TP2 : Seuillage couleur (Matlab/C/GPU)

IUT Cachan

N. Gac

1 NVIDIA GPU Computing

Toutes les cartes graphiques NVIDIA (depuis la génération 8) même celles à bas prix, permettent de faire de la programmation CUDA. Il suffit d'installer pour cela le *driver* de la carte, le *cudatoolkit* (contenant le compilateur NVIDIA) et le *SDK* (exemples de code pour débuter la programmation CUDA). Tous ces logiciles sont disponibles sur le site de Nvidia pour développeurs : http://developer.nvidia.com/object/gpucomputing.html

Figure 1: Carte Nvidia Geforce 9500 GT (prix < 50 €)

1.1 Exemples du SDK

Le SDK (Software Developpement Kit) de Nvidia offre plusieurs executables utilisant la puissance des cartes graphiques. Le code CUDA est donné à titre d'exemple pour débuter dans la programmation CUDA. Lancez les applications disponibles à partir du raccourci NVIDIA sur le bureau : *Ocean Simulation*, *Particles*, *Smoke Particles*...

1.2 Device Query

Identifiez les caractéristiques de votre carte graphique (Geforce 9500 GT) grâce à l'executable *Device Query*. Combien y a t il de multiprocesseurs ? Combien de coeurs par multiprocesseurs ? Quelle est la fréquence des coeurs ? Calculez en GFLOPS la puissance de calcul de la carte graphique.

Remarque : Un coeur effectue jusqu'à 3 opérations sur des float par cycle d'horloge (MADD + MULT).

1.3 Préparation pour la suite du TP

a) Créez un répertoire *C:\travail \TP_GPU_votre_nom*. Récupérer sous Dokeos (Organisation S4-TI) le repertoire zippé files_TP2.zip. Copiez le dans votre répertoire *C:\tmp \TP_GPU_votre_nom*.

Les images d'entrée et de sortie de la ferrari se trouve dans le répertoire *TP_GPU \Image*.

2 Seuillage sous Matlab

L'image couleur *ferrari.jpg* est une image RGB (Red Green Blue). Cette image RGB est stockée dans trois tableaux 2D, chacun correspondant à une des trois couleurs Rouge, Vert et Bleu.

Travaillez sous Matlab dans le répertoire *seuillage_Matlab*. Utilisez comme script de départ le fichier *seuil_Matlab.m*.

2.1 Ferrari rouge

Seuillez l'image I en fonction de η_R niveau de rouge de chaque pixel de l'image I :

$$I_s(xn, yn) = \begin{cases} I(xn, yn) \text{ si } \eta_R > 0.7 \\ 0 \text{ sinon} \end{cases}$$

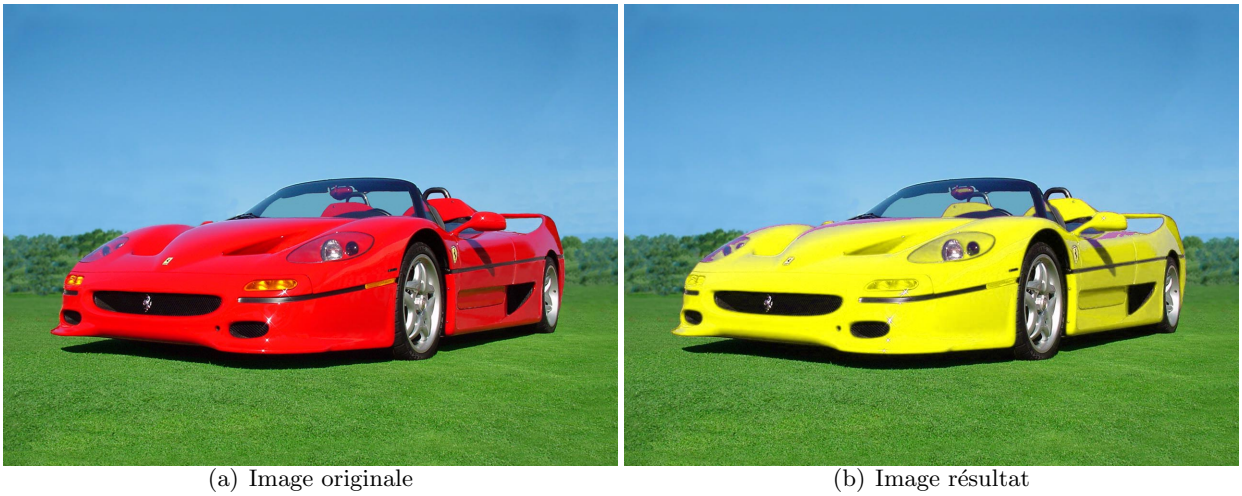
$$\eta_R = \frac{I_R}{\sqrt{I_R^2 + I_G^2 + I_B^2}}$$

2.2 Ferrari jaune

Jaune=Rouge+Vert...

(a) Image originale I (b) Image seuillée I_s

Figure 2: Extraction de la Ferrari rouge par seuillage du niveau de rouge



(a) Image originale (b) Image résultat

Figure 3: Changement de couleur de la Ferrari

2.3 Réduction du temps d'exécution sous Matlab

Contrairement au C qui ets un langage compilé (apres compilation, le code binaire est prêt à être exécuté), Matlab est un langage interprété. Chaque ligne de code est traduite un mini executable. Cette te analyse de chaque ligne de code implique un surcoût en temps d'execution. Optimisez votre code en supprimant au maximum les boucles for qui ralentissent les performances des codes Matlab.

3 Seuillage en C sous Visual Studio

a) Ouvrez le projet visual studio*seuillage_etudiant.sln* contenu dans le répertoire *seuillage_etudiant*.

b) Codez la fonction *seuillage_C_ij* contenu dans *seuillage_C.cpp*. Cette fonction est lancée depuis le main contenu dans le fichier *seuillage_main.cu*. Vérifiez si vous obtenez le bon seuillage en observant sous ImageJ, l'image de sortie *Image\ferrari_out_CPU.raw*. Comparez la avec celle obtenue sous Matlab *Image\ferrari_out_Matlab.raw*.

c) Notez le temps de calcul, comparez le avec celui obtenu sous Matlab.

d) Inversez l'ordre des boucles (boucle selon les lignes et colonnes de l'image) dans la fonction *seuillage_C_ji.cpp* . Notez le temps de calcul. Comparez le temps de calcul pour les deux ordres de boucle. Pourquoi existe t il une différence ?

4 Seuillage couleur en CUDA sous Visual Studio

a) A vous de compléter le code à trou ! Dans le fichier *seuillage_main.cu*, il manque les étapes d'allocation mémoire, les transferts mémoire entre le PC hôte et la carte GPU, le découpage en threads et le lancement du kernel. Dans le fichier *seuillage_GPU_kernel.cu*, il manque la définition d'un thread. Inspirez vous pour cela du code de multiplication de matrice vu en cours.

b) Vérifiez si vous obtenez le bon seuillage en observant sous ImageJ, l'image de sortie *Image \ferrari_out_GPU.raw*. Comparez la avec celle obtenue sous Matlab *Image \ferrari_out_Matlab.raw*.

d) Mesurez le temps de calcul et comparez le avec celui obtenu en C et Matlab. Quel facteur d'accélération obtenez vous ? Quel serait ce facteur avec une carte ayant 2880 coeurs (freq. 745 Hz) comme la Tesla K40 ?

5 Déctection de contours en C/CUDA sous Visual Studio

Reprenez le TP1 de détection de contours et faites un code C puis en CUDA pour la convolution d'un filtre 2D. Mesurez les temps de calcul pour différentes taille de filtre (3*3, 5*5 .. 21*21) et comparez les avec ceux obtenus en Matlab.