INFORMATIQUE PSI P2B

Année 2024/25

TP n° 6

Traitement d'images

Table des matières

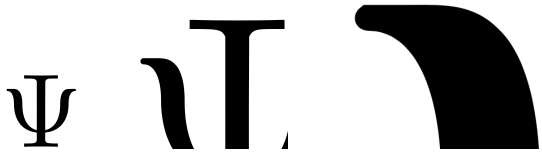
Vincent Rohart

http://vrohart.e-monsite.com/

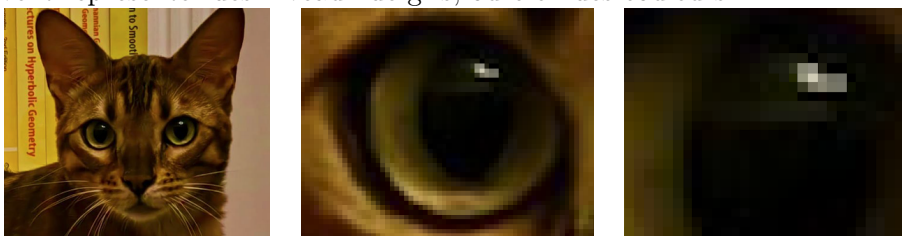
1 Représentation d'une image matricelle

Les images en informatique peuvent être de deux types :

— *vectorielles* : les données sont définies par des courbes paramétrées. Leur qualité n'est pas altérée par un agrandissement.

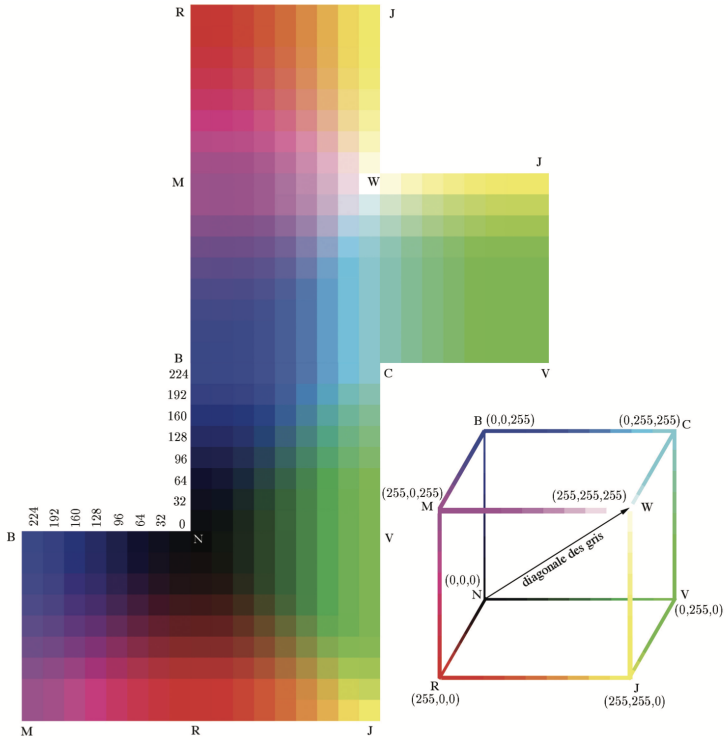


— *matricielles* (ou *bitmap*) : elles sont constituées d'un ensemble de *pixels* (de *picture elements*). Ceux-ci peuvent représenter des niveaux de gris, ou bien des couleurs.



Le système RGB (*red-green-blue*) est utilisé pour les formats JPEG, TIFF [1], etc. Un pixel est représenté par une 3-liste `[r,g,b]` constituée de trois entiers codés sur 8 bits, soit des entiers compris entre 0 et $2^8 - 1 = 255$.

— le pixel `[0,0,0]` (absence de couleur) code le noir.
— le pixel `[255,255,255]` code le blanc.
— le pixel `[255,0,0]` code le rouge.
— le pixel `[0,255,0]` code le vert.
— le pixel `[0,0255]` code le bleu.
— le pixel `[n,n,n]` est en gris (plus ou moins foncé) quel que soit $n \in [\![0, 255]\!]$.



1. Joint Photographic Experts Group ; Tagged Image File Format. Les formats PNG (Portable Network Graphics) sont de nature différente.

La Commission Internationale de l'Éclairage propose une nuance de gris, dit naturel, donnée par la formule suivante :

$$n = [0,2125\mathrm{R} + 0,7154\mathrm{V} + 0,0721\mathrm{B}]$$

où $[x]$ est l'entier le plus proche de x (`round` en `Python`). Cette nuance semble plus naturelle à l'œil humain qui a une plus grande sensibilité dans le vert. Mais on peut choisir $n = [0,3333\mathrm{R} + 0,3333\mathrm{V} + 0,3333\mathrm{B}]$ comme autre nuance de gris.

2 Modules à importer et fonctions utiles

Le fichier sur lequel porteront les question de ce TP est `simbad.jpeg` : il est à télécharger sur la page web où se trouve l'énoncé du TP. Vous mettrez ce fichier dans le même répertoire que votre fichier `.py` sur lequel vous compilez vos scripts `Python`.

On importera les trois modules suivants :
- `matplotlib.pyplot as plt`
- `matplotlib.image as mpimg`
- `numpy as np`

Fonctions utiles
- `chat = mpimg.imread("simbad.jpg")` charge le fichier dans un tableau (que l'on a choisi de nommer `chat`) de taille $p_\mathrm{L} \times p_\mathrm{C} \times 3$ pour une image RGB (de taille $p_\mathrm{L} \times p_\mathrm{C}$ en niveau de gris). Le nombre p_L est le nombre de lignes de l'image : c'est donc le nombre de pixel en hauteur. De même, p_C le nombre de pixels en largeur.
- `chat.shape` renvoie le tuple `(pL, pC, 3)`.
- `plt.imshow(chat)` affiche l'image dont `chat` est la représentation matricielle.

```
>>> chat = mpimg.imread("simbad.jpeg")

>>> chat.shape
(1671, 1016, 3)

>>> chat[0,0]
array([193, 148,  89], dtype=uint8)
# C'est le pixel en haut à gauche de l'image.
```

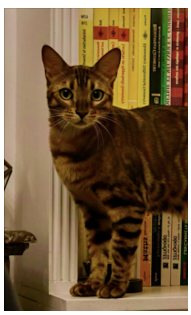
3 Questions

1. Ouvrir le fichier `galette.png` et expliquer quel est le type de ce fichier image. En particulier, donner les différences avec un fichier au format JPEG.

   Dans les questions qui suivent, le fichier à traiter est `simbad.jpeg`.

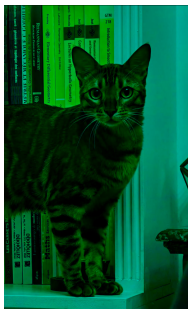
2. Afficher le pixel en haut à gauche. Et celui qui est en haut à droite ?

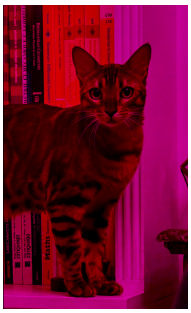
3. Afficher, après avoir écrit un programme `Python`, l'image renversée, comme celle que l'on pourrait observer dans un miroir.

4. Créer une fonction `decomp` qui prend une image et renvoie les trois listes L_R, L_G, L_B des entiers codant les couleurs rouge, verte et bleue.

5. Afficher l'image sans sa composante rouge (mettre 0 en lieu et place des entiers codant les nuances de rouge). Faire de même en enlevant la composante verte, et enfin la composante bleue.


Sans rouge

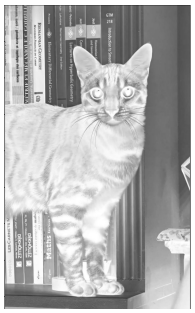

Sans vert


Sans bleu

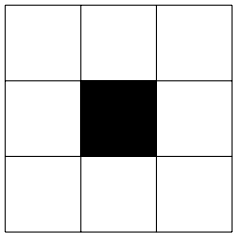
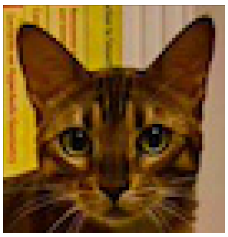
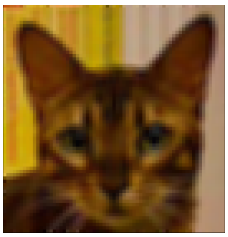

En gris

6. Rédiger un programme transformant une image en couleur en niveau de gris. On appliquera ce résultat à l'image mise en ligne. Comparer avec un niveau de gris $\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$.
7. Créer une fonction `reduction(im, p)` qui réduit la taille d'une image `im` en ne sélectionnant qu'une ligne sur p et qu'une colonne sur p. Afficher le résultat pour $p = 10$.

8. Programmer une fonction `inverser` s'appliquant à une image en nuances de gris qui inverse les niveau de gris (le noir devient blanc et réciproquement). Afficher le résultat.




9. Rapprochement des extrêmes : remplacer chaque pixel `[r,g,b]` par `[r',g',b']` où $x' = |x-127|$. Interpréter graphiquement et afficher le résultat.
10. Le filtrage est une technique pour lisser l'apparence d'une image dégradée (par exemple trop pixélisée). Il en existe de nombreux types. Un filtrage simple (et rudimentaire) consiste à remplacer chaque pixel $[r, g, b]$ par sa moyenne (arrondi à l'entier près) des pixels qui lui sont voisins, lui compris (il y a donc 9 pixels concernés en général, sauf pour les pixels de bord).

Mise en garde. Ne pas faire $(R_{1,1} + R_{2,1} + \ldots + R_{3,3})/9$ mais $R_{1,1}/9 + R_{2,1}/9 + \ldots + R_{3,3}/9$. Pourquoi à votre avis ?

4 Stéganographie

Réaliser une *stéganographie* (du grec στεγανός : étanche) par la méthode LSB (*least significant bit*[2]) consiste à considérer chaque pixel (R, G, B) (avec R, G, B des entiers entre 0 et 255 écrits en base 2) et à remplacer le dernier bit (« l'unité », c'est-à-dire le bit le plus à droite) par un bit de son choix. Cette modification entraîne un changement de teinte du pixel qui est invisible pour l'œil humain.

Par exemple, considérons une image (rudimentaire) composée de 2×2 pixels :

11100011 00000000 11101100	11111110 00001111 10101010
11111111 00000000 11101110	00000001 10000001 10000000

Si l'on veut cacher le message **111000111000**, on modifiera l'image précédente en

1110001**1** 0000000**1** 1110110**1**	1111111**0** 0000111**0** 1010101**0**
1111111**1** 0000000**1** 1110111**1**	0000000**0** 1000000**0** 1000000**0**

On peut bien sûr imaginer inclure de cette manière une image cachée à l'intérieur d'une image donnée plus grande.

1. Choisir une image de votre choix et la cacher par stéganographie LSB dans l'image de Simbad de ce TP.
2. Construire une fonction `Python` permettant de décalquer l'image ainsi cachée et de la mettre au clair.

FIGURE 1 – Un chat caché par stéganographie dans une image

Ce procédé de stéganographie très basique peut être *stérilisé* par compression puis décompression des images avec perte partielle d'information (typiquement une réduction de format ou de qualité).

Dans la pratique, tous les pixels ne se valent pas pour être remplacés : le bit le moins significatif est un essai naïf.

2. Le bit le moins significatif