TD – Fichiers

ING1 – Programmation C

Génie Informatique

1 Images

Une image en couleur correspond à une matrice de pixels. Chaque pixel est composé de 3 composantes (rouge, vert et bleu) allant généralement de 0 à 255. La variation de l'intensité de chacune des composantes permet de générer une grande palette de couleurs. Le pixel est gris lorsque les 3 composantes ont la même intensités.

Le *portable pixmap file format* (PPM), le *portable graymap file format* (PGM) et le *portable bitmap file format* (PBM) sont des formats de fichier graphique en ASCII.

Les fichiers sont composés comme suit :

- Un nombre magique : P1 pour le fichier PBM, P2 pour le fichier PGM et P3 pour le fichier PPM
- Un caractère d'espacement (espace, tabulation, nouvelle ligne)
- Largeur de l'image
- Un caractère d'espacement
- Hauteur de l'image
- Un caractère d'espacement
- La valeur maximale utilisée pour coder les couleurs ou le gris, cette valeur doit être inférieure à 65536 (uniquement pour les images PGM et PPM)
- Un caractère d'espacement (uniquement pour les images PGM et PPM)
- Données ASCII de l'image :
  - L'image est codée ligne par ligne en partant du haut
  - Chaque ligne est codée de gauche à droite
  - Format PBM : Un pixel noir est codé par un caractère 1, un pixel blanc est codé par un caractère 0, précédées et suivies par un caractère d'espacement.
  - Format PGM : Chaque pixel est codé par une valeur, précédée et suivie par un caractère d'espacement. Un pixel noir est codé par la valeur 0, un pixel blanc est codé par la valeur maximale et chaque niveau de gris est codé par une valeur entre ces deux extrèmes, proportionnellement à son intensité.
  - Format PPM : Chaque pixel est codé par trois valeurs (rouge vert bleu), précédées et suivies par un caractère d'espacement.

— Aucune ligne ne doit dépasser 70 caractères.

— Toutes les lignes commençant par # sont ignorées.

```
P3
# The P3 means colors are in ASCII, then 3 columns and 2 rows,
# then 255 for max color , then RGB triplets
3 2
255
# Begining of the image
255 0 0
255 255 0
0
255
0
0 0 255
255 255 255
0
0
0
```

1 **Charger une image**

Définir les structures suivantes `sPixel`, `sImagePPM`, `sImagePGM` et `sImagePBM`.

Écrire les fonctions qui permettent de charger en mémoire une image PPM, PGM et PBM. □

2 **Niveau de gris**

Écrire une méthode qui permet de transformer une image couleur en une image en niveau de gris.

On utilisera la formule suitenace pour passer de la couleur en gris : $g = 0.299 \times r + 0.587 \times v + 0.114 \times b$. □

3 **Seuillage**

Á partir d'une image en niveau de gris, le seuillage d'image peut être utilisé pour créer une image comportant uniquement deux valeurs, noir ou blanc.

Le seuillage d'image remplace un à un les pixels d'une image à l'aide d'une valeur seuil fixée. Ainsi, si un pixel a une valeur supérieure au seuil, il prendra la valeur *blanc*, et si sa valeur est inférieure, il prendra la valeur *noir*.

Écrire la méthode qui permet de seuiller une image en niveau de gris. □

(4) **Sauvegarder une image**

Écrire les méthodes qui permettent de sauvegarder les images dans les différents formats. □

(5) **Programme principal**

Le programme utilisera les arguments afin de lancer le programme avec les bons choix :

-in *nomImage1* : précise l'image en entrée

-out *nomImage2* : précise l'image en sortie

-gris : transformation en niveau de gris

-seuil *valeur* : seuillage

L'ordre des options n'a pas d'importance. Lorsque l'option du seuillage est demandé et que c'est une image en couleur qui est donné, le niveau de gris est implicite. □

2 Pour aller plus loin ...

Dans le cadre du traitement d'image, la plupart des filtres (floutage, détection des contours, ...) utilisent des matrices de convolution. Les filtres diffèrent en fonction des valeurs dans la matrice de convolution.

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

(a) Augmente le contraste

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

(b) Flou

-2	-1	0
-1	1	1
0	1	2

(c) Repoussage

FIGURE 1 – Exemples de matrices de convolution

Une convolution est un traitement d'une matrice par une autre appelée matrice de convolution ou "noyau", que nous noterons par ⊗. Elle se fait par une sommation de multiplications. Par exemple :

$$\begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix} \otimes \begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix} = \sum_{i=1}^{9} p_i \times m_i$$

Dans le cas plus général, le filtre étudie successivement chacun des pixels de l'image. Pour chaque pixel, que nous appellerons "pixel initial", il multiplie la valeur de ce pixel et de chacun des 8 pixels qui l'entourent par la valeur correspondante dans le noyau. Il additionne l'ensemble des résultats et le pixel initial prend alors la valeur du résultat final [1]. Si la valeur est négative, on assignera 0 et si la valeur est

1. Cette opération se fera pour chacune des cimposantes couleurs de l'image.

suppérieur à la valeur maximale, on assignera cette dernière. De plus, quand le pixel initial est sur un bord, une partie du noyau porte en dehors des limites de l'image. On gèrera ce problème de bordure en assignant 0 à ces pixels.

114	158	46	43	74	141
202	39	63	67	21	123
87	113	146	131	234	93
10	188	67	216	224	32
151	71	36	213	18	191
245	132	94	36	175	168

$\otimes$

0	-1	0
-1	4	-1
0	-1	0

=

0	0	0	0	0	0
0	0	0	10	0	0
0	0	210	0	255	0
0	255	0	229	255	0
0	0	0	255	0	0
0	0	0	0	0	0

FIGURE 2 – Exemple de produit de convolution

(6) **Charger un noyau**

Écrire une méthode qui permet de charger en mémoire un noyau stocké dans un fichier. Le fichier sera constitué d'une première valeur qui correspond à la dimension de la matrice, puis des différentes valeurs du noyau. □

(7) **Produit de convolution**

Écrire la méthode `produitConvolution` qui, à partir de la matrice *image* et la matrice *noyau*, effectue le produit de convolution.

On les appellera avec l'option **-filtre *nom*** où *nom* correspond au nom du fichier du noyau de convolution. □