Rapport CUDA

Groupe PETRAGALLO Tom - ELIAS Raghdah

Algorithme C++ (convolution dans main.cpp)

Cet algorithme permet d'appliquer une matrice de convolution à une image donnée en entrée. Il va parcourir les pixels de l'image d'origine et va effectuer la somme des produits entre la matrice et les pixels alentour. Il peut travailler sur des images en nuances de gris ou en RVB mais dans la suite, nous ne discuterons que du travail sur des nuances de gris.

À partir de cet algorithme de base, on peut effectuer plusieurs convolutions en changeant simplement la matrice donnée en paramètre. Voici 2 exemples de matrices qu'on utilise pour appliquer un filtre :

- Le flou (blur) :

0.0625 0.125 0.0625
0.125 0.25 0.125
0.0625 0.125 0.0625

- La détection de lignes (edges) :

-1 -1 -1
-1 8 -1
-1 -1 -1

** Ici, ce sont 2 matrices 3x3 mais avec l'algorithme CUDA, il est également possible d'utiliser des matrices plus grandes.*

Algorithme CUDA (convolution dans main.cpp)

Dans cet algorithme, l'image est découpée en blocs (par défaut ils sont de taille 32 x 32). Cet algorithme appelle le kernel convolution_gpu qui va appliquer la matrice à l'image.

Pour les mesures qui suivent, nous avons exécuté chaque algorithme plusieurs sur 3 images :

- "ingray.jpg" de taille 1920x1080 px
- "lighthouse.jpg" de taille 3000x4000 px
- nebuleuse.jpg de taille 14400*7200 px

Nous avons ensuite pris une moyenne approximative des différents temps.
Voici les résulats de nos mesures en fonction de l'algorithme utilisé, du filtre choisi et de la taille des blocs définies :

main.cpp :

Image	Filtre	Temps
ingray	id	36 ms
ingray	blur	38 ms
ingray	edges	40 ms
lighthouse	id	150 ms
lighthouse	blur	160 ms
lighthouse	edges	160 ms
nebuleuse	id	1.5 s
nebuleuse	blur	1.5 s
nebuleuse	edges	1.9 s

main.cu :

Image	Filtre	blocs 32x32	blocs 16*8	blocs 13*5	blocs 1*1
ingray	id	0.89 ms	0.94 ms	1.38 ms	41 ms
ingray	blur	0.93ms	0.95 ms	1.39 ms	57 ms
ingray	edges	0.80 ms	0.94 ms	1.38 ms	45 ms
lighthouse	id	4.3 ms	4.6 ms	6.9 ms	210 ms
lighthouse	blur	4.5 ms	4.6 ms	7.0 ms	266 ms
lighthouse	edges	4.3 ms	4.6 ms	6.9 ms	212 ms
nebuleuse	id	36 ms	39 ms	58 ms	1.8 s
nebuleuse	blur	38 ms	39 ms	59 ms	2.3 s
nebuleuse	edges	36 ms	39 ms	59 ms	1.8 s

** A noter que des tailles de blocs de 32*4 et 16*16 ont donné des résultats très similaires à ceux de 32 x 32*

Discussion des résultats :

Sans surprise, le calcul avec GPU est bien plus rapide que sur CPU (jusqu'à 50 fois plus rapide pour la nébuleuse avec le filtre edges)

La principale difficulté rencontrée est la gestion des zones entourant les blocs habituels de pixels si l'on utilise de la mémoire shared. En effet nous avons trouvé pertinent de conserver le résultat des pixels d'un bloc dans de la mémoire shared lorsque l'on veut enchaîner plusieurs convolutions. Cela permettrait ainsi de réaliser plusieurs filtres sans avoir à rapatrier les données sur le CPU et les renvoyer entre chaque filtre. Cependant, nous n'avons pas réussi à finaliser cette implémentation, les erreurs que l'on rencontre à l'exécution viennent probablement d'une mauvaise gestion des indices entre la mémoire shared et les données de base de l'image.