Travaux dirigés d'infographie n°5

Cours de synthèse d'image I

—IMAC première année—

Traitement d'Images - Filtres & Convolutions

Cette séance porte sur le traitement des images (Image Processing). Nous verrons d'abord comment manipuler les images au format `.pgm` puis nous aborderons les convolutions et notamment les filtres de flou moyen et de détection de contours.

▸ Exercice 1. *Chargement et sauvegarde d'une image*

Sur la page `http://igm.univ-mlv.fr/~boulengu/IMAC1/infographie.html` *se trouve l'image* `lena.pgm` *au format Portable Gray Map*[1]*:*

- *la première ligne contient un code indiquant que l'image est en niveaux de gris (*`P5`*),*
- *la deuxième ligne contient la longueur et la hauteur de l'image en pixels,*
- *la troisième ligne contient la valeur maximum d'un niveau de gris (généralement 255),*
- *enfin, les niveaux de gris des pixels sont renseignés, en binaire, les uns à la suite des autres, en partant du pixel en haut à gauche jusqu'au pixel en bas à droite.*

Si le niveau de gris maximum est 255, le niveau de gris d'un pixel est représenté par un `unsigned char`*.*

Voici un exemple de fichier `pgm`*:*

```
P5
152 65
255
@^@@]@ @!@&...
```

- *Écrire la fonction:*

```
unsigned char *ouvrir_image_pgm(char *nom_fichier, int *w, int *h)
```

[1] Le format PGM est décrit en détail sur la page http://netpbm.sourceforge.net/doc/pgm.html

qui renvoie l'image contenue dans le fichier `nom_fichier`*, sous la forme d'un tableau de taille* `w*h` *contenant les niveaux de gris des pixels. La fonction renvoie NULL en cas de problème.*

- *Écrire la fonction:*

```
int sauver_image_pgm(char *nom_fichier, unsigned char *image, int w, int h)
```

qui sauvegarde l'image `image` *dans un fichier PGM de nom* `nom_fichier`*. La fonction renvoie* 0 *en cas de problème,* 1 *sinon.*

- *Écrire la fonction:*

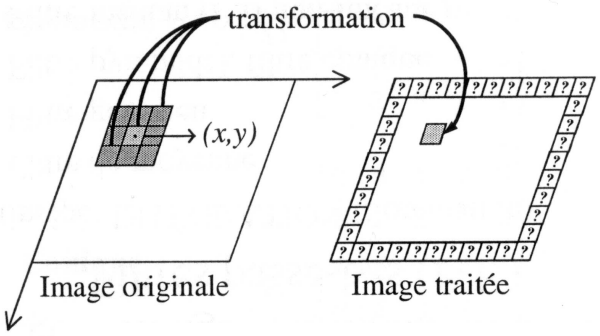
```
void miroir_image(unsigned char *image, int w, int h)
```

qui effectue une rotation 'miroir horizontal' de l'image `image`*.*

Vous devez bien évidement tester chacune de ces fonctions en les ouvrant avec un logiciel de dessin par exemple.

▸ **Exercice 2.** ***Filtres & Convolutions***

Un filtre linéaire d'image modifie la valeur d'un pixel en la remplaçant par une combinaison linéaire de sa valeur et de celles de ses voisins.



Les pixels voisins et leurs coefficients sont identifiés par un "masque". Dans la suite nous travaillons avec des masques de taille 3x3. Appliquer un filtre à une image, c'est réaliser une convolution de l'image avec le masque et le filtre est appelé "masque de convolution".

- *Comment stocker le masque ?*
- *Quels sont les pixels de l'image qu'il n'est pas possible de traiter ?*

- *Écrire une fonction qui renvoie une nouvelle image contenant l'image passée en paramètre convolué avec le filtre "moyenne locale" :*

$$\begin{vmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{vmatrix} = 1/9 \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix} \quad (1)$$

- *Écrire une fonction de convolution générique : elle fonctionne pour tout les masques 3x3 possibles. Tester les masques suivants :*
  - *Masque de flou gaussien :*

$$1/16 \begin{vmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{vmatrix} \quad (2)$$

  - *Masque de détection des contours (verticaux) (filtre de Sobel) :*

$$\begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{vmatrix} \quad (3)$$

  - *Masque de détection des contours (horizontaux) (filtre de Sobel) :*

$$\begin{vmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{vmatrix} \quad (4)$$

Pour les deux derniers masques, il faut "normaliser" le résultat...

▸ **Exercice 3.** ***Traitement d'images en*** *openGL*

Nous allons écrire une application glut pour afficher et sauvegarder une image filtrée. Pour remplir la zone de dessin avec un tableau de pixel, l'image, on peut utiliser les fonctions openGL suivantes :

```
glPixelStorei(GL_UNPACK_ALIGNMENT,1);
glDrawPixels(width,height,GL_LUMINANCE,GL_UNSIGNED_BYTE,image);
```

La fonction `glDrawPixels` *permet de remplir une zone rectangulaire de l'écran avec les données contenues dans le tableau pointé par* `image`. `width` *est la largeur de la zone,* `height` *sa hauteur. Un entier indique ensuite ce que sont les données du tableau: de la couleur, un niveau de gris... Enfin* `GL_UNSIGNED_BYTE` *indique le type des données du tableau.*

Attention : les pixels doivent être écris dans le buffer openGL à partir du coin 'bas à gauche' jusqu'au coin en 'haut à droite'. Vous utiliserez la fonction `image_miroir` *pour retourner les images.*

- *Écrire une application glut qui charge une image dont le nom est le premier argument au programme et l'affiche.*

- *La touche '`c`', affiche la convolution de l'image avec le masque de flou gaussien.*
- *On peut appliquer plusieurs fois la même convolution.*
- *La touche '`r`', permet d'annuler les changements.*
- *Un menu glut permet de choisir le filtre de convolution.*
- *Les filtres de détection de contours réalisent un "seuillage" : un pixel supérieur à un seuil devient blanc, s'il est inférieur il devient noir (on obtient ainsi une image binaire). Sachant que tous les filtres n'utilisent pas forcement des seuils, modifiez votre fonction de convolution afin qu'elle soit générique. Puis faites en sorte qu'à l'appui sur la touche '`+`', le seuil soit augmenté de 1 et sur la touche '`-`' qu'il diminue de 1.*