MI3.22 – Advanced Programming for HPC
Labwork 5 – *Project: image segmentation, Voronoï, glass effect*

Voronoï diagram can be used for image segmentation. It produces convex region into the resulting image (one criterion for a good segmentation being the convexity of region!), leading to some nice effect like the "*Glass effect*", or helping for face recognition.

If you do not know what a Voronoï diagram is, just use Internet and Wikipedia for instance. In this project, we will use equivalently seed or center of the region, and region or convex polygon.

The project have to be made starting with a labwork dedicated to the median filter (exercise 1). Extend this skeleton in order to realize the different version into 4 executable, one per exercise.

Exercise 1: Preliminary: median filter

The median filter is similar to the mean filter, except that it replaces the current color with the median value considering a neighboring (the size of the windows is given as parameter).

- Extract the array of Values from the image (in RGB), like in Labwork 4.
- By-column filtering.
  THe filtering can be efficiently made using the following algorithm (function `filter`). You need to use a CUDA kernel here, with one block for one image's column[1]. Each block starts by loading the neighboring of the first (top) pixel (line 0). For a pixel, you kernel must compute the histogram of neighboring values (function `fill_shared_memory`), and then do an *inclusive scan* to obtain a CDF (*Cumulative Distribution Function*, and the median value search (a simple MAP!, function `apply_filter`). Then, this algorithm continue with the next column pixel, up to the bottom of the column. In others words, **your kernel manage a full image's column**.

  You should observe that the neighboring of the next pixel is almost the same, except the top line (to remove) and the bottom one (to add) ... Hence, the hitogram calculation can be simplified: remove the values from the (old) top line, add values from the (new) bottom line (function `update_histo`)).

  To resume, the functions to implement are:

  - `fill_shared_memory` that computes the first (full) histogram (line 0);
  - `scan` that does a SCAN on the histogramme, without modifying it (so, starting by doing a local copy in shared memory ...);
  - `apply_filter` that applies the median filter (extracting it from the CDF);
  - `update_histo` that updates the histogram (that was not modified by the SCAN, fortunately).
- RGB normalization.
  At least, transform the RGB input image by modifying its values (HSV model). Each RGB component should be multiplied by the ratio of the new value by the old value. Take care about the overflows (a component R/G/B cannot be greater to 255 – they are `unsigned char`). Use CUDA and/or Thrust as you need.

Exercise 2: First version

Convex regions can be obtained quite simply if you already have the center of each of them. Hence, it suffices to fill the region!

For that, a very simple solution is to compute the distance between any pixel and the nearest region seed. It can be done iteratively, by propagating the nearest distance to the neighbors (take the minimum of the neighboring distance added to one, each). Here, the distance is then a Manhattan distance! Initialize this process with infinity except for seeds (with 0, the distance to itself). This part is very easy to implement. You may stop the loop early (when no modification is made, then the result is stable)

The problem, so, consists to find the seeds. You need to find "a good number" of seeds: to few and the result will be very ugly, to much and it will be over-segmented. A simple solution for this exercise consist to filter the image (median filter), and then to work on the values (again in HSV). First, each pixel considers itself as a seed. Then, iteratively, the regions are merged: a region relies to the minimum and the maximum of its values. You will merge two regions when the min and max values of the cumulative range is less than a threshold (cumulative range results from the union of the ranges of the two regions to merge ...). In the main loop, the threshold will grow. This loop ends when the number of seeds becomes less than a given limit, or the threshold becomes to high (using parameters in command line to control that).

[1] Why by column and not by line? You should be able to answer this question by yourself ;-)

Exercise 3: Second version

Here, the idea is to read the article provided on the Moodle, and to modify the first version in order to follow this new method. Notice that is relies on histogram! Ha, histograms ...

Compare the results, and the computation time.

Report.