Projet PROG 2006

Lecture, écriture et traitement d'images

Le but de ce projet est de développer quelques fonctions de manipulation d'images. Rappelons qu'une image peut être codée, soit par un tableau à deux dimensions (non recommandé ici), soit par un tableau unidimensionnel (fortement recommandé), contenant la juxtaposition des lignes de l'image. Chaque ligne contenant N pixels, l'image possédant M lignes, le vecteur en question est de dimension $N \times M$. Le projet se décompose en deux parties : écriture de fonctions d'éntrées/sorties permettant la lecture et l'écriture d'images au format `PGM` et écritures de fonctions de traitement d'images.

On rappelle ici que le plan image est défini dans un repère cartésien inversé, c'est à dire que l'axe des ordonnées (dit axe des lignes) est dirigé vars le bas. En revanche, l'axe des abscisses (dit axe des colonnes) est le même.
Pour chacune des fonctions demandées, on utilisera les pointeurs pour passer les adresses des images, filtres, histogrammes, entiers ... qu'on leur transmet. De même, si la fonction donne une image en résultat, cette image sera transmise à la fonction via un pointeur, puis modifiée à l'intérieur de cette fonction. D'une manière générale, on aura des définitions du type `void fonction(pointeurs sur entrées, pointeurs sur sorties)`. Il ne faut donc pas oublier les allocations de mémoire éventuels avant l'appel de fonction.

Travail à rendre pour le lundi 08 janvier 2007, minuit au plus tard

Vous devez fournir une archive `tar` compressée (vous lui donnerez comme nom le votre) contenant :
- un rapport dans lequel vous devez expliquer et justifier vos choix de programmation pour chaque question, faire des commentaires sur les résultats et répondre aux questions éventuelles. Toute initiative personnelle d'enrichissement du rapport sera également la bienvenue ;
- les sources de vos programmes : un par question. Ces sources doivent être clairement commentés.
- éventuellement un `Makefile` pour lancer les applications.

Partie 1 : lecture/écriture d'images au format `PGM`

Le format `PGM` est reconnu par la plupart des logiciels d'imagerie (`The Gimp`, `PaintShop Pro`, `Photoshop`, *etc.*). Un fichier `PGM raw` est constitué d'un entête `ASCII` suivi des octets de l'image parcourus ligne par ligne (dans notre cas, on se limitera à des valeurs comprises entre 0 et 255). L'entête a le format suivant :

```
P5
# Commentaires
# Commentaires
# ...
256 256
255
```

La première ligne est toujours `P5`, pour une image en niveaux de gris (on indique `P6` pour le format `PPM`, qui est un `PGM` couleur dans lequel chaque pixel est codé sur trois octets consécutifs représentant les composantes `RVB`). On trouve ensuite des lignes de commentaires (optionnelles), commençant par `#`. La ligne suivante donne les dimensions (largeur puis hauteur) de l'image. La dernière ligne contient la dynamique des valeurs. Ensuite le fichier contient les valeurs des pixels, dans leur ordre lexicographique dans l'image (i.e. de haut

en bas, de gauche à droite).

Les trois champs indispensables (dans le cadre de ce projet) pour caractériser une image sont sa largeur, sa hauteur et bien sur ses pixels. Aucune contrainte de programmation n'est donnée dans le cadre de ce projet : vous pouvez, soit utiliser une structure, contenant la largeur, la hauteur et les pixels de l'image, soit 2 entiers (largeur et hauteur) et un tableau (pixels) indépendants. Toutefois, vous devrez faire apparaître très clairement vos choix dans le compte-rendu.

Question 1 : lecture de format `PGM`

Écrire une fonction qui permet, à partir d'un fichier `PGM`, de lire une image. Il faudra donc récupérer les valeurs de largeur et hauteur ainsi que les pixels de l'image et les stocker dans des champs prévus pour.

Question 2 : écriture au format `PGM`

Écrire une fonction qui permet de sauvegarder une image dans un fichier au format `PGM`.

PARTIE 2 : traitements d'images

Maintenant que nous savons lire et écrire des images, nous pouvons effectuer des traitements sur celles-ci. Pour chacune des fonctions qui suivent, vous les testerez sur le jeu d'images fourni sur le site du Master et vous visualiserez le résultat avec un logiciel d'imagerie supportant le format `PGM`, tel que `The Gimp`.

Question 3 : seuillage binaire d'images

Le seuillage binaire d'images est un traitement visant à ramener le nombre de niveaux de gris d'une image à deux. Ainsi, pour chaque pixel $I(x, y)$ de l'image, ayant un niveau de gris k, le résultat du seuillage binaire k' de cette valeur est donné par :

$$k' = \begin{cases} k_1 & \text{si } k \leq S \\ k_2 & \text{si } k > S \end{cases}$$

où k_1, k_2 et S (seuil) sont des niveaux de gris.

Écrire une fonction qui effectue le seuillage binaire d'une image passée en arguments. Elle recevra en paramètres l'image d'entrée et l'image de sortie, ainsi que S, k_1 et k_2, définis selon l'équation plus haut.

Question 4 : inversion d'images

Le principe de l'inversion consiste à inverser les valeurs des pixels d'une image par rapport à l'intervalle de départ. Par exemple, pour une image en niveaux de gris (valeurs comprises entre 0 et 255), un pixel ayant initialement une valeur de 255 aura comme valeur 0, celui ayant pour valeur 254 aura pour valeur 1, *etc.*

Déterminer dans un premier temps la fonction mathématique d'inversion de valeurs dans un intervalle. Puis écrire la fonction qui permet d'inverser une image. Cette fonction recevra les images d'entrée et de sortie en paramètres.

Question 5 : histogramme d'une image

L'histogramme d'une image permet de donner la répartition des niveaux de gris d'une image. Il est défini comme suit :

$$H(k) = \text{Card}\{0 \leq x \leq N-1, 0 \leq y \leq M-1 : I(x, y) = k\} = n_k$$

L'histogramme est donc une fonction à une dimension qui se représente dans le plan cartésien tel que l'axe des abscisse contienne les niveaux de gris k (compris entre 0 et 255) et en ordonnée les valeurs $H(k)$ correspondant au nombre n_k de pixels dans l'image qui ont pour niveau de gris k.

Écrire une fonction qui, à partir d'une image, calcule son histogramme. Cette fonction recevra en paramètres l'image d'entrée et l'histogramme.

Question 6 : affichage d'histogramme

On veut représenter "graphiquement" l'histogramme à l'écran exactement de la manière suivante :

```
k=000 : *****
k=001 : ***************
...
k=255 : ***
```

où chaque * représente exactement 10 pixels. Sur l'exemple précédent, le niveau de gris $k = 0$ est représenté par entre 50 et 59 pixels dans l'image.

Écrire la fonction qui affiche l'histogramme qui lui est transmis en paramètres.

Question 7 : étirement d'histogramme

Dans le cas où l'intervalle de variation des niveaux de gris (dynamique) de l'image est réduit (on a alors un faible contraste), l'étirement d'histogramme permet de rétablir cette dynamique entre 0 et $(L-1)$ (par exemple $L = 256$). Ainsi, si les niveaux de gris de I appartiennent à l'intervalle $[a, b]$, et qu'on veut étirer cet intervalle à $[0, L-1]$, alors on a :

$$k' = \frac{L-1}{b-a}(k-a)$$

Ici, tous les pixels qui ont dans l'image originale pour valeur k, auront dans l'image destination la valeur k'.

Écrire une fonction qui effectue l'étirement de l'histogramme d'une image transmise. Il faudra donc dans un premier temps, calculer l'histogramme (se servir de la fonction écrite dans la question précédente), calculer l'histogramme étiré, puis calculer l'image après étirement d'histogramme. La fonction recevra en paramètres l'image d'entrée et l'image de sortie (après étirement d'histogramme).

Question 7 : filtrage par convolution

Le principe de la convolution d'image par un filtre sur une image, vis à, pour chaque pixel de l'image :

– centrer le filtre sur ce pixel ;

– calculer le produit de convolution entre les coefficients du filtre et les pixels qu'il recouvre dans l'image.

Soit une image I, et $I(x, y)$ l'intensité du pixel de coordonnées (x, y) dans le plan image (x est une colonne de l'image, et y une ligne). Ainsi, si on considère le filtre suivant $h = \begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{pmatrix}$ avec w_i les coefficients réels du filtre. La convolution $I'(x, y)$ au pixel (x, y) de I est donnée par :

$$\begin{aligned} I'(x,y) &= w_1 f(x-1, y-1) + w_2 f(x, y-1) + w_3 f(x+1, y-1) \\ &+ w_4 f(x-1, y) + w_5 f(x, y) + w_6 f(x+1, y) \\ &+ w_7 f(x-1, y+1) + w_8 f(x, y+1) + w_9 f(x+1, y+1) \end{aligned}$$

D'une manière générale, si on considère une image I de taille $N \times M$ (N colonnes et M lignes) et un filtre h de taille $d \times d$ (d impair), l'algorithme du filtrage par convolution de I par h est donné par :

/* Algorithme du calcul de convolution */
for $i = 0$ to M **do**
  for $j = 0$ to N **do**
    for $k = -\frac{d}{2}$ to $\frac{d}{2}$ **do**
      for $l = -\frac{d}{2}$ to $\frac{d}{2}$ **do**
        $f_y = k + \frac{d}{2}$
        $f_x = l + \frac{d}{2}$
        $I'[j + i * N] += I[j + i * N] * h[f_x + f_y * d]$
      end for
    end for
  end for
end for

Remarque 1. Avec les problèmes aux bords (cas ou des coefficient du filtre ne couvrent pas des pixels de l'image). Une solution est de ne pas traiter les pixels tels que, lorsque le filtre est centré sur eux, il recouvre des parties hors de l'image (ce qui correspond à une couronne de l'image dont l'épaisseur dépend de la taille du filtre.
Remarque 2. Les valeurs $I'(x, y)$ peuvent être réelles, négatives ou supérieure à 255 (en fonction de la valeur des coeficients du filtre). Dans ce cas, il faut rétablir la dynamique de l'image entre 0 et 255.
Remarque 3. On rappelle ici que l'on manipule, non pas des matrices ($2D$), mais des vecteurs qui correspondent à la juxtaposition des lignes de l'image. Cette représentation s'applique pour les images comme pour les filtres. Par exemple, pour le filtre défini plus haut, on manipule en fait un vecteur (tableau) $h = \left[\begin{array}{cccccccccc} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 & w_9 \end{array} \right]$. Ainsi, en `C` (les indices commencent à 0), l'accés au pixel de la ligne j et colonne i de l'image I est $I[j + Ni]$ (i.e. on passe i lignes de N pixels et les j premiers pixels de cette ligne).

Écrire une fonction qui, à partir d'une image et d'un filtre carré (de taille impaire), calcule l'image convoluée. Le fonction recevra en paramètres l'image d'entrée, le filtre et l'image de sortie (*i.e.* l'image filtrée).

On veut tester cette fonction avec le filtre $h_1 = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$, qu'on appelle aussi filtre moyenneur. Visualiser le résultat et dire ce que vous remarquez.

Mêmes questions avec un filtre h_2 de taille 5×5 donc chaque coefficient est $\frac{1}{25}$. Quelles différences constatez-vous sur l'aspect de l'image apresè le filtrage par h_1 et celui par h_2 ? Qu'en déduisez-vous ?