

## PROJET 1 - PROJETS EN C L3IF - ENS LYON 2010-2011

BOGDAN PASCA <BOGDAN.PASCA@ENS-LYON.FR>  
MARIN BOUGERET <MARIN.BOUGERET@IMAG.FR>  
KEVIN PERROT <KEVIN.PERROT@ENS-LYON.FR>

### INTRODUCTION

- Choisir un projets à réaliser par groupe de 2 personnes au maximum.
- **Rapport (1 à 2 pages au format PDF)** : Présenter votre approche et expliquer vos choix d'implémentation. Expliquer comment se servir du programme. Dire quelques mots sur les avantages et les désavantages du C pour ce projet.
- **Conseils** : Penser à la propreté du code, découper le projet en fonctions claires et bien commenter les sources.
- **Rendu** : à kevin.perrot@ens-lyon.fr
- **Deadline session 1** : **Vendredi 15 octobre 2010 23h59**
- **Deadline session 2** : **Vendredi 26 novembre 2010 23h59**
- **Deadline session 3** : **Mardi 11 janvier 2011 23h59**

## 1. AUOMATES CELLULAIRES (À LA WOLFRAM) ★

## 1.1. Présentation.

Nous utiliserons la définition suivante.

Un automate cellulaire en une dimension consiste en une suite fini de  $N$  cellules contenant chacune un *état* (0 ou 1). L'état de la cellule d'indice  $i$  au temps  $t + 1$  est fonction de l'état des cellules  $i - 1 \bmod[N]$ ,  $i$  et  $i + 1 \bmod[N]$  au temps  $t$  (on considère une structure torique, c'est à dire que le voisin gauche de la cellule la plus à gauche est la cellule la plus à droite et inversement).

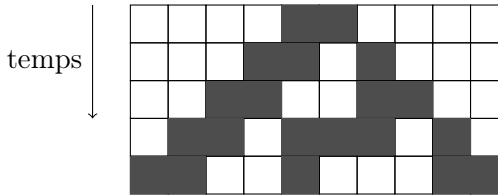
Chacune des cellules pouvant prendre deux états, il existe  $2^3 = 8$  configurations (ou motifs) possibles d'un tel voisinage. Pour que l'automate cellulaire fonctionne, il faut définir quel doit être l'état, à la génération suivante, d'une cellule pour chacun de ces motifs. Il y a  $2^8 = 256$  façons différentes de s'y prendre, soit 256 automates cellulaires différents de ce type.

Les automates de cette famille sont dits "élémentaires". On les désigne souvent par un entier entre 0 et 255 dont la représentation binaire est la suite des états pris par l'automate sur les motifs successifs 111, 110, 101, etc.

Par exemple l'automate cellulaire élémentaire 30 (00011110 en binaire) est défini par les règles de transition suivantes :

111	110	101	100	011	010	001	000
0	0	0	1	1	1	1	0

Les 4 premières itérations pour  $N = 10$  et la configuration initiale 0000110000 donnent :



## 1.2. Projet.

Ecrire un simulateur d'automate cellulaire dans lequel on peut choisir :

- la règle de 0 à 255 ;
- le nombre  $N$  de cellules ;
- la configuration initiale ou une configuration aléatoire ;
- le nombre d'itérations à afficher.

La sortie peut être présentée sous forme de lignes de 0 et 1 représentant les configurations successives. Expliquer dans le rapport comment se service du simulateur.

## 2. DÉTECTION DE CONTOURS ★★

## 2.1. Présentation.

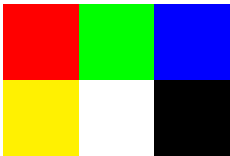
Le but de ce projet est de réaliser un programme transformant une image couleur au format PPM en une image noir et blanc au format PBM représentant ses contours.

## 2.2. Les formats.

## 2.2.1. PPM - extension ".ppm".

Voici un exemple d'image au format PPM (à gauche le source, à droite le rendu)

```
P3
3 2
255
255 0 0 0 255 0 0 0 255
255 255 0 255 255 255 0 0 0
```



- En première ligne on trouve la version du format utilisée, P3.
- En seconde ligne on a le nombre de lignes et le nombre de colonnes séparés par un espace.
- En troisième ligne on voit la valeur maximale d'une composante RVB.
- Ensuite on lit des triplets d'entiers représentant les composantes Rouge Vert et Bleu d'un pixel. L'organisation théorique est la suivante : sur une ligne se suivent les composantes RVB de tous les pixels de la ligne. **Attention**, dans la pratique il vaut mieux confondre les retours à la ligne et les espaces car on peut trouver des retours à la ligne tous les 4 pixels qui ne correspondent pas à la fin d'une ligne de l'image.

Pour convertir une image *image.extension* en PPM version P3 avec la suite *ImageMagick* on peut utiliser la commande

```
convert image.extension -compress none image.ppm
```

2.2.2. PBM. Voici un exemple d'image au format PBM (à gauche le source, à droite le rendu)

P1					
5 7					
0 0 0 0 0					
0 1 0 1 0		■		■	
0 1 0 1 0		■		■	
0 0 0 0 0					
1 0 0 0 1	■				■
0 1 1 1 0		■	■	■	
0 0 0 0 0					

- En première ligne on trouve la version du format utilisée, P1.
- En seconde ligne on a le nombre de lignes et le nombre de colonnes séparés par un espace.
- Ensuite on lit des 1 pour noir et 0 pour blanc. Une ligne du source code une ligne de pixels, les lignes sont décrites de haut en bas.

### 2.3. L'algorithme.

On utilisera l'algorithme de Canny simplifié avec mesure de Prewitt suivant

- (1) On calcule les mesures de Prewitt  $G_x$  et  $G_y$  de chaque pixel

$$\begin{aligned} \blacksquare G_x &= \frac{1}{3} \sum_{c \in \{R,V,B\}} \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \begin{bmatrix} a_{i-1,j-1} & a_{i,j-1} & a_{i+1,j-1} \\ a_{i-1,j} & a_{i,j} & a_{i+1,j} \\ a_{i-1,j+1} & a_{i,j+1} & a_{i+1,j+1} \end{bmatrix} \\ \blacksquare G_y &= \frac{1}{3} \sum_{c \in \{R,V,B\}} \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * \begin{bmatrix} a_{i-1,j-1} & a_{i,j-1} & a_{i+1,j-1} \\ a_{i-1,j} & a_{i,j} & a_{i+1,j} \\ a_{i-1,j+1} & a_{i,j+1} & a_{i+1,j+1} \end{bmatrix} \end{aligned}$$

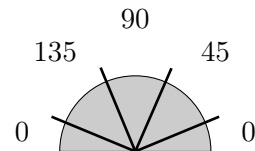
pour un pixel en position  $a_{i,j}$ , où  $*$  est<sup>1</sup> définie pour deux matrices  $A$  et  $B$  de taille 3 par 3 par  $A * B = \sum_{i,j \in \{0,1,2\}} a_{i,j} b_{i,j}$ .

Si un pixel du voisinage est en dehors de l'image d'origine, sa valeur est considérée comme nulle.

- (2) On calcule le gradient  $G$  et la direction  $\Theta$  de l'arête en chaque pixel

$$\blacksquare G = \sqrt{G_x^2 + G_y^2}$$

$$\blacksquare \Theta = \arctan\left(\frac{G_x}{G_y}\right), \text{ en arrondissant la direction à un des 4 angles : } 0, 45, 90, 135.$$



- (3) On sélectionne les arêtes maximales (appelées *arêtes maigres*)

1. En imagerie, on parle de convolution.

- Si  $\Theta = 0$ , l'arête est sélectionnée si son gradient est supérieur à celui de ses voisins **est** et **ouest**.
- Si  $\Theta = 45$ , l'arête est sélectionnée si son gradient est supérieur à celui de ses voisins **nord-est** et **sud-ouest**.
- Si  $\Theta = 90$ , l'arête est sélectionnée si son gradient est supérieur à celui de ses voisins **nord** et **sud**.
- Si  $\Theta = 135$ , l'arête est sélectionnée si son gradient est supérieur à celui de ses voisins **nord-ouest** et **sud-est**.

On obtient une image binaire des arêtes.

- (4) On applique un seuillage à hystérésis pour faire le tri dans les arêtes obtenues. On définit un seuil haut et un seuil bas (il n'existe pas de méthode générique pour déterminer les seuils, faites des essais...) et pour chaque point détecté précédemment comme une arête, si son gradient est
- Supérieur au seuil haut, le point est accepté comme formant un contour ;
  - Inférieur au seuil bas, le point est rejeté ;
  - Entre le seuil bas et le seuil haut, le point est accepté s'il est connecté (par le *voisinage de Moore*) à un point déjà accepté.

Cette dernière étape peut nécessiter plusieurs passes.

- (5) On peut proposer en option d'élargir en nombre de pixels les contours obtenus pour un meilleur rendu.

#### 2.4. **Projet.**

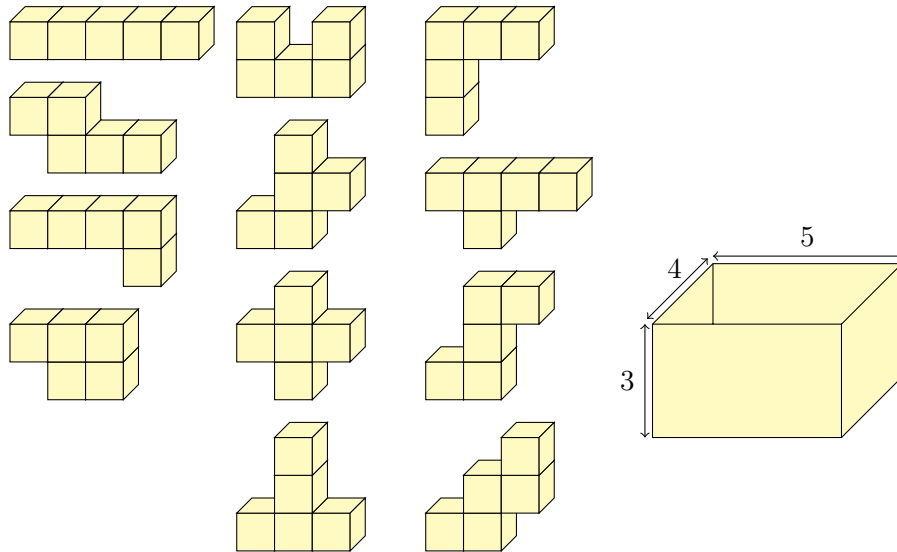
Implémenter cet algorithme de détection de contour, prenant en entrée un fichier au format PPM et retournant un fichier au format PBM.

- Vous pouvez apporter des modifications à l'algorithme en précisant leur motivation dans le rapport.
- Joindre à votre rapport quelques exemples d'images traitées.

## 3. CASSE-TÊTE ★★★

## 3.1. Présentation.

Ce casse tête en 3D consiste à trouver comment faire tenir 12 pièces de bois dans une boîte.



## 3.2. Projet.

Ecrire un programme pour résoudre ce casse-tête qui donne la solution sous une forme intelligible. N'oubliez pas d'expliquer le codage dans le rapport. Proposer une ou plusieurs stratégies.

**Bonus :** Donner la solution dans un fichier si vous l'avez trouvée.