Tutorial: Image Processing with OpenCL

Note

This tutorial comes with a skeleton of code that helps concentrate on the main thing: the kernel code.

1 Image manipulation with OpenCL

The sample code provided executes a copy of an image file into another. It is coded in C++. It consists in some utilities (utils.hpp and and utils.cpp) and one main file (imageCopyFilter.cpp, containing the main function).

This tutorial supposes a linux computer (with ubuntu, already configured).

- Open a terminal.
- Unzip the file with

```
unzip imageCopyFilter.zip
```

- Generate the makefile by

```
cmake -g .
```

- Compile the project

```
make
```

- Execute it

```
./imageCopyFilter
```

Spend some time studying this code. The kernel is defined in the file copyimage.cl.

2 Mean filter by convolution

A mean filter M of an image f is defined by $g = M(f)$, with the definition of an operational window Ω. If p is a pixel, $\mathcal{D}$ a definition domain and vol represents the n-dimensional volume:

$$g(p) = \frac{1}{vol(\Omega)} \int_{\Omega} f(q)$$

In a discrete 2D domain, the formulation is:

$$p \in \mathcal{D}, g(p) = \frac{1}{\#\Omega} \sum_{q \in \Omega} f(q)$$

where $\#\Omega$ denotes the cardinal of Ω.

3 Gaussian filter by convolution

If f is the original image, and g the filtered image, defined in a 2D domain of size N, then the Gaussian filter is defined by:

$$g(i,j) = \sum_{l=-N}^{N} \sum_{m=-N}^{N} \underbrace{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{l^2+m^2}{2\sigma^2}}}_{\text{Gaussian function}} \cdot f(i+l, j+m)$$

- Code this filter by using the built-in function exp.
- Then, optimize your code. It has to be as fast as possible. Your assignment is to speed up the code by any mean you might find. You should send your code to gavet@emse.fr before the deadline. It should compile and execute under linux without any problem (provide as well the CMakeList.txt file).
- The final mark will take into account the speed up strategy and gain as well as the validity of the result (of course).