M02 – Licence Pro ASSUR
Mini-projet de programmation

Camille Coti – camille.coti@iutv.univ-paris13.fr

2012-2013

*Ce mini-projet est à rendre pour le **15 novembre 2012** avant 23:59 (heure de Paris). Tout jour de retard sera comptabilisé par un point retranché à la note obtenue. La note de ce mini-projet pourra être utilisée comme note de contrôle final. Si vous le souhaitez, vous pourrez passer également le contrôle final : la meilleure des deux notes sera retenue.*

1 Présentation

Le but de ce mini-projet est d'implémenter un programme de traitement d'image qui applique un algorithme de détection de contour sur une image lue dans un fichier d'entrée et écrit le résultat dans un fichier de sortie.

2 Format de fichiers d'images

Les fichiers manipulés sont au format PBM (Portable BitMap) ou PGM (Portable GrayMap). Il s'agit d'un format non compressé, où l'image est décrite pixel par pixel. Chaque pixel est représenté par un octet indiquant sa couleur : il n'y a dont que 256 valeurs possibles, soit 256 couleurs pour le PBM et 256 niveaux de gris pour le PGM.

Le format de ce type de ce fichier est le suivant :

- Un "magic number" : P1 ou P4 pour PBM, P5 pour PGM
- Un séparateur (tabulation, retour à la ligne, espace...)
- La largeur de l'image, en nombre de pixels, en décimal
- Un séparateur (tabulation, retour à la ligne, espace...)
- La hauteur de l'image, en nombre de pixels, en décimal
- Un séparateur (tabulation, retour à la ligne, espace...)
- Pour PGM : la valeur maximale possible dans cette image
- Pour PGM : un séparateur (tabulation, retour à la ligne, espace...)
- Les lignes, données pixel par pixel, en utilisant un octet pour représenter chaque pixel
- Les lignes commençant par # sont considérées comme des commentaires et ignorées

Vous pouvez convertir une image vers le format PBM en utilisant un logiciel de traitement d'image (par exemple GIMP) ou plus simplement, l'utilitaire en ligne de commande `convert` :

```
$ convert monimage.png monimage.pbm
```

Votre programme devra comporter une fonction de lecture du fichier d'entrée, prenant en paramètre le nom du fichier, lisant le contenu d'un fichier respectant le format PBM ou PGM et

retournant un tableau à deux dimensions contenant des octets (type `byte`) constituant l'image lue. Il doit supporter les deux formats PGM et PBM.

En cas de problèmes lors de la lecture de l'image, votre fonction devra soulever une exception :

- S'il s'agit d'une exception renvoyée par une fonction de lecture que vous appelez, celle-ci doit être remontée à l'appelant (votre fonction de lecture relève l'exception levée par la fonction appelée).
- S'il s'agit d'une erreur parce que le fichier que vous lisez est mal formé, vous devez lever une exception spécifique écrite par vos soins.

Votre programme devra également comporter une procédure qui écrit l'image résultant du calcul dans un fichier de sortie. Cette procédure reçoit comme paramètres le tableau d'octets contenant l'image à écrire et le nom du fichier dans lequel le programme doit écrire l'image de sortie. Les exceptions soulevées par les fonctions d'écriture appelées doivent être re-levées pour être remontées à l'appelant.

Le fichier de sortie doit être au format PBM ou PGM décrits ci-dessus. Le "magic number" de l'image de sortie doit être identique à celui de l'image d'entrée.

3 Algorithme de Sobel

L'algorithme de Sobel est un algorithme de traitement d'images utilisé pour la détection de contours dans une image. Il fonctionne en traitant l'image pixel par pixel, et en tenant compte des pixels situés immédiatement autour de chaque pixel.

Pour cela, il applique deux filtres successifs, appelés *opérateurs de Sobel*, qui approximent le gradient d'intensité dans la zone entourant le pixel. Les deux opérateurs de Sobel sont des matrices de dimension 3×3 contenant des poids, représentés figure 1. Le gradient d'un pixel est approximé en prenant la moyenne des valeurs du pixel dans l'image d'entrée et des 8 pixels environnant, pondérée par les poids de l'opérateur appliqué.

+1	0	-1
+2	0	-2
+1	0	-1

(a) Opérateur horizontal G_x

+1	+2	+1
0	0	0
-1	-2	-1

(b) Opérateur vertical G_y

Figure 1: *Opérateurs de Sobel*

La valeur du pixel de l'image de sortie est égale à la somme des valeurs obtenues en appliquant les deux gradients G_x et G_y au pixel de l'image d'entrée.

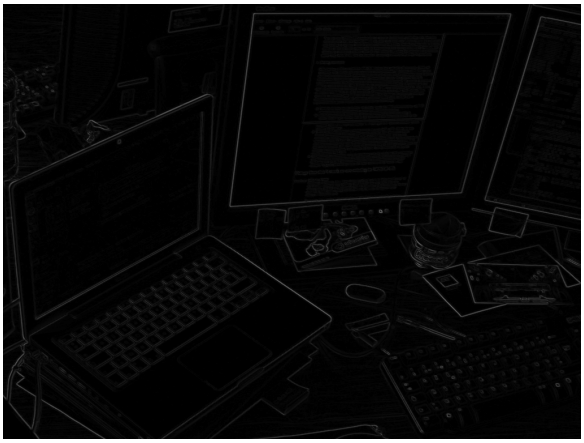
En passant l'algorithme de Sobel sur chaque pixel de l'image d'entrée, on obtient une image de sortie. C'est cette image calculée que l'on écrit dans le fichier de sortie.

La figure 2 montre un exemple obtenu avec l'algorithme de Sobel, où l'on voit que l'image 2(b) ne comporte en clair que les contours des objets de la figure 2(a).

Le travail demandé ici consiste à appliquer l'algorithme de Sobel sur l'image lue depuis le fichier d'entrée, puis d'écrire le résultat dans le fichier de sortie.

Vous pourrez bien entendu vérifier la correction de votre programme en regardant le fichier de sortie.

(a) Image d'entrée



(b) Image de sortie (détail)

Figure 2: *Résultats calculé par l'algorithme de Sobel : image d'entrée et image de sortie*

4 Exécution et livraison du travail

Vous devrez m'envoyer par email une archive compressée (gzippée) contenant :

- Un petit rapport (maximum 8 pages) décrivant votre travail et les choix d'implémentation que vous y avez fait
- Le code source de votre travail (fichiers .py)

Le code doit pouvoir être exécuté par un interpréteur Python "récent" (2.4, 2.6 ou 3.x). Si vous utilisez des fonctionnalités spécifique à un interpréteur, merci de préciser la version à utiliser.

Si le script comprenant le module principal (le point d'entrée dans le programme) s'appelle sobel.py, votre programme devra s'exécuter des deux façons suivantes :

- Soit en fournissant le chemin vers le fichier d'entrée et le fichier de sortie, dans l'ordre suivant :

```
$ ./sobel.py in.pgm out.pgm
```

- Soit en ne fournissant quel le nom du fichier d'entrée, auquel cas le fichier de sortie sera appelé out_<fichier d'entrée> et on appelera le programme de la façon suivante :

```
$ ./sobel.py in.pgm
```

- Si aucun fichier n'est passé (le programme étant appelé sans paramètre), il doit afficher un message d'erreur donnant la syntaxe correcte.