

TP Modèles Géométriques pour l'Image : Interprétation d'Images en Niveaux de Gris (PGM)

Dans ce TP vous mettrez en œuvre différentes notions et algorithmes que vous avez rencontrés dans le cours de *Modèles Géométriques pour l'image*, afin d'interpréter des images en niveaux de gris. Vous programmerez en C ou C++, et visualiserez vos images par un logiciel ad hoc tel que xv. Vous trouverez à l'adresse suivante les fichiers nécessaires à la réalisation de ce TP :

<http://liris.cnrs.fr/isabelle.sivignon/cours.html>

Vous rendrez un document rédigé exposant en quelques mots les solutions que vous avez proposées, illustrées par les images que vous avez obtenues.

Vous enverrez également aux enseignants vos sources dans une archive .tar.gz aux adresses suivantes :

Cedric.Gerot@gipsa-lab.inpg.fr Isabelle.Sivignon@gipsa-lab.grenoble-inp.fr

Vos programmes seront testés sur les images fournies ainsi que sur d'autres images afin de tester leur robustesse.

Préambule : Structures de données et entrées/sorties

Nous vous invitons à adopter la structure de données suivante pour la manipulation des images :

```
struct image {
    int largeur, hauteur, valmax ;
    int* buffer ;
}
```

où *largeur* et *hauteur* sont les dimensions de l'image, *valmax* est la plus grande valeur de niveaux de gris et *buffer* est un tableau mono-dimensionnel contenant les valeurs de niveaux de gris des *largeur*hauteur* pixels de l'image.

Les images manipulées seront au format PGM P2 (valeurs de niveaux de gris données en ASCII et non en binaire) :

```
P2
# image.pgm
17 3
6
0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 6
```

Cette image, nommée *image.pgm*, a une *largeur* de 17 pixels, une *hauteur* de 3 pixels, et la valeur maximale de niveaux de gris est 6.

Votre première tâche est d'écrire deux fonctions d'entrées/sorties :

```
struct image LireImagePGM( char* nomFichier ) ;

int EcrireImagePGM(char* nomFichier, struct image im) ;
```

La première prend en entrée un nom de fichier PGM de type P2 valide, en renvoie la structure *image* associée. En cas d'erreur, le champ *buffer* de la structure *image* renvoyée est égal à NULL.

La deuxième prend en entrée le nom du fichier *nomFichier* que l'on crée et la structure *image* dont on va écrire le contenu dans *nomFichier*. En cas d'erreur, 0 est renvoyé, 1 sinon.



Traitements élémentaires

Certains traitements élémentaires sur une image vous seront utiles pour répondre aux deux questions principales de ce TP. Nous vous en donnons une description sommaire dans les pseudo-algorithmes suivants ; à vous de les écrire convenablement (quitte à ne pas suivre ces pseudo-algorithmes à la lettre si vous trouvez une forme plus efficace !) :

```
int SeuillerImage( struct image* im, int seuil)
{
    im->valmax=0 ;
    parcours des éléments p de im->buffer
        si p < seuil alors
            p = 0
        sinon
            p = 255 et im->valmax=255
    si tout s'est bien passé, retourner 1, 0 sinon.
}
```

```
int NegatifImage( struct image* im)
{
    int max = im->valmax ;
    int->valmax=-1;
    parcours des éléments p de im->buffer
        p = max-p ;
        si int->valmax < p
            int->valmax=p ;
    si tout s'est bien passé, retourner 1, 0 sinon.
}
```

L'étiquetage en composantes connexes utilise un masque dépendant de la connexité choisie (indiquée par le paramètre d de la fonction *ComposantesConnexes*).

Les prédécesseurs d'un pixel P sont ainsi définis par : 4-connexité  8-connexité 

```
int ComposantesConnexes( struct image im, struct image* im_cc, char d)
{
    vérifier que *im_cc (c'est l'image des étiquettes) est une image créée aux mêmes dimensions que im
    // im[i,j] fait référence à l'élément im.buffer[i*im.largeur+j]

    int T[im.hauteur*im.largeur] ; //tableau d'équivalence des étiquettes initialisées à 0: T[k] contient l'étiquette
    la plus petite équivalente à k.

    int k=0 ;
    pour i = 0 à im.hauteur-1
        pour j = 0 à im.largeur-1
            si im[i,j] est non nul alors // pixel objet à étiqueter
                si tous les préd. de im[i,j] sont à 0 alors // nouvelle composante connexe
                    k++ ; im_cc[i,j] = k ; T[k]=k ;
                sinon
                    si tous les préd. non nuls de im[i,j] ont la même étiquette e dans im_cc alors
                        im_cc[i,j] = e ;
                    sinon // deux composantes connexes sont fusionnées en une seule
                        e = min{ T[im_cc[s]], s étant un préd. non nul de im[i,j]}
                        im_cc[i,j] = e ;
                        pour tous les préd. non nuls de im[i,j] et d'étiquette f telle que T[f] != e
                            tant que T[f] != e // propagation du minimum pour les étiquettes équivalentes
                                m=T[f] ; T[f]=e ; f=m ;

            pour i = 0 à im.hauteur-1
                pour j = 0 à im.largeur-1
                    im_cc[i,j] = T[im_cc[i,j]]

    si tout s'est bien passé, retourner 1, 0 sinon.
}
```

Exercice 1 : Reconnaître des objets différents dans une image PGM

On vous fournit l'image objets.pgm suivante :



Il s'agit d'écrire un programme qui permette :

1. d'indiquer le nombre d'objets contenus dans cette image
2. pour chaque objet, indiquer le nombre de trous
3. pour chaque objet, quelques caractéristiques géométriques (épaisseur moyenne, longueur...)

Pour cela, vous serez probablement amenés à utiliser les traitements élémentaires suivants :

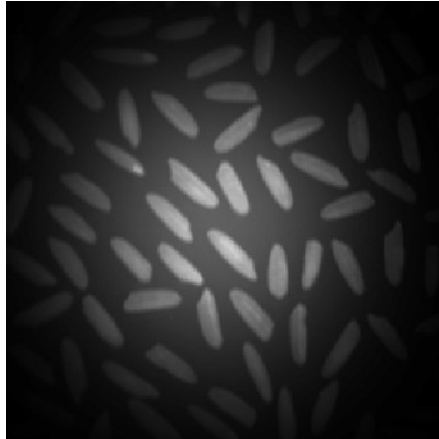
- seuillage
- négatif
- étiquetage en composantes connexes

ainsi qu'un des deux squelettes discrets vus en cours :

- squelette binaire
- axe médian

Exercice2 : Compter un nombre de grains sur une image mal éclairée

On vous fournit l'image riz.pgm suivante :



Il s'agit d'écrire un programme qui permette de compter le nombre de grains de riz dans cette image.

Pour cela, vous serez probablement amenés à utiliser les traitements élémentaires suivants :

- seuillage
- étiquetage en composantes connexes

ainsi que les opérateurs morphologiques vus en cours :

- dilatation
- érosion
- ouverture
- fermeture

Si vous en avez le temps, vous pourrez donner des caractéristiques géométriques moyennes sur l'ensemble des grains de riz, en composant les deux programmes que vous venez d'écrire.