

Devoir maison Facultatif n°1 - Corrigé

Problème : Autofocus sur les appareils photos numériques

Question 1 : La valeur d'une composante est représentée par un entier allant de 0 à 255. Il y a donc $256 = 2^8$ valeurs possibles, c'est codable sur 8 bits = 1 octet.

L'espace mémoire nécessaire pour stocker la valeur d'une composante est donc un octet.

Pour chaque pixel, il y a 3 composantes (RVB).

L'espace mémoire nécessaire pour stocker la valeur d'un pixel est donc 3 octets.

Pour une image, il y a 48 MPixels, et chaque pixel nécessite 3 octets.

L'espace mémoire nécessaire pour stocker la valeur d'une image est donc 144 Mo ($48 \times 3 = 144$).

Question 2 :

```
1 def Clinear(val):
2     if val <= 0.04045:
3         return val/12.92
4     else:
5         return ((val+0.055)/1.055)**2.4
```

Question 3 :

```
1 def Y(pix): #pix est une liste de trois valeurs
2     Ylin=0.2126*Clinear(pix[0]) + 0.7152*Clinear(pix[1]) + 0.0722*Clinear(pix[2])
3     if Ylin <= 0.0031308:
4         return 12.92*Ylin
5     else:
6         return 1.055*Ylin**(1/2.4) - 0.055
```

Question 4 :

```
1 def NiveauxGris(I): #I est un tableau à trois dimensions
2     hauteur=len(I) #hauteur de l'image
3     largeur=len(I[0]) #largeur de l'image
4     gris=[[0 for i in range(largeur)] for i in range(hauteur)]
5     #un tableau à deux dimensions rempli de 0
6     for i in range(hauteur):
7         for j in range(largeur):
8             gris[i][j]=Y(I[i][j]) #I[i][j] est bien une liste de trois valeurs
9     return gris
```

Question 5 :

```
1 def convolution (A,B):
2     s=0
3     for i in range(3):
4         for j in range(3):
5             s+=A[i][j]*B[i][j]
6     return s
```

Question 6 :

```
1 def contraste_pixel (l,i,j): #l est un tableau à deux dimensions
2   Gx = [[-1,0,1],[-2,0,2],[-1,0,1]]
3   Gy = [[-1,2,-1],[0,0,0],[1,2,1]]
4   ltab=np.array(l) #transformation en tableau numpy pour pouvoir faire du slicing
5                   #sur les lignes et les colonnes
6   le=ltab[i-1:i+2][j-1:j+2] #on utilise du slicing sur les lignes et les colonnes
7   return int (sqrt (convolution (le,Gx)**2+convolution(le,Gy)**2))
```

Question 7 :

```
1 def contraste(l): #l est un tableau à deux dimensions
2   c=0
3   hauteur=len(l) #hauteur de l'image
4   largeur=len(l[0]) #largeur de l'image
5   for i in range(1,hauteur-1): #on ne prend pas les pixels du bord
6     for j in range(1,largeur-1):
7       c+=contraste_pixel(l,i,j)
8   return c/((hauteur-2)*(largeur-2)) #on fait la moyenne en divisant par
9                                     #le nombre total de pixel
```

Question 8 :

```
1 def extraction (L1,L2,dec):
2   if dec==0:
3     return L1,L2
4   if dec>0:
5     return L1[:len(L1)-dec],L2[dec:]
6   if dec<0:
7     return L1[-dec:],L2[:len(L2)+dec]
```

Question 9 :

```
1 def comparaison(L1,L2):
2   if len(L1)!=len(L2):
3     return False
4   for i in range(len(L1)): #si on arrive ici c'est que les listes ont même longueur
5     if L1[i]!=L2[i]:
6       return False #si deux éléments diffèrent, inutile de continuer
7   return True
```

Question 10 :

```
1 def recherche_décalage(L1,L2):
2   for dec in range(-80,81): #on va tester toutes les valeurs de décalage possibles
3     test=comparaison(extraction(L1,L2,dec))
4     if test :
5       return dec
6   return None #si on arrive ici c'est qu'on n'a trouvé aucun décalage qui convienne
```

Question 11 :

Dans le meilleur des cas : le décalage cherché est le 1er testé : $dec = -80$, on a alors :

- la fonction recherche_décalage n'utilise qu'une fois la fonction comparaison, et qu'une fois la fonction extraction.
- les deux sous-listes testées sont de longueur $(n-80)$.
- la fonction comparaison, qui retournera True par hypothèse, effectue $(2 \times (n-80) + 1)$ comparaisons (à chaque fois, comparaison entre compteur et $\text{len}(L1)$, et entre $L1[\text{compteur}]$ et $L2[\text{compteur}]$, et la comparaison finale entre compteur et $\text{len}(L1)$).
- La fonction extraction effectue 2 comparaisons.

Il y aura alors $2 + (2 \times (n-80) + 1) = O(n)$ comparaisons (n est grand, supposé plus grand que 80).

Dans le pire des cas : il faut alors tester les m valeurs possibles pour dec . En ordre de grandeur, il y aura $2nm = O(nm)$ comparaisons.

Question 12 :

```
1 def erreur(L1,L2): #on suppose que les deux listes ont même longueur
2     e=0
3     for i in range(len(L1)):
4         e+=(L1[i]-L2[i])**2
5     return e/len(L1)
```

Question 13 :

```
1 def recherche_décalage_2(L1,L2):
2     dec_min=-80 #le décalage qui minimisera l'erreur
3     e_min=erreur(extraction(L1,L2,dec_min)) #l'erreur minimum
4     for dec in range(-80,81): #on va tester toutes les valeurs de décalage possibles
5         e=erreur(extraction(L1,L2,dec))
6         if e<e_min:
7             e_min=e
8             dec_min=dec #si on trouve une erreur plus petite on met à jour la valeur de décalage
9     return dec_min
```