Filtrage d'images

Matériel Récupérez l'archive `td-filter.tgz`

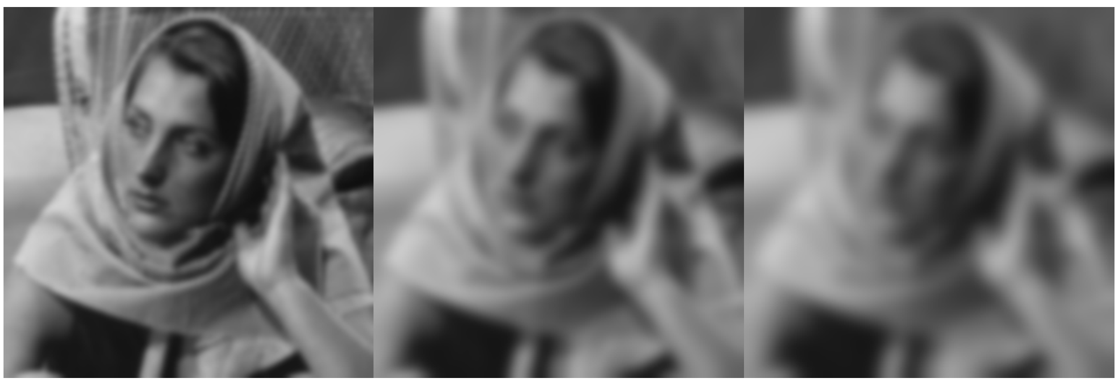
1 Heat Equation

La première équation aux dérivées partielles (EDP) a avoir été étudiée dans le domaine du traitement des images est certainement l'équation de diffusion de la chaleur : $\partial_t u(x,y) = \Delta u((x,y),t)$ avec $t \geq 0$, $u(.,0) = u_0(.)$ et $u_0 : \Omega \subset R^2 \rightarrow R$ l'image initiale vue comme une fonction continue et Δu le Laplacien de u.

Écrire le programme `heat_equation <n> <ims> <imd>` qui réalise l'équation de la chaleur. Utiliser les schémas numériques d'approximation suivant. On note τ la discrétisation temporelle ($\tau = 0.25$)

- $u^n_{i,j}$ désigne la valeur $u((i,j), n\tau)$.
- Le terme $\partial_t u((i,j), n\tau)$ est approximé par $\tau^{-1}(u^{n+1}_{i,j} - u^n_{i,j})$.
- Le Laplacien est discrétisé par différences finies. $\Delta u^n_{i,j} = u^n_{i+1,j} + u^n_{i-1,j} + u^n_{i,j+1} + u^n_{i,j-1} - 4u^n_{i,j}$

La figure 1 montre des exemples de filtrage avec l'équation de la chaleur. Nous pouvons remarquer que l'équation de la chaleur débruite l'image mais a tendance à la rendre floue. Les zones homogènes sont restaurées mais les contours sont érodés. Il s'agit là d'un filtre par diffusion isotrope (diffusion identique dans la même direction).



(a) `n`=10 (b) `n`=50 (c) `n`=100

FIGURE 1 – Exemples d'exécution du programme `heat_equation.c`.

2 Anisotropic Diffusion

Pour améliorer l'équation de la chaleur, [1] ont proposé d'intégrant une fonction de détection de contours basée sur la norme du gradient.

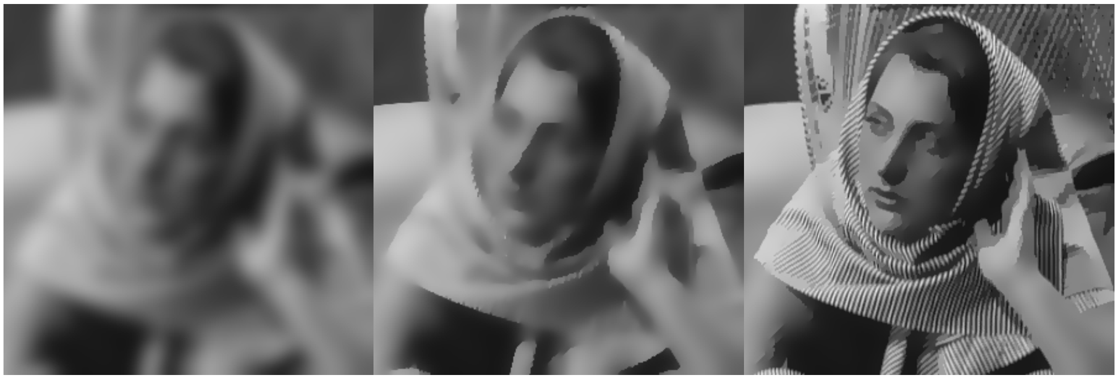
L'équation de la chaleur se réécrit donc de la manière suivante : $\partial_t u = div(c(|\nabla u|)\nabla u)$ avec $u(.,0) = u_0(.)$ où c est une fonction décroissante[1]. La fonction c est une fonction valant 1 en 0 et tend vers 0 en l'infini. Cela a pour effet de ralentir (ou stopper) la diffusion au point où la norme du gradient est élevée, i.e., les points de contours. [1] proposent deux fonctions de détection de contours : $c_1(s) = (1 + (s/\lambda)^2)^{-1}$ et $c_2(s) = \exp(-(s/\lambda)^2)$ où le paramètre λ permet de contrôler la diffusion.

Écrire le programme `anisotropic_diffusion <n> <lambda> <function> <ims> <imd>` qui réalise la diffusion anisotrope de Perona-Malik. Le paramètre `function` $\in \{c_0, c_1, c_2\}$ avec les discrétisations suivantes :

- le gradient $\nabla u = (\delta^1 u, \delta^2 u)$, par différences finies à droite, $\delta^1 u_{i,j} = u_{i+1,j} - u_{i,j}$ et $\delta^2 u_{i,j} = u_{i,j+1} - u_{i,j}$
- sa norme : $|\nabla u_{i,j}| = \sqrt{(\delta^1 u_{i,j})^2 + (\delta^2 u_{i,j})^2}$.
- par les différences finies à droite, l'opérateur de divergence est donnée, pour un vecteur $p = (p^1, p^2)$, par $div(p)_{i,j} = p^1_{i,j} - p^1_{i-1,j} + p^2_{i,j} - p^2_{i,j-1}$

La figure 2 montre des exemples de filtrage pour les fonctions c_0, c_1 et c_2. Nous pouvons remarquer que nous retrouvons bien le même filtrage qu'avec l'équation de la chaleur dans le cas de la fonction c_0 (fig. 1(d) et 2(b)).

1. Si $c(.) = c_0(.) = 1$ alors nous retrouvons bien l'équation de la chaleur car $div(\nabla u) = \Delta u$



(a) n=100, c_0 (b) n=100, c_1, $\lambda = 10$ (c) n=100, c_2, $\lambda = 10$

FIGURE 2 – Exemples d'exécution du programme `anistropinc_diffusion.c`

3 Bilateral Filter

Le filtre bilatéral [2] est un filtre non itérative permettant un filtrage préservant les contours. L'idée de combiner informations spatiales et colorimétriques dans un processus de filtrage par moyennes pondérées. Ainsi, si p est un pixel de l'image I et que I' est l'image restaurée peut s'écrire

$$I'(p) = \frac{\sum_{x \in V(p)} G_{\sigma_s}(|x-p|).G_{\sigma_g}(|I(x)-I(p)|).I(x)}{\sum_{x \in V(p)} G_{\sigma_s}(|x-p|).G_{\sigma_g}(|I(x)-I(p)|)} \quad \text{où } G_\sigma(k) = \exp(-k^2/2\sigma^2). \tag{1}$$

Le voisinage V est déterminé grâce à l'écart type σ_s.

Écrire le programme `bilateral <sigma_s> <sigma_g> <ims> <imd>` réalisant le filtre bilatéral. La figure 2 montre des exemples de filtrage avec le filtre bilatéral pour différentes valeurs de σ_g et σ_s.

(a) $\sigma_s = 3, \sigma_g = 30$ (b) $\sigma_s = 3, \sigma_g = 100$ (c) $\sigma_s = 10, \sigma_g = 30$ (d) $\sigma_s = 10, \sigma_g = 100$

FIGURE 3 – Exemples d'exécution du programme `bilateral`

4 NL-Means

Le filtre à moyennes non locales [3] peut être vu comme une extension du filtre bilatéral en intégrant la notion de similarité de patchs dans le processus de filtrage.

Le NLmeans peut s'écrire de la manière suivante :

$$I'(p) = \frac{1}{C(p)} \sum_{q \in V(p,r)} w(p,q) I(q) \quad \text{avec} \quad C(p) = \sum_{q \in V(p,r)} w(p,q) \tag{2}$$

où $V(p,r)$ est le voisinage centré sur p et de taille $(2r+1)^2$. $C(p)$ est un facteur de normalisation et $w(p,q)$ est un poids qui dépend de la distance euclidienne $d = d(P(p), P(q))$ entre des patchs P de taille $n \times n$ centrés sur les pixels p et q :

$$d(P(p), P(q)) = \frac{1}{n^2} \sum_{j}^{n^2} (I(p+j) - I(q+j))^2 \tag{3}$$

Comme fonction de poids, vous pouvez prendre une Gaussienne de type $w(p,q) = \exp(-d(P(p), P(q))/2\sigma^2)$ avec σ la variance estimée du bruit.

Écrire le programme `nlmeans <sigma> <ims> <imd>` réalisant le filtre à moyenne non locales. La figure 2 montre des exemples de filtrage pour différentes valeurs de σ.

Les résultats obtenus dans la figure 4 utilisent un voisinage $V(p,r) = 11 \times 11$ et des patchs de tailles 5×5.

(a) `barbara.ppm` (b) résultat sur (a), $\sigma = 5$ (c) `barbara_g.ppm` (d) résultat sur (c), $\sigma = 15$

FIGURE 4 – Exemples d'exécution du programme `nlmeans`

5 Filtres dans le domaine fréquentiel

Écrire le programme `butterworth.c` qui réalise le filtrage dans le domaine des fréquences. Pour rappel, les filtres usuels de Butterworth sont les suivants. Dans tous les cas, n correspond à l'ordre du filtre et $d(u,v) = \sqrt{u^2 + v^2}$.

passe bas et passe haut :

$$lp(u,v) = \frac{1}{1 + \left(d(u,v)/d_0\right)^{2n}} \quad \text{et} \quad hp(u,v) = \frac{1}{1 + \left(d_0/d(u,v)\right)^{2n}} \tag{4}$$

où d_0 correspond à la fréquence de coupure.

rejet de bande et passe bande :

$$br(u,v) = \frac{1}{1 + \left(d(u,v).w/\left(d^2(u,v) - d_0^2\right)\right)^{2n}} \quad \text{et} \quad bp(u,v) = 1 - br(u,v) \tag{5}$$

où d_0 et w correspondent au rayon et à la largeur de la bande.

rejet d'encoche :

$$no(u,v) = \frac{1}{1 + \left(d_0^2/\left(d_1(u,v)d_2(u,v)\right)\right)^{2n}} \tag{6}$$

avec $d_1(u,v) = \sqrt{(u-u_0)^2 + (v-v_0)^2}$ et $d_2(u,v) = \sqrt{(u+u_0)^2 + (v+v_0)^2}$, u_0 et v_0 sont les coordonnées du centre et d_0 le rayon.

La figure 5 montre les spectres d'amplitudes pour les images de test `lena.ppm` et `lena_sin.ppm`. Avec un zoom, sur l'image (d) nous pouvons voir deux pics (sur l'axe de fréquence v) qui correspondent à la trame ajoutée à l'image `lena.ppm` pour donner l'image `lena_sin.ppm`.

La figure 6 montre des résultats d'exécution du programme `butterworth.c` réalisant des filtrages fréquentiels avec les images `lena.ppm` et `lena_sin.ppm` avec `n`=2.

6 Method Noise

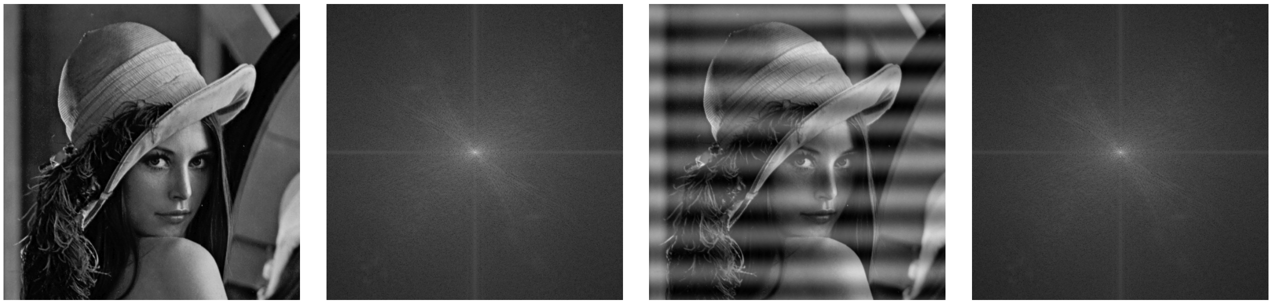
Afin d'évaluer visuellement la qualité des filtres, une des méthodes classique est la méthode « Method Noise », qui consiste à soustraire l'image initiale à la solution du filtre. La figure 7 montre une comparaison des différents filtres.

Références

[1] P. Perona J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, 1990.

[2] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images", in *Proc. IEEE ICCV*, 1998, pp. 839-846.

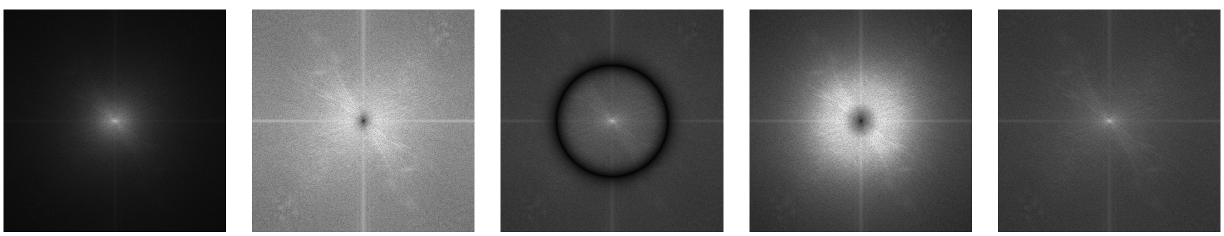
[3] A. Buades, B. Coll and J.-M. Morel, "A review of image denoising algorithms, with a new one" *SIAM Multiscale Model. Simul.*, vol. 4, no. 2, pp. 490–530, 2005.



(a) `lena.ppm` (b) spectre de `lena.ppm` (c) `lena_sin.ppm` (d) spectre de `lena_sin.ppm`

FIGURE 5 – Images de tests « lena » et leur spectre d'amplitude dans le domaine des fréquences.

(a) *lp*, d_0=32 (b) *hp*, d_0=32 (c) *br*, d_0=128, w=64 (d) *bp*, d_0=64, w=64 (e) *no*, d_0=1, u_0=0, v_0=8



(f) (g) (h) (i) (j)

FIGURE 6 – Exemples d'exécution du programme `butterworth.c` sur `lena.ppm` et `lena_sin.ppm`

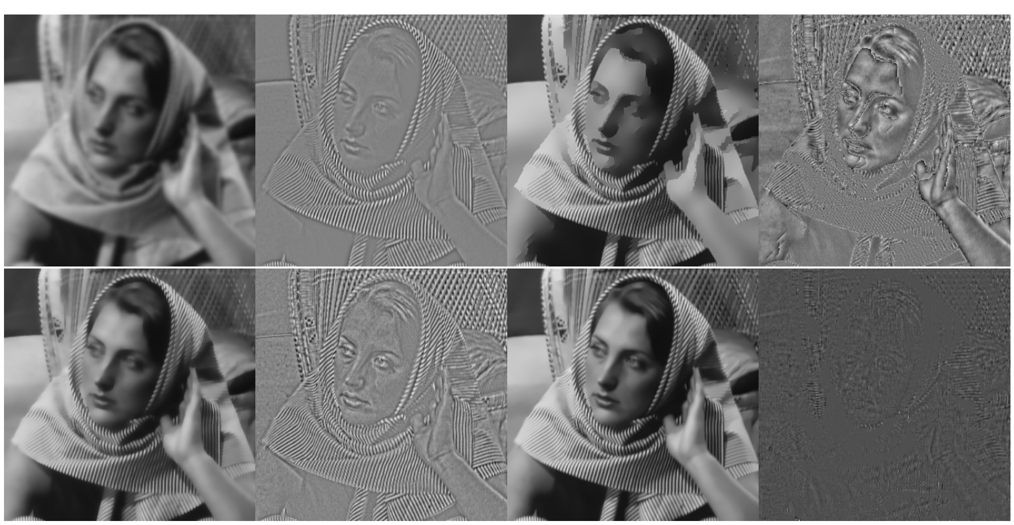


FIGURE 7 – Comparaison par Method Noise. Colonne de gauche, de haut en bas : Figs. 1(a), 2(c), 3(a), 4(a).