TP n° 2

Programmation OpenMP

L'adresse email pour l'envoi des comptes rendus à utiliser est `pablo.de-oliveira-castro@cea.fr`. Le compte rendu de TP est à fournir au plus tard le 12 décembre. Dans le compte rendu, inclure les codes ainsi que les traces d'exécution.

Dans ce TP vous mettrez en pratique les bases de programmation OpenMP. Le sujet est divisé en deux parties. La partie I vous permettra de vérifier que la chaîne de compilation OpenMP marche correctement. Dans la partie II on s'intéressera à la parallélisation d'un chaîne de traitement d'image.

I) Préliminaires

Exercice 1 [**Hello World OpenMP**]

Écrivez un programme OpenMP qui affiche pour chaque thread le message : "`Hello from thread : <rang du thread courant>`". Le processus de rang 0 devra afficher également le nombre total de processus crées.

Pour compiler un programme OpenMP on utilisera la commande :

```
gcc -fopenmp -o hello_world hello_world.c
```

II) Extraction de droites dans une image

Il existe diverses méthodes pour extraire les droites d'une image. Ici nous proposons une méthode utilisant une des variantes de l'algorithme de Hough (proposé par P. Hough en 1962).

L'idée est que chaque point de l'image vote pour l'ensemble des droites du plan qui pourraient le traverser. Une fois que tous les points ont voté, les droites ayant le score maximum correspondent aux droites les plus probables dans l'image.

Pour que l'algorithme de Hough soit efficace on n'exécute celui-ci que sur les points correspondant aux contours des objets dans l'image, c'est à dire les points à gradient maximal. Pour repérer les contours utilisera l'algorithme de Sobel (1968).

Le but de cette partie c'est d'implémenter ces deux algorithmes et des les paralléliser en utilisant OpenMP.

a) Lecture et écriture des images

Nous travaillerons sur des images en 255 niveaux de gris, stockées en format pgm plain (P2). Récupérer les fichiers pgm.h, pgm.c et train.pgm à l'adresse http ://www.sifflez.org/tp-openmp-fichiers.tar.gz

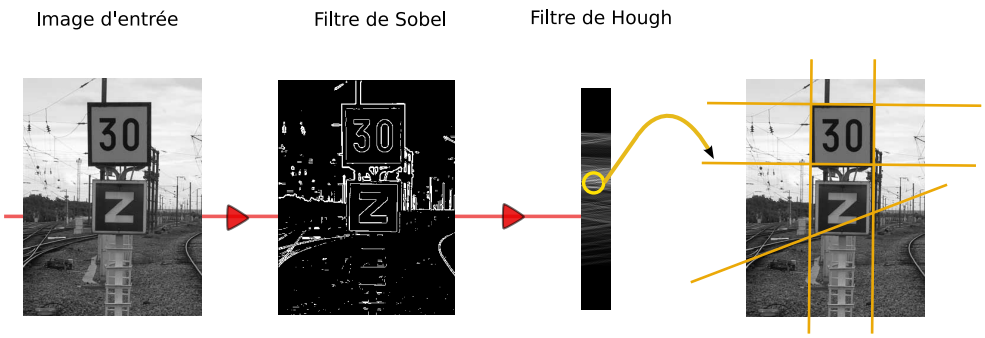

FIG. 1 – Extraction des droites (photographie de Aurélien Braida).

pgm.h permet de charger, modifier et écrire des images pgm. On charge une image en utilisant la fonction :

```
image * read_pgm(char * file);
```

Cette fonction nous retourne une structure qui contient :
- `w,h` les dimensions de l'image
- `colors` le nombre de niveaux de gris
- `*img` un tableau dont la case $x*h+y$ contient la couleur du pixel (x, y), c'est à dire un entier entre 0 et *colors*.

On dispose également de la fonction :

```
void write_pgm(char * file, image * im);
```

qui nous permet d'écrire l'image `im` vers le fichier `file`.

Exercice 2 [**Charger une image et la modifier**]
Écrire un programme en utilisant l'interface proposée dans pgm.h qui charge en mémoire l'image train.pgm, normalise l'image à une valeur passée en paramètre, et l'écrit sur un fichier de sortie.

b) Filtre de Sobel

Maintenant que nous savons lire et écrire des images nous pouvons enfin entrer dans le vif du sujet. Il s'agit tout d'abord de trouver les bords des objets dans l'image que l'on notera I.

Pour cela on calcule les gradients horizontaux et verticaux (deux filtres convolutifs) :

$$G_x = \begin{vmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{vmatrix} * I$$

$$G_y = \begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix} * I$$

Autrement dit, G_x (respectivement G_y) est une matrice de la taille de l'image, dont chaque point (x, y) est défini par :

$$\begin{array}{rcccccc} G_x(x,y) = & 1*I(x-1,y-1) & + & 0*I(x,y-1) & - & 1*I(x+1,y-1) \\ + & 2*I(x-1,y) & + & 0*I(x,y) & - & 2*I(x+1,y) \\ + & 1*I(x-1,y+1) & + & 0*I(x,y+1) & - & 1*I(x+1,y+1) \end{array}$$

Une fois obtenus les gradients on peut calculer la sortie du filtre de Sobel en tout point de la manière suivante :

$$S(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

Exercice 3 [**Parallélisation de l'algorithme de Sobel**]
Écrire l'algorithme de Sobel. Proposer au minimum deux méthodes de parallélisation, les comparer et implémenter celle qui vous parait la meilleure.

c) Algorithme de Hough

L'algorithme de Hough a pour sortie un espace de taille $WH * RMAX$. (Où $WH = 100$ et $RMAX = \sqrt{w^2 + h^2}/2$).

Les abscisses sont notées θ et les ordonnées r. Le point $H(\theta, r)$ correspond à la droite faisant un angle θ par rapport à l'horizontale et de distance à l'origine r.

Avec cette représentation l'algorithme de Hough consiste à :

1. On initialise à zéro l'espace de sortie de $H(\theta, r)$.
2. Pour tous les points (x, y) tels que $S(x, y) > seuil$, c'est à dire les bords :
   (a) Pour WH échantillons de θ entre 0 et $\pi/2$:
      i. On calcule : $r = (x - w/2) * cos(\theta) + (y - h/2) * sin(\theta)$
      ii. On incrémente $H(\theta, r)$.

Exercice 4 [**Parallélisation de l'algorithme de Hough**]

1. Écrire l'algorithme de Hough.
2. Proposer au minimum deux méthodes de parallélisation, les comparer et implémenter celle qui vous parait la meilleure.
3. Votre parallélisation garantit-elle la cohérence des données lors de l'incrémentation de $H(\theta, r)$?
4. Expliquez le mécanisme mis en place pour garantir cette cohérence.

d) Bonus

Par l'algorithme de votre choix extrayez les zones les plus lumineuses en sortie de l'algorithme de Hough, elle correspondent aux droites les plus probables dans l'image source. Tracez ces droites sur l'image d'origine pour vérifier le bon fonctionnement de la chaîne.