UNIVERSITÉ DE CAEN **L2**
U.F.R. des Sciences **Calcul scientifique**

TP 2 : Traitement d'image à l'aide de l'histogramme de niveau de gris

---

1 Introduction

L'objectif de ce TP est de se familiariser avec des opérations standards des librairies numpy, scipy et matplotlib. Nous réaliserons dans ce TP des fonctions basiques de traitement d'image.

> Sauf si cela est précisé, vous ne devez pas utiliser de boucle (for, while) ou de branchement conditionnel (if) durant ce TP.

2 L'histogramme des niveaux de gris

En imagerie numérique, l'histogramme représente la distribution des intensités (ou des couleurs) de l'image. C'est une représentation graphique donnant pour chaque niveau de gris (ou couleur) le nombre de points de l'image ayant ce niveau de gris (couleur).

L'histogramme permet ainsi de visualiser la répartition des différents niveaux de gris de l'image. Pour chaque niveau de gris entre 0 (noir) jusqu'à 255 (blanc) en abscisse, vous avez le nombre de pixels de l'image ayant cette valeur en ordonnée.

Comme dans le TP 1, nous allons charger l'image *ascent* pour effectuer des traitements. Vous pouvez pour cela utiliser la commande im = sc.misc.ascent().

Nous allons maintenant afficher l'histogramme des niveaux de gris de cette image, grace à la fonction suivante :

```
def hist(im):
  x,h = np.unique(im,return_counts=True)
  plt.fill_between(x,0,h)
  plt.axis((0,255,0, np.max(h)))
```

Questions :

1. Expliquez à quoi sert la fonction `np.unique`. Idem avec la fonction `plt.fill_between`.
2. Que manque t-il à la fonction `hist` pour que l'affichage se fasse ?
3. Que pouvez vous dire sur l'image en regardant son histogramme ?

3 Éclaircir et assombrir une image

Nous allons dans un premier temps chercher à éclaircir l'image. Pour cela, nous allons augmenter l'ensemble des valeurs des niveaux de gris des pixels. Il faut néanmoins faire en sorte que les valeurs restent entre 0 et 255.

1. Soit n notre paramètre de réglage de l'effet. n Varie entre 0 (aucun effet) et 255 (effet maximal, l'image devient entièrement blanche). Réalisez un programme qui sature (mettre à 255) les pixels dont la valeur de départ est supérieure à $255 - n$ et augmente de n les autres pixels.
2. Affichez l'histogramme avant et après l'éclaircissement et regarder les conséquences du traitement pour différentes valeurs de n.
3. Réalisez maintenant l'effet inverse. Les valeurs inférieures à n doivent être ramenées à 0 et les autres doivent être réduites de n.

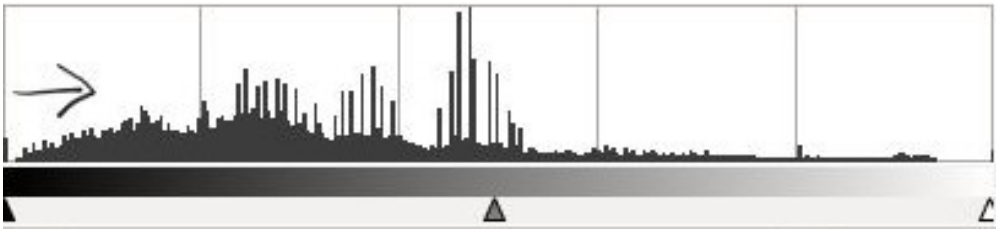

FIGURE 1 – Éclaircir l'image

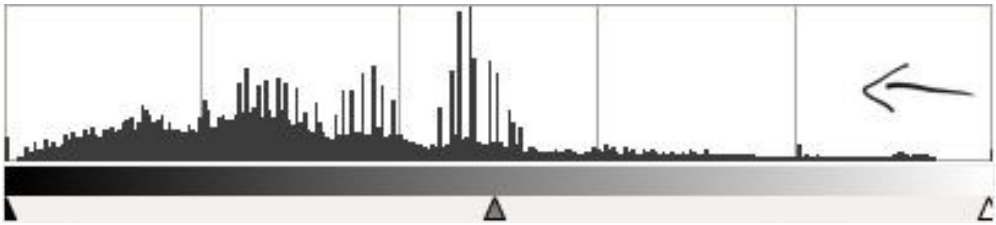

FIGURE 2 – Assombrir l'image

4 Étirement d'histogramme (comparaison numpy/-python seul)

L'objectif de cet exercice est d'étirer l'histogramme d'une image, en appliquant aux niveaux de gris des pixels une transformation affine telle que le plus petit niveau de gris aura, pour valeur 0 et le plus grand 255, après transformation des niveaux de gris.

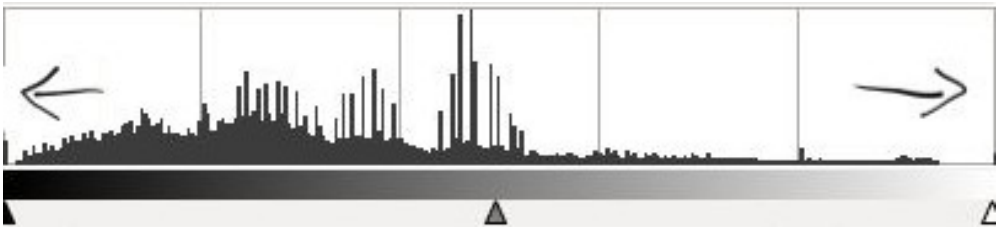

FIGURE 3 – Étirement d'histogramme

Nous utiliserons pour cela deux méthodes et comparerons l'efficacité de ces deux méthodes.

Mais auparavant, vous commencerez par charger l'image face_gris.png (fournie sur ecampus) en mémoire dans la variable `im` et vous l'afficherez à l'écran. La lecture d'une image sur disque peut se faire au moyen de la fonction `sc.ndimage.imread` de la librairie `scipy`. L'affichage peut se faire à l'aide de la fonction `plt.imshow(im,cmap='gray',vmin=0,vmax=255)` de matplotlib. L'option cmap='gray' permet de préciser que l'image est en noir et blanc. vmin = 0 et vmax = 255 permet de préciser que les pixels sont définis par des valeurs allant de 0 (noir) à 255 (blanc).

Première méthode : utilisation de boucles

1. Commencez par visualiser l'histogramme de cette image.
2. À l'aide d'instructions `if` et `for` (en utilisant que python3 de base) trouvez la valeur minimale et maximale des pixels de l'image. Nous les noterons p_{min} et p_{max} dans la suite du sujet.

3. À l'aide d'instructions `if` et `for` créer une nouvelle image dont les pixels correspondent à l'équation $255\frac{p_{ij} - p_{min}}{p_{max} - p_{min}}$ où p_{ij} est le pixel de la ligne i et de la colonne j. Cette opération consiste à ramener les valeurs effectives des pixels entre 0 et 255 et forcer ainsi l'utilisation de toute la plage de valeur possible. Pour cette question vous calculerez les nouveaux niveaux de gris, pixel par pixel, à l'aide de l'image d'origine. Pour simplifier l'initialisation, vous pouvez commencer avec une image noire avec l'instruction : im2 = np.zeros(im.shape).
4. Affichez cette image et mesurez le temps d'exécution de votre programme. Vous devriez voir l'image "cachée" du fichier face_gris.png .

Seconde méthode : utilisation de numpy Utilisez maintenant les fonctions numpy : `np.min` et `np.max` pour calculer la valeur minimale et maximale des pixels de l'image. Effectuez le même traitement que pour la méthode précédente, mais cette fois-ci sans boucle, en traitement l'ensemble de l'image en même temps grâce aux opérations mathématiques sur les array numpy.

Comparez les images obtenues avec les deux approches. Que pouvez-vous dire des temps d'exécutions dans les deux cas ?

5 Rehaussement de contraste

Pour rehausser les contrastes d'une image, il faut accentuer l'écart entre les tons clairs et les tons foncés. Pour cela, il faut assombrir davantage les pixels sombres et éclaircir les pixels clairs.

Afin d'obtenir ce résultat, nous allons appliquer aux pixels de l'image une fonction qui augmente les valeurs déjà fortes et diminue les valeurs plus faibles. Généralement on prend pour faire cette opération une courbe en forme de "S".

1. Affichez sur la même figure les fonctions $y = x$ et $y = 0.5\cos((1 - x)\pi) + 0.5$ entre 0 et 1.
2. Prenez une des deux images vues dans ce TP et divisez toutes ces valeurs par 255.0 pour avoir des valeurs comprises entre 0 et 1.
3. Prenez l'image de la question précédente et appliquez une courbe en S comme celle vu dans la question 1.
4. Repassez toutes les valeurs entre 0 et 255 en multipliant l'image de la question précédente par 255.
5. Visualisez l'effet obtenu sur l'image et sur son histogramme.