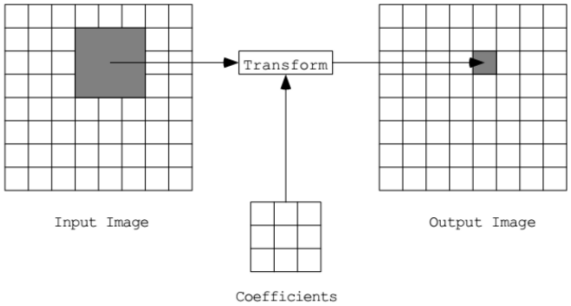
COMPUTAÇÃO DE ALTO DESEMPENHO

2021/2022

Project 1 - CUDA

Image filters, like the blur filter used in lab 3, are examples of matrix processing that can be taken to the GPU. Consider two classes of filters: area filters, that considers an area around each pixel to define the final value of that pixel; and the point filters, that applies some function on the pixel value. For this project you need to implement a sequence of two filters with some parameters, that could be applied to any given image.
For the area filter, considering just the direct neighbors, we get a 3x3 grid of pixels with the original pixel at its center.



The final value for that pixel will be the weighted average defined by a 3x3 matrix of coefficients. The following examples of coefficients allows the implementation of several types of filters:

$$\begin{bmatrix}0 & 0 & 0\\0 & 1 & 0\\0 & 0 & 0\end{bmatrix} \quad \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} \quad \begin{bmatrix}0 & -1 & 0\\-1 & 5 & -1\\0 & -1 & 0\end{bmatrix} \quad \begin{bmatrix}-1 & -1 & -1\\-1 & 8 & -1\\-1 & -1 & -1\end{bmatrix}$$

(a) Original (b) Blur (c) Emboss (d) Edge detection

For the point filter, we pretend to gray the image using the following expression:

(r,g,b) = alpha*(c,c,c) + (1-alpha)(r,g,b), where c = 0.3 r + 0.59 g + 0.11 b

Alfa is a value in [0 .. 1] that defines the color shift to grayscale.

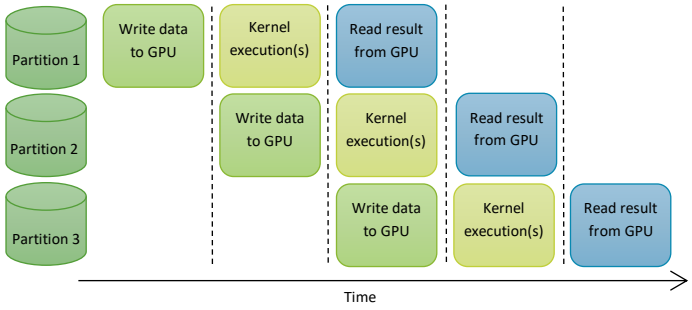
A sequential code example is provided for reference. The main objective is to achieve the best performance, particularly for big images. For convenience, this work is presented in several stages (number 5 is mandatory):

1. (30%) Implement a CUDA or OpenCL solution that
   a. Parallelizes both filter operations
      Suggestion: experiment with different parallel strategies. Examples: just one kernel with both filters vs. one kernel per filter.
   b. Experiment with different grid layouts and block sizes.
   Study if using a local shared area can improve your kernel(s) performance (nvprof and section 8 of CUDA Best Practices Guide[1] may help you evaluate any improvement).

2. (20%) Evaluate these solutions against
   a. The sequential version
   b. Using different grid arrangements and thread block sizes
   c. Using/not using shared memory
   Note: ignore file I/O times but include in your timings memcopy to/from device.

[1] https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html#performance-metrics

3. (15%) Complement your solution with the ability to overlap computation with communication by partitioning the image space to process into a given number of partitions (NP). Assigning each of such partitions to a dedicated stream. Consider the following example with NP=3.



While kernel(s) is(are) being applied over Partition 2, Partition 3 can be simultaneously uploaded to the GPU and, possibly, the result of processing Partition 1 can be simultaneously downloaded from the GPU.

4. (15%) Evaluate this solution against the others.

5. (20%) Write a report (max of 5 pages A4 11pt font) that presents
    a. Tested approaches and final solution
    b. Relevant implementation details
    c. Your evaluation results (include times and/or graphs to compare and justify your solution)
    d. An analysis and interpretation of these results

Other relevant optimizations may also be accounted in the final grade.