Faculté des Sciences
Département d'Informatique

# INFO-F-404 – Real-Time Operating Systems

## Parallel computing project

## Image processing

December 8, 2020

## 1 Project objectives

In the scope of the new French law that forbids to upload photos and videos of policemen where their faces are not blurred, President Macron himself mandates you to create an efficient platform that will allow people to easily blur people's faces.

## 2 Project details

This project in is individual and will be implemented **in C or in Rust** with the **MPI library** in order to be as CPU-efficient as possible, as well as distributed since many people are likely to use your platform.

You are required to implement a **parallel** blurring algorithm based on the key principles of MPI message passing.

### 2.1 Image processing basics

A raw grayscale image of size $W \times H$ is a matrix of $W \times H$ subsequent bytes. Consequently, an image can be loaded in an array as shown in the below listing.

```
unsigned char image[W * H];
FILE* file = fopen("police1.raw", "rb");
fread(image, 1, W * H, file);
```

In the scope of this project, you can assume that all the images are of size $1280 \times 720$.

### 2.2 Blurring an image

One of the many ways to blur an image is to average each pixel in a neighbourhood of a given size. With a neighbourhood of size $N$, the pixel at index $(i, j)$ will be the average of all pixels from $(i-N, j-N)$ up to $(i+N, j+N)$ included (i.e. the *main* pixel is at the center of the neighbourhood).

In the scope of this project, you will take the size of the neighbourhood as argument on the command line.

#### Borders

Blurring pixels that occur on borders is a problem with regard to the algorithm that is presented here above. It is up to you to choose how to tackle this issue.



MEMBRE DE L'ACADÉMIE UNIVERSITAIRE WALLONIE-BRUXELLES ET DU PÔLE UNIVERSITAIRE EUROPÉEN BRUXELLES WALLONIE

### 2.3 Blurring mask

Since we do not want to blur a whole image but only people's faces, your program should only blur rectangles that are specified in a blurring mask file. This file contains rows that describe the top left and bottom right corners of the rectangles to blur in terms of pixel coordinate as shown here below.

`<start_i> <start_j> <stop_i> <stop_j>`
A file example is given here below[1].

```
250 325 350 415
225 500 350 615
50 825 185 980
```

### 2.4 Program inputs

Your program will take at least the following mandatory inputs, and can take additional optional inputs to your convenience.

- the neighbourhood size
- the image file to blur
- the mask to apply
- the output file name

As a result, the program can be launched as shown here below.

```
$ mpirun -n <ncores> blur <neighbourhood_size> <image> <mask> <output>
```

## 3 Report

You have to write a short report that contains at least

- a short description of what the report is about (the project, the context, the scope, ...)
- all you have done in addition to the requirements of this document (additional options, task generator, additional scheduler, ...)
- a description of your implementation choices (master/slave work dispatch, structure, ...)
- if any, a section where you describe difficulties that you met during this project and how you overcame them
- anything you think is relevant to show the work you have provided

## 4 Evaluation criteria

This project will be rated out of 20 points and aims at evaluating your ability to design and implement algorithms for distributed or parallel architectures. You will be evaluated on

- The algorithm is parallel and makes correct usage of MPI primitives /6

[1] This is the content of file `mask1` that is suitable to the image `police1.raw`.



MEMBRE DE L'ACADÉMIE UNIVERSITAIRE WALLONIE-BRUXELLES ET DU PÔLE UNIVERSITAIRE EUROPÉEN BRUXELLES WALLONIE

- The blurring algorithm works as expected /4
- The report /6
  - contains relevant information /4
  - is well structured /1
  - contains a user manual /1
- The code is readable, well structured and properly documented /4

**Total /20**

Any implementation in addition to the strict requirements of this sheet can grant you additional marks.

## 5 Handing in your project

You have to hand in a zip file containing at least

- your source code
- your report in PDF

You have to upload this zip file on the "Université Virtuelle" no later than the 13$^{th}$ of December, 23:59.

### Delays

Late projects will also be submitted on the UV with a penalty of 1 point for every 4h with at most 24h of delay.

### Questions

Please have a look at the FAQ on the project assignment page on the UV before asking questions by email (yannick.molinghen@ulb.be).



MEMBRE DE L'ACADÉMIE UNIVERSITAIRE WALLONIE-BRUXELLES ET DU PÔLE UNIVERSITAIRE EUROPÉEN BRUXELLES WALLONIE

# Additional information

## Visualisation

You can easily visualise raw images on `https://rawpixels.net/` by selecting width = 1280, height = 720 and "Grayscale 8bit" as format.

## Provided files

You are provided with four images and one mask in the data folder. The provided mask is calibrated for the image `police1.raw`.

## Installing MPI

### Install from package repository

Check with the package repository of your distribution how to install openmpi. On raspberry pi for instance, you can do the following.
`sudo apt install openmpi-bin openmpi-common libopenmpi3 libopenmpi-dev`.

On windows, you should have a look at this page `https://www.microsoft.com/en-us/download/details.aspx?id=100593`.

### Compile from source

A source installation is also rather simple by downloading openmpi on the official website `https://www.open-mpi.org/software/ompi/v4.0/`. Uncompress the file, then follow and then run the below commands.

```
$ ./configure [--prefix="/the/prefix/if/not/root"]
$ make -j4
$ make install # Can require sudo depending on the installation directory
```

### Use the provided installation

You can download a compiled version of MPI for Linux x64 on the page of the project, and put it where somewhere on your system. You should then change the configuration of the provided `Makefile` to match your configuration as shown in the example below.

```
# Directory where openmpi library is located
MPI_DIR=/home/yann/openmpi
INC_DIR=$(MPI_DIR)/include
LIB_DIR=$(MPI_DIR)/lib
BIN_DIR=$(MPI_DIR)/bin
#Compiling flags
FLAGS = -I$(INC_DIR) -L$(LIB_DIR)

all: blur.c blur.h
gcc -o blur blur.c $(FLAGS) -lmpi

run:
export PATH=$(PATH):$(BIN_DIR) && mpirun -n 4 blur
```