Michele Zoncheddu – 545227

OpenCV assignment report

To choose the best parallel implementation, I measured the times of the four stages (reading, Gaussian filter, Sobel filter, writing), that took an average of 71, 12, 120 and 63ms per image. Then I put those times in rplsh, and I chose the normal form: a farm in which every worker computes the composition of the four stages.

The program can be executed with the “nw” and “iterations” parameter, to choose how many times all the images need to be read. The array containing all the names of the files is read in a circular way, simulating a longer array. The other parameters are the source and destination folder.

I made four versions of the farm, with different types of scheduling, only for testing purposes:

1. Static scheduling (fixed chunks): it didn’t performed very well, because of the unbalanced work (some images are heavier than others);
2. Round-robin scheduling: it performed even worse, for the same reason;
3. Dynamic scheduling with blocking communication: it was the best solution, to uniformly spread the work among the workers;
4. Dynamic scheduling with non-blocking communication: same performance of the previous version, until we get to a parallel degree that exceed 64, the the number of real cores of the remote Xeon.

Even with the dynamic scheduling, the speedup was not ideal, and I think that at that point the bottleneck was the input/output of the disk.

To have a reference and confirm that suspect, I made a version with OpenMP, and until 44 workers, the speedup is more or less the same, and also with a pardegree between 72 and 88. These are the results: