Traitement d'images
TP5 : modélisation du bruit - filtre des moyennes non locales

Benoît Naegel, Julien Haristoy

1 Bruit synthétique

1.1 Bruit impulsionnel

Écrire une fonction permettant d'ajouter du bruit impulsionnel (ou *poivre et sel*) à une image. La probabilité p qu'un pixel de l'image soit corrompu, avec $0 \leq p \leq 1$, sera passée en paramètre. Chaque pixel corrompu peut être blanc (valeur 255) ou noir (valeur 0) avec une probabilité 0.5 (une chance sur deux).

1.2 Bruit gaussien

Écrire une fonction permettant d'ajouter un bruit gaussien de moyenne μ et d'écart-type σ à une image.
Dans ce modèle de bruit, tous les pixels de l'image sont corrompus : la nouvelle valeur d'un pixel est calculée en ajoutant à la valeur d'origine une valeur aléatoire suivant une loi de probabilité gaussienne $\mathcal{N}(\mu, \sigma^2)$. On effectue une troncature sur l'image bruitée : si un pixel a une valeur négative, on la ramène à 0 ; si un pixel a une valeur qui dépasse 255, on la ramène à 255.
Vous pourrez utiliser la classe `std::normal_distribution` du C++ 11.

1.3 Mean Square Error

On introduit une mesure, notée MSE, de l'écart entre deux images I et I' de mêmes dimensions $N \times M$, définie par :

$$\mathrm{MSE}(I, I') = \frac{\sum_{x=0}^{N-1} \sum_{y=0}^{M-1} (I(x,y) - I'(x,y))^2}{NM}$$

Cette grandeur permet en particulier de mesurer l'écart entre une image I et une version bruitée I' de I, ce qui justifie la terminologie d'*erreur quadratique moyenne*, ou *mean square error*, et l'acronyme MSE.

Écrire une fonction `computeMSE` qui retourne la valeur de MSE entre deux images.

1.4 Tests

Générer les images suivantes à partir de l'image `barbara.pgm` :

1. Bruit impulsionnel de 15% ;
2. Bruit impulsionnel de 40% ;
3. Bruit gaussien de moyenne $\mu = 0$ et d'écart-type $\sigma = 15$;
4. Bruit gaussien de moyenne $\mu = 0$ et d'écart-type $\sigma = 30$.

Calculer ensuite le MSE entre l'image bruitée et l'image de référence puis entre l'image débruitée et l'image de référence en utilisant les filtres suivants :

1. Filtre médian de taille 3 ;
2. Filtre médian de taille 7 ;
3. Filtre moyenneur de taille $2N + 1 = 3$; $2N + 1 = 7$;
4. Filtre gaussien avec $\sigma = 1$; $\sigma = 2$.

Faire un tableau résumant les résultats obtenus ; le meilleur résultat pour chaque type d'image sera mis en valeur (par exemple fonte grasse ou colorée).

2 Débruitage par moyennes non-locales

2.1 Principe

Le débruitage par moyennes non-locales est une méthode introduite par Buades et al. [1] particulièrement efficace pour réduire le bruit. Contrairement aux filtres par convolution qui utilisent un voisinage et des poids fixes, cette méthode utilise un voisinage théoriquement très grand (l'ensemble du domaine de l'image) et des poids variables, qui s'adaptent au contexte local du point traité. L'idée de la méthode est de tirer profit de la redondance présente dans les images pour estimer la valeur débruitée de chaque point (voir figures 1 et 2).

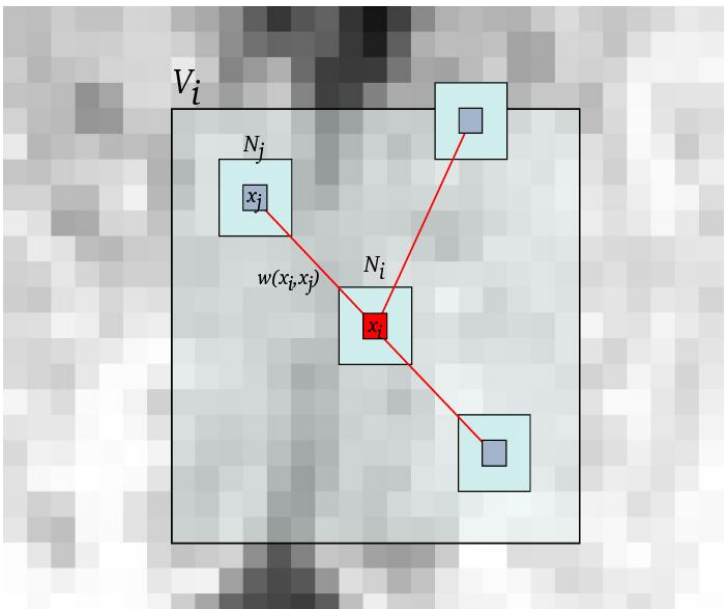

FIGURE 1 – La valeur du point central est calculée à partir d'une combinaison linéaire des valeurs de tous les points contenus dans l'espace de recherche. Plus la configuration du voisinage autour des deux points est similaire, plus la contribution (poids) de la valeur au point est importante.

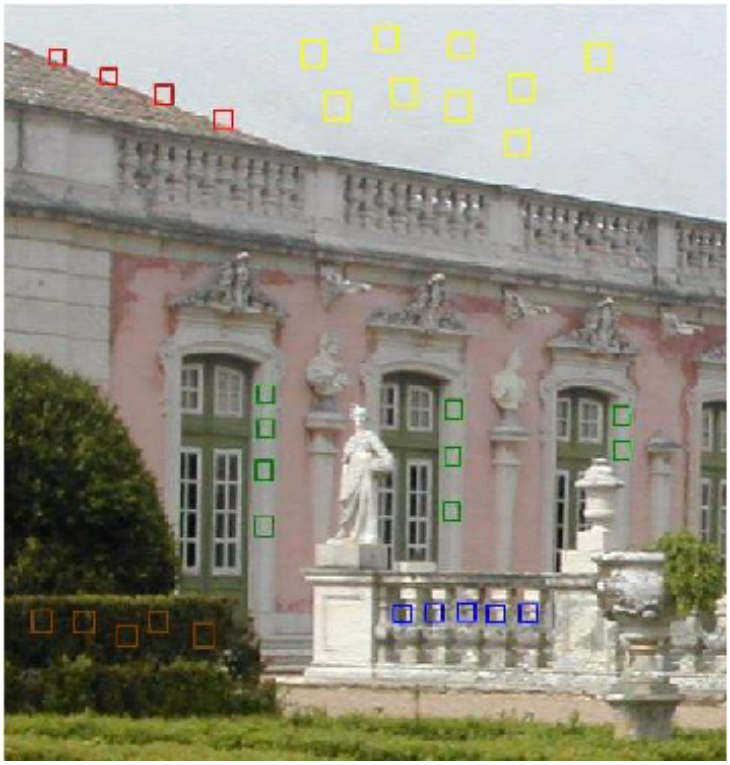

FIGURE 2 – *Patchs* d'images similaires (ayant la même configuration de voisinage) dans une image.

Plus précisément, pour une image donnée f, de support E, on calcule une nouvelle image $\mathrm{NL}(f)$, dont la valeur au point $x_i \in E$ est obtenue par combinaison linéaire des valeurs de tous les autres points x_j ($j \neq i$) de l'image, pondérées par une mesure de similarité entre le point à traiter x_i et le point x_j :

$$NL(f)(x_i) = \sum_{x_j \in E} w(x_i, x_j) f(x_j).$$

Le poids $w(x_i, x_j)$ assigné à chaque valeur $f(x_j)$ est déterminé ainsi :

1. on fixe un entier $D \geq 0$, et on considère les sous-images (*patchs*) $\mathbf{f}(x)$ de f, de support le carré de côté $2D+1$ centré en x ;
2. on calcule alors une mesure de similarité (norme euclidienne, en fait) entre les voisinages $\mathbf{f}(x_i)$ et $\mathbf{f}(x_j)$:

$$\|\mathbf{f}(x_i) - \mathbf{f}(x_j)\|_2^2 = \sum_{x=0}^{2D} \sum_{y=0}^{2D} \big(\mathbf{f}(x_i)(x-D, y-D) - \mathbf{f}(x_j)(x-D, y-D)\big)^2$$

3. on fixe un paramètre réel h ; la contribution de la valeur $f(x_j)$ au filtrage de x_i est proportionnelle à la quantité :

$$\exp\left(-\frac{\|\mathbf{f}(x_i) - \mathbf{f}(x_j)\|_2^2}{h^2}\right)$$

4. on calcule un facteur de normalisation sommant les contributions des mesures de similarité relativement à x_i, pour tous les points x_j :

$$Z(x_i) = \sum_j \exp\left(-\frac{\|\mathbf{f}(x_i) - \mathbf{f}(x_j)\|_2^2}{h^2}\right)$$

5. on peut finalement définir le poids de la contribution de x_j au filtrage de x_i :

$$w(x_i, x_j) = \frac{1}{Z(x_i)} \exp\left(-\frac{\|\mathbf{f}(x_i) - \mathbf{f}(x_j)\|_2^2}{h^2}\right)$$

Cette méthode est très coûteuse algorithmiquement : complexité en $\mathcal{O}(N^4(2D+1)^2$, où N^2 est la taille de l'image et $(2D+1)^2$ est la taille de la fenêtre de similarité, si bien que certains aménagements de l'algorithme sont nécessaires en pratique :

- Généralement, l'ensemble des points x_j utilisé pour filtrer le point x_i est restreint à une fenêtre carrée S, de taille $(2M+1)^2$, centrée autour du point x_i, appelée *espace de recherche*.
- Réduire la taille de l'espace de recherche permet d'augmenter significativement la rapidité de l'algorithme, mais réduit la probabilité de trouver des patchs d'image similaires, ce qui réduit donc l'efficacité du débruitage.
- Une autre optimisation, introduite par Coupé et al. [2], consiste à supprimer de l'espace de recherche tous les points x_j dont la similarité avec le point x_i est trop faible. L'idée est de précalculer, avant filtrage, moyenne et écarts types sur des voisinages de chaque point de l'image.

2.2 Implantation

La fonction principale `computeNLMeans` pourra s'appuyer sur les fonctions intermédiaires suivantes :

- `computeSimilarity` qui retourne la similarité entre deux patchs d'image centrés en (i_1, j_1) et (i_2, j_2). Si les patchs sont parfaitement identiques, cette fonction retourne 0. On considère que les points situés en dehors de l'image ont une contribution nulle dans le calcul ;
- `computeWeight` qui calcule (à un facteur près) le poids $w(x_1, x_2)$ avec $x_1 = (i_1, j_1)$ et $x_2 = (i_2, j_2)$ en utilisant la formule donnée plus haut, et s'appuie sur la fonction `computeSimilarity`.

2.2.1 Fonction `computeSimilarity`

Cette méthode calcule la similarité entre deux patchs (sous-images). La similarité se fonde sur le calcul de l'écart quadratique $\|\mathbf{f}(x_i) - \mathbf{f}(x_j)\|^2$, où $\mathbf{f}$ est déduit de f au moyen d'un patch de dimension `patchSize`.

2.2.2 Fonction `computeWeight`

Cette méthode calcule le poids défini par $w(x_1, x_2) = \exp\left(-\frac{\mathrm{sim}(i_1, j_1, i_2, j_2)}{h^2}\right)$, où $\mathrm{sim}(i_1, j_1, i_2, j_2)$ est la valeur de retour de la fonction `computeSimilarity`.

2.2.3 Fonction principale `computeNLMeans`

Cette fonction calcule, pour tous les points x_i de l'image source, la nouvelle valeur affectée à l'image de sortie selon la formule donnée *supra* :

$$NL(f)(x_i) = \sum_{x_j \in S} w(x_i, x_j) f(x_j).$$

L'espace de recherche S est une fenêtre centrée en x_i, de taille `windowSize`×`windowSize`.
La valeur $w(x_i, x_j)$ est, pour tout $x_j \neq x_i$ la valeur de retour de la fonction `computeWeight` normalisée par la somme des poids obtenus lors du filtrage de x_i (facteur Z). Dans le cas particulier où $x_j = x_i$, on prend pour valeur de $w(x_i, x_i)$ la valeur maximale des poids calculés dans la fenêtre S pour $x_j \neq x_i$.

Les paramètres d'entrée de la fonction `computeNLMeans` sont donc :

- l'image d'entrée ;
- `int windowSize` : entier impair définissant la taille de l'espace de recherche S ;
- `int patchSize` : entier impair définissant la taille d'un patch (`patchSize=2D+1`, avec les notations précédentes) ;
- `double h` : paramètre h (cf. plus haut) définissant la « force » du filtrage. Plus l'écart-type du bruit gaussien est grand, plus ce paramètre doit être élevé.

2.3 Tests

Testez votre filtre avec une image corrompue par du bruit gaussien d'écarts types $10, 15, 25$. Vous pourrez utiliser les paramètres suivants : `patchSize=7`, `windowSize=21`, `h=50.0`
Étudiez l'effet de l'augmentation de ces paramètres sur l'effet du filtre. Reprendre le tableau de la section 1.4 et le compléter avec les résultats du filtre NL-means.

Références

[1] Buades, A., Coll, B., Morel, J., 2005. A review of image denoising algorithms, with a new one. Multiscale Modeling and Simulation 4 (2), 490–530.

[2] Coupé, P., Yger, P., Prima, S., Hellier, P., Kervrann, C., Barillot, C., 2008. An optimized blockwise non local means denoising filter for 3D magnetic resonance images. IEEE Transactions on Medical Imaging 27 (4), 425–441.