Projet CUDA - Traitement d'images

L'objectif est de tester les performances d'algorithmes de traitement d'images. Le rendu sera constitué d'une archive contenant les différents codes ainsi qu'un rapport (2-3 pages) contenant une description de vos codes, de leurs performances (tableaux ou graphes), et des difficultés rencontrées.

Consignes :

- Gestion des erreurs : les appels aux fonctions CUDA devront être vérifiés.
- Utiliser les évènements CUDA ou la commande nvprof pour profiler le code GPU.
- Utiliser std::chrono pour profiler le code CPU.
- Prendre des images très grandes pour les tests
- Analyser systématiquement les performances des codes et comparer les versions entre elles.
- Tester différentes configuration (2D) de grilles et blocs et identifier les meilleures configurations.

1. Niveaux de gris

- Vérifier que le code grayscale fonctionne sur GPU. Le code se trouve sur la machine 192.168.80.64 dans le dossier /home/CUDA/examples.

2. Détection des bords

- Créer un kernel CUDA pour la détection de bords d'une image basé sur le code C du dossier /home/CUDA/examples/sobel. Attention, il convient d'utiliser une variable temporaire de types short (16 bits) car les valeurs peuvent être négatives et/ou dépasser la valeur maximale d'un unsigned char (8bits). La valeur temporaire devra être saturée avant d'être reconvertie en unsigned char : si la valeur est supérieure à 255 alors elle est fixée à 255, si elle est inférieure à 0 alors elle est fixée à 0.

- Vérifier et résultat par rapport à celui obtenu par la version CPU. Comparer les performances de ce kernel par rapport à la version CPU.
- Essayer d'optimiser ce kernel en utilisant la mémoire shared, les streams et toute autre optimisation qui vous semble pertinente. Pour chaque optimisation, comparer par rapport à la version non optimisée et discuter des performances.

3. Séquence et fusion

- Appliquer les 2 kernels séquentiellement à l'image pour produire l'image de sortie. Pour cela il faut modifier le kernel de détection de contours pour qu'il prenne en entrée une image en noir et blanc à la place d'une image couleur. Tester et comparer les performances.
- Créer un kernel unique effectuant en une même passe la conversion et la détection des bords. Quels sont les problèmes rencontrés et comment les résoudre. Tester et comparer les performances.