# 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://gipsa-lab.grenoble-inp.fr/~isabelle.sivignon/enseignement.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.grenoble-inp.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 entrés/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 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 sont préliminaires à la suite des Tps. Voici les spécifications de ces fonctions. A vous de les écrire convenablement !

**1. Seuiller une image** : étant donnés une image et un seuil, met à 0 tous les pixels de valeur inférieure au seuil, et à *valmax* ceux de valeur supérieure.

**int** SeuillerImage( **struct** image* im, int seuil)

**2. Négatif d'une image** : étant donnée une image, remplace la valeur *p* d'un pixel par *valmax-p.* Pensez à remettre *valmax* à jour.

**int** NegatifImage( **struct** image* im)

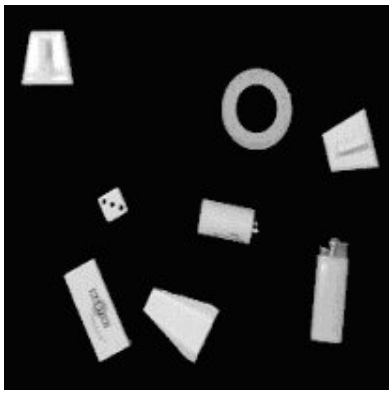
**3. Calcul des composantes connexes d'une image** : étant donnée une image binaire, étiqueter les pixels de l'image en composantes connexes. Le résultat sera stocké sous la forme d'une nouvelle image. Reportez-vous au cours pour le détail de l'algorithme. L'algorithme prend aussi en paramètre la connexité choisie pour l'image.

**int** ComposantesConnexes( **struct** image im, **struct** image* im_cc, **char** d)

**4. Calcul du bord d'un objet** : étant donnée une image contenant un objet, calculer le bord de cet objet. Le bord sera codé par son code de Freeman. L'algorithme prend aussi en paramètre la connexité choisie pour l'image.

## 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 :

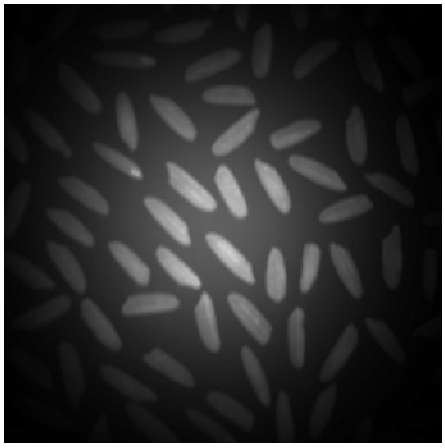
seuillage
négatif

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.