \underbrace{\int_{\Omega} \|\nabla u(x)\|^2 d\Omega}_{\text{régularisation}}.$$

L'équation d'Euler-Lagrange associée à ce problème de minimisation est la suivante (cf cours) : $u(x) - f(x) - \lambda \Delta u = 0$. Elle n'admet pas de solution explicite simple et on décide donc de définir une séquence minimisante en introduisant une variabilité temporelle ("descente de gradient") :

$$\frac{\partial u}{\partial t}(x, t) = -(u(x, t) - f(x) - \lambda \Delta u(x, t)),$$

qu'on peut discrétiser ainsi :

$$\begin{aligned} \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} &= -(u_{i,j}^n - f_{i,j} - \lambda \Delta u_{i,j}^n) \\ u_{i,j}^{n+1} &= u_{i,j}^n - \Delta t (u_{i,j}^n - f_{i,j} - \lambda \Delta u_{i,j}^n) \end{aligned}$$

Exercice 3

Implémentez cette évolution, qui minimise l'énergie $J(u)$ pour un λ donné.

- le laplacien sera calculé en utilisant les fonctions `div` et `grad` fournies.
- Au cours de l'évolution, calculez et sauvegardez les valeurs successives de l'énergie $J(u)$ et affichez à la fin de l'évolution les valeurs de cette énergie. Vérifiez qu'elle décroît bien à chaque instant et qu'elle converge.
- Le critère d'arrêt pourra être donné par le fait que l'image `u` ne bouge plus suffisamment.

Exercice 4 (a) Vous disposez également de la version non bruitée de l'image (`gatlin.png`) et d'une fonction `snr` qui calcule le rapport signal à bruit (Signal to Noise Ratio). Cherchez (approximativement) le λ permettant d'obtenir une image au SNR maximal.

(b) Testez cet algorithme de débruitage sur l'image d'un carré blanc sur fond noir :

```
X = zeros(100,100);
X(30:70,30:70)=1;
% on ajoute du bruit
sigma = .1;
X_bruit = X + sigma*randn(size(X));
```

Choisissez un λ qui permet de faire disparaître la quasi-totalité du bruit. Que remarque-t-on ?

3 Restauration Ψ (Rudin-Osher-Fatemi Model)

On considère maintenant le problème :

$$\inf_u J(u) = \inf_u \int_{\Omega} (u(x) - f(x))^2 d\Omega + \lambda \int_{\Omega} \Psi(\|\nabla u(x)\|) d\Omega$$

où Ψ est une fonction moins pénalisante pour les gradients élevés que la fonction quadratique. Nous allons ici utiliser la fonction $\Psi(x) = \sqrt{\varepsilon + x^2}$ (avec ε petit, 10^{-3} par exemple).

L'équation d'Euler-Lagrange associée à ce problème de minimisation est la suivante $2(u - f) - \lambda \operatorname{div}\left(\frac{\Psi'(\|\nabla u\|)}{\|\nabla u\|} \nabla u\right) = 0$. Comme précédemment elle n'admet pas de solution explicite et on décide de définir une séquence minimisante ainsi :

$$\frac{\partial u}{\partial t} = 2(f - u) + \lambda \operatorname{div}\left(\frac{\Psi'(\|\nabla u\|)}{\|\nabla u\|} \nabla u\right), \quad \text{avec ici : } \frac{\Psi'(\|\nabla u\|)}{\|\nabla u\|} \nabla u = \frac{\nabla u}{\sqrt{\varepsilon + \|\nabla u\|^2}}$$

Exercice 5

Implémentez l'évolution correspondante. Comme précédemment, vous calculerez et afficherez les valeurs successives de $J(u)$ pour s'assurer de la décroissance et de la convergence.

Exercice 6

- Testez ce nouvel algorithme sur le carré de l'exercice 4.b et comparez les résultats.
- Pour `gatlinoisy.png`, cherchez (approximativement) le λ permettant d'obtenir une image au SNR maximal. Comparez au résultat de l'exercice 4.
- Vous pourrez également comparer au débruitage obtenu grâce à BM3D (<https://webpages.tuni.fi/foi/GCF-BM3D/>).

4 Inpainting

Lorsque qu'on a perdu la valeur de certains pixels d'une image (ou qu'on veut "effacer" un objet de l'image), l'inpainting (ou désocclusion) consiste à essayer de donner une valeur à ces pixels qui soit cohérente avec le reste de l'image.

On peut modéliser ce problème de la manière suivante :

$$\inf_u J(u) = \inf_u \int_{\Omega} (\mathcal{R}u(x) - f(x))^2 d\Omega + \lambda \int_{\Omega} \Psi(\|\nabla u(x)\|) d\Omega$$

où \mathcal{R} est un opérateur de masquage qui supprime les pixels du domaine $D \subset \Omega$:

$$(\mathcal{R}u)(x) = \begin{cases} 0 & \text{si } x \in D, \\ u(x) & \text{si } x \notin D \end{cases}$$

On pourrait implémenter l'équation d'Euler-Lagrange vu en cours (de la même manière que dans les parties précédentes) :

$$\frac{\partial u}{\partial t} = 2\mathcal{R}^*f - 2\mathcal{R}^*\mathcal{R}u + \lambda \operatorname{div}\left(\frac{\Psi'(\|\nabla u\|)}{\|\nabla u\|} \nabla u\right)$$

Cependant, si l'on suppose qu'il n'y a pas de bruit dans f , on n'a pas envie de modifier la valeur des pixels connus. On aimerait donc avoir

$$\int_{\Omega} (\mathcal{R}u(x) - f(x))^2 d\Omega = 0.$$

C'est à dire $\mathcal{R}u(x) = f(x)$ (en dehors du masque on ne veut pas modifier l'image).

Ceci revient à faire tendre λ vers zéro, mais dans ce cas la convergence est très lente. Pour accélérer l'évolution, on modifie un peu l'algorithme : à chaque itération, on effectue

1. un pas dans la direction opposée au gradient du critère de régularité **seul**
2. un reprojection sur les données en fixant les valeurs connues de f :
`u(mask==0) = f(mask==0);`

Exercice 7

- Implémentez cette évolution (un squelette `inpainting.m` est fourni).
- Pour la tester chargez une image (exemple `cameraman.png`), créez un masque et appliquez le à l'image :

```
I = double(imread('cameraman.png'));
% pourcentage de pixels perdus
rho = .5;
%masque aléatoire , mask==1 pour les pixels supprimés
mask = zeros(size(I));
sel = randperm(numel(I)); sel = sel(1:round(rho*numel(I)));
mask(sel) = 1;
I_masked = I.*(1-mask);
```

- Testez également sur l'image `parrot.png` en lui appliquant `parrot-mask.png`.
- Vous pourrez aussi tester sur une image en couleur en appliquant la même méthode sur chaque composante.

5 Déconvolution

On considère le problème :

$$\inf_u J(u) = \inf_u \int_{\Omega} (G_{\sigma} * u(x) - f(x))^2 d\Omega + \lambda \int_{\Omega} \Psi(\|\nabla u(x)\|) d\Omega$$

où G_{σ} est un noyau gaussien d'écart-type σ .

Exercice 8 Implémentez la "descente de gradient" vue en cours, en prenant $\lambda = 0$ (absence de bruit). Testez sur différentes images et pour divers niveaux de flou (différent σ). Pour définir un noyau gaussien : `h = fspecial('gaussian', hsize, sigma);`. Pour effectuer la convolution par ce noyau : `Ru = conv2(u,h,'same');`