Informatique - TD 08

Algorithmes sur les images

Nous allons travailler sur des images *ponctuelles*, c'est à dire des images décrites par la valeur de chacun de leurs *pixels*. Ainsi, on représentera nos images comme des tableaux à deux dimensions, où chaque case contient la couleur du pixel correspondant.

De plus, on ne travaillera que sur des images en *niveau de gris* (« en noir et blanc », mais il n'y a pas que du noir et du blanc, mais bien plusieurs niveaux (ou nuances) de gris). On code le niveau de gris par un entier compris entre 0 et 255 : 0 correspond au noir total, et 255 au blanc total. En résumé, notre image sera un tableau à deux dimensions contenant des entiers entre 0 et 255.

Manipulation d'une image

Question 1. Installez pyzo et PIL. Puis, rendez vous dans hlabrande.fr/BCPST1 ; vous trouverez deux images, `lovelace.jpg` et `hopper.jpg` ; choisissez une des deux images et mettez-la dans votre répertoire de travail (le même que celui où votre fichier de code est enregistré).

Question 2. Quel est l'effet de chacune des lignes du programme suivant ?

```
from PIL import Image
img = Image.open("lovelace.jpg")                # ou "hopper.jpg"
pixels = img.load()
hauteur=img.height; largeur=img.width
img.show()
```

Modifier directement le tableau `pixels` revient à modifier l'image, et les modifications apparaîtront à l'affichage. Attention cependant : le tableau `pixels` est un type spécial (`PixelAccess`), et il faut taper `pixels[i,j]` (au lieu de `[i][j]`) pour accéder à l'élément $M_{i,j}$.

Question 3. Modifier l'image de sorte à ce que la colonne 200 soit entièrement noire. Afficher le résultat. Rendre aussi la ligne 250 blanche, et afficher le résultat.

Algorithmes à modifications "sur place"

Nous allons commencer par les algorithmes pour lesquelles les modification se font "sur place", c'est à dire ceux où il suffit de balayer l'image (avec deux boucles `for`) et de modifier les valeurs des pixels à mesure qu'on les rencontre. Dans tout ce qui suit, $I_{i,j}$ désigne la valeur du pixel de coordonnées (i, j) dans l'image, et $T_{i,j}$ la valeur du pixel de coordonnées (i, j) après transformation.

Question 4. Écrire une fonction `negatif` qui inverse le sombre et le clair. (La formule est une fonction affine qui transforme 0 en 255 et 255 en 0).

Question 5. Écrire une fonction `seuil` qui prend un argument $s \in \mathbb{N}$ en paramètre et qui transforme un pixel en un pixel blanc si $I_{i,j} > s$ et en un pixel noir sinon. Faire des essais avec plusieurs paramètres.

Question 6. Écrire une fonction `assombrir` qui prend un argument p en paramètre tel que $p \in [0;1]$, et qui renvoie l'image transformée telle que $T_{i,j} = (1-p)I_{i,j}$. On n'oubliera pas d'arrondir le résultat, pour bien obtenir un entier. Tester la fonction pour différentes valeurs du paramètre p (en particulier, 0 et 1).

Question 7. Écrire une fonction `eclaircir` qui prend un argument p en paramètre tel que $p \in [0;1]$ et qui renvoie l'image transformée telle que $T_{i,j} = 255 - (1-p)(255 - I_{i,j})$. On n'oubliera pas d'arrondir le résultat, pour bien obtenir un entier. Tester la fonction pour différentes valeurs du paramètre p (en particulier, 0 et 1).

Question 8. Écrire une fonction `compression` qui transforme l'image de sorte à ce que seulement 16 nuances de gris (espacées régulièrement) soient utilisées. On utilisera un arrondi de $\frac{I_{i,j}}{16}$ pour déterminer le numéro de la nuance à utiliser pour un pixel.

Algorithmes à modification différée

Les formules dans les algorithmes suivants dépendent des valeurs des pixels voisins ; on ne peut ainsi pas modifier les valeurs des pixels "sur place", car cela affectera les calculs futurs (sur les voisins à droite ou en bas). Il faut ainsi travailler sur un tableau auxiliaire, et le remplir peu à peu jusqu'à avoir obtenu le résultat final.

Question 9. Écrire une commande qui crée un tableau `p` rempli de zéros et avec les mêmes dimensions que l'image choisie.

On travaillera sur ce tableau ; pour mettre à jour l'image, on utilisera

```
img.putdata([p[i][j] for i in range(0,hauteur) for j in range(0,largeur)])
```

Question 10. Écrire une fonction `flou` qui prend un paramètre $n \in \mathbb{N}$ en argument, et qui floute l'image de la façon suivante : chaque pixel prend la valeur moyenne des pixels qui l'entourent dans une boîte de rayon n :

$$T_{i,j} = \frac{1}{(2n+1)^2} \sum_{k \in [\![i-n, i+n]\!]} \sum_{\ell \in [\![j-n, j+n]\!]} I_{k,\ell}$$

Question 11. Écrire une fonction `contours` qui prend un paramètre $s \in \mathbb{N}$ en argument, et qui réalise l'algorithme suivant : on calcule

$$T_{i,j} = |I_{i+1,j} - I_{i,j}| + |I_{i-1,j} - I_{i,j}| + |I_{i,j+1} - I_{i,j}| + |I_{i,j-1} - I_{i,j}|$$

puis on applique un filtre de paramètre s (c'est à dire le pixel devient noir si sa valeur est plus grande que s, et blanc sinon). Appliquer cette fonction à l'image choisie pour plusieurs valeurs de paramètres, puis déterminer le meilleur paramètre pour votre image.