Chapitre 6 :
Traitement d'images 2

I Stéganographie

La stéganographie est un moyen de dissimuler un message dans un autre message. Ici, on va dissimuler une image dans une autre image.

1.1 Utilisation de l'écriture binaire

Une image est représentée par un tableau, chaque élément du tableau correspond à un pixel, et chaque pixel est représenté par un triplet de nombres entre 0 et 255, ces nombres représentants les composantes rouge, verte et bleu.

Un nombre entre 0 et 255 est représenté en binaire par un nombre stocké sur 8 bits (car $2^8 = 256$). Par exemple, en écriture binaire 10110011 représente le nombre dont l'écriture décimale est $1\times2^7+0\times2^6+1\times2^5+1\times2^4+0\times2^3+0\times2^2+1\times2^1+1\times2^0 = 179$.

Le principe de la dégradation par perte de bits d'information consiste à éliminer les bits de poids faible. Par exemple :

- si on élimine 1 bit : en écriture binaire 10110011 devient 10110010, ainsi, en écriture décimale 179 devient 178,
- si on élimine 2 bits : en écriture binaire 10110011 devient 10110000, ainsi, en écriture décimale 179 devient 176,
- ...

1.2 Application aux images

En éliminant des bits, on perd des informations mais le résultat sur les images n'est pas toujours visible. On écrit une fonction qui permet de dégrader une image `im` en enlevant `n` bits.

```
import numpy as np
from PIL import Image

def image_degradee(im,n) :
    t=np.array(im)
    h,l,r=t.shape
    for i in range(h) :
        for j in range(l) :
            for k in range(3) :
                t[i][j][k]=2**n*(t[i][j][k]//2**n)
    return Image.fromarray(t)
```

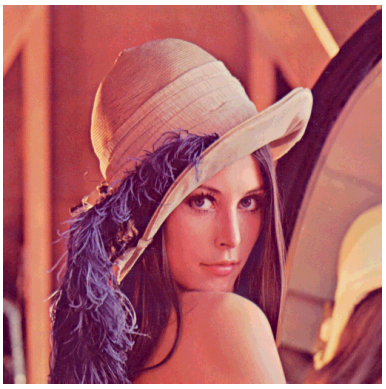
On teste cette fonction avec l'image de Lena.

image_degradee(lena,0)

image_degradee(lena,1)

image_degradee(lena,2)

image_degradee(lena,3)

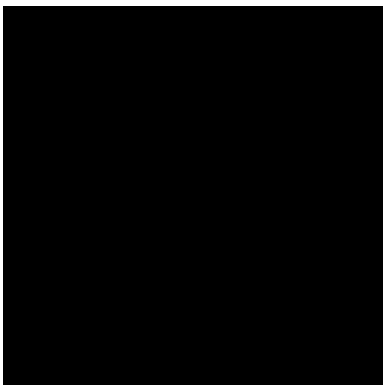


image_degradee(lena,4)

image_degradee(lena,5)

image_degradee(lena,6)

image_degradee(lena,7)



image_degradee(lena,8)

On remarque que jusqu'à une perte de 4 bits, le résultat est très proche de l'image originale.

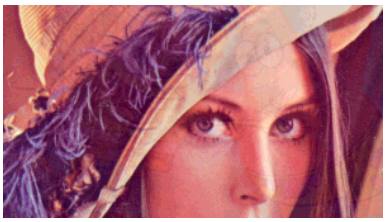
1.3 Image cachée

On vient de remarquer qu'on peut libérer 4 bits de poids faible dans une image sans l'affecter de façon visible. On va utiliser ces 4 bits libres afin de cacher les 4 bits de poids fort d'une autre image.

Par exemple, on veut cacher le nombre 11010110 dans le nombre 10101100, on utile le nombre 10101101.

En TD, on utilisera les images suivantes :

mystere1.png



mystere2.png

II Modification d'une image par convolution

Le traitement d'image par convolution est une méthode générale qui permet de faire différentes modifications sur les images.

2.1 Méthode générale

On considère une image matricielle dont les composantes de chaque pixel sont représentées par des matrices $(p_{i,j})$ (on ne différenciera pas dans les notations les composantes rouge, verte et bleue).

On considère également une matrice de taille 3×3 appelée noyau de convolution :

$$\begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix}$$

Le principe de la convolution est de remplacer chaque valeur de $p_{i,j}$ par :

$$a_1 p_{i-1,j-1} + a_2 p_{i-1,j} + a_3 p_{i-1,j+1} + a_4 p_{i,j-1} + a_5 p_{i,j} + a_6 p_{i,j+1} + a_7 p_{i+1,j-1} + a_8 p_{i+1,j} + a_9 p_{i+1,j+1}.$$

On remarque que cette formule de n'applique pas aux bords de l'image, on choisira de les laisser inchangés.
Le résultat obtenu par cette formule n'est pas nécessairement un entier entre 0 et 255. On devra donc tronquer la formule de façon à s'y ramener.

2.2 Applications

2.2.1 Lissage

Afin de lisser une image, on effectue la moyenne de tous les pixels voisins. Le noyau de convolution est donc :

$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$


avec filtre moyenneur

On peut aussi choisir de faire une moyenne pondérée afin de prendre en compte davantage le pixel central et moins les pixels diagonaux. On utilise le noyau gaussien :

$$\frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}.$$


avec filtre Gaussien

2.2.2 Renforcement des contours

Un contour est un endroit de l'image où les pixels varient beaucoup.
Pour renforcer les contours, on peut donc utiliser un filtre réhausseur :

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$


avec filtre réhausseur

2.2.3 Détection de contours

Pour détecter les contours, on peut donc utiliser un noyau laplacien :

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$


avec filtre Laplacien

avec filtre Laplacien et négatif