A3 – Image Processing – RGB to GrayScale, Tiled MM

Deadline: August 13, 9AM.

Q1. [Ref. PMPP 3e. Chapter 3. Scalable Parallel Execution.]

The purpose of this lab is to convert an RGB image into a gray scale image. The input is an RGB triple of float values. You have to convert the triplet to a single float grayscale intensity value. A pseudo-code version of the algorithm is shown below:

```
for ii from 0 to height do
  for jj from 0 to width do
    idx = ii * width + jj
    # 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)  // converts 3 r g b values to a single grayscale value.
  end
end
```

Image Format

The input image is in PPM P6 format while the output grayscale image is to be stored in PPM P5 format. You can create your own input images by exporting your favorite image into a PPM image. On Unix, **bmptoppm** converts BMP images to PPM images (you could use **gimp** or similar tools too).

Run command: **`./ImageColorToGrayscale_Template -e <expected.pbm> -i <input.ppm> -o <output.pbm> -t image`**

where `<expected.pbm>` is the expected output, `<input.ppm>` is the input dataset, and `<output.pbm>` is an optional path to store the results. Questions.

1. How many floating operations are being performed in your color conversion kernel?
2. Which format would be more efficient for color conversion: a 2D matrix where each entry is an RGB value or a 3D matrix where each slice in the Z axis representes a color. I.e. is it better to have color interleaved in this application? can you name an application where the oposite is true?
3. How many global memory reads are being performed by your kernel?
4. How many global memory writes are being performed by your kernel?
5. Describe what possible optimizations can be implemented to your kernel to achieve a performance speedup.

Q2. Perform Matrix Multiplication of two large integer matrices in CUDA. Answer the following questions.

1. How many floating operations are being performed in your matrix multiply kernel?
2. How many global memory reads are being performed by your kernel?
3. How many global memory writes are being performed by your kernel?
4. Describe what possible optimizations can be implemented to your kernel to achieve a performance speedup.

Q3. [Ref. Chapter 4. Memory and Data Locality.]

Implement a tiled dense matrix multiplication routine using shared memory. Use the template code. Run command:
`./TiledMatrixMultiplication_Template -e <expected.raw>  -i <input0.raw>,<input1.raw> -o <output.raw> -t matrix`

where `<expected.raw>` is the expected output, `<input0.raw>,<input1.raw>` is the input dataset, and `<output.raw>` is an optional path to store the results.

1. How many floating operations are being performed in your matrix multiply kernel? explain.
2. How many global memory reads are being performed by your kernel? explain.
3. How many global memory writes are being performed by your kernel? explain.
4. Describe what further optimizations can be implemented to your kernel to achieve a performance speedup.
5. Compare the implementation difficulty of this kernel compared to the previous MP. What difficulties did you have with this implementation?
6. Suppose you have matrices with dimensions bigger than the max thread dimensions. Sketch an algorithm that would perform matrix multiplication algorithm that would perform the multiplication in this case.
7. Suppose you have matrices that would not fit in global memory. Sketch an algorithm that would perform matrix multiplication algorithm that would perform the multiplication out of place.