TP IMAGES

Dans ce TP, les traitements ne s'appliquent qu'à des images en niveaux de gris.

Voilà deux images en niveau de gris qui vont nous servir de base pour les traitements :


Vous les récupérerez sur le site **angeliquerenaud.com** ainsi que le code de départ à compléter.
Il contient le code de la fonction *OuvrirImagegris(path)* qui prend en entrée un chemin d'accès et renvoie le tableau numpy contenant les valeurs de niveau de gris des pixels.
Il contient également le code de la procédure *CreerImgGris(matrix,path)* qui prend en entrée un tableau de données et un chemin d'accès et crée le fichier image correspondant aux données à l'endroit indiqué.

Par souci de clarté, on utilise les valeurs numériques de la taille des images fournies (640 par 480 pour la fleur ou 450 par 600 pour le portrait). Le code sera à adapter à l'image utilisée en modifiant les valeurs de ces paramètres. Si cela ne vous pose pas de difficultés, vous pouvez utiliser des variables.

Attention, la taille d'une matrice/tableau numpy est donnée sous la forme nombre de lignes × nombre de colonnes, c'est à dire hauteur × largeur. Or les dimensions des images sont généralement données sous la forme largeur × hauteur. Il faut donc être vigilant.

petit rappel sur les tableaux numpy :

```
1  import numpy  #importation de la bibliothèque
2
3  #matrices en liste de liste
4  T=[[0,1],[2,3]]
5  U=[[4,5],[6,7]]
6  print (T+U)  #concatenation
7  print (2*T)  #concatenation
8
9  #tableaux numpy
10 T=numpy.array(T)
11 U=numpy.array(U)
12 print (T+U)  #addition terme a terme
13 print (2*T)  #multiplication de tous les elements par 2
14
15 T1=T #deux noms pour le même objet
16 T2=numpy.copy(T) #on peut modifier T sans changer les valeurs de T2
```

```
In [1]: (executing lines 1 to 6 of "rappelnumpy.py")
[[0, 1], [2, 3], [4, 5], [6, 7]]
[[0, 1], [2, 3], [0, 1], [2, 3]]


In [2]: (executing lines 9 to 16 of "rappelnumpy.py")
[[ 4  6]
 [ 8 10]]
[[0 2]
 [4 6]]
```

1 Un exemple de traitement pixel par pixel : modifier les contrastes

On souhaite pouvoir modifier le contraste d'une image en utilisant la fonction de réajustement linéaire des valeurs des pixels suivante :

$$f(x) = \begin{cases} y = x + 0.4(x - 127) & \text{si } y \in [\![0; 255]\!] \\ 0 & \text{si } y < 0 \\ 255 & \text{si } y > 255 \end{cases} \quad (1)$$

Question 1 : Ecrire une fonction qui prend en entrée un tableau de données et renvoie un autre tableau de données dans lequel le contraste a été modifié selon la fonction fournie. Quel est son effet ?

Question 2 : Ecrire une fonction qui prend en entrée un tableau de données et renvoie un autre tableau de données correspondant à l'image symétrique verticalement.

2 Un exemple de traitement local : les filtres

Filtrer une image consiste à lui appliquer une transformation mathématique modifiant la valeur des pixels. Un filtre F est défini par une matrice carrée d'ordre n impair, qui se déplace sur l'image en modifiant la valeur du pixel central selon les valeurs de ses voisins.

prenons l'exemple d'un filtre F de taille 3. $F = \begin{pmatrix} f_{0,0} & f_{0,1} & f_{0,2} \\ f_{1,0} & f_{1,1} & f_{1,2} \\ f_{2,0} & f_{2,1} & f_{2,2} \end{pmatrix}$
Appliquons le à la matrice A d'une image.

On superpose le filtre à la matrice carrée extraite de A centrée sur la valeur $a_{i,j}$.
On effectue les produits terme à terme.
On « remplace » $a_{i,j}$ par la somme de ces produits terme à terme.

$a_{i,j} \to b_{i,j} = f_{0,0} \times a_{i-1,j-1} + f_{0,1} \times a_{i-1,j} + f_{0,2} \times a_{i-1,j+1} + f_{1,0} \times a_{i,j-1} + f_{1,1} \times a_{i,j} + f_{1,2} \times a_{i,j+1} + f_{2,0} \times a_{i+1,j-1} + f_{2,1} \times a_{i+1,j} + f_{2,2} \times a_{i+1,j+1}$

On parle de convolution discrète (opérateur *).

remarque : Les bords de l'image est toujours exclu du traitement local !

Question 3 : Programmez une fonction filtrage(A,F) prenant en argument deux tableaux numpy (A celui de l'image et F celui du filtre) et renvoyant un tableau numpy B, de même taille que A, contenant les valeurs de A filtrées par F.

2.1 application à la restauration d'image

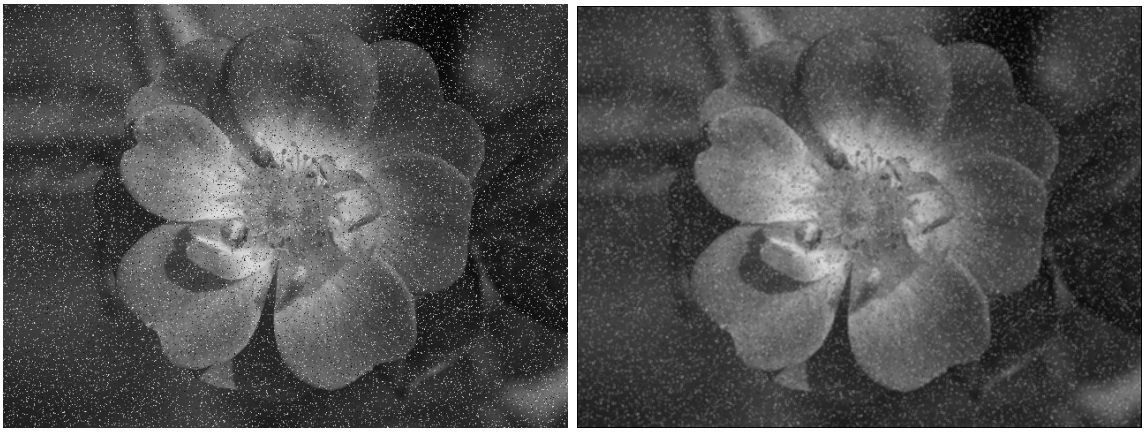
Parfois certains défauts (poussières, fluctuation de l'intensité électrique, erreurs de transmissions...) conduisent à avoir une image « abimée ». Les valeurs de certains pixels sont « erronée »., d'où des taches de petites tailles à certains endroits aléatoires de l'image. On parle alors de bruit.

Le filtre de moyenne d'ordre 3 remplace chaque pixel par la moyenne de ses 9 voisins (lui compris).

La matrice associée est donc : $F = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$

Question 4 : Appliquez la fonction filtrage(A,F) précédente afin de restaurer l'image abimée proposée.

exemple :



A droite, l'image obtenue après restauration par le filtre de la moyenne. On constate que le bruit est bien partiellement réduit mais que le filtrage provoque un certain flou.

Il existe d'autres filtres plus efficaces, comme le filtre médian (qui consiste à prendre la médiane au lieu de la moyenne) mais qui ne correspond pas à une convolution.

2.2 Détection des contours

La notion de contour est intimement liée à celle des variations de valeurs entre un pixel et ses voisins. Sans une partie homogène (par exemple, le ciel), les valeurs des pixels sont proches. Par contre, un contour provoque de forte variation donc une augmentation locale du gradient.

Si l'on traitait de fonction continue $f : \mathbb{R}^2 \to \mathbb{R}^2$ avec $f(x, y)$ la valeur du pixel, le gradient de f au point (x, y) serait donné par $\vec{grad}f = \frac{\partial f}{\partial x}\overrightarrow{u_x} + \frac{\partial f}{\partial y}\overrightarrow{u_y}$.

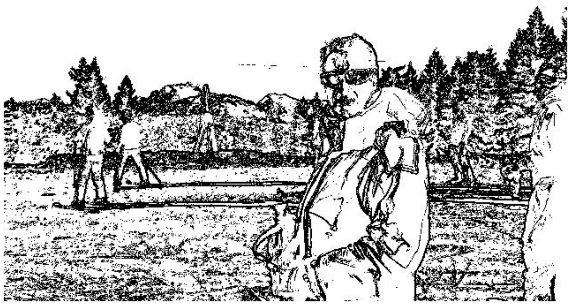
Mais comme nous sommes en discret, un première approche du gradient peut être d'assimiler :
$\frac{\partial f}{\partial x}$ à ..... $\frac{\partial f}{\partial y}$ à .....

donc on utilisera les filtres :

$G_x =$ $G_y =$

Une première approximation de la norme du gradient peut donc être $G = \sqrt{(G_x * A)^2 + (G_y * A)^2}$.
On ne retient que les pixels donc la norme du gradient est supérieure à un certain seuil comme contour potentiel.

Question 5 : Ecrire un programme permettant d'obtenir l'image ci-dessous (On a retenu un seuil de 15 : Si le gradient en (i, j) est superieur à 14, le pixel (i, j) est noir, sinon, il est blanc).



3 Pour aller plus loin... jouer avec la couleur

Pour encoder en bit map une image couleur, on utlise non plus un octet par pixel mais trois : un pour le rouge, un pour le vert, un pour le bleu.
Vous trouverez les versions couleur des images du tp sur le site, ainsi que le code qui vous permet de recuperer le contenu d'une image couleur et celui de la fonction permettant d'enregistrer une image couleur.
Dans un premier temps ouvrez une image et observez la structure des données.
Votre objectif est de jouer à Andy Warhol, créateur du pop Art en créant une l'oeuvre d'art suivante :

Vous vous aiderez de trois fonctions composanterouge(), composanteverte(), composantebleu() à écrire, ainsi que de la methode numpy.concatenate.