PROGRAMMATION C
TP7 - TRAITEMENT D'IMAGE (1)

LICENCE MATHS-INFO
12 MARS 2012

Introduction

Ce TP est le premier d'une série sur le traitement des images. Pour commencer, vous allez créer un type pour manipuler les images, des fonctions de lecture et d'écriture d'images, et les premiers traitements.

Vous n'aurez besoin que des bibliothèques standards que vous connaissez déjà et de deux images disponibles sur le site du module [1] : `feep.pgm` pour développer les fonctions, et `homer_ascii.pgm` pour les tester avec une image plus intéressante. Le programme de test est également sur le site.

1. Définition du type Image

Nous considérons uniquement les images en niveaux de gris. Une image est vue comme une matrice dont chaque élément est un flottant correspondant à un pixel. Ces nombres correspondent aux niveaux de gris, codés de 0 (noir) à 1 (blanc). Une matrice de M lignes et N colonnes sera stockée dans un tableau à une dimension de longueur MN.

Dans un fichier `types_image.h`, définissez une type `Image` avec les champs suivants :
- l'entier correspondant à la largeur de l'image (nombre de pixels) ;
- l'entier correspondant à la hauteur de l'image (nombre de pixels) ;
- un pointeur vers la zone où sont stockées les données (la valeur des pixels).

2. Fonctions élémentaires de manipulation d'image

Dans un fichier `image_utils.c`, écrivez les fonctions suivantes :
- `Image *create_image(int width, int height)` : crée une variable de type `Image` et renvoie un pointeur vers cette variable en ayant alloué un espace mémoire pour stocker une image de largeur `width` et de hauteur `height` ;
- `void free_image(Image *p_image)` : libère la mémoire allouée par `create_image` ;
- `Image *copy_image(Image *p_image)` : copie une image en dupliquant la variable de type `Image` et les données de l'image ;
- `int sub2ind(int i, int j, int width)` : pour une image de largeur `width`, calcule et renvoie l'indice dans un tableau à une dimension correspondant au pixel situé ligne `i` et colonne `j`, ;
- `int ind2row(int k, int width)` : pour une image de largeur `width`, calcule et renvoie le numéro de ligne correspondant à l'indice `k` d'un pixel dans un tableau à une dimension ;

1. `http://www.lif.univ-mrs.fr/~vemiya/?page=L2C_2011_2012`

– `int ind2col(int k, int width)` : idem pour retrouver un numéro de colonne.

3. Fonctions de lecture/écriture d'image pgm

Nous travaillerons sur des images au format `pgm`, défini ainsi [2] :
– l'identifiant `P2` et un caractère d'espacement ;
– la largeur de l'image et un caractère d'espacement ;
– la hauteur de l'image et un caractère d'espacement ;
– la valeur maximale des niveaux de gris et un caractère d'espacement ;
– des entiers codant la valeur des pixels, ligne par ligne, séparés par un caractère d'espacement.

Les caractères d'espacement peuvent être des espaces, des fabulations et des nouvelles lignes. Un exemple de fichier pgm :

```
P2
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0  0  0  0  0 0 0  0  0  0  0
0 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0  0  0  0 0 15 0  0  15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0  0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0  0  0  0 0 15 0  0  0  0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 0  0  0  0
0 0 0 0 0 0 0 0 0 0 0 0 0 0  0  0  0  0 0 0  0  0  0  0
```

Dans un fichier `image_io.c`, écrivez les fonctions suivantes :
– `Image * pgm_read(char * filename)` : lit une image pgm à partir du nom du fichier `filename` ;
– `void pgm_write(char * filename, Image *p_image)` : écrit une image pgm dans le fichier `filename`.

Ces fonctions devront gérer la conversion des valeurs des pixels (entiers pour le format pgm, flottant entre 0 et 1 pour le type `Image`).

Pour faire des tests en affichant partiellement une image, vous pouvez ajouter, dans `image_utils.c`, une fonction `void affiche_image_ascii(Image * p_image)` : pour les pixels situés au maximum dans les 7 premières lignes et les 25 premières colonnes, elle affiche selon la valeur v du pixel : `' '` si $v = 0$, `'@'` si $v = 1$, `'+'` si $0 < v < 0.5$, `'#'` si $0.5 \leq v < 1$.

4. Effets élémentaires

Dans un fichier `image_effects.c`, écrivez les effets suivants, qui modifient directement l'image pointée par l'argument :
– `void binarization_effect(Image *p_image)` : arrondit la valeur de chaque pixel selon la valeur 0 ou 1 la plus proche.
– `void negative_effect(Image *p_image)` : crée le négatif en appliquant à chaque pixel l'opération $x \mapsto 1 - x$.

2. Ce format est probablement le plus simple à manipuler. Nous nous restreignons ici à un seul type de fichier pgm (pgm ASCII) et à une description incomplète du format (par exemple, on ne décrit pas la façon d'insérer des commentaires avec '#'). Cf. `http://fr.wikipedia.org/wiki/Portable_pixmap`.