Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Dév GPGPU

Projet Image

Application parallèle de filtre sur des images avec CUDA

Gestion des images

Pour ce projet, nous aurons besoin de lire et d'écrire des images en mémoire :

- nous utiliserons le format PPM, « *Portable Pixel Map* », qui est un format d'image très simple, non compressé, qui permet de traiter des images composées de pixels sur 3 composantes R,V & B quantifiée, chacune, sur un octet ;
- le module C « `ppm_lib` », composé des fichiers :
  - `ppm_lib.h`, dispo. à `http://p-fb.net/fileadmin/GPGPU/ppm_lib.h`
  - `ppm_lib.c`, dispo. à `http://p-fb.net/fileadmin/GPGPU/ppm_lib.c`
- un exemple d'utilisation, contenu du fichier `test_ppm_lib.c` :

```c
#include <stdio.h>
#include <stdlib.h>
#include "ppm_lib.h"

void changeColorPPM(PPMImage *img)
{   int i;
    if(img){
         for(i=0;i<img->x*img->y;i++){
              img->data[i].red=RGB_COMPONENT_COLOR-img->data[i].red;
              img->data[i].green=RGB_COMPONENT_COLOR-img->data[i].green;
              img->data[i].blue=RGB_COMPONENT_COLOR-img->data[i].blue;
         }
    }
}

int main(){
    PPMImage *image;
    image = readPPM("mon_image.ppm");
    changeColorPPM(image);
    writePPM("mon_image2.ppm",image);
}
```

Pour convertir une image de n'importe quel format, vers le format PPM, vous pouvez utiliser Python :

```
>>> from PIL import Image
>>> img=Image.open("mon_image.jpg")
>>> img.save("image_convertie.ppm")
```

Vous trouverez une photo de la gare Limoges-Bénédictins au format PPM en 1000 × 500 à : `http://p-fb.net/fileadmin/GPGPU/gare_parallelisme2.ppm`

Application de filtre sur une image

Pour appliquer un « filtre » sur une image :

- ▷ il faut faire une copie de l'image en mémoire.
  Elle servira de destination pour l'application du filtre.
- ▷ disposer d'un codage pour un pixel suivant ses 3 composantes pour une image couleur ;
- ▷ pour chaque pixel de l'image :
  - ◇ pour ce pixel et ceux qui l'entourent ;
    - ⋆ obtenir la valeur de ce pixel et la multiplier par la valeur correspondante de la grille du filtre ;
    - ⋆ ajouter ce résultat au total ;
  - ◇ diviser le nombre total par le « facteur de division », *divide factor*, et ranger la valeur obtenue dans le pixel de l'image destination.

Ce qui donne le parcours de l'image suivant :

```
for(y=top; y<=bottom; y++)
   for(x=left; x<=right; x++)
           {
             …
           }
```

- ▷ Exemple de filtre :

```
0 0 0 0 0
0 1 3 1 0
0 3 5 3 0
0 1 3 1 0
0 0 0 0 0

Divide: 21
```

Ce filtre rend l'image plus douce, « softer ».
Le *divide factor* correspond à la somme des différents coefficients de la matrice du filtre.
Le « pixel traité » est celui **au centre** de la matrice du filtre, *ici 5*, les pixels adjacents sont ceux autour de ce 5.

Ce pixel possède la valeur la plus grande du filtre car c'est celui qui est le plus important et qui possède le plus de poids dans le calcul du total.

Une fois le total calculé, on le divise par le *divide factor* et on met le résultat où était le « pixel traité ».

L'application du filtre donne le code suivant :

```
for(y=top; y<=bottom; y++)      // for each pixel in the image
   for(x=left; x<=right; x++)
   { gridCounter=0;      // reset some values
     final = 0;
     for(y2=-2; y2<=2; y2++)     // and for each pixel around our
       for(x2=-2; x2<=2; x2++)   //  "hot pixel"...
       { // Add to our running total
         final += image[x+x2][y+y2] * filter[gridCounter];
         // Go to the next value on the filter grid
         gridCounter++;
       }
     // and put it back into the right range
     final /= divisionFactor;
     destination[x][y] = final;
   }
```

D'autres filtres :

```
Soften (a lot)
 1   1   1   1   1
 1   1   1   1   1
 1   1   1   1   1
 1   1   1   1   1
 1   1   1   1   1
 Divide: 25

Diagonal "shatter"
 1   0   0   0   1
 0   0   0   0   0
 0   0   0   0   0
 0   0   0   0   0
 1   0   0   0   1
 Divide: 4

Horizontal Sobel
 1   2   0   -2   -1
 4   8   0   -8   -4
 6   12  0  -12   -6
 4   8   0   -8   -4
 1   2   0   -2   -1
 Divide: 1
```

```
Sharpen (medium)
 -1   -1   -1   -1   -1
 -1   -1   -1   -1   -1
 -1   -1   49   -1   -1
 -1   -1   -1   -1   -1
 -1   -1   -1   -1   -1

 Divide: 25

Horizontal Blur
 0   0   0   0   0
 0   0   0   0   0
 1   2   3   2   1
 0   0   0   0   0
 0   0   0   0   0
 Divide: 9

Vertical Sobel
-1  -4  -6  -4  -1
-2  -8 -12  -8  -2
 0   0   0   0   0
 2   8  12   8   2
 1   4   6   4   1
 Divide: 1
```

Travail à réaliser

a. Adapter le programme précédent pour traiter alternativement chaque composante de couleur R, V & B.

b. Proposer une version OpenMP de ce traitement.

c. Écrire une version parallèle CUDA avec une répartition du travail entre les threads que vous choisirez afin de maximiser l'exploitation du parallélisme.

d. Lors du traitement des « bords » des zones distribuées, on se rend compte que les threads entre en compétition pour l'accès aux pixels partagés.

   Donner une version améliorée permettant de limiter ces chevauchements d'accès entre threads (cette version améliorée devra être le plus efficace possible quant à sa mise en œuvre).

   Vous essaierez de tirer parti de la mémoire partagée exploitable.

e. Vous mesurerez et noterez le temps pris par ces différentes versions, ainsi que le gain obtenu, en appliquant de 1000 à 100 000 fois, un ou des filtres sur l'image proposée ou sur des images plus grandes que vous proposerez.

Complément d'information sur l'application du filtre de Sobel

D'après les informations fournies par Nicolas Pavie :
Le filtre de Sobel n'a pas de facteur de normalization ou alors un facteur de division à 1.

Le but de ce filtre est de calculer directement les variations locales de couleurs, ce qui va créer des valeurs proches du noir dans les zones où il n'y a pas de variations (typiquement dans les objets).

Il n'y a que les bords qui resortent vraiment dans l'image après application du filtre.

Si on applique un seuil a 128, après transformation du résultat en niveau de gris, on se retrouves avec uniquement les bords des objets (comme si on négligeait les micros variations de teinte dans l'image).

Pour la transformation en niveau de gris entre le sobel et le seuil, on peut utiliser la formule de calcul de la luminance d'une couleur :

$$Y = 0.2126 * R + 0.7152 * G + 0.0722 * B$$

Remise du travail

Le travail devra être remis sous forme d'une archive en utilisant `https://filesender.renater.fr`

Pour les mesures et commentaires sur vos programmes vous joindrez un document PDF (pas un rapport) à cette archive contenant des copies d'écran de vos résultats.