TP5 – Restauration d'images

Débruitage

L'objectif du *débruitage* consiste, à partir d'une image bruitée $u_0 : \Omega \to \mathbb{R}$ (image en niveaux de gris), à trouver une image u telle que les deux images u et u_0 soient « proches », et que la nouvelle image u soit « lisse ». Le modèle variationnel de débruitage de type « Tikhonov » revient à minimiser l'énergie suivante :

$$E_{\text{Tikhonov}}(u) = \frac{1}{2} \iint_\Omega \left\{ [u(x,y) - u_0(x,y)]^2 + \lambda \, |\nabla u(x,y)|^2 \right\} dx\, dy \tag{1}$$

où $\lambda > 0$ est fixé par l'utilisateur. Le minimiseur de (1) est solution de l'équation d'Euler-Lagrange :

$$(I - \lambda \, \Delta)\, u(x,y) = u_0(x,y), \qquad \forall\, (x,y) \in \Omega \tag{2}$$

où I désigne l'opérateur identité et Δ le laplacien. Après discrétisation par différences finies, (2) se réécrit :

$$\underbrace{\left[\mathbf{I}_N - \lambda \, \left(-\mathbf{D}_x^\top \, \mathbf{D}_x - \mathbf{D}_y^\top \, \mathbf{D}_y\right)\right]}_{\mathbf{A}} \mathbf{u} = \underbrace{\mathbf{u}_0}_{\mathbf{b}} \tag{3}$$

où $\mathbf{u}$ et $\mathbf{u}_0$ sont des vecteurs de taille $N \times 1$ (N désigne le nombre de pixels) contenant les valeurs des niveaux de gris u et u_0, et $\mathbf{I}_N$ est la matrice identité (de taille $N \times N$). Enfin, $\mathbf{D}_x$ et $\mathbf{D}_y$ sont deux matrices bi-diagonales permettant d'approcher les composantes de ∇u par différences finies, dont la définition a été vue en cours.

Lancez le script `exercice_0`, qui implémente le modèle de débruitage (1) par résolution de l'équation discrète (3). Ce script utilise les fonctions `speye` pour construire la matrice identité, `spdiags` pour construire les matrices $\mathbf{D}_x$ et $\mathbf{D}_y$ (`sp` caractérise les fonctions pour matrices « creuses », *sparse* en anglais), et l'opérateur \ (*backslash*) pour la résolution du système. Même en jouant sur le paramètre λ, vous constatez que ce modèle ne préserve pas convenablement les contours.

Exercice 1 : modèle de débruitage par variation totale

On préfère généralement à la régularisation quadratique $\frac{1}{2} \iint |\nabla u(x,y)|^2 \, dx\, dy$ du modèle (1), la « variation totale » $\iint |\nabla u(x,y)| \, dx\, dy$. L'énergie devient alors non différentiable, mais on peut utiliser l'approximation :

$$E_{\text{TV}}(u) = \iint_\Omega \left\{ \frac{1}{2} \, [u(x,y) - u_0(x,y)]^2 + \lambda \, \sqrt{|\nabla u(x,y)|^2 + \epsilon} \right\} dx\, dy \tag{4}$$

où le paramètre ϵ permet de garantir la différentiabilité en 0. L'équation d'Euler-Lagrange qui en résulte est non linéaire. Elle peut néanmoins être résolue par un schéma itératif de type « point fixe » :

$$\left[I - \lambda \, \nabla \cdot \left(\frac{1}{\sqrt{|\nabla u^{(k)}(x,y)|^2 + \epsilon}} \, \nabla \right) \right] u^{(k+1)}(x,y) = u_0(x,y), \qquad \forall\, (x,y) \in \Omega \tag{5}$$

où l'opérateur $\nabla \cdot$ désigne la *divergence*. Après discrétisation, l'itération (5) s'écrit :

$$\underbrace{\left[\mathbf{I}_N - \lambda \, \left(-\mathbf{D}_x^\top \, \mathbf{W}^{(k)} \, \mathbf{D}_x - \mathbf{D}_y^\top \, \mathbf{W}^{(k)} \, \mathbf{D}_y\right)\right]}_{\mathbf{A}^{(k)}} \mathbf{u}^{(k+1)} = \underbrace{\mathbf{u}_0}_{\mathbf{b}} \tag{6}$$

où $\mathbf{W}^{(k)}$ est la matrice diagonale des coefficients $\frac{1}{\sqrt{|\nabla u^{(k)}(x,y)|^2 + \epsilon}}$ (`spdiags` avec 0 en deuxième argument).

Écrivez la fonction `debruitage`, appelée par le script `exercice_1`, permettant d'appliquer le modèle (4) à l'image `cameraman_avec_bruit.tif`, en effectuant les itérations de point fixe (6) avec $\epsilon = 0,01$, tant que $\|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\|_{L_2(\Omega)} / \|\mathbf{u}^{(k)}\|_{L_2(\Omega)} > 10^{-3}$, en ajustant au mieux la valeur de λ dans l'intervalle $[10, 15]$. La commande `drawnow nocallbacks` permet de forcer l'affichage de la solution à chaque itération. La préservation des contours devrait être nettement meilleure. Testez ensuite ce script sur l'image `lena_avec_bruit.bmp`.

Inpainting

L'*inpainting* (que l'on peut traduire en français par « retouche d'images ») est une technique qui permet de restaurer une image sur un certain domaine. Il existe deux cas de figure assez différents, selon que le domaine à restaurer est déjà identifié ou non. Deux types d'applications sont possibles : la restauration de zones endommagées, par exemple les rayures sur des photographies anciennes ; la « réalité diminuée » (par opposition à la « réalité augmentée »), qui consiste à retirer certains objets d'une image. Nous nous intéressons dans l'exercice 2 à la restauration par *inpainting* (cf. figure 1).

FIGURE 1 – Restauration par *inpainting* : à gauche, image originale ; au centre, image dégradée u_0 (le domaine D à restaurer est indiqué en jaune) ; à droite, image u restaurée par *inpainting*.

Exercice 2 : modèle d'*inpainting* par variation totale

Le modèle d'*inpainting* par variation totale consiste en une modification très simple du modèle de débruitage (4). Supposons que la donnée u_0 ne soit pas fiable sur un domaine $D \subset \Omega$. Le terme d'attache aux données ne doit alors être calculé que sur $\Omega \backslash D$. Par conséquent, le modèle (4) devient :

$$E_{Inpainting}(u) = \frac{1}{2} \iint_{\Omega \backslash D} [u(x,y) - u_0(x,y)]^2 \, dx \, dy + \lambda \iint_{\Omega} \sqrt{|\nabla u(x,y)|^2 + \epsilon} \, dx \, dy \tag{7}$$

ce qui conduit aux itérations de point fixe suivantes :

$$\underbrace{\left[\mathbf{W}_{\Omega \backslash D} - \lambda \left(-\mathbf{D}_x^\top \mathbf{W}^{(k)} \mathbf{D}_x - \mathbf{D}_y^\top \mathbf{W}^{(k)} \mathbf{D}_y\right)\right]}_{\mathbf{A}^{(k)}} \mathbf{u}^{(k+1)} = \underbrace{\mathbf{W}_{\Omega \backslash D} \, \mathbf{u}_0}_{\mathbf{b}} \tag{8}$$

Dans cette expression, $\mathbf{W}_{\Omega \backslash D}$ est la matrice diagonale ayant pour éléments diagonaux les valeurs $1 - \chi_D(x,y)$, où χ_D désigne la fonction caractéristique de D : $\chi_D(x,y) = 1$ si $(x,y) \in D$, $\chi_D(x,y) = 0$ sinon.

Faites une copie du script `exercice_1`, de nom `exercice_2`, et une copie de la fonction `debruitage`, de nom `inpainting`, que vous modifierez de manière à restaurer l'image `fleur_avec_defaut.png` (cf. figure 1), pour laquelle D est fourni dans l'image binaire `defaut_fleur.png`. Observez l'influence de λ, en testant différentes valeurs de ce paramètre, en particulier vis-à-vis du lissage en dehors du domaine D.

Une autre solution consiste à déterminer D par segmentation. Faites une copie du script `exercice_2`, de nom `exercice_2_bis`, que vous modifierez de manière à déterminer le domaine D en utilisant la couleur comme critère. Il est rappelé que le jaune n'est pas une couleur primaire, mais un mélange de rouge et de vert ne comportant pas de bleu. Testez ensuite ce script sur l'image `grenouille_avec_texte.png`, afin de « gommer » le texte affiché en surimpression.

Faites enfin une nouvelle copie du script `exercice_2`, de nom `exercice_2_ter`, à laquelle vous apporterez les modifications suivantes : fixez le paramètre `lambda` à 1 ; remplacez les fichiers `fleur_avec_defaut.png` et `defaut_fleur.png` par `randonneur.jpg` et `masque_randonneur.png`, respectivement. Vous observez que le résultat n'est pas satisfaisant. Pour l'améliorer, il faut recourir à une autre méthode d'*inpainting*, qui constitue la partie facultative de ce TP.

Partie facultative : réalité diminuée

La méthode d'*inpainting* de l'exercice 2 s'appelle l'*inpainting par diffusion*. Elle permet de restaurer des zones de faible épaisseur, comme les rayures sur une photographie ancienne, mais ne permet pas de répondre aux besoins de la « réalité diminuée », dont le but est de supprimer un ou plusieurs objets d'une scène. La réalité diminuée nécessite d'utiliser des méthodes d'*inpainting* qui, contrairement à l'*inpainting* par diffusion, ne découlent pas d'une approche variationnelle.

Si Ω désigne l'ensemble des pixels de l'image, et D le domaine à compléter, l'*inpainting* « par rapiéçage » (*patch-based inpainting*) consiste à rechercher, en tout pixel $\mathbf{p}$ de la frontière ∂D de D, le pixel $\widehat{\mathbf{q}} \in \overline{D} = \Omega \backslash D$ (complémentaire de D à Ω) dont le voisinage $V(\widehat{\mathbf{q}})$ « ressemble le plus » au voisinage $V(\mathbf{p})$ de $\mathbf{p}$. En pratique, le voisinage est une fenêtre centrée de taille $(2\,t+1) \times (2\,t+1)$, $t > 0$. Si $\mathbf{p} = (i_\mathbf{p}, j_\mathbf{p})$ est un pixel de ∂D, notons $R(\mathbf{p})$ l'ensemble des *indices relatifs* (i, j) des pixels voisins $\mathbf{p}' = (i_\mathbf{p} + i, j_\mathbf{p} + j)$ se trouvant hors de D :

$$R(\mathbf{p}) = \left\{(i,j) \in \{-t, \ldots, t\}^2 \,/\, (i_\mathbf{p} + i, j_\mathbf{p} + j) \in \overline{D}\right\} \tag{9}$$

Comme au moins un voisin de $\mathbf{p} \in \partial D$ appartient à $\overline{D}$, il s'ensuit que $\mathrm{Card}(R(\mathbf{p})) > 0$. Pour une image en couleur u, la *dissemblance* $d(\mathbf{p}, \mathbf{q})$ entre le voisinage d'un pixel $\mathbf{p} \in \partial D$ et le voisinage d'un pixel $\mathbf{q} \in \overline{D}$ est définie comme suit :

$$d(\mathbf{p}, \mathbf{q}) = \frac{1}{\mathrm{Card}(R(\mathbf{p}))} \sum_{(i,j) \in R(\mathbf{p})} \|u(i_\mathbf{p} + i, j_\mathbf{p} + j) - u(i_\mathbf{q} + i, j_\mathbf{q} + j)\|^2 \tag{10}$$

Pour trouver $\widehat{\mathbf{q}}$, il est inutile de calculer $d(\mathbf{p}, \mathbf{q})$ en chaque pixel $\mathbf{q} \in \overline{D}$, car une image est généralement constituée de régions de texture homogène. Nous nous contentons de rechercher $\widehat{\mathbf{q}}$ dans une fenêtre $F(\mathbf{p})$ centrée en $\mathbf{p}$, de taille $(2\,T+1) \times (2\,T+1)$, avec $T > t$. Il suffit même de mener cette recherche pour les pixels $\mathbf{q} \in F'(\mathbf{p})$ tels que $V(\mathbf{q}) \subset \big(F(\mathbf{p}) \cap \overline{D}\big)$, comme le montre la figure 2. Le pixel $\widehat{\mathbf{q}}$ associé à un pixel $\mathbf{p} \in \partial D$ est donc tel que :

$$\widehat{\mathbf{q}} = \underset{\mathbf{q} \in F'(\mathbf{p})}{\mathrm{argmin}} \{d(\mathbf{p}, \mathbf{q})\} = \underset{\mathbf{q} \in F'(\mathbf{p})}{\mathrm{argmin}} \sum_{(i,j) \in R(\mathbf{p})} \|u(i_\mathbf{p} + i, j_\mathbf{p} + j) - u(i_\mathbf{q} + i, j_\mathbf{q} + j)\|^2 \tag{11}$$

Le rapiéçage consiste à remplacer les pixels manquants de $V(\mathbf{p})$ par les pixels de mêmes positions dans $V(\widehat{\mathbf{q}})$.

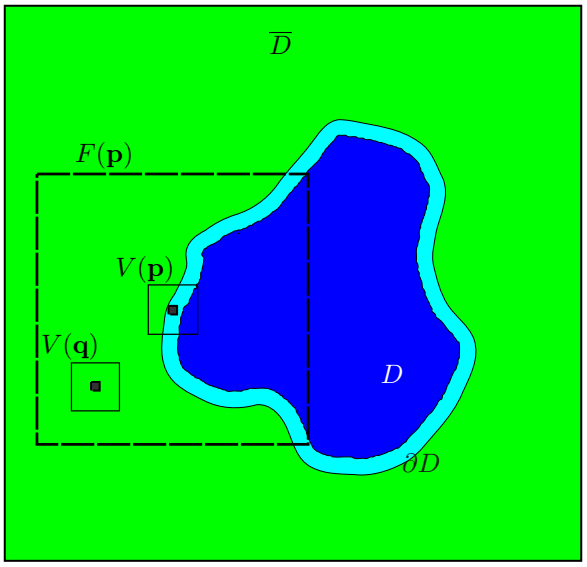

FIGURE 2 – Principe de l'*inpainting* par rapiéçage.

Exercice 3 : *inpainting* par rapiéçage (facultatif)

Une manière de coder l'*inpainting* par rapiéçage consiste à répéter la boucle suivante, tant que le domaine D n'est pas vide :

1. Choisir un pixel $\mathbf{p}$ de la frontière ∂D par tirage aléatoire.
2. Rechercher le pixel $\widehat{\mathbf{q}}$ défini en (11) en testant tous les pixels $\mathbf{q} \in F'(\mathbf{p})$. Pour chaque $\mathbf{q}$:
   - Calculer la dissemblance $d(\mathbf{p}, \mathbf{q})$ définie par (10).
   - Si $d(\mathbf{p}, \mathbf{q}) < \widehat{d}$, alors $\widehat{\mathbf{q}} \leftarrow \mathbf{q}$ et $\widehat{d} \leftarrow d(\mathbf{p}, \mathbf{q})$.
3. Utiliser $V(\widehat{\mathbf{q}})$ pour compléter les pixels manquants de $V(\mathbf{p})$ par rapiéçage.
4. Mettre à jour D et ∂D.

Écrivez les fonctions `d_min` et `rapiecage`, appelées par le script `exercice_3`, permettant de reproduire cet algorithme. En le testant sur l'image `randonneur.jpg`, vous constatez que `exercice_3` fournit un résultat beaucoup plus réaliste que `exercice_2_ter`. Le script `exercice_3_bis` est identique à `exercice_3`, à ceci près que le domaine D à compléter doit être sélectionné par l'utilisateur sous la forme d'un polygone déterminé par une série de clics (double-clic pour terminer).

Conclusion

Dans le cas où la zone à compléter coupe les arbres de l'arrière-plan, le résultat du script `exercice_3_bis` perd de son réalisme. Cela vient de ce que le prochain pixel $\mathbf{p} \in \partial D$ à traiter est tiré aléatoirement. Pour améliorer les résultats, il semble inévitable de mettre en place un ordre de priorité. Cette idée a été proposée par Criminisi, Pérez et Toyama, qui ont publié en 2003 la première méthode d'inpainting par rapiéçage, dans un article intitulé *Region filling and object removal by exemplar-based image inpainting*. Bien sûr, de nombreuses améliorations ont été proposées depuis. Une version commerciale de l'inpainting par rapiéçage est proposée dans le logiciel *PaintShop Pro*. Le *plugin* de GIMP de nom *PatchMatch*, disponible dans la boîte à outils G'MIC développée à l'Université de Caen, constitue une version libre de cette méthode d'*inpainting*.