Introduction au filtrage des images

1. Introduction

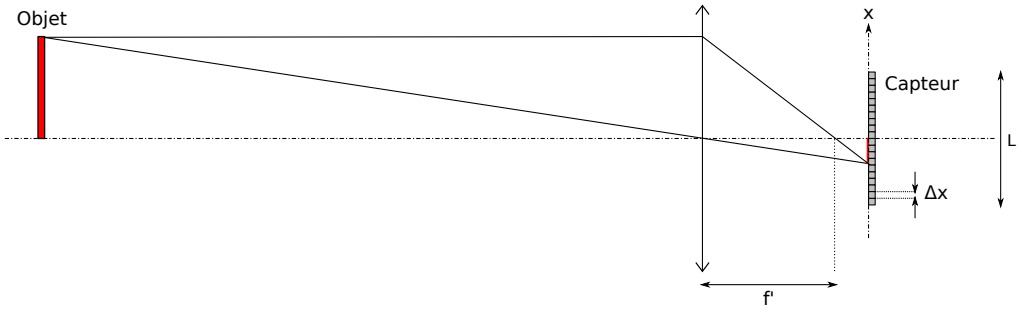
Ces travaux pratiques constituent une introduction au filtrage des images numériques. Les deux types de filtrages linéaires sont le filtrage par convolution et le filtrage par transformée de Fourier. Le filtrage permet de réduire le bruit dans une image (filtre passe-bas), d'accentuer les détails (filtre passe-haut) ou encore de détecter les contours des objets (filtre dérivateur).

Par définition, une image à niveaux de gris est une fonction de deux variables d'espace, à valeurs réelles. Le niveau de gris au point de coordonnées (x, y) est noté $V(x, y)$ (lettre V comme valeur). Il s'agit d'un nombre réel appartenant par convention à l'intervalle $[0, 1]$. La valeur 0 représente le noir, la valeur 1 représente le blanc, la valeur $0{,}5$ représente le gris moyen.

2. Échantillonnage d'une image et fréquences spatiales

Une image numérique est une image échantillonnée. Elle est représentée par une matrice $V_{j,i}$, dont chaque élément est nommé *pixel*. Le premier indice désigne une ligne de l'image ($0 \leq j \leq N_y - 1$), le second désigne une colonne ($0 \leq i \leq N_x - 1$).

Considérons le cas d'une image photographique, générée par un système optique et une matrice de photodiodes (capteur de type CCD). La figure suivante représente un système optique constitué d'une lentille de distance focale f'.



Chaque photodiode fournit le niveau de gris correspondant à l'énergie lumineuse qui lui parvient (pendant le temps d'exposition). L'échantillonnage consiste à attribuer ce niveau de gris au point situé au centre de la photodiode. On obtient ainsi un pixel de l'image numérique. Le pixel d'indices (j, i) correspond à la photodiode d'indices (j, i) de la matrice. Notons Δx la distance entre deux centres de photodiodes voisines du capteur, dans la direction x. Notons Δy la distance dans la direction y et supposons que $\Delta y = \Delta x$. Δx est la période d'échantillonnage spatiale, définie sur le capteur. Pour un capteur de taille donnée, la finesse des détails de l'objet visibles sur l'image sera d'autant plus faible que cette période est faible. La fréquence d'échantillonnage est une fréquence spatiale, égale à $1/\Delta x$. Si L_x désigne la largeur du capteur dans la direction x et L_y sa largeur dans la direction y, la fréquence d'échantillonnage est :

$$f_e = \frac{1}{\Delta x} = \frac{N_x}{L_x} = \frac{1}{\Delta y} = \frac{N_y}{L_y} \tag{1}$$

Considérons une image présentant une variation sinusoïdale dans la direction x :

$$V(x,y) = g(y)\sin\left(\frac{2\pi}{\lambda_x}x\right) \tag{2}$$

λ_x est la période spatiale et $1/\lambda_x$ est la fréquence spatiale. D'après la condition de Nyquist-Shannon, cette variation sinusoïdale est échantillonnée correctement si sa fréquence est strictement inférieure à la moitié de la fréquence d'échantillonnage :

$$\frac{1}{\lambda_x} < \frac{1}{2\Delta x} \tag{3}$$

Si cette condition n'est pas vérifiée, la sinusoïde peut donner un phénomène de repliement de spectre, qui peut se traduire par l'apparition d'un motif périodique dans l'image numériques, de fréquence inférieure à la fréquence de Nyquist (phénomène *d'aliasing*). Les systèmes optiques (comme les objectifs photographiques) se comportent naturellement comme des filtres passe-bas. Pour éviter le repliement, il faudra que la fréquence d'échantillonnage soit supérieure au double de la fréquence de coupure du système optique. Pour une taille de capteur donnée, la fréquence d'échantillonnage est d'autant plus grande que le nombre N_x de pixels (sur une ligne) est grand. Les fréquences d'échantillonnage des capteurs CCD sont de l'ordre de 200 pixels par millimètre, ce qui correspond à des photodiodes espacées de 5 micromètres. Cette fréquence est suffisante pour la plupart des systèmes optiques. Il faudra néanmoins faire attention à la condition de Nyquist-Shannon lorsqu'on voudra réduire le nombre de pixels d'une image numérique dans un logiciel.

La taille du capteur est une information généralement non disponible. Dans le cas d'une photographie scientifique, on s'intéresse en fait aux fréquences spatiales dans le plan de l'objet. Soit L_{xo} la largeur de l'image ramenée au plan objet. Elle peut être déterminée en plaçant dans ce plan une règle graduée ou un objet de taille connue. La fréquence d'échantillonnage dans le plan objet est N_x/L_{xo}. Lorsque l'objet est à l'infini, la fréquence d'échantillonnage est donnée en pixel par radian et peut se déduire de la distance focale de l'objectif.

Lorsqu'on traite une image numérique, les informations qui permettraient de connaître la fréquence d'échantillonnage soit au niveau du capteur soit au niveau de l'objet ne sont généralement pas connues, ou sont simplement ignorées. On définit alors des fréquences spatiales sans dimensions, en posant $L_x = N_x$ et $L_y = N_y$, c'est-à-dire en attribuant aux dimensions de l'image les tailles en nombre de pixels. Avec cette convention, la fréquence d'échantillonnage vaut 1 (sur les deux axes). Un motif périodique de fréquence f est un motif périodique dont la période spatiale est de $1/f$ pixels. La plus grande fréquence est la moitié de la fréquence d'échantillonnage, soit $1/2$, ce qui correspond à un motif périodique de période spatiale égale à 2 pixels.

3. Quantification des niveaux de gris

La quantification est l'opération de conversion d'un niveau de gris, défini comme un nombre réel dans l'intervalle $[0,1]$, en un nombre entier. Dans le cas d'une image photographique, la quantification est accomplie par le convertisseur analogique/numérique associé au capteur CCD. Par exemple, un convertisseur 12 bits donne des nombres entiers compris entre 0 et 4095. Lorsqu'une image numérique est stockée dans un fichier de type JPEG ou PNG, les entiers sont codés sur 8 bits et les niveaux de gris sont alors des entiers de 0 à 255. De même, si une image

est produite numériquement, elle subit la quantification au moment de son enregistrement dans un fichier.

Lors d'un traitement d'image, les calculs doivent être faits avec des nombres à virgule flottante afin d'éviter que la quantification introduise des erreurs d'arrondi trop grandes. Un nombre à virgule flottante représente un nombre réel, mais il s'agit en réalité d'un nombre entier auquel on adjoint un exposant (par exemple $3{,}1415926$ est le nombre 31415926 associé à l'exposant -7). Un flottant simple précision (32 bits) comporte un nombre de 23 bits (la mantisse), un exposant 8 bits et un bit pour le signe. Les calculs faits avec des flottants 32 bits introduisent des erreurs d'arrondis relatives de l'ordre de $2^{-23} = 10^{-7}$, ce qui est négligeable par rapport à la quantification des images dont les niveaux sont représentés en 8 bits. Pour les images numérisées en 14 bits (le maximum sur les appareils photo actuels), il peut être nécessaire de faire les calculs avec des flottants double précision (64 bits), qui comportent une mantisse de 53 bits.

Une conséquence de la quantification est que le nombre de niveaux de gris représenté est faible. Pour une image 8 bits, il y a 256 niveaux de gris. Cette valeur est suffisante pour la vision humaine, à condition que les 256 niveaux soient effectivement présents dans l'image. Par exemple, si on utilise seulement les 128 premiers niveaux, l'image photographique apparaît trop sombre. Pour éclaircir l'image, on est alors amené à multiplier toutes les valeurs par 2 ; les valeurs finales sont bien étalées de 0 à 255 mais seules les valeurs paires sont utilisées, ce qui fait seulement 128 niveaux utilisés. Si l'image était codée en 16 bits, il resterait à la fin 32768 niveaux de gris, ce qui est très largement suffisant. Le stockage des images en 16 bits est possible mais il est peu utilisé car les fichiers seraient deux fois plus volumineux. Il ne se justifie que si l'image doit subir un traitement numérique important.

4. Modèle d'image à une dimension

Dans ce travail, on s'interesse à une image ne présentant qu'une variation de niveau de gris dans la direction x. Autrement dit, on se ramène à un signal à une dimension, similaire aux signaux rencontrés en électronique. Ce modèle permettra d'introduire simplement le principe du filtrage, tout en permettant une implémentation simple et rapide. La généralisation à une image quelconque ne pose pas de problème car le filtrage sur un axe x est indépendant du filtrage sur l'axe perpendiculaire y.

L'image considérée est donc une fonction $V(x)$, par convention définie sur l'intervalle $[0, 1]$ et à valeurs dans $[0, 1]$, que l'on échantillonne avec N pixels. Elle sera stockée dans un tableau de type `numpy.ndarray` à une dimension et de taille N, contenant des flottants 32 bits (`numpy.float32`).

On rappelle l'existence de la fonction `numpy.arange(N,dtype=numpy.float32)`, qui permet de générer un tableau contenant les entiers (sous forme de flottants 32 bits) de 0 à N-1, et de la fonction `numpy.zeros(N,dtype=numpy.float32)` qui génère un tableau initialisé à 0.

[1] Écrire une fonction `sinus(periode,N)` qui renvoie un tableau de taille `N` représentant un motif sinusoïdal dont la période en nombre de pixels est donnée par `periode`. La valeur moyenne de la sinusoïde est égale à $0{,}5$ et l'amplitude est $0{,}4$. Tester la fonction avec $N = 1000$ et différentes valeurs de la période. Tracer la courbe représentant V en fonction de l'indice. Quelle est la période minimale pour respecter la condition de Nyquist-Shannon ?

[2] Écrire une fonction `creneau(periode,N)` qui génère un motif en créneau de période donnée (même valeur moyenne et même amplitude que précédemment).

Les images générées par les capteurs CCD comportent du bruit, c'est-à-dire que les niveaux de gris sont perturbés par des petites fluctuations aléatoires qui s'ajoutent à l'image optique. Le bruit est essentielement d'origine thermique et il est d'autant plus présent que le temps d'exposition est grand. Pour modéliser le bruit, on fait généralement l'hypothèse qu'il est de type gaussien, c'est-à-dire que les perturbations aléatoires obéissent à une loi de probabilité normale (ou loi gaussienne). La génération d'un tableau de `N` nombre aléatoires à distribution normale, de valeur moyenne `mu` et d'écart type `sigma` est obtenue par : `numpy.random.normal(mu,sigma,N)`.

[3] Ajouter un faible bruit gaussien aux deux images générées précédémment et observer le résultat sur les courbes.

Afin de voir l'effet de la quantification, considérons la conversion en entiers 8 bits. Pour convertir un tableau `V` de flottants dans l'intervalle $[0, 1]$ en un tableau de nombres entiers codés sur 8 bits, il faut écrire :

```
numpy.array(V*255,dtype=numpy.uint8)
```

Le type `uint8` désigne un entier non signé 8 bits, compris entre 0 et 255.

[4] Effectuer la quantification sur les images générées (sans et avec bruit) et observer l'effet de la quantification.

Dans toute la suite du travail, on utilisera les images non quantifiées (nombres flottants 32 bits, dont la quantification est négligeable).

5. Filtre de convolution

Un filtre de convolution est défini par des coefficients (constituant le noyau du filtre). On adopte généralement un nombre de coefficients impair $(2P + 1)$. Les coefficients sont notés b_k avec l'indice k variant de $-P$ à $+P$. Le produit de convolution de l'image V par ce noyau consiste à créer une nouvelle image V' dont les pixels sont définis par :

$$V'_i = \sum_{k=-P}^{P} V_{i+k} b_k \tag{4}$$

Autrement dit, on remplace chaque pixel par une moyenne pondérée de lui-même et de ses voisins. La convolution est centrée sur le pixel, c'est-à-dire qu'elle se fait avec P pixels situés à sa gauche et avec P pixels situés à sa droite. On peut comparer cela à la convolution effectuée en électronique numérique sur un signal filtré en temps réel : la convolution se fait alors seulement sur les échantillons du passé (à gauche) puisque l'avenir n'est pas connu.

La suite des coefficients b_k constitue aussi la réponse impulsionnelle du filtre. Le filtrage par convolution ne doit pas modifier le niveau de gris moyen de l'image. Pour cela, il faut que les coefficients du filtre soient normalisés. Si tous les coefficients du filtre sont positifs, la normalisation consiste à multiplier les coefficients par un facteur afin que leur somme soit égale à 1 :

$$\sum_{k=-P}^{P} b_k = 1 \tag{5}$$

Le calcul du produit de convolution (4) pose un problème sur les bords de l'image. Il faut en effet que le pixel d'indice $i+k$ appartienne à l'image pour k variant de $-P$ à P : il faut donc que

i varie entre P et $N - P - 1$. Si on se limite à ces pixels, il reste P pixels non filtrés à gauche et P pixels non filtrés à droite. Pour éviter cela, une première approche consiste à calculer les produits de convolution aussi pour ces pixels mais en utilisant seulement les pixels de V disponibles. Par exemple, le produit de convolution du premier pixel ne fait intervenir que ce pixel lui-même et les P pixels situés à sa droite. L'inconvénient de cette méthode est que pour ces pixels la condition de normalisation n'est plus respectée, ce qui aura pour conséquence un léger assombrissement de l'image près des bords.

[5] Écrire une fonction `convolution(V,N,B,P)` qui calcule le tableau V' et le renvoie. Le tableau `V` contient l'image de taille `N` et le tableau `B`, de taille `2*P+1`, contient les coefficients du filtre.

[6] Définir un filtre moyenneur simple, c'est-à-dire un filtre dont tous les coefficients sont égaux (tout en respectant la condition de normalisation). On pourra utiliser la fonction `numpy.ones`. Tester ce filtre sur les images bruitées pour différentes valeurs de P (de 1 à 10).

La réponse fréquentielle du filtre est obtenue facilement avec la fonction `scipy.signal.freqz`. Le code suivant permet de tracer le gain du filtre en fonction de la fréquence spatiale (définie avec $L_x = N_x$ comme expliqué plus haut).

```
figure()
w,H = freqz(B)
plot(w/(2*numpy.pi),numpy.absolute(H))
grid()
xlabel("f")
```

[7] Tracer la réponse fréquentielle du filtre moyenneur pour $P = 5$. En déduire par lecture graphique sa fréquence de coupure (définie par un gain 1/2). À quelle période (exprimée en pixels) cette fréquence de coupure correspond-elle ?

Le filtre moyenneur simple fonctionne mais sa réponse fréquentielle n'est pas optimale car elle présente des variations non monotones dans la bande atténuante. On lui préfère généralement le filtre gaussien.

6. Filtre passe-bas gaussien

Un filtre passe-bas a pour fonction de réduire les composantes de haute fréquence de l'image. Par exemple, un filtre passe-bas de fréquence de coupure $0{,}2$ atténue les variations périodiques de période inférieure à 5 pixels. Les hautes fréquences d'une image sont associées aux détails les plus fins ou aux bords nets des objets.

Un filtre passe bas gaussien (à une dimension) est défini par la réponse impulsionnelle suivante :

$$h(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{6}$$

Cette fonction permet de définir les coefficients du filtre. Pour définir un filtre à $2P + 1$ coefficients on pose $b_k = h(k)$ pour $k = -P, -P + 1, ..., P$. Le paramètre σ est choisi de telle mznière que les coefficients du bord du filtre (b_{-P} et b_P) soient très faibles. Notons ϵ cette valeur, on obtient :

$$\sigma = \frac{P}{\sqrt{-2\ln(\epsilon)}} \tag{7}$$

Après avoir calculé σ puis les coefficients du filtre par la relation $b_k = h(k)$, il faut calculer leur somme afin de procéder à la normalisation.

[8] Écrire une fonction `passeBas(P,epsilon)` qui renvoie le tableau des coefficients du filtre gaussien passe-bas.

[9] Pour $P = 5$ et $\epsilon = 0,05$, représenter graphiquement le filtre avec la fonction `matplotlib.pyplot.stem`.

[10] Tracer la réponse fréquentielle (gain en fonction de la fréquence) pour $P = 5$. Comparer au filtre moyenneur et conclure.

[11] Trouver la valeur de P nécessaire pour avoir une fréquence de coupure de $0{,}1$ (correspondant à une période de 10 pixels).

[12] Tester le filtre sur les images bruitées, pour différentes valeurs de P.

7. Filtre dérivateur

La dérivation est couramment utilisée en traitement d'image car elle permet de faire ressortir les bords des objets, qui sont caractérisés par des variations rapides de niveau de gris. Voici les deux formes les plus simples de filtres dérivateurs :

$$b = (-\frac{1}{2}, \frac{1}{2})$$
$$b = (-\frac{1}{2}, 0, \frac{1}{2})$$

On préfère en général le second, qui permet de calculer une convolution centrée.

[13] Tracer la réponse fréquentielle du filtre dérivateur puis l'appliquer aux deux images bruitées. On remarquera que l'image résultante contient des valeurs négatives. Quel est l'inconvénient du filtre dérivateur ?

La dérivation s'accompagne d'une augmentation du bruit. Pour réduire cet effet, on associe le filtre dérivateur à un filtre passe-bas. On pourrait appliquer tout d'abord la dérivation puis un filtre passe-bas gaussien (ou l'inverse), mais il est plus efficace d'utiliser un filtre qui fait en même temps la dérivation et le filtrage passe-bas. Cela se fait à partir de la fonction suivante :

$$h(x) = x \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{8}$$

Le paramètre σ est calculé pour que la valeur sur les bords du filtre soit ϵ, ce qui conduit à la relation :

$$\sigma = \frac{P}{\sqrt{2(\ln(P) - \ln(\epsilon))}} \tag{9}$$

La normalisation consiste à diviser les coefficients pour que la somme des coefficients positifs soit égale à 1, et la somme des coefficients négatifs égale à -1.

[14] Écrire une fonction `derivateurPasseBas(P,epsilon)` qui renvoie le tableau des coefficients du filtre gaussien dérivateur et passe-bas.

[15] Tester ce filtre (pour $P = 5$) sur les deux images bruitées et comparer au dérivateur simple.

8. Filtrage par transformée de Fourier

Le filtrage par convolution est couramment utilisé lorsque le nombre de coefficients n'est pas trop grand. Pour obtenir des effets de filtrage plus forts, correspondant à une convolution avec un grand nombre de coefficients, on utilise plutôt le filtrage par transformée de Fourier, qui consiste à filtrer dans le domaine fréquentiel.

La première étape consiste à calculer la transformée de Fourier discrète de l'image (TFD). Dans le cas d'une image à une dimension que nous traitons ici, la TFD se calcule comme la TFD des signaux échantillonnés en électronique :

```
from numpy.fft import fft
N=len(V)
tfd = fft(V)*2/N
```

Le tableau obtenu comporte N nombres complexes, chacun correspondant à une certaine fréquence spatiale. `tfd[0]` correspond à la fréquence nulle, c'est-à-dire au niveau de gris moyen de l'image. `tfd[1]` correspond à la fréquence $1/N$, `tfd[2]` à la fréquence $2/N$, etc. jusqu'à `tfd[N-1]` qui correspond à la fréquence $(N-1)/N$, c'est-à-dire la fréquence d'échantillonnage moins $1/N$. Par ailleurs, `tfd[N-i]` est le complexe conjugué de `tfd[i]`, ce qui signifie que le terme de fréquence i/N a le même module que le terme de fréquence $1 - i/N$.

[16] Calculer la TFD des deux images bruitées et tracer son module en fonction de la fréquence. Pour le spectre de l'image sinusoïdale, vérifier la position et la hauteur des raies.

Le filtrage s'effectue sur la TFD. Il s'agit de multiplier les valeurs de la TFD par un gain qui dépend de la fréquence (le déphasage est nul). Soit $G(f)$ la fonction de gain à appliquer, la fréquence f étant définie dans l'intervalle $[0, 1/2]$. Il faut appliquer le gain aux éléments de la TFD de fréquences 0 à $1/2$ mais aussi aux éléments de fréquences $1/2$ à 1. Soit F_i l'élément d'indice i de la TFD. La transformation à appliquer est :

$$F'_i = G\left(\frac{i}{N}\right) F_i \text{ pour } 0 \leq i < \frac{N}{2} \tag{10}$$

$$F'_i = G\left(1 - \frac{i}{N}\right) F_i \text{ pour } \frac{N}{2} \leq i < N \tag{11}$$

Considérons par exemple un filtre gaussien, dont le gain est défini par :

$$G(f) = \exp\left(-\frac{f^2}{2\sigma^2}\right) \tag{12}$$

Soit f_c la fréquence de coupure, pour laquelle le gain vaut $1/2$. Le paramètre σ est donné par :

$$\sigma = \frac{f_c}{\sqrt{2\ln(2)}} \tag{13}$$

[17] Écrire une fonction `filtrageFourierPasseBas(tfd,N,fc)` qui effectue la multiplication de la TFD par le gain de ce filtre passe-bas. Elle renvoie la nouvelle TFD. Appliquer cette fonction à la TFD de l'image créneau et représenter le nouveau spectre pour $f_c = 0{,}03$.

La dernière étape consiste à calculer la transformée discrète inverse de la nouvelle TFD, de la manière suivante :

```
V_filtre = numpy.fft.ifft(tfd)*N/2
```

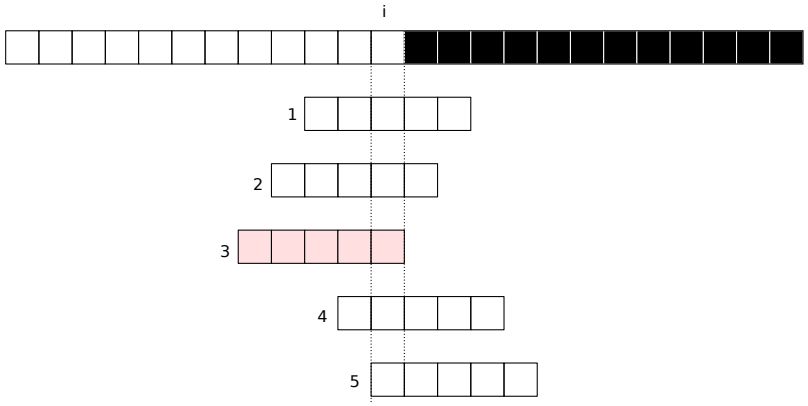
[18] Effectuer le filtrage de l'image en créneau pour $f_c = 0{,}03$.

[19] Déterminer la taille d'un filtre gaussien de convolution qui aurait le même effet. Comparer l'efficacité du filtrage par convolution et du filtrage par transformée de Fourier, sachant que les fonctions `fft` et `ifft` implémentent l'algorithme de transformée de Fourier rapide, de complexité $O(Nln(N))$.

9. Filtre réducteur de bruit avec conservation de la netteté

Le filtre de convolution passe-bas étudié dans la partie 6 permet de réduire le bruit présent dans une image. Il a cependant l'inconvénient de réduire aussi sa netteté. En effet, les pentes des transitions de l'image en créneau (correspondant à des bords nets) sont réduites par ce filtrage. Le filtrage par convolution peut être adapté pour effectuer un filtrage du bruit sans trop altérer la netteté.

La figure suivante montre des pixels au voisinage d'un bord net avec des pixels blancs à gauche du bord et des pixels noirs à droite. Supposons que l'on cherche à calculer le produit de convolution pour le pixel d'indice i, situé juste à gauche du bord, avec un fitre à 5 coefficients. Si la convolution est centrée sur ce pixel, elle conduira nécessairement à remplacer le bord net par une transition plus douce, ce qui donnera un bord flou. L'idée est de décentrer la convolution. Pour un filtre à 5 coefficients, il y a *a priori* 5 positions possibles pour l'application de la convolution.



Parmi toutes les positions possibles, on choisit celle qui contient les pixels dont les niveaux de gris sont les plus proches de celui du pixel i. Il s'agit de sélectionner la fenêtre de 5 pixels (contenant le pixel i) dont la variance des niveaux de gris est la plus faible. Sur cet exemple, il s'agit de la position 3 : la convolution pour le pixel i est alors calculée avec le pixel i et avec les 4 pixels situés à sa gauche. La variance d'un ensemble de valeurs v_i est définie comme la moyenne des valeurs au carré moins le carré de la moyenne des valeurs :

$$\sigma^2 = < v_j^2 > - < v_j >^2 \tag{14}$$

Pour chaque pixel de l'image, il faut donc disposer de la variance des pixels de toutes les fenêtres à $2P+1$ pixels contenant ce pixel. Il suffit de connaître, pour chaque pixel, la variance

d'une fenêtre de largeur $2P+1$ centrée sur ce pixel. Ces variances peuvent être calculées avant le filtrage et être stockées dans un tableau de la même taille que l'image. On constate par ailleurs que le calcul du tableau contenant les moyennes ($< v_j >$) se fait simplement en appliquant à l'image le filtre de convolution moyenneur de taille $2P+1$, déjà étudié dans la partie 5. Il s'agit d'un filtre dont les coefficients sont égaux à $1/(2P+1)$. De même, la moyenne du carré ($< v_j^2 >$) se calcule en filtrant le carré de l'image par le filtre moyenneur.

L'algorithme est donc le suivant, pour une image stockée dans le tableau V. On commence par calculer un tableau `variance` de taille N de la manière suivante :

- ▷ Calcul du tableau contenant les carrés des valeurs des pixels, noté V^2.
- ▷ Application du filtre moyenneur de taille $2P+1$ à V^2, ce qui donne un tableau A.
- ▷ Application du filtre moyenneur de taille $2P+1$ à V, ce qui donne un tableau B.
- ▷ Calcul du tableau contenant les carrés des valeurs de B, noté C.
- ▷ Calcul du tableau des variances `variance=A-C`.

Pour filtrer le pixel d'indice i, on consulte dans le tableau de variance les éléments d'indices $i-P$ à $i+P$. Soit j l'indice de l'élément dont la variance est la plus faible. Cet indice correspond au centre de la convolution qu'il faut appliquer, c'est-à-dire qu'il faut calculer le produit de convolution suivant :

$$V'_i = \sum_{k=-P}^{P} V_{j+k} b_k \tag{15}$$

[20] Écrire une fonction `calculVariance(V,N,P)` qui renvoie le tableau des variances de l'image `V`. On rappelle que l'opérateur $\star$ appliqué à deux tableaux `numpy.ndarray` permet de les multiplier terme à terme.

[21] Calculer le tableau des variances pour l'image en créneau bruitée puis le représenter graphiquement.

[22] Écrire une fonction `minimumVariance(variance,N,P,i)` qui renvoie l'indice du centre de la fenêtre (de largeur $2P+1$ contenant le pixel i) donnant la variance minimale. On fera attention à bien traiter le cas des pixels proches des bords, pour lesquels seulement certaines positions parmi les $2P+1$ sont à explorer.

[23] Écrire une fonction `filtrageAdaptatifPasseBas(V,N,B,P)` qui effectue un filtrage passe-bas gaussien en utilisant cet algorithme.

[24] Appliquer le filtrage adaptatif à l'image en créneau bruitée avec $P=10$ et comparer avec le filtrage par convolution centrée. Tracer les deux images filtrées sur la même figure et conclure.

10. Solution

filtrage-images.py