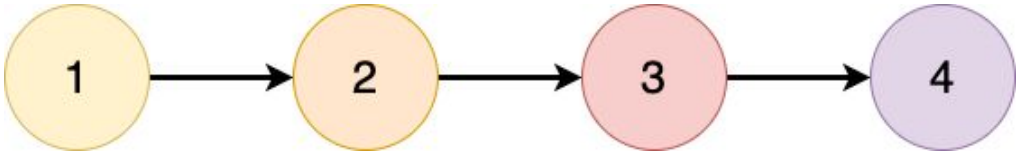
Assignment 4: simple image processing with opencv

We were asked to implement an algorithm capable of applying two filters in sequence over a set of images. In particular, the two filters were respectively the Gaussian and the Sobel. Hence, the task consisted of a pipeline as follows:

1. open the image
2. apply the Gaussian filter
3. apply the Sobel filter
4. save the image



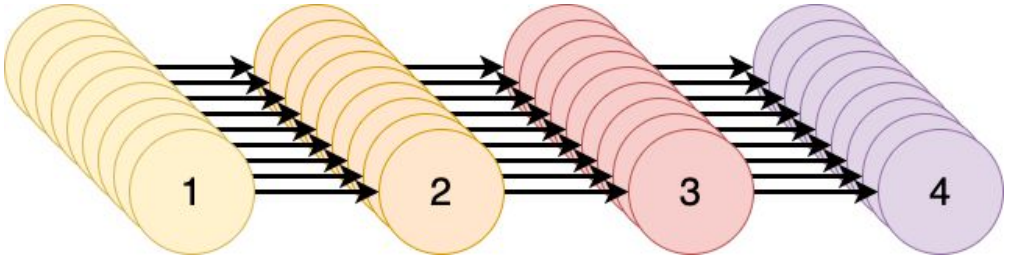
As far as the sequential version is concerned, the average time[1] (%) for each task is described in the following table:

READ I/O	54,62%
GAUSS FILTER	1,50%
SOBEL FILTER	24,28%
WRITE I/O	19,60%

There are a couple of details to discuss. First, the time measurements strictly depend on the image dimension, indeed each I/O lasts a time proportional to the size of the image. Second, it's evident that the Sobel filter is the most time-consuming operation (without considering I/Os).

[1] These tests have been performed on a personal dataset which can be found here: /home/bruno25-spm19/img

To realise the parallel version of the code, I decided to exploit the KISS software design principle, therefore the Parallel For has been chosen as the skeleton for this implementation. In my opinion, the Parallel For is the most suitable pattern for this kind of task, since the main goal is to read, apply some functions and finally store the data.



The speedups achieved by all the implementations highlight that the "bottleneck" is the number of parallel access to the disk. In fact, until 64 workers the speedup grows linearly, whereas, from 64 on, there's a flattening of the gain.