M2-BIG DATA
GPGPU Chapter 3

Exercice 2

Réalisé par :
Gaby Maroun

Encadré par :
Dr. Etancelin JM

11 novembre 2020

Objectives

The purpose of this exercice is to convert an RGB image into a gray scale image. The input is an RGB triple of float values and the program will convert that triple to a single float grayscale intensity value. A pseudo-code version of the algorithm is shown bellow :

```
for i from 0 to height do
    for j from 0 to width do
        idx = i * width + j
        # here channels is 3
        r = input[3*idx]
        g = input[3*idx + 1]
        b = input[3*idx + 2]
        grayImage[idx] = (0.21*r + 0.71*g + 0.07*b)
    end
end
```

Instructions

Edit the given code *Chap3_Ex2-imgTograyscale.cu* to performs the followings steps :

```
#include <iostream>
#include <cstdlib>
#include <string>

#include "img_utils.hpp"
#define CHANNELS 3

using namespace std;

//@TODO@ : Write the kernel here

__global__ void colorConvert(float * grayImage, float * rgbImage, int width, int
    height) {

 int x = threadIdx.x+ blockIdx.x* blockDim.x;
 int y = threadIdx.y+ blockIdx.y* blockDim.y;

 if (x < width && y < height) {

 int grayOffset= y * width + x;

  int rgbOffset = grayOffset*CHANNELS;
  float r = rgbImage[rgbOffset]; // red value for pixel
  float g = rgbImage[rgbOffset+ 1]; // green value for pixel
  float b = rgbImage[rgbOffset+ 2]; // blue value for pixel

  grayImage[grayOffset] = 0.21f * r + 0.71f * g + 0.07f * b;
```

```
27
28  }
29 }
30
31 int main(int argc, char **argv)
32 {
33   if(argc!=3) {
34   cout<<"Program takes two image filenames as parameters"<<endl;
35   exit(3);
36   }
37
38   float *imgIn, *imgOut;
39   int nCols, nRows, channels;
40   // Allocate images and initialize from file
41   imgIn = read_image_asfloat(argv[1],&nCols, &nRows, &channels);
42     if(channels!=3){
43     cout<<"Input image is not a colored image"<<endl;
44     exit(4);
45     }
46   imgOut = (float *)calloc(nCols*nRows, sizeof(float));
47
48   // Allocates device images
49   float *d_imgIn, *d_imgOut;
50
51   //@TODO@ : Complete for device allocations
52    cudaMalloc(&d_imgIn, nCols * nRows * channels * sizeof(float));
53    cudaMalloc(&d_imgOut, nCols * nRows * sizeof(float));
54
55
56   // Copy input data
57   //@TODO@ : Complete for data copy
58    cudaMemcpy(d_imgIn, imgIn, nCols * nRows * channels * sizeof(float),
      cudaMemcpyHostToDevice );
59
60   // Call the kernel
61   //@TODO@ : Compute threads block and grid dimensions
62   dim3 DimGrid((nRows-1)/32+ 1, (nCols-1)/32+1, 1);
63   dim3 DimBlock(32, 32, 1);
64
65   //@TODO@ : Call the CUDA kernel
66   colorConvert<<<DimGrid,DimBlock>>>(d_imgOut, d_imgIn, nCols, nRows);
67
68   // Copy output data
69   //@TODO@ : Complete for data copy
70   cudaMemcpy(imgOut, d_imgOut, nCols * nRows * sizeof(float), cudaMemcpyDeviceToHost
       );
71
72   // Write gray image to file
73   write_image_fromfloat(argv[2], imgOut, nCols, nRows, 1);
74
75
```

```
76    // Free memory
77    //@TODO@ : Free host and device memory
78
79    // Free device memory
80    cudaFree(d_imgIn);
81    cudaFree(d_imgOut);
82
83    // Free host memory
84    free(imgIn);
85    free(imgOut);
86
87    return 0;
88 }
```

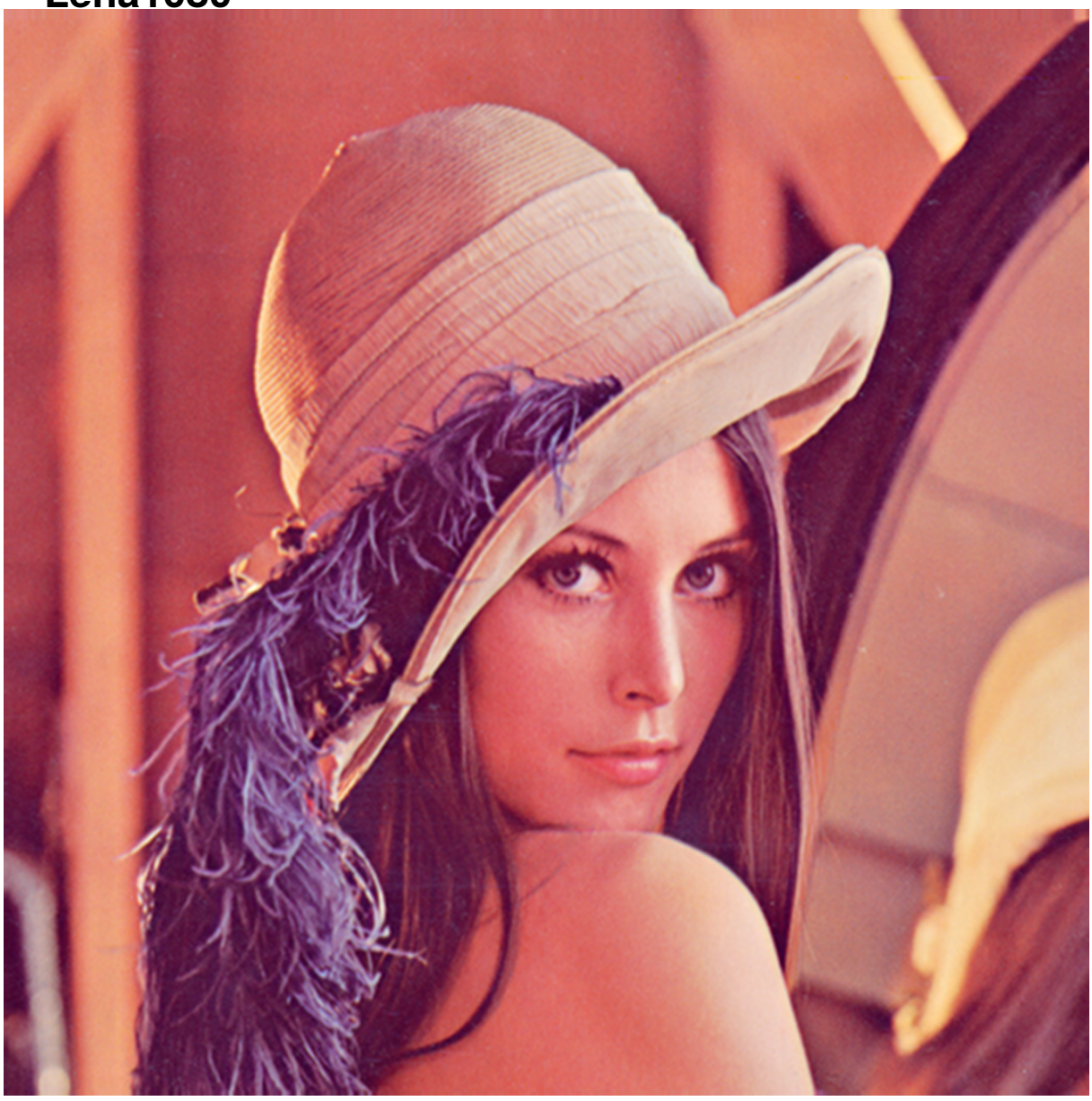
The execution is performed giving the 2 filenames as program arguments : *./Chap3_Ex2-imgTograyscale inputImg.png outputImg.png* The input image must be a colored image. The reading and writing is performed by two functions *read_colored_image_asfloat* and *write_gray_image_fromfloat*. In this exercise, images are represented as RGB or gray with float values.
The CUDA thread grid must be computed from the image dimensions. A common approach is to define squared blocks (called tile), and the grid size accordingly. Mind the arbitrary image dimensions which are generally not a multiple of tile dimensions. Use the given images $Lena.png$ then the $Lena1080.png$ or any RGB image you want.

The resulted images came as follow :

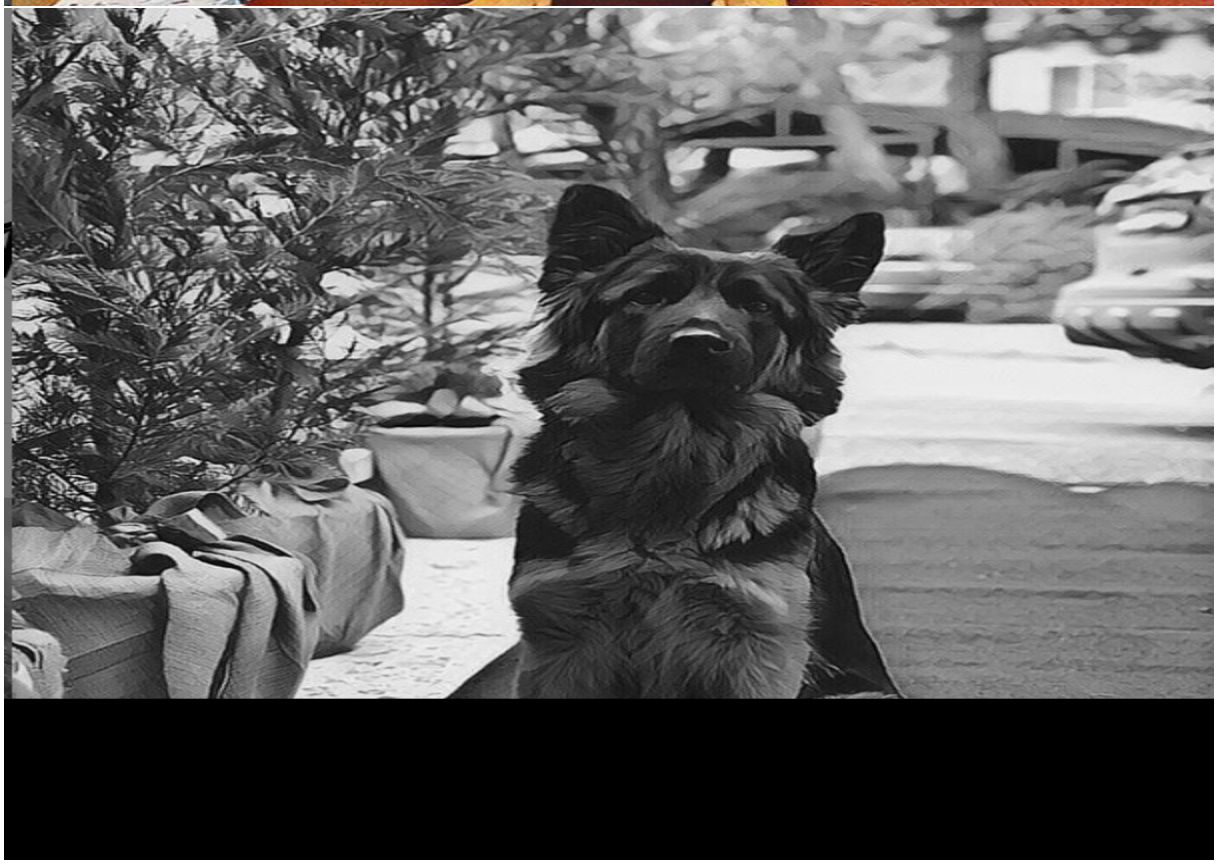
```
gmaroun@scinfe058:/import/etud/3/gmaroun/Bureau/stockage/Semestre 3/GPGPU/Chap3Ex2$ make
nvcc -c  2-imgToGrayScale.cu -o 2-imgToGrayScale.o
2-imgToGrayScale.cu: In function 'int main(int, char**)':
2-imgToGrayScale.cu:86:26: warning: format '%f' expects argument of type 'double', but argument 2 has type 'float*' [-Wformat=]
  printf("diO %f",d_imgOut);
                          ^
2-imgToGrayScale.cu:87:25: warning: format '%f' expects argument of type 'double', but argument 2 has type 'float*' [-Wformat=]
  printf("diI %f",d_imgIn);
                         ^
2-imgToGrayScale.cu:88:23: warning: format '%f' expects argument of type 'double', but argument 2 has type 'float*' [-Wformat=]
  printf("iO %f",imgOut);
                       ^
2-imgToGrayScale.cu:89:25: warning: format '%f' expects argument of type 'double', but argument 2 has type 'float*' [-Wformat=]
  printf("iI %f \n",imgIn);
                         ^
nvcc  2-imgToGrayScale.o img_utils.o -o 2-imgToGrayScale   `pkg-config --libs opencv` -lm
gmaroun@scinfe058:/import/etud/3/gmaroun/Bureau/stockage/Semestre 3/GPGPU/Chap3Ex2$ ./2-imgToGrayScale Lena.png LenaBW.png
Read image of size 512x512 3 channels
Write image 512x512 1 colors into LenaBW.png
diO 0.000000diI 0.000000iO 0.000000iI 0.000000
gmaroun@scinfe058:/import/etud/3/gmaroun/Bureau/stockage/Semestre 3/GPGPU/Chap3Ex2$ ./2-imgToGrayScale Lena1080.png.png LenaBW.png
libpng warning: Image width is zero in IHDR
libpng warning: Image height is zero in IHDR
libpng error: Invalid IHDR data
Read image of size 0x0 1 channels
Input image is not a colored image
gmaroun@scinfe058:/import/etud/3/gmaroun/Bureau/stockage/Semestre 3/GPGPU/Chap3Ex2$ ./2-imgToGrayScale Lena1080.png LenaBW1080.png
Read image of size 1080x1080 3 channels
Write image 1080x1080 1 colors into LenaBW1080.png
```

Lena

Lena1080



Ivy



Questions

1. **How many floating operations are being performed in your color conversion kernel ? EXPLAIN.**

   *There are $nCols * nRows * 5$ floating operations being performed, the number of the operations (3 * et 2+)*

2. **How many global memory reads are being performed by your kernel ? EXPLAIN.** *It's equal to $nCols * nRows * 3$ to read the whole image with its different channels of colors as the condition in the if is to search the whole width and height while reading the rgb*

3. **How many global memory writes are being performed by your kernel ? EXPLAIN.** *It's equal to $nColsnRows1$ to write the whole new image using the gray color as the only channel*

4. **How to you choose the CUDA thread grid dimensions ? EXPLAIN.** *To choose the CUDA thread grid dimensions, we divide the width and height (or number of rows and columns) on the number of blocks to cover all elements. Each block cannot have more than 512/1024 threads in total and the number of threads per block should be around multiple of the warp size, which is 32 on most of the current hardware.*

La fin.