TP2 : Filtrage non linéaire

Pour palier le problème soulevé par les filtres linéaires (filtrage identique sur le bruit et sur les contours), on utilise des filtres non linéaires, destinés à supprimer le bruit sans détériorer les contours.

L'objectif du tp est de tester et comparer différentes méthodes de lissages non linéaires.

1 Filtre médian

Il consiste à remplacer le pixel central par la valeur médiane du voisinage (taille impaire) dans lequel il se trouve. Ce filtre ne tient pas compte de la hauteur du bruit, au contraire des filtres linéaires, puisqu'il opère un classement relatif.

Par exemple, dans l'image suivante $\begin{bmatrix} 40 & 50 & 60 \\ 50 & 82 & 75 \\ 80 & 90 & 95 \end{bmatrix}$, le pixel central de valeur 82 est remplacé par la valeur 75 par un filtrage médian de taille 3×3.

Tout point hors de la norme est donc rejeté, quelque soit sa valeur. Il n'introduit donc pas de nouvelles valeurs d'intensité et réduit la variance des intensités. Il permet d'éliminer les valeurs incorrectes isolées. Il permet d'éliminer les oscillations d'intensité de période inférieure à la taille de la fenêtre ainsi que les bruits impulsionnel.

La fenêtre utilisée pour le filtrage peut être une fenêtre carrée $n \times n$, une croix $n \times n$, une ligne ou une colonne. Les effets de ce filtre varient significativement en fonction de cette fenêtre. Pour des grandes tailles, le filtre utilisant une fenêtre carrée a tendance à éliminer (en les arrondissant) les angles et les coins des contours, car toute impulsion inférieure à la taille du filtre est éliminée, quelque soit sa direction (un angle est une impulsion dans une direction donnée).

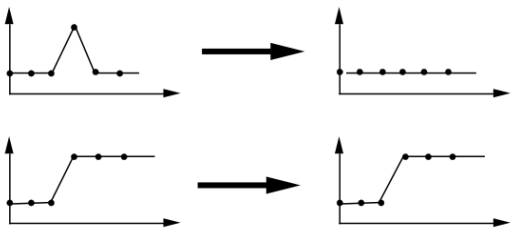

FIGURE 1 – 2 signaux filtrés par la médiane : en haut, le bruit impulsionel est supprimé ; En bas, le contour est conservé

1.1 Implémentation

Le temps d'exécution de ce filtre peut etre long car il impose un tri sur $(2n+1).(2n+1)$ valeurs, et augmente rapidement avec la taille du voisinage. Une manière élégante de réduire les temps d'exécution d'un médian de grande taille consiste à utiliser l'histogramme cumulé qui s'obtient facilement entre 2 positions voisines de la fenêtre de traitement. L'algorithme est alors le suivant :

Algorithme 1 : Median par histogramme

input : im :image, N : taille de l'image, n :demi taille du filtre
output : ims :image débruitée
for *chaque ligne i de l'image (à partir de la ligne n)* **do**
  Calculer l'histogramme $h(k)$ d'une fenêtre $(2.n+1).(2.n+1)$ autour du point d'indice i, n;
  $ims(i,j) = k$ tel que $\sum_{l=0}^{k} h(l) \geq 2n^2 + 2n = \frac{(2.n+1)^2}{2}$;
  for *tous les points d'indice i, j de la ligne i* **do**
    Supprimer de l'histogramme h les valeurs des points de la colonne $j - n - 1$;
    Ajouter à l'histogramme h les valeurs des points de la colonne $j + n$;
    $ims(i,j) = k$ tel que $\sum_{l=0}^{k} h(l) \geq 2n^2 + 2n = \frac{(2.n+1)^2}{2}$;
  end
end

Cet algorithme présente un temps d'exécution linéaire en fonction de la taille du filtre. Il a la particularité de n'effectuer aucun tri.

2 Filtres adaptatifs

Ces filtres sont des opérateurs de lissage qui modifient leur comportement ou leurs coefficients en fonction du contexte local. Ils comportent généralement des seuils qui leur permettent de s'adapter aux valeurs de bruits à éliminer. Leur inconvénient est la connaissance requise à priori du type de bruit et des valeurs du bruit.

2.1 Filtre adaptatif récursif

L'idée suivie par ces auteurs est la suivante : le filtrage consiste d'une part à réduire la variance du bruit dans les zones homogènes, d'autre part à conserver les fronts correspondant à bords des objets. Il s'agit donc de filtrer uniquement les zones homogènes pour éviter l'adoucissement des contours. Malheureusement, l'emplacement de ces contours est inconnu, auquel cas notre problème serait résolu. Ils proposent donc de filtrer une image par une fonction dépendant du gradient de l'image, de telle sorte que le lissage soit fort pour les gradients faibles (zones relativement homogènes) et faible pour les gradients forts (présence probable de contours).

Le filtrage gaussien peut de plus être approximé par une succession de filtres moyenneurs de taille 1,

mais passés successivement plusieurs fois. Si $S^0(x)$ est le signal à filtrer, le filtre itératif devient :

$$S^{t+1}(x,y) = \frac{\sum_{i=-1}^{1}\sum_{j=-1}^{1} w^t(x+i,y+j).S^t(x+i,y+j)}{\sum_{i=-1}^{1}\sum_{j=-1}^{1} w^t(x+i,y+j)}$$

$$w^t(x,y) = e^{-\left(\frac{\left(S^t(x+1,y)-S^t(x-1,y)\right)^2+\left(S^t(x,y+1)-S^t(x,y-1)\right)^2}{2k^2}\right)}$$

Ce filtrage est répété itérativement jusqu'à ce que l'image filtrée soit stationnaire.

Il y a deux opérations distinctes réalisées par le filtrage : d'une part, le rehaussement des contours qui doivent rester présents dans l'image finale et d'autre part le lissage des régions homogènes. Le premier effet est obtenu juste après quelques itérations tandis que le second peut être très long (100 à 200 itérations au moins). La valeur du paramètre k correspond à peu près au seuil pour lequel les contours seront conservés et accentués si leur gradient est supérieur à ce seuil k. Il est donc directement lié au niveau de bruit de l'image.

Remarques

1. on peut utiliser une version dégradée en considérant $w^t = w^0$. Le résultat est légèrement moins bon avec cette version, mais beaucoup plus rapide.
2. ce filtre est très proche du filtrage par diffusion isotrope non linéaire. En 1D, il s'écrit en

$$\begin{aligned} S^{t+1}(x) &= c^t(x+1).S^t(x+1) + c^t(x).S^t(x) + c^t(x-1).S^t(x-1) \\ &\text{avec} \quad c^t(x+1) + c^t(x) + c^t(x-1) = 1 \\ &= c^t(x+1).S^t(x+1) + \left(1 - c^t(x+1) - c^t(x-1)\right).S^t(x) + c^t(x-1).S^t(x-1) \end{aligned}$$

$$\text{D'où} \quad S^{t+1}(x) - S^t(x) = c^t(x+1).\left(S^t(x+1) - S^t(x)\right) - c^t(x-1).\left(S^t(x) - S^t(x-1)\right)$$

$$\text{et} \quad \frac{\partial S}{\partial t} = \nabla(c.\nabla I) \text{ qui est l'équation de la chaleur}$$

Si c est une constante, la solution de cette équation est le lissage gaussien, et le temps t est lié à la largeur du lissage par $\sigma = \sqrt{2t}$. Un nombre infini d'itérations conduit à une image lissée constante. Dans notre cas, c est un scalaire qui dépend des caractéristiques locales de l'image et le lissage converge vers une image constante par morceaux.

2.2 Filtre bilatéral

Les coefficients de ce filtre dépendent non seulement de la distance euclidienne au point à lisser (comme pour la gaussienne), mais aussi de la ressemblance des intensités entre le pixel central et le pixel du masque.

$$S(x,y) = \frac{\sum_{i=-3\sigma_1}^{3\sigma_1}\sum_{j=-3\sigma_1}^{3\sigma_1} e^{-\frac{i^2+j^2}{2\sigma_1^2}}.e^{-\frac{(I(x+i,y+j)-I(x,y))^2}{2\sigma_2^2}}.I(x+i,y+j)}{\sum_{i=-3\sigma_1}^{3\sigma_1}\sum_{j=-3\sigma_1}^{3\sigma_1} e^{-\frac{i^2+j^2}{2\sigma_1^2}}.e^{-\frac{(I(x+i,y+j)-I(x,y))^2}{2\sigma_2^2}}}$$

De ce fait, les pixels se ressemblant provoqueront un fort lissage (ie des zones homogènes) alors que des pixels ne se ressemblant pas provoqueront de faibles lissages.

L'exemple figure 2 montre l'image à gauche, les coefficients du masque situé à 2 pixels sur la droite de la frontière, et l'image lissée.

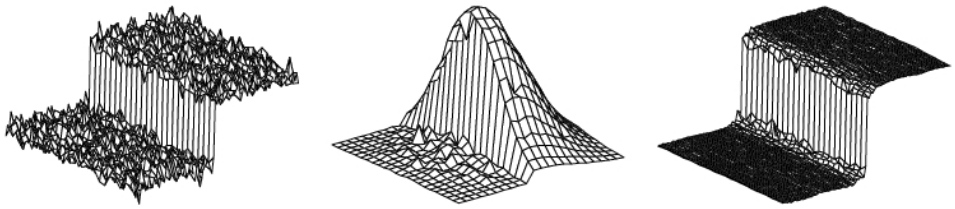

FIGURE 2 – Filtre bilateral : à gauche, une image avec un contour ; Au centre, la valeur des coefficients du filtre bilatéral ; A droite, l'image lissée par le filtre bilateral

Remarques

1. Ce filtre peut etre lui aussi itéré plusieurs fois. Il tend alors à donner des images constantes par morceaux.

2. Si la valeur de σ_2 est liée au bruit de l'image, celle de σ_1 est comme pour la gaussienne classique liée à la taille des détails que l'on souhaite préserver.

3. Dans le cas d'un bruit poivre et sel, le bruit modifie beaucoup les valeurs de l'image, mais de manière éparse dans l'image. L'influence du terme de distance entre les intensités devient alors très faible. Pour pallier ce problème, il est courant de remplacer, dans le calcul des distances entre intensités, ces intensités par une version lissée par un médian, qui est un estimateur robuste. Le filtre devient :

$$S(x,y) = \frac{\sum_{i=-3\sigma_1}^{3\sigma_1} \sum_{j=-3\sigma_1}^{3\sigma_1} e^{-\frac{i^2+j^2}{2\sigma_1^2}} . e^{-\frac{(Med(I(x+i,y+j))-Med(I(x,y)))^2}{2\sigma_2^2}} . I(x+i,y+j)}{\sum_{i=-3\sigma_1}^{3\sigma_1} \sum_{j=-3\sigma_1}^{3\sigma_1} e^{-\frac{i^2+j^2}{2\sigma_1^2}} . e^{-\frac{(Med(I(x+i,y+j))-Med(I(x,y)))^2}{2\sigma_2^2}}}$$

4. On peut aussi remplacer la fonction gaussienne par une fonction lorentzienne $f(x) = \frac{\frac{2}{\pi\gamma}}{1+(\frac{x}{\gamma/2})^2}$ ou une fonction de Huber

2.3 Méthodes à base de patch : moyenne non locale

Le filtrage classique réalise une moyenne pondérée des pixels voisins, sous l'hypothèse que la plupart d'entre eux appartiennent à des zones homogènes et doivent donc etre semblables du point de vue statistique. De ce fait, le SNR est amélioré, mais les contours sont atténués.

L'idée du filtre Nlmeans est d'utiliser la redondance présente dans la plupart des images, cette redondance n'étant pas spécifiquement locale. De ce fait, il est possible de faire le lissage (moyenne pondérée) de pixels qui ne sont pas voisins. Le patch est un voisinage de petite taille autour d'un pixel. La région est un voisinage de plus grande taille autour de ce pixel. Le principe de cette méthode, pour traiter un pixel, est donc de rechercher d'autres pixels dans la région qui ont le même environnement ou patch. Le lissage consiste à réaliser la moyenne pondérée de ces pixels dont les patchs se ressemblent, la pondération étant proportionnelle à la dissemblance entre 2 patchs. L'algorithme s'écrit :

Algorithme 2 : Nl-Means

input : im :image, t :taille des régions, r :taille des patches ;
output : ims :image débruitée
for *tous les pixels p de coordonnées* X, Y *de l'image im* **do**
  soit P le patch de taille $(2r+1).(2r+1)$ centré sur le pixel p;
  soit $R_t(p)$ la région de taille $(2t+1).(2t+1)$ centré sur le pixel p;
  for *tous les pixels* $q \in R_t(p)$ *de coordonnées* x, y **do**
    soit Q le patch de centre q de taille $(2r+1).(2r+1)$;
    $d^2(P,Q) = \sum_{i=-r}^{r} \sum_{j=-r}^{r} \left((im(x+i, y+j))^2 - (im(X+i, Y+j))^2\right)$;
    w(p,q)=$e(-\frac{d^2(P,Q)}{2.\sigma^2})$;
  end
  $ims(p) = \frac{\sum_{q \in R_t(p)} w(p,q).im(q)}{\sum_{q \in R_t(p)} w(p,q)}$;
end

3 Estimation du bruit

Il existe de nombreuses méthodes pour estimer le bruit. Nous utiliserons ici la methode du pourcentile, adaptée à un bruit gaussien stationnaire. Cette méthode est basée sur le calcul de la variance locale dans des blo cs de petite taille de l'image. Cette variance correspond essentiellement au bruit gaussien de l'image, perturbée à certaines positions par les contours ou la texture de l'image. Comme les contours sont peu nombreux, l'histogramme des variances locales est essentiellement un direac centré sur la variance du bruit, et n'est affecté que dans sa partie la plus à droite (les plus fortes variances) par les contours. La méthode propose donc de calculer la variance à partir d'un très faible pourcentile de l'histogramme des variances locales. Il faut dans un premier temps supprimer les variations lentes dues au signal utile grâce à un filtre passe-haut, comme les filtres basés sur les opérateurs différentiels (gradient, laplacien) ou sur des ondelettes.

L'algorithme devient alors :

Algorithme 3 : Estimation du bruit

input : Entrées : im :image, t :taille des blocs, p :pourcentile ;
output : La valeur de la variance du bruit
Filtrage passe-haut : convolution par un masque $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$;
for *tous les pixels de l'image filtrée* **do**
  calcul de la variance locale du bloc de taille $(2t+1).(2t+1)$ centré sur le pixel;
  Mise à jour de l'histogramme des variances locales;
end
variancefinale : 1.13* variance qui atteint p% de l'histogramme cumulé;

La variance calculée est naturellement un peu inférieure à la variance réelle. Le facteur 1.13 est un facteur multiplicatif destiné à corriger cette sous estimation. Il est obtenu de manière empirique. La taille des blocs et la valeur de pourcentile dépendent de la variance réelle du bruit et du filtre passe-haut utilisé. Pour des variances inférieures à 60, une taille de bloc de $t = 10$ et une valeur de $p = 0.5$ sont de bons paramètres. Au dela de 60, une taille de bloc de $t = 15$ et une valeur de $p = 50$ sont de meilleurs paramètres.

Comme la variance réelle est inconnue, on peut estimer une première fois le bruit avec un jeu de

paramètres standard $t = 10, p = 0.5$, puis avec le jeu le plus approprié selon cett e première estimation.

4 Travail à réaliser

L'objectif est de comparer quantitativement les différents filtres en calculant le PSNR et le temps d'exécution sur des images sur lesquelles un bruit connu est ajouté, et qualitativement sur des images naturelles.

- Programmer les différents filtres non linéaires présentés, ainsi que l'estimation du bruit
- Proposer une estimation des paramètres des différents filtres
- Etablir et comparer PSNR et temps CPU de ces filtres sur les images formes2yyxx.pgm et etablir un classement de performances
- Comparer d'un point de vue qualitatif l'effet de ces filtres sur les autres images