Licence Informatique **INF158 – Projet de programmation 1**
Université Bordeaux 1 **Feuille 4**

Des fichiers d'images de test (format PGM binaire) vous sont donnés. Téléchargez les à l'aide de la commande : wget http://dept-info.labri.fr/ENSEIGNEMENT/projetprog1/src/fichiers-4.tar.gz

Le but de cette séance est d'implémenter les algorithmes de traitement d'image demandés dans le projet. Il est souhaitable d'avoir terminé la feuille 3. Si ce n'est pas le cas, vous pouvez tout de même écrire complètement le module de traitement d'image (il suffit de disposer des fonctions de lecture et écriture d'un fichier PGM binaire et d'un afficheur d'images comme gimp)

Comme structure de données pour représenter une image en mémoire, vous avez le choix entre :
- un tableau 1D de valeurs entre 0 et 255 (les niveaux de gris des pixels). Dans ce cas, il faut également stocker la largeur et la hauteur de l'image.
- un tableau 2D de valeurs entre 0 et 255.

Attention, le choix de votre structure de donnée a une influence sur les algorithmes . . .

Placez vous dans votre répertoire `projet`. Ouvrez un fichier de nom `traitement.py`. La structure de données contenant l'image à traiter sera passée en paramètre de chaque fonction de traitement. Elle sera modifiée par la fonction. Il faut impérativement tester chaque fonction avant de passer à la suivante.

Module `traitement`

Augmentation de la luminosité

Pour augmenter la luminosité, il suffit d'ajouter une valeur constante à chaque niveau de gris. Si le niveau de gris d'un pixel est supérieur à 255 après cette modification, on le ramène à 255.

Complément: *l'utilisateur pourra choisir la valeur de l'augmentation dans une boite de dialogue. Il pourra également autoriser une diminution de la luminosité.*

Inversion vidéo

Pour obtenir un effet de « *négatif* », il faut remplacer la valeur d'un pixel de niveau de gris `ng` par 255 − `ng` (255 est la valeur maximum des niveaux de gris).

Flou

Dans un premier temps, le voisinage considéré pour chaque pixel est un carré de côté 3 centré sur le pixel. On remplace la valeur de chaque pixel par la valeur moyenne de son voisinage. Attention au traitement des pixels du bord de l'image.

Complément: *Étendre la taille de voisinage à un carré de côté* `2n+1`. *L'utilisateur pourra choisir la valeur de* `n` *dans une boite de dialogue. On vérifiera que la valeur de* `n` *choisie est acceptable.*

Filtrage médian

Dans un premier temps, le voisinage considéré pour chaque pixel est un carré de côté 3 centré sur le pixel. Les neuf valeurs de ce voisinage sont triées. On appelle valeur médiane celle qui apparaît en $5^{\text{ème}}$ position dans la suite des valeurs triées). On remplace la valeur de chaque pixel par la valeur médiane de son voisinage.

Complément: *Étendre la taille de voisinage à un carré de côté* `2n+1` *comme pour le flou.*

Augmentation de contraste

Les valeurs de niveau de gris de l'image initiale sont comprises entre $\texttt{ng}_{\min}$ et $\texttt{ng}_{\max}$. Si ces deux valeurs sont très proches l'une de l'autre, il y aura peu de différence entre les différents niveaux de gris de l'image et donc peu de contraste. Une méthode simple pour augmenter le contraste est de ramener linéairement l'intervalle de niveaux de gris de $[\texttt{ng}_{\min}, \texttt{ng}_{\max}]$ à $[0, 255]$.

Complément: *Que se passe-t-il si l'image est uniforme. Comment traiter ce cas ?*

Symétrie

Cette fonction réalise une symétrie autour d'une droite horizontale centrée dans l'image (échange entre haut et bas).