SÉANCE 9

Objectif

Le but de cette séance est de poursuivre la mise en œuvre des techniques de programmation modulaire. Si vous n'avez pas terminé les exercices de la séance 8, commencez par le faire avant de passer à la suite décrite ci-dessous. Nous vous rappelons que vous devez travailler en dehors des séances.

Exercice

Exercice (Module de traitement d'images)

L'objectif de cet exercice est d'écrire un module `ti` d'opérateurs de traitement d'images de niveaux de gris qui utilise les modules `matrice` et `image` de la séance précédente.

Ce module ne définira pas une nouvelle structure de données (il utilisera celles des modules `matrice` et `image`), mais il proposera des fonctions de traitement d'images.

Vous devez écrire :

- le fichier d'en-tête `ti.h` ;
- le fichier d'implémentation `ti.c` ;
- le fichier `tp9ex.c` qui va contenir la fonction principale permettant de tester le module ;
- le fichier de description des dépendances permettant de produire l'exécutable `tp9ex`.

Les fonctions publiques que vous devez programmer sont décrites ci-dessous.

1. `Image Rotation(Image Im)`
   Cette fonction doit créer une image qui est le résultat de la rotation de 90° de l'image `Im`. Cette fonction retourne la nouvelle image ou `NULL` en cas de problème. Vous pouvez réutiliser une partie de la fonction principale que vous avez écrite pour tester le second exercice de la séance 8.

2. `Image AppliquerTable(unsigned char Table[256], Image Im)`
   La fonction `Rotation` est un exemple de transformation géométrique d'une image. Avec la fonction `AppliquerTable`, il s'agit d'un autre type de transformation d'image : les transformations ponctuelles. Le résultat d'une telle transformation est une autre image dont le niveau de gris de chaque pixel est fonction du niveau de gris du pixel situé au même endroit dans l'image initiale. Les niveaux de gris étant des entiers compris entre 0 et 255, cette transformation peut être complètement décrite par une « table de transcodage » que vous implémenterez par un tableau de 256 valeurs de type `unsigned char`. L'indice de chaque case de cette table indique l'ancien niveau de gris et son contenu indique le nouveau niveau de gris. Par exemple, si la case d'indice 112 contient la valeur 184, cela signifie que tous les pixels dont le niveau de gris dans l'image initiale était égal à 112 vont avoir le niveau de gris 184 dans l'image finale.

   Pour obtenir l'image résultat, il suffit donc de parcourir l'image initiale et, pour chaque pixel, mettre dans l'image finale le niveau de gris contenu dans la case de la table de transcodage dont l'indice est égal au niveau de gris de l'image initiale.

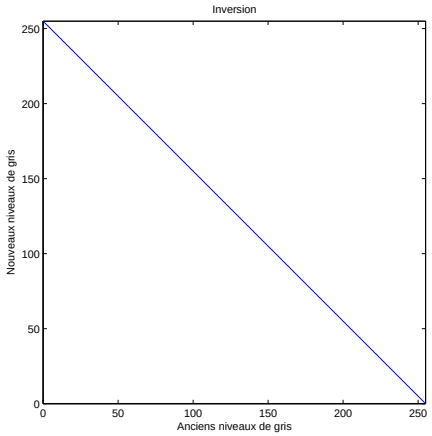
   L'effet visuel obtenu dépendra des valeurs contenues dans la table de transcodage. Par exemple, si elle est représentée par le tableau `Table` et si `Table[0]==255`, `Table[1]==254`, `Table[2]==253`, $\cdots$, `Table[254]==1`, `Table[255]==0`, c'est-à-dire si `Table[g]==255-g`, alors l'image finale sera le « négatif » (au sens de la photographie) de l'image initiale.

   La fonction `AppliquerTable` prend en entrée la table de transcodage `Table` et l'image initiale `Im`. Elle crée et retourne l'image finale.

3. `void RemplirTableInversion(unsigned char Table[256])`
   Cette fonction remplit le tableau `Table` pour créer la table de transcodage correspondant à la transformation d'inversion des niveaux de gris (obtention du négatif) :

$$F(g) = 255 - g \tag{1}$$

   où g désigne l'ancien niveau de gris et $F(g)$ le nouveau niveau de gris.

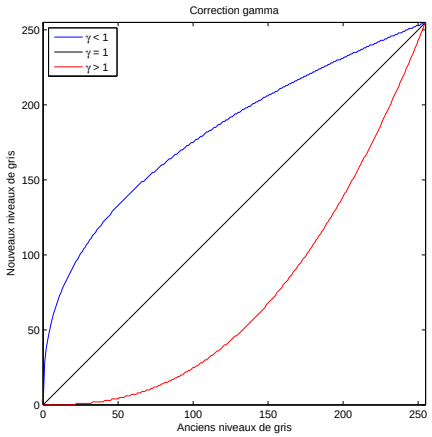


   Testez les deux fonctions précédentes par exemple sur l'image `chien.pgm`.

4. `void RemplirTableGamma(double Gamma, unsigned char Table[256])`
   Cette fonction crée la table de transcodage correspondant à la transformation appelé « correction gamma » :

$$F(g) = \left[255 \left(\frac{g}{255}\right)^{\gamma}\right] \tag{2}$$

   où $[x]$ désigne l'entier le plus proche de x. Si $\gamma = 1$ alors la transformation sera sans effet. Si $\gamma < 1$ alors l'image finale sera plus claire que l'image initiale (généralement, on choisit $\gamma \in [0.4, 0.5]$). Si $\gamma > 1$ alors l'image finale sera plus sombre que l'image initiale (généralement, on choisit $\gamma \in [2, 2.5]$).



   La fonction `double round(double x)` retourne l'entier le plus proche de `x`.
   La fonction `double pow(double x, double y)` retourne x^y.
   Ces deux fonctions font partie de la bibliothèque standard mathématique. Elles nécessitent donc l'inclusion de `math.h` et l'ajout de `-lm` lors de l'édition de liens.
   Testez votre programme sur l'image `paysage.pgm` avec plusieurs valeurs pour γ, par exemple 0.4 et 2.5.