Ce TP est en très grande partie issue du livre "Informatique pour tous en classes préparatoires aux grandes écoles" de Wack et al aux éditions Eyrolles

---

Exercice 1 :

Sur la page web du cours télécharger dans un répertoire personnel les fichier **image.py** et **chromosome.pgm**.

(1) Analyser le fichier **image.py**.

Il apparaît que pour beaucoup de traitement de l'image auront la même structure que la fonction **negatif**.

On peut donc commencer par rendre leur écriture plus efficace

(2) Écrire une fonction **traitement** qui prend pour arguments une image et une fonction puis applique cette fonction sur la valeur de chacun des pixels de l'image.

(3) Redéfinir **negatif** à l'aide de **traitement**.

(4) On joue maintenant sur la luminosité de l'image : puisque les valeurs les plus hautes sont les plus claires, il suffit d'augmenter les valeurs des pixels pour éclaircir l'image et de les diminuer pour l'assombrir. On est alors tenté d'appeler **traitement** avec une fonction comme

```
def lumi(x):
        return x + 50
```

On s'aperçoit vite que l'effet obtenu n'est pas celui escompté : tous les pixels dont la valeur était supérieure à 205 reçoivent une valeur entre 255 et 305. Comme les valeurs des pixels sont comptées modulo 255, ces pixels deviennent en fait presque noirs.

Écrire une fonction qui éclaircit une image en s'assurant par un test que la valeur des pixels est plafonnée à 255.

(5) La solution précédente n'est pas satisfaisante car elle crée de grands aplats blancs dans les zones les plus claires (on pourra augmenter la valeur des pixels de 100 ou 150 pour s'en convaincre).

À l'aide de la fonction racine carrée, construire une bijection de $[0;255]$ dans $[0;255]$. Programmer cette fonction pour qu'elle manipule et renvoie des entiers (ce n'est alors évidemment plus une bijection). Passer cette fonction en paramètre à traitement et vérifier le résultat obtenu.

(6) Construire de même une solution pour assombrir une image.

(7) Tracer le graphe de la fonction $f : x \mapsto 128+\text{signe}(x-128)\times\sqrt{128|x-128|}$ sur $[0;255]$. À l'aide de ce graphe, prévoir l'effet qu'aura cette fonction sur l'image, puis tester votre hypothèse.

(8) Expérimenter avec des fonctions linéaires, non monotones, etc.

Exercice 2 :

On s'intéresse maintenant à des traitements qui font intervenir plusieurs pixels voisins. La fonction **traitement** écrite dans l'exercice précédent n'est donc plus utilisable.

(1) Écrire une fonction qui produit une image floutée. Pour cela, il suffit de remplacer chaque pixel par la moyenne des pixels qui l'entourent.

On obtient différents degrés de floutage selon qu'on considère les 9 pixels immédiatement adjacents, ou bien un carré de 5 voire 7 pixels de côté centré sur le pixel à calculer.

(2) À l'inverse, on peut chercher des contours dans une image ; il s'agit de détecter les endroits où les pixels changent brutalement de couleur. Pour cela, on calcule pour chaque pixel une moyenne pondérée des pixels qui l'entourent, par exemple avec les coefficients :

-1	-1	-1
-1	8	-1
-1	-1	-1

Quel est le résultat de ce calcul pour un pixel qui est entouré d'autres pixels d'une couleur proche ? Et si au contraire il est d'une couleur nettement différente de celle d'une partie de ses voisins ?

Programmer cette méthode de détection des contours.

(3) Expérimenter avec d'autres coefficients. On retrouve ainsi encore d'autres fonctionna- lités des logiciels de traitement d'image.

Exercice 3 :

On peut enfin chercher à modifier l'image dans sa globalité.

(1) Écrire une fonction qui agrandit une image d'un facteur $k > 1$ donné. Pour k entier, il suffit de remplacer chaque pixel par un bloc $k \times k$ pixels. Si k est décimal, il faudra procéder à des interpolations pour calculer la couleur des pixels de l'image agrandie.

(2) Écrire une fonction qui divise par deux les dimensions d'une image. Chaque bloc de 2×2 pixels sera donc remplacé par un unique pixel dont la valeur est la moyenne des quatre pixels originaux.

(3) Généraliser ce principe pour écrire une fonction qui réduit la taille d'une image d'un facteur $k < 1$.

(4) Que se passe-t-il si on prend deux images de même taille et qu'on calcule leur différence pixel par pixel ?

Exercice 4 :

Pour finir, on va travailler un peu sur des images en couleurs. On rappelle que pour ces images, chaque pixel est représenté par un triplet (r, v, b) de nombre compris entre 0 et 255 représentant successivement l'intensité du rouge, du vert et du bleu.

Écrire les fonctions suivantes permettant de transformer une image couleur en une image en noir et blanc.

(1) **Clarté** , où le niveau de gris de chaque pixel est la moyenne entre le minimum et le maximum des trois composantes RVB. Si par exemple (R,V,B) = (122,200,147), cette moyenne vaut (122+200)/2 = 161, et le résultat est (R,V,B) = (161,161,161).

(2) **Luminosité** , où le niveau de gris correspond à R = 0,21*R+0,71*V+0,07*B, et V = R, B = R.

(3) **Moyenne**, où R = (R+V+B)/3 et V = R , B = R

(4) **Noir et blanc**, trouver une méthode pour que l'image devienne noir et blanc.