HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Alberto Dassatti, Bruno
Da Rocha Carvalho

Laboratoire de High Performance Coding

semestre printemps 2024

High Performance Coding (HPC)

Temps à disposition: 4 périodes (deux séances de laboratoire)

Récupération du laboratoire dans l'archive sur Cyberlearn

1 Objectifs de ce laboratoire

Vous allez écrire en langage C tout au long de ce cours. Ce laboratoire vise à vous remettre dans le bain et surtout à vous créer une ligne de base pour les futurs travaux pratiques.

2 Convolution

Il vous est proposé de mettre en place un traitement d'images appliquant des filtres de convolution pour déterminer les contours composant une image. Le filtre utilisé dans ce laboratoire est l'un des plus simples qui soit : le filtre de Sobel.

Le programme met à votre disposition des fonctions qui vous permettront de charger en mémoire une image RGB(A)/niveaux de gris au format PNG ainsi que d'enregistrer ces images sur le disque.

Pour appliquer la détection de contours, vous devez :

- Convertir l'image en niveaux de gris.
- Appliquer un filtre gaussien pour adoucir les contours de l'image. Cette opération permettra de supprimer le bruit et d'obtenir de meilleurs résultats.
- Appliquer le filtre de Sobel pour ne garder que les contours de l'image.

Les filtres (gaussien et Sobel) utilisent des matrices (ou masques) de convolution. L'idée de la convolution est que pour chaque pixel de l'image de départ, on accumule la valeur pondérée des voisins immédiats à la valeur pondérée de lui-même. La pondération est déterminée par la matrice de convolution qui, centrée sur le pixel de départ, agit comme un masque.

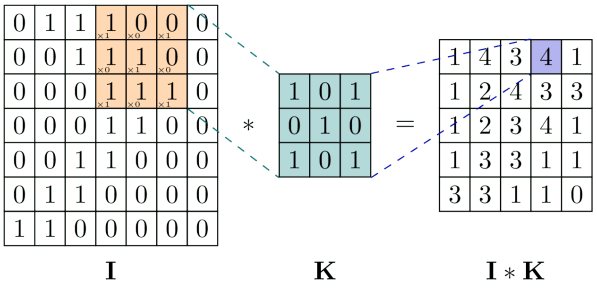

Figure 1: Exemple d'une convolution avec un masque

Le filtre gaussien est relativement simple à utiliser : une fois la valeur d'accumulation calculée, celle-ci est multipliée par un facteur de normalisation, puis le résultat est stocké dans la nouvelle image à la même position que le pixel d'origine.

Le filtre de Sobel consiste à calculer deux valeurs d'accumulation que l'on nomme "gradient" horizontal et vertical. Il y a donc deux masques de convolution à utiliser. Intuitivement, ces gradients représentent le taux de variation dans une direction horizontale/verticale de l'intensité des voisins d'un pixel. Ensuite, la norme dans la vraie direction du gradient est calculée. Si la norme dépasse un seuil prédéfini, alors on applique un pixel blanc à l'image finale, sinon un pixel noir.

3 Travail à faire

Votre tâche consiste à implémenter les fonctions présentes dans le fichier `sobel.c`. Vous remarquerez que certaines fonctions sont dupliquées. Vous devez vous intéresser à implémenter ces fonctions pour le traitement de deux structures de données différentes : une première `img_1D_t` qui contient un tableau C à une dimension et une seconde qui contient une liste chaînée de pixel `img_chained_t`. *Attention le premier pixel de la liste chaînée et celui en bas à droite.*

Voici ce qui doit être présent dans chacune des fonctions que vous devez développer :

- `rgb_to_grayscale`: Prenez un à un les pixels et stockez-les dans une nouvelle image. Pour la conversion, vous devrez multiplier chaque composante R, G et B par leur facteur respectif et les additionner les uns aux autres pour obtenir une nuance de gris.
- `gaussian_filter`: Appliquez le filtre gaussien sur les pixels qui ne sont pas présents dans les bords, pour ceux dans les bords, copiez simplement leur valeur actuelle. Ensuite, multipliez chaque pixel par la matrice et additionnez leurs valeurs, puis finalement divisez la valeur totale par la pondération gaussienne proposée.
- `sobel_filter`: Appliquez le filtre de Sobel sur les pixels qui ne sont pas présents dans les bords, pour ces derniers, copiez simplement leur valeur actuelle. Ensuite, sur chaque pixel, appliquez la matrice horizontale (résultat H) et verticale (résultat V) comme pour l'application d'un filtre gaussien, c'est-à-dire la somme de chaque pixel multiplié par la matrice superposée. Pour décider si un pixel est un "bord", il vous suffira de calculer la norme des résultat H et V et de les comparé au seuil. Si cette dernière est supérieure au seuil donné dans le code, vous pourrez définir le pixel comme noir, sinon blanc.
- `edge_detection`: Fonction qui va instancier la logique, les allocations, les appels aux fonctions et retourner le résultat de la détection de bords.

4 Indications

Le makefile fourni vous permettra de construire un binaire : `lab01`. Il prends en paramètres le chemin vers un fichier d'entrée sous format PNG et un chemin vers l'image finale. Par example `./lab01 img_src img_dest`.

5 Mesue du temps d'exécution

Lorsque l'on cherche à connaître la performance d'un programme, le premier critère d'évaluation à observer est son temps d'exécution. Un outil que vous avez à connaître pour évaluer ce critère est /usr/bin/time. Nous vous invitons à lire sa documentation et à l'exécuter sur votre programme pour le prendre en main. N'hésitez pas a essayer d'utiliser hyperfine également, un outil bien plus

performant. Une fois que vous savez mesurer le temps d'exécution, nous allons nous intéresser à l'évolution des temps d'exécution des deux implémentations en fonction de la taille des données fournies. Choisissez des images de tailles différentes, exécutez les deux implémentations, récupérer les temps d'exécution puis générez un graphe comparant les deux algorithmes. Analysez vos résultats et concluez.