L'École Normale Supérieure de l'Enseignement Technique de Mohammedia
Filières Ingénieurs-
COMPÉTENCES NUMÉRIQUES ET INFORMATIQUE
— TD N°9: —
Traitement des images

Problème I:

Contexte du problème :

Actuellement, les images ont envahi notre vie, que ce soit dans les loisirs (cadres photos numériques, jeux vidéo avec une caméra détectant les mouvements...) ou dans un cadre professionnel. Citons par exemple l'utilisation croissante de la vision dans des domaines variés comme la médecine (IRM), la vidéo surveillance (sécurité, analyse routière...), la robotique, la défense (missiles, détection, véhicules autonomes...), l'astronomie... .
De ce fait, le traitement d'images est introduit comme une discipline informatique qui étudie les images numériques. En effet, cette nouvelle discipline représente un ensemble de méthodes et de techniques opérant sur les images, dans le but d'améliorer leur aspect visuel et d'en extraire des informations jugées pertinentes.
Parmi les aspects du traitement d'image nous citions :

- Filtrage / déconvolution (ou filtrage inverse)
- Compression
- Segmentation
- Restauration / reconnaissance
- etc.

Nous nous focalisons dans ce problème sur les différents aspects de la représentation des images sur machine.

Le codage des images

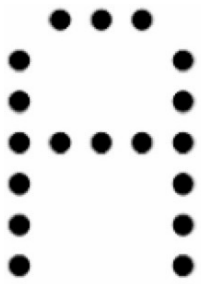
Une manière de représenter une image sur machine, est de l'échantillonner en petits éléments appelés pixels (picture element= pixel) supposés de couleur uniforme, et cela suivant un quadrillage. L'image est alors représentée par une **matrice de couleurs** appelée bitmap, chaque case de la matrice donnant la **couleur du pixel** correspondant du quadrillage. Il existe trois façons pour représenter un pixel dans une matrice :

Noir et blanc: dans ce cas chaque pixel est représenté sur un bit, 1 pour noir, 0 pour blanc. C'est de ce mode que vient le nom "bitmap".

Niveau de gris: chaque pixel est représenté sur un octet (256 valeurs) dont la valeur varie du plus obscur 0 (noir) au plus clair 255 (blanc) en passant par 254 nuances de gris.

Couleur RGB: chaque pixel est représenté par 3 octets, chacun donnant l'amplitude des 3 couleurs fondamentales additives : R (rouge), G (vert), B (bleu). Le noir est alors (0; 0; 0) et le blanc (255; 255; 255).

Exemple:



0 0 0 1 1 1 0 0
0 0 1 0 0 0 1 0
0 0 1 0 0 0 1 0
0 0 1 1 1 1 1 0
0 0 1 0 0 0 1 0
0 0 1 0 0 0 1 0
0 0 1 0 0 0 1 0

On convient de numéroter les colonnes en partant de la gauche, et les lignes en partant du haut de l'image. Un pixel de l'image est repéré par un couple (i, j) où i et j sont respectivement les numéros de la ligne et de la colonne à l'intersection desquelles se trouve le pixel.
Dans ce problème, nous traitons quelques algorithmes relatifs au traitement d'images numériques matricielles, en noir et blanc, en niveau de gris et en RGB.

Préliminaire : Les opérations d'entrée/sortie sur les images

1. La fonction **AfficherImg(Img)** ci-dessous prend en argument une image ensuite elle l'affiche sur l'écran

```
def AfficherImg(img):
        plt.axis("off")
        plt.imshow(img, interpolation="nearest")
        #plt.imshow(img, cmap = "gray")#palette predefinie pour afficher une image plt.show()
```

2. La fonction **ouvrirImage(Img)** ci-dessous prend en argument une image Img, ensuite retourne une matrice contenant le codage de cette matrice

```
def ouvrirImage(chemin):
    img=plt.imread(chemin)
    return img
```

3. La fonction **saveImage(Img)** ci-dessous prend en argument une image Img, ensuite elle enregistre cette image dans le disque dur.

```
def saveImage(img):
    plt.imsave("image1.png",img)
```

Partie I. Les images Noir et blanc

Dans cette partie nous développons quelques fonctions pour manipuler les images en mode noir et blanc.

Question 1. Créer une image noir

Ecrire une fonction d'entête **def image_noire(h, l):** qui permet de générer une image en noir de hauteur et de largeur l **.** Pour ce faire, la fonction retourne une matrice contenant **h** lignes et **l** colonnes dont la valeur de chaque pixel est initialisée par 0

Question 2. Créer une image blanche

Ecrire une fonction d'entête **def image_blanche(h, l):** qui permet de générer une image en noir de hauteur et de largeur l **.** Pour ce faire, la fonction retourne une matrice contenant **h** lignes et **l** colonnes dont la valeur de chaque pixel est initialisé par 1

Question 3. Créer une image noir et blanc

Ecrire une fonction **def creerImgBlancNoir(h,l):**qui retourne une image noir et blanc dont le contenu de chaque pixel (i,j) est donné par (i+j)%2

Question 4. Négatif

Ecrire une fonction d'entête **def negatif(Img)** qui construit le négatif de l'image représentée par une matrice Img. Pour cela, il suffit d'inverser les valeurs de la matrice t (0 devient 1 et 1 devient 0)
Question

Partie II. *Les images en niveau de gris*

Dans cette partie nous développons quelques fonctions pour manipuler les images en niveau de gris.
Question 5. La luminance (ou brillance) d'une image est définie comme la moyenne de tous les pixels de l'image. Écrire une fonction d'entête **def luminance(Img)** qui retourne la luminance d'une image Img.
Question 6. Le contraste peut être défini comme la variance des niveaux de gris (N nombre de pixels dans l'image). Ecrire une fonction d'entête **def constrast(Img)** qui retourne le contraste d'une image t.

$$\frac{1}{N}\sum_{i=0}^{l-1}\sum_{j=0}^{c-1}(I(i,j)-Moy)^2$$

Question 7. Ecrire une fonction d'entête **def profondeur(Img)** qui retourne la valeur maximale d'un pixel dans l'image Img

Question 8. **Écrire** une fonction d'entête **def Ouvrir(Img)** qui retourne la matrice représentant l'image A.

Image A

Partie III. *Opérations élémentaires sur les images en mode gris*

Image B

Image C

Image D

Image E

FIGURE 1 – Opérations élémentaires.

Question 9. Écrire une fonction **inverser(img)** qui renvoie l'image inverse de l'image img, c'est- à-dire que le ton d'un pixel de la nouvelle image est p − v où v est le ton du pixel correspondant de l'image d'origine. Par exemple, l'image B de la figure 1 résulte de l'application de inverser à l'image A de l'introduction.

Question 10. Écrire une fonction **flipH(img)** qui renvoie la transformée de l'image img par la symétrie d'axe vertical passant par le milieu de l'image. Par exemple, l'image C de la figure 1 résulte de l'application de flipH à l'image A de

Question 11. Écrire une fonction **poserV(img1,img2)** qui prend en arguments deux images img1 et img2 de même largeur et profondeur, et qui renvoie la nouvelle image obtenue en posant img1 sur img2. Par exemple, l'image D de la figure 1 résulte de l'application de poserV aux images B et C.

Question 12. Écrire une fonction **poserH(img1, img2)** qui prend en arguments deux images img1 et img2 de même hauteur et profondeur, et qui renvoie la nouvelle image obtenue en posant img2 à droite de img1. Par exemple, l'image E de la figure 1 résulte de l'application de poserH aux images B et C.

Partie IV. Transferts

Certaines transformations des images sont simplement l'application d'une fonction aux tons, dont on rappelle qu'ils sont des entiers compris entre 0 et p (profondeur de l'image) au sens large. Une telle fonction de transfert peut s'appliquer vers des images dont la profondeur n'est pas nécessairement p, mais une nouvelle profondeur q. Une fonction de transfert est représentée par la donnée de la profondeur cible q et d'un tableau d'entiers t de taille p + 1, dont les cases contiennent des entiers entre 0 et q au sens large.

Question 13. Écrire une fonction **transferer(img, q, t)** qui prend en arguments une image img, ainsi qu'une fonction de transfert donnée par un entier q et un tableau d'entiers t. La fonction transferer renvoie une nouvelle image, de même taille que img, de nouvelle profondeur q et dont chaque pixel (i, j) a pour ton t[img[i][j]].

Question 14. Écrire une nouvelle fonction **inverser** (Question 1) qui utilise la fonction transferer de la question précédente.

FIGURE 2 – Transferts.

L'histogramme d'une image de profondeur p est un tableau hi d'entiers de taille p + 1 tel que h_i [v] compte le nombre de pixels de l'image dont le ton est v.

Question 15. Écrire une fonction **histogramme(img)** qui prend en argument une image img et retourne l'histogramme de celle-ci.

Soit img une image de hauteur h, de largeur l et de profondeur p. Soit h_i son histogramme et soit v_{min} le ton de gris le plus sombre se trouvant dans l'image img. On égalise les tons en transformant chaque ton v en v' défini ainsi :

$$v' = p \times \frac{\left(\sum_{k=0}^{v} h_i[k]\right) - h_i[v_{\min}]}{h \times l - h_i[v_{\min}]} \qquad \text{pour } v_{\min} \leqslant v \leqslant p.$$

On note que v' n'est pas défini pour v tel que 0 <=v <= v_{min} , ce qui n'a pas d'importance, ces tons v étant absents de l'image. On note surtout que la valeur de v' ci-dessus n'est généralement pas un entier. On rappelle à cette occasion que la commande **int(round(x, 0))** renvoie l'entier le plus proche de x.

Question 16. Écrire la fonction **egaliser(img)** qui prend une image img en argument et qui renvoie une nouvelle image qui est img dont les tons de gris sont égalisés. Par exemple, l'image F de la figure 2 résulte de l'application de egaliser à l'image A.

Question 17. Que renvoie la fonction **egaliser** appliquée à une image uniformément blanche ?

On cherche maintenant à réduire la profondeur d'une image de p vers q avec q <=p. Une technique consiste à remplacer un ton v' tel que v'/q est le plus proche possible de v/p.

Question 18. Écrire une fonction **reduire(img, q)** qui renvoie une nouvelle image dont la profondeur est réduite à q. Par exemple, les images G, H et I de la figure 2 résultent des réductions à 1, 4 et 16 de la profondeur de l'image A.

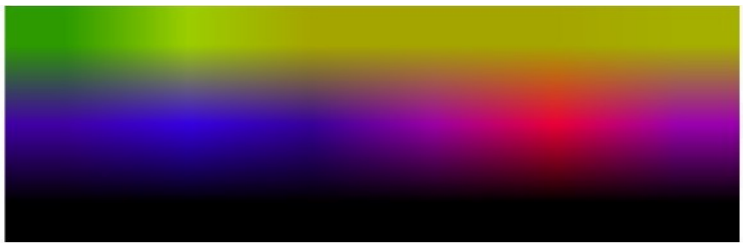
Partie V. Les images RGB

Les images RGB en couleur constituée de trois couches (RGB). Les données de l'image sont stockées dans un tableau à trois dimensions : la première dimension correspond à la couleur (rouge, vert ou bleu, indice 0, 1 ou 2). Ainsi, les dimensions du tableau sont : 3 × n × m où n le nombre des lignes et m le nombre des colonnes. La valeur associée à chaque pixel est un entier compris entre 0 et 255.

Exemple

Pour n=3 et m=6

```
>>>M=[[[210, 100, 255],[100, 50, 255],[90, 90, 255],[90, 90, 255],[90, 90, 255],[90, 80, 255]],
[[190, 255,89],[ 201, 255,29],[200, 255,100],[100, 255,90],[20, 255,200], [100, 255,80]],
[[255,0, 0],[ 255,0, 0],[255,0, 0],[255,0, 0],[255,0, 0], [255,0, 0]] ]]
>>>import matplotlib.pyplot as plt
>>>plt.imshow(M)
>>>plt.axis ("off")
>>>plt.show ()
```



Question 19. Donnez les valeurs des expressions suivantes

```
>>>print(M[0][1][1])
    ……..
>>>print(M[1][0][1])
    ……..
>>>print(M[2][1][0])
    ……..
```

Question 20. Donner la quantité de mémoire nécessaire en octets(8 bits) pour stocker le tableau représentant une image RGB, justifier votre réponse?

Question 21. Ecrire une fonction **def initImageRGB(imageRGB)** permettant d'initialiser et de renvoyer le tableau imageRGB à trois dimensions , l'initialisation se fera d'une manière aléatoire suivant la fonction ***randrange(p)***

Question 22. Ecrire une fonction **def symetrie(img)** qui retourne une image *symétrique* à l'image img par rapport à l'axe horizontal (même question pour l'axe vertical)

L'algorithme de conversion d'une image RGB en une image de niveau de gris se présente comme ceci:

1. Chercher le maximum et le minimum pour chaque pixel sur les trois couches
2. Calculer la partie entière de la moyenne de ces valeurs maximales et minimales
3. Obtenir la valeur du nouveau pixel en niveaux de gris.

Question 23. Ecrire une fonction **def grayscale(imageRGB)** qui renvoie une image en niveaux de gris, sous forme de tableau à deux dimensions de taille n × m contenant des valeurs entières comprises entre 0 et 255, en suivant l'algorithme décrit ci-dessus.

Partie VI. Stéganographie

L'objet de la stéganographie est de dissimuler un message dans un contenant a priori anodin. On va étudier ici comment dissimuler un message dans une image. Le message en question sera une suite de 0 et de 1. Pour obtenir une telle suite à partir d'un texte, on peut utiliser un tableau de correspondance du type suivant :

Caractère	Numéro	Binaire	Caractère	Numéro	Binaire
_	0	00000	P	16	10000
A	1	00001	Q	17	10001
B	2	00010	R	18	10010
C	3	00011	S	19	10011
D	4	00100	T	20	10100
E	5	00101	U	21	10101
F	6	00110	V	22	10110
G	7	00111	W	23	10111
H	8	01000	X	24	11000
I	9	01001	Y	25	11001
J	10	01010	Z	26	11010
K	11	01011	,	27	11011
L	12	01100	:	28	11100
M	13	01101	.	29	11101
N	14	01110	'	30	11110
O	15	01111	-	31	11111

Par soucis de simplicité, on a uniquement considéré des lettres majuscules, l'espace et quelques signes de ponctuation. Chaque caractère est ainsi représenté par un nombre de 5 chiffres en binaire. Pour coder un texte en une suite de 0 et de 1, il suffit mettre bout à bout les codages de chaque caractère du texte.

Question 24. Écrire le texte BONJOUR sous forme d'une suite de 0 et de 1.

Question 25. Quel est le texte représenté par la suite 01010011110110001001 ? On fournit les fonctions suivantes :

- La fonction binaire(text) construit et renvoie la liste L contenant des 0 et des 1 associée à la chaîne de caractères text ;
- La fonction texte(L) construit et renvoie la chaîne de caractères

correspondant au texte décrit par la liste L de 0 et de 1.

On va maintenant modifier chaque pixel de l'image pour représenter la suite de 0 et de 1. Pour simplifier à nouveau, on supposera que l'image contient au moins autant de pixels qu'il y a d'éléments dans la suite. Considérons tout d'abord un exemple, la matrice :

$$A = \begin{bmatrix} 0 & 127 & 0 & 27 \\ 255 & 0 & 127 & 200 \\ 0 & 255 & 0 & 255 \\ 255 & 0 & 255 & 200 \end{bmatrix}$$

représente l'image de départ et la suite de 0 et de 1 à coder est 0011011101. On dispose cette suite dans une matrice C de même taille que A en complétant avec des 0 si besoin (les 0 ajoutés sont soulignés) :

$$C = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & \underline{0} & \underline{0} \\ \underline{0} & \underline{0} & \underline{0} & \underline{0} \end{bmatrix}$$

À partir de ces deux matrices, on construit une nouvelle matrice B de même taille dans laquelle on va dissimuler le message contenu dans C. On impose que :

- Chaque coefficient B[i , j] soit assez proche de A[i , j] pour que l'image décrite par B soit visuellement identique à celle décrite par A. On aura en fait B[i , j]- A[i , j]=-1, 0 ou 1
- Chaque coefficient B[i , j] puisse permettre de retrouver le coefficient C[i ,j] associé. On convient de prendre B[i , j] pair lorsque C[i , j] =0 et B[i ,j] impair sinon.

Ceci conduit à définir B[i ,j] de la manière suivante :

- Si A[i , j] est pair, alors on pose B[i , j]= A[i , j]+C[i , j] ;
- Si A[i , j] est impair, alors on pose B[i , j] = A[i , j]-1+C[i, j].

On obtient sur l'exemple considéré :

$$B = \begin{bmatrix} 0 & 126 & 1 & 27 \\ 254 & 1 & 127 & 201 \\ 0 & 255 & 0 & 254 \\ 254 & 0 & 254 & 200 \end{bmatrix}$$

Question 15. Écrire une fonction **disposer(L,A)** qui prend en paramètre une liste L contenant des 0 et des 1 et qui construit et renvoie la matrice C de même taille que A obtenue en disposant les éléments de L et en complétant si besoin par des 0.

Question 17. Écrire une fonction **dissimuler(A,C)** qui construit et renvoie la matrice B obtenue à partir des matrices A et C supposées de même taille.

Question 18. Écrire une fonction **retrouver(B)** qui prend en paramètre la matrice B (qui a été construite en codant un texte dans une matrice A) et renvoie la liste de 0 et de 1 associée.

Partie VII. Segmentation d'image

Extrait de WIKIPÉDIA :

La segmentation d'image est une opération de traitement d'images qui a pour but de rassembler des pixels entre eux suivant des critères pré-définis. Les pixels sont ainsi regroupés en régions, qui constituent un pavage ou une partition de l'image. Il peut s'agir par exemple de séparer les objets du fond. Si le nombre de classes est égal à deux, elle est appelée aussi binarisation.

Question 19. Méthode manuelle de segmentation.

(a) On considère un nombre entier v appelé seuil. On définit une nouvelle image B, de même taille que A, en posant :

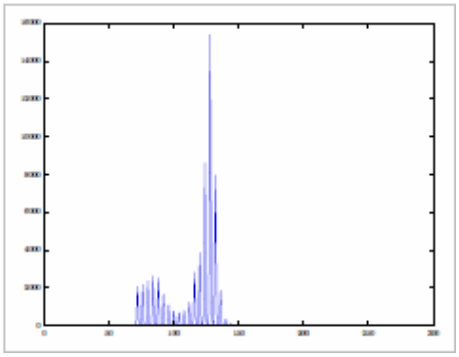
$$B[i,j] = 0 \quad \text{si} \quad A[i,j] \leqslant v$$
$$= 255 \quad \text{sinon}$$

On dit

que B est l'image monochrome obtenue à partir de A par seuillage. Écrire une fonction segmentation(A,v) qui construit et renvoie l'image B ainsi définie.

(b) L'histogramme de l'image A est la liste H de taille 256 telle que H[i] est égal au nombre de cases de valeur i dans A. Écrire une fonction histogramme(A) qui construit et retourne la liste H. On pourra ensuite effectuer le tracé de l'histogramme avec (par exemple).

```
plt.plot(histogramme(Cellules))
plt.show()
```



c) Examiner l'histogramme de l'image Cellules.png afin de déterminer un seuil pertinent pour effectuer le seuillage et isoler les cellules du fond de l'image.

Question 20. Méthode automatique.

(a) On donne l'algorithme suivant permettant d'obtenir automatiquement un seuil v pertinent. Cet algorithme utilise un paramètre eps>0 donné.

- On définit une première valeur de v en posant v=(min(A)+max(A))/2
- On détermine la moyenne u1 des éléments de A qui sont strictement inférieurs à v ainsi que la moyenne u2 des éléments de A qui sont supérieurs ou égaux à v ;
- On considère la nouvelle valeur v'=(u1+u2)/2
- Si |v-v'|<=eps, alors on arrête en renvoyant la valeur v0. Sinon on recommence en prenant v=v'.

Programmer une fonction **seuil_automatique(A,epsilon)** qui met en oeuvre cet algorithme.

(b) Programmer une fonction **segmentation_automatique(A)** qui utilise cet algorithme