ALGORITMI PARALELI ȘI DISTRIBUIȚI
Tema #3 Image processing with MPI

Termen de predare: 14 Decembrie 2018

Responsabili Temă: **Cristian Chilipirea, Andrei Crividenco, Andrei Stancu**

Objectives

The goal of this homework is to build a scalable MPI program that can apply multiple filters on images. Filters are an elemental and basic part of image processing. Many algorithms use a workflow of predefined filters in order to enhance certain characteristics of an image.
In order to apply a filter to an image, we update the value of each pixel with the sum of the resulting values by multiplying each element from the filter kernel matrix with the value of each pixel and its surrounding neighbors. The number of neighbors depends on the size of the kernel matrix. For this homework we only consider filters of 3×3.

Part 1. The filters

Your program will take as input an image and will apply any combination of the following filters on it: "smooth", "blur", "sharpen", "mean" and "emboss".
Considering that filters are applied for every pixel and its neighbors, what happens with the border? Pixels on the border don't have all the required eight neighbors. It is good practice to add another border of a certain color, black or white or neutral, so the filter can be applied on the entire image. However, for this homework we simply leave the border pixels untouched.
The identity filter:

$$K = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Imagine multiplying all the values from this matrix with all the values in a sliding window that goes over the image. We then sum all the elements in the result and obtain the value of the center pixel. This is why this filter is called the identity filter. If we apply it, we leave the entire image unchanged.

A. Smoothing filter The goal here is to create a modify the pixel so that it takes the mean value of itself and all the pixels around it:

$$K = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

B. Approximative Gaussian Blur filter The goal of this filter is to reduce image noise and detail:

$$K = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

C. Sharpen The opposite of Blur, the goal here is to make differences more obvious:

$$K = \frac{1}{3} \cdot \begin{bmatrix} 0 & -2 & 0 \\ -2 & 11 & -2 \\ 0 & -2 & 0 \end{bmatrix}$$

D. Mean removal Similar with Sharpen, but this filter uses the diagonal pixels as well:

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

D. Emboss This filter is specifically used for edge detection. The following matrix is focused on vertical edges and it is one of multiple emboss versions:

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Part 2. To consider

The homework takes as input .pnm or .pgm pictures and outputs an image in the same format. The size of the output image should be equal to the size of the input image.
Your solution needs to scale. Because communication is a major factor in distributed systems, you should make sure that you try to limit it as much as possible.
Before you start implementing, try to consider what happens at the border between the responsibilities of two processes. After it starts applying the filters. does one process require data from others?

Running, input/output

Your program will be run in the following way:

```
mpirun -np N ./homework image_in.pnm image_out.pnm filter1 filter2 ... filterX
```

There can be any number of filters and the output image should be printed only after all of them are applied. An image file will have the **.pgm/.pnm** format. The homework only needs to process a simplified version of the file formats, as listed below:

```
P(5 or 6)\n
width height\n
maxval (max 255)\n
height * width * numcolors bytes representing the image
```

P5 specifies that the picture is grayscale, while P6 means that the picture is color. For grayscale images, each pixel is given as a single value between 0 and *maxval*, whereas for color images, there is a byte for each RGB channel. If the image is color, filter computations will be performed for each channel separately.

Points

The homework needs to be uploaded on the checker system. The points are as follows:

- 16 points - correct filter implementation
- 24 points - correct output regardless of the number of processes
- 60 points - scalability

A homework that does not compile gets 0 points.

Archive

Your archive should contain **only** the files “homework.c” and “Readme”.

Any attempt to abuse the system will result in 0 points for all homeworks.