Institut National des Sciences Appliquées et de Technologie

Traitement d'images, TP3
Filtrage spatial - Convolution
Wided MILED - RT4 -

1 Dégradations dans une image

1.1 Rappels de cours

Lors de l'acquisition, de la transmission ou de la compression d'une image, il peut apparaître de nombreuses dégradations. Un des domaines principaux en traitement d'image consiste à traiter et corriger ces dégradations pour obtenir une image de meilleure qualité. On s'intéresse ici à deux types de dégradations fréquemment rencontrées dans les images :

— Le **bruit gaussien**, qui affecte tous les pixels de l'image. Dans ce TP, nous considérerons un bruit blanc additif Gaussien, de moyenne nulle et de variance σ^2. Il s'agit d'un modèle fréquemment utilisé en première approximation pour modéliser le bruit d'acquisition. Le bruit gaussien est caractérisé par sa variance σ^2 : plus σ^2 est élevé, plus l'image est dégradée.

— Le **bruit impulsionnel**, n'affecte que certains pixels de l'image. Le bruit sel et poivre est une dégradation de l'image sous la forme de pixels noirs et blancs répartis au hasard. Ce bruit est dû soit à des erreurs de transmission de données, soit à la défaillance d'éléments du capteur CCD, soit à la présence de particules fines sur le capteur d'images. On le caractérise par le pourcentage p de pixels modifiés : plus p est élevé, plus l'image est dégradée.

1.2 Travail demandé

1. Charger l'image `Lena.png` dans une matrice`imGray`.
2. Appliquer sur cette image un bruit blanc Gaussien de variance $\sigma^2 = 0.01$ et stocker le résultat dans une matrice `imGauss`. Vous pouvez utiliser la fonction random_noise de skimage. Faire varier σ^2 et commenter.
3. Appliquer sur l'image `imGray` un bruit sel et poivre avec un pourcentage $p = 0.05$ de pixels modifiés et stocker le résultat dans une matrice `imSP`. Vous pouvez utiliser la fonction random_noise de skimage. Faire varier p et commenter.
4. Afficher sur une même figure les images `imGray`, `imGauss` et `imSP`. Comparer les effets des deux dégradations et commenter.

2 Filtrage spatial d'une image

2.1 Rappels de cours

Le filtrage peut être vu comme une opération transformant une image en une autre image ayant des propriétés spatiales et fréquentielles différentes. On distingue deux types de filtrage :

— Le **filtrage linéaire** est une opération de convolution en 2D transformant une image en une autre en général de même taille. Il est défini par une matrice $h(m, n)$ de taille $M_h \times N_h$ appelée **masque de convolution** (en général $M_h = N_h$). Le filtrage linéaire revient à remplacer la valeur de chaque pixel par une moyenne pondérée calculée avec les pixels voisins. Le masque contient les coefficients de pondérations de chacun des pixels.

— Il existe également des **filtres non-linéaires** utilisés par exemple pour diminuer un bruit spécifique. Il s'agit encore une fois de remplacer la valeur de chaque pixel à partir des pixels voisins. En revanche, contrairement au filtrage linéaire, l'opération réalisée sur les pixels voisins est cette fois ci non-linéaire (par exemple une médiane).

2.2 Travail demandé

1. Reprendre les images `mGray`, `imGauss` et `imSP` précédemment définies. Pour l'image `imGauss` on prendra $\sigma^2 = 0.005$, et pour `imSP`, on prendra $p = 0.05$.
2. Tentez de restaurer l'image originale en appliquant les filtres adéquats. Au lieu de créer vous-même les noyaux, utilisez les fonctions prédéfinies d'OpenCV pour effectuer le filtrage adéquat (cv2.blur, cv2.GaussianBlur, cv2.medianBlur). Testez différentes tailles de noyau pour le filtrage moyen (3x3, 5x5, 7x7, etc.), différentes valeurs d'écart-type pour le filtre gaussien.
3. Evaluez les restaurations en calculant les deux mesures de qualité PSNR et MSE dans les différents cas. Commentez.

3 Méthodes à base de patch : moyenne non locale

3.1 Description

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 être semblables du point de vue statistique. De ce fait, le PSNR 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.

3.2 Algorithme

Algorithme : 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

3.3 Travail demandé

Appliquer le filtre NLMeans pour débruiter l'image originale dégradée par un bruit gaussien. Comparer d'un point de vue quantitatif et qualitatif l'effet de ce filtre avec le filtre moyenneur et le filtre Gaussien.