Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'hamed Bougara de Boumerdès
Faculté des Sciences, Département : Informatique

TP 2 ► Module : TAI	**Année :** S1-Master I2A Pro.	GACEB

Une image acquise, transmise, compressée, ou transformée peut avoir des dégradations (bruit, mauvais contraste, flou, skew, slant, faible résolution, etc.) qui altèrent sa qualité et impactent la performance de toute la chaine de traitement, d'analyse et de reconnaissance de son contenu. L'objectif de prétraitement d'image consiste à corriger (ou atténuer au maximum possible ces dégradations) pour avoir une image de meilleure qualité.
L'Objectif de ce TP *est de développer en python quelques techniques de base de prétraitement d'image.*

Partie 1 : Bruit, Convolution et Filtrage d'image

Exercice 1 : Bruit d'image

Dans une image numérique, on appelle bruit numérique toute fluctuation parasite ou dégradation que subit l'image de l'instant de son acquisition jusqu'à son enregistrement. Dans cet exercice, on s'intéresse à deux types très fréquents de bruits : Le bruit additif et le bruit impulsionnel.
Pour le bruit additif qui affecte tous les pixels de l'image. On s'intéresse à étudier le bruit blanc additif Gaussien, de moyenne nulle et de variance σ^2, plus σ est élevée, plus l'image est bruitée. Ce bruit, affecte les basses et les hautes fréquences dans une image. Il est très utilisé comme modélisation approximative de bruit d'acquisition et de lecture d'image.
Pour le bruit impulsionnel, seuls quelques pixels de l'image sont affectés. Nous allons étudier ici deux variétés de ce type de bruit : un bruit à effet de sel et à effet de poivre, qui se manifestent par l'apparition de pixels blancs et/ou noirs, dispersés aléatoirement. Ce bruit est causé soit par des erreurs dans la transmission d'image, soit par une défaillance de certains composants du capteur CCD, soit par la présence de particules microscopiques sur le capteur d'images. On décrit cela par le pourcentage p de pixels altérés : plus p est grand, plus l'image est dégradée.
Travail à faire :

1) Dans un nouveau fichier *«TP2_Exo1.py »*, saisir l'instruction qui permet de lire les niveaux de gris de l'image *«Images/cameraman.bmp»* dans un tableau 2D avec le nom *im1*.
2) En utilisant OpenCV, appliquer sur l'image *im1* un bruit blanc Gaussien d'une variance $\sigma^2=100$ et stocker le résultat dans une matrice *im2*. Afficher l'image originale et l'image bruitée. Enregistrer cette image dans le fichier *«Images/ cameraman_bruit_gauss_sig0_001.bmp»*. Faire varier σ^2, puis noter vos observations.
3) On souhaite appliquer sur l'image *im1* un bruit à effet de poivre et de sel avec un pourcentage *p* de *10%* de pixels modifiés et stocker le résultat dans une matrice *im3*. Pour cela, développer la fonction python *sp_noise(image, prob)* qui renvoi en sortie l'image bruitée résultante. Appliquer cette fonction avec les paramètres *im1 et p=0.1* et récupérer le résultat dans im3. Afficher l'image originale et l'image bruitée. Enregistrer cette image dans le fichier *« Images/cameraman_bruit_sel_poivre_p_10.bmp»*. Faire varier *p,* puis noter vos observations.
4) Afficher les images *im1, im2 et im3*. Comparer les effets des deux types de bruits. Noter vos observations.

Exercice 2 : Filtrage d'une image dans le domaine spatial (cartisien)

Le filtrage linéaire (ex. Filtre moyenne, Gaussien,…) est une opération de convolution 2D qui modifie une image en produisant généralement une autre image de taille similaire. On le caractérise par une matrice *M(m, n)* de dimension *hm×wm* qu'on appelle masque de convolution (En général, *hm=wm*, prend des valeurs impaires, M prend les dimensions suivantes : *3×3, 5×5, 7×7, 9×9, 11×11, etc.*).
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.
On trouve aussi des filtres non linéaires (Médian, Moyenne adaptative, bélatéral), utilisés par exemple pour réduire un bruit particulier. Cela consiste à remplacer la valeur de chaque pixel en se basant sur celle des pixels voisins. Cependant, l'opération appliquée sur les pixels voisins est ici non linéaire (par exemple, recherche de la valeur médiane ou une autre opération ad hoc).
Travail à réaliser :

1) Dans un nouveau fichier *«TP2_Exo2.py »*, saisir l'instruction qui permet de lire les niveaux de gris de l'image *«Images/cameraman.bmp»* dans la matrice *im1*, de l'image *«Images/cameraman_bruit_gauss_sig0_001.bmp»* dans la matrice *im2* et de l'image *« Images/cameraman_bruit_sel_poivre_p_10.bmp»* dans la matrice *im3*. Les deux dernières images sont issues du premier exercice.
2) Appliquer un filtre moyenneur de taille *3×3* (en utilisant une fonction opencv) sur l'image *im2* et stocker le résultat dans *imf2*. Appliquer ce filtre en utilisant votre propre fonction *« filtreMoyenne(image d'entrée, taille de masque)*. Afficher les figures *im1, im2 et imf2*. Le bruit a-t-il été atténué?
3) Appliquer un filtre médian de taille *3×3* sur l'image *im3 et im2* et stocker le résultat dans *imf3 et imfm2*. Appliquer ce filtre en utilisant une fonction OpenCV, puis en développant votre propre fonction *« filtreMedian(image d'entrée, taille de voisinage) »*. Afficher les figures *im1, im3, imf3, im2 et imfm2*. Le bruit a-t-il été atténué?
4) Pour pouvoir évaluer et quantifier la qualité du l'image bruitée ou filtrée par rapport à son image d'origine, on utilise la métrique PSNR (Peak Signal to Noise Ratio) ou Rapport Signal/Bruit de Crête. Il s'exprime en décibels (dB) et compare la puissance maximale possible d'un signal (l'image originale) avec la puissance du bruit de corruption (l'erreur introduite). Un PSNR élevé indique que l'image modifiée est très similaire à l'image originale et que la qualité de reconstruction est bonne. Un PSNR infini signifie que les deux images sont identiques. Un PSNR faible indique une différence significative entre les deux images, correspondant à une dégradation importante de la qualité. Le calcul du PSNR se fait en deux étapes : d'abord le calcul de l'erreur quadratique moyenne (MSE), puis l'application de la formule du PSNR. Les deux images comparées, Im (l'image originale) et imf (l'image modifiée), doivent avoir les mêmes dimensions.

Etape 1 : Calcul du MSE (Mean Squared Error) : Le MSE mesure l'erreur cumulée entre les deux images. Pour deux images Im et imf de taille HxW, la formule est la suivante :

$$MSE = \frac{1}{H \times W} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \left(\mathrm{Im}[i, j] - \mathrm{Im\,f}[i, j]\right)^2$$

Etape 2 : Une fois le MSE calculé, la formule du PSNR est :

$$PSNR = 10\log_{10}\left(\frac{R^2}{MSE}\right)$$

R est la dynamique du signal (valeur maximale possible pour un pixel, dans notre cas R=255).

- Calculer et comparer les PSNR suivants : *PSNR (im1, im2) et PSNR (im1, imf2), PSNR(im1, im3), PSNR(im1, imf3) et PSNR(im1, imfm2)* en utilisant la fonction PSNR d'OpenCV, puis votre propre fonction *calculate_psnr(image1, image2)* à développer. Analyser les résultats, les résultats vous semblent-ils logiques?

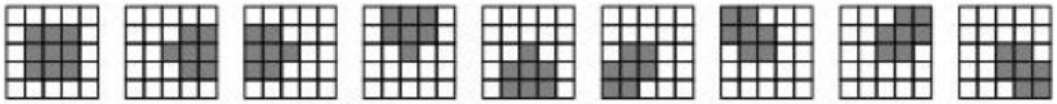
5) Tester les filtres suivants sur *im2*, puis sur *im3*. Lequel donne les meilleures performances sur *im2*? sur *im3*?
   a) Filtre moyenneur : *3×3, 5×5 et 7×7*
   b) Filtre Gaussien de taille *3×3, 5×5, sigma= 2, 1.5, 1 et 0.5*
   c) Filtre médian : *3×3, 5×5 et 7×7 (*réduit le bruit impulsionnel tout en préservant les contours d'image nets)
   d) Filtre bilatérale : *d=7, sigmaColor=40, sigmaSpace=40 (*le filtre bilatéral réduit le bruit additif tout en préservant les contours d'image nets).

Donner dans un tableau le PSNR de chaque filtre sur im2 et im3 par rapport à im1 et le temps d'exécution. Proposer des solutions pour réduire le temps d'exécution du filtre de plus grande taille en améliorant la qualité.
Devoir noté à faire:

a) Coder quelques solutions permettant d'accélérer un filtre moyenne de grande taille
b) Appliquer un filtre moyenne adaptative sur les images im2 et im3, comparer ses performances par rapport aux autres filtres, en utilisant la mesure PSNR.

Exercice 3 : Filtre de Nagao

Le filtre de Nagao est un filtre non linéaire de lissage préservant les contours de l'image, qui n'est pas inclus dans les fonctions de filtrage de base d'OpenCV. Afin d'utiliser ce filtre, il faut l'implémenter soi-même ou utiliser une bibliothèque tierce qui le propose.
Ce filtre fonctionne en analysant une fenêtre locale autour de chaque pixel (souvent une fenêtre 5x5, voir la figure ci-dessous) pour choisir la sous-région (qui est souvent en forme d'angle ou de ligne) qui présente la plus petite variance. La valeur du pixel au centre est donc calculée par la moyenne de cette sous-région choisie.



Travail à faire : dans un nouveau fichier *«TP2_Exo3.py »),* reprendre le code de la première question de l'exercice précédent. Développer votre propre fonction filtreNagao (im, taille de filtre). Appliquer ce filtre sur les images bruitées im2 et im3 pour différentes tailles de fenêtre (*5×5, 7×7, 9×9),* puis calculer et comparer les PSNR entre im1 et les images filtrées. Analyser les résultats.

Exercice 4 : Morphologie mathématique

Dans un nouveau fichier *«TP2_Exo4.py »,* charger l'image *« Globules_rouges.jpg»* dans *im1* en niveaux de gris, binariser cette image avec un bon seuil et stocker le résultat dans imb1.

1) Appliquer les différentes opérations morphologiques (érosion, dilatation, ouverture et fermeture morphologique avec un élément structurant 3×3, 5×5, 7×7) en utilisant des fonctions OpenCV, puis vos propres fonctions que vous devez développer. Analyser les résultats.
2) Mettre en œuvre des techniques simples en combinant ces opérations morphologiques pour séparer et faciliter le comptage des globules rouges dans im1.
3) L'implémentation de l'algorithme de squelettisation de Zhang-Suen (ou amincissement, thinning en anglais) est proposée dans son module ximgproc, qui fait partie des modules "contrib" (contributions supplémentaires). Pour utiliser cette fonction, vous devez avoir installé le package opencv-contrib-python. En utilisant cette fonction, appliquer une squelettisation morphologique à l'image imb1.