Rapport CUDA :

MOULAOUI Kamilia 2174585.

1/ Transformation d'image en niveaux de gris :

1 méthode :

- Vérification que le code grayscale fonctionne sur GPU.
- les resultat CPU et GPU sont les même .mais en terme de temps et d'espace l'utilisation du kernel c'est plus intéressant .
- gestion des erreur , ainsi que du temps GPU.

2eme méthode :

-J'ai testé un autre type d'image (.ppm) et la bibliothèque netpbm m'a permis de lire et écrire des fichiers ppm. (http://netpbm.sourceforge.net/doc/libppm.html).

- J'ai proposé une version Cuda du programme en 1D.
- et une autre version en 2D.

- En s'appuyant sur cette bibliothèques et à l'aide du type PIXEL, j'utilise les fonctions de la bibliothèques pour manipuler ce type par exemple PPM_ASSIGN(img_out[y*cols+x], gray, gray, gray).

- Performance :

J'ai limité la taille des blocs pour maximiser le parallélisme, ici, un bloc = 16*16 = 256 threads,(des blocs de 32*32 pour gray méthode 1), sur une image de 512*512 pixels, on lancera donc au moins 1024 (512*512/256) blocs.

2. Détection des bords:

- Utilisation d'un kernel sobel pour la version GPU avec des blocs de 32*32.
- Gestion des erreur et de synchronisation ainsi du temps GPU.

3. Séquence et fusion:

- **Sequence :**

- Lancer kernel gray puis modification des paramètre d'entré du kernel de détection de contours ,il prend en entrée une image en noir et blanc retourné par gray et applique la détection des contours.
- Utilisation de cudaDeviceSynchronize() qui bloque le CPU tant que l'ancien appel kernel gray n'est pas terminé, puis sobel se lance.
- le temps pris est considérable.

- **Kernel unique :**
    - cet unique kernel fait en une même temps la conversion et la détection des bords.
    - version FusionPremierEssaie: j'ai essayé de fusionner les deux kernel en appliquant gray et sobel dans le même kernel et j'ai eu une image avec des traits blancs sur chaque pixel.
        - les problèmes rencontrés :

            - Fusion de 2 kernels, gestion des blocs et des threads.

            - Problème d'indices.

        - Résolution:
    - version 1 FusionFinal: j'ai commencé par mettre au niveau de gris tous les pixels contenant des trait blancs , puis les 6 voisins qui interviennent (3 voisins en horizontal et 3 en vertical) dans le kernel sobel.
    - version 2 FusionFinal: positionner les thread dans des endroit précis , cela permet à gray et sobel de s'exécuter en parallèle mais pas sur les même blocs .les threads exécutant gray sont alors dans un bloc plus haut.

- le temps d'exécution est beaucoup plus intéressant.