Universidad Carlos III de Madrid

Lab assignment 1

Software developtment, implementation and optimization in C++ and OpenMP

Área de Arquitectura y Tecnología de Computadores
Departamento de Informática

October 2016

Authors:
Estefanía Serrano, Alberto Cascajo, David Del Rio, Jose Daniel Garcia

Chapter 1

Introduction

The main goal of this project is to gain knowledge about parallel programing in shared memory platforms. To do so, the work in this project is to develop an aplication using the language C++ and the OpenMP parallel framework.

This aplication is comprised by different utilities for image processing, that are stored in a specific format: ARCfmt.

This document describes the application requirements and the necessary information for its development. Additionally, this document presents the qualification criteria and work submission conditions.

Chapter 2

Application requirements

The work associated to this project should comprise different image processing utilities. Also, the capabilities to read and store those images in a specific format are required: ARCfmt. The specification of this format and the utilities to be developed are described in the following sections.

2.1 Description

The aplication will be implemented using C++ and parallelized leveraging the OpenMP parallel framework. The main requiremens are the following ones:

- The execution of the aplication is through command line.
- The aplication is targeted to Linux systems.
- It should be able to read and write images in the ARCfmt format.
- The application should permit to transform the image to gray scale and compute histograms.
- It should be able to return the maximum and minumum values for each color channel.
- It should be able to transform part of the selected image to grey scale.
- The aplication should provide the capabily to rotate an image in specific angles.
- Finally, it should be able to read a mask file (in the same format: ARCfmt) and apply it to the image.

In order to complete the whole project, it is required to implement **two applications** fulfilling the aformentioned requirements: a sequential application that do not use any kind of parallelism and a parallel version leveraging the OpenMP framework to improve the performance.

2.2 Command line execution

Both applications will be executed by command line and their names are the following:

ARCfmtut_seq For the sequential version.

ARCfmtut_par For the parallel version.

Additionaly, both applications should require the argument to specify which utility will be executed. This argument must be specified through the option *-u*.

-u 0 Gray scale histogram.

-u 1 Maximums and minumums.

-u 2 Apply a mask.

-u 3 Image rotation.

-u 4 Selective gray scale conversion.

Additionaly, all the aformentioned execution modes should receive two additional arguments, **-i** and **-o**, to specify the input and output image files, respectively.

In the following subsections detail the specific arguments that should be introduced for the different execution modes.

2.2.1 Additional arguments for the Histogram

-t <integer number> Number of intervals of the histogram.

2.2.2 Additional arguments to apply a mask

-fm <path> File path that contains the mask.

2.2.3 Aditional arguments for image rotation

-a <decimal number> Sexagesimal degrees to rotate the image.

2.2.4 Aditional arguments for the selective gray scale conversion

-r <positive decimal number> Radius in pixels of the circle that selects the part of the image that is not transformed to gray scale.

2.3 Reading and writing files in ARCfmt format

The aplication uses a image file in a specific format: ARCfmt. This format encodes non compressed images in the RGB color space.

The ARCfmt format specification is the following one:

- The **first four bytes** specify the height in pixels.
- The **next four bytes** specify the image width in pixels
- Next, all the Red (R) values are stored consecutively, then the Green (G) values and, finally, all the Blue (B) values. Each value is represented by an unsigned byte that takes values from 0 to 255.

Therefore, the image total size in byte will be:

$$size = 4 + 4 + (height * width) * 3 \tag{2.1}$$

Figure 2.1 shows an example of an image in this format with a size of 2 (height) x 3 (width).

In this format the image is ordered by rows, for example, focusing on the red values, the first 3 bytes store the red values of the first row and the next three bytes contain the values of the second row.

2.4 Histogram

This utility should transform the input image to gray scale and, then, compute the histogram. The equation to transform from RGB to gray scale is the following one:

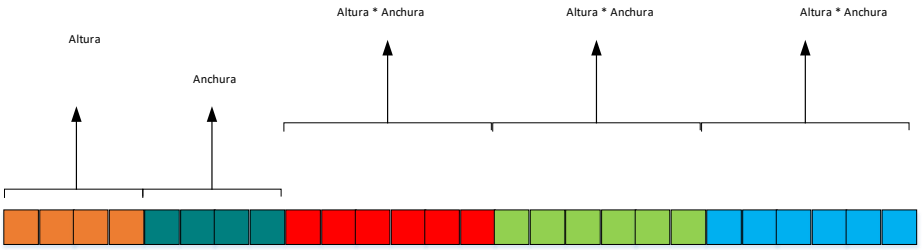

FIGURE 2.1: Example of an ARCfmt file with a size of 2x3 pixels. Each byte is represented by a square.

$$g_s = (r_i * 0.3 + g_i * 0.59 + b_i * 0.11) \tag{2.2}$$

Afterwards, in order to compute the histogram, the range of possible values (0 to 255) should be split in the number of intervals defined as an argument.

Then, the output of this utility should be a text file in which the computed values for each interval are separated by spaces. For example, the output file of the resultant histogram, computed for a square image of size of 32 pixels, where all the pixels are black, and using 8 intervals, should be the following one:

1024 0 0 0 0 0 0 0

2.5 Maximum and minimum values

In this execution mode, the application should compute the maximum and minimum pixel value for each one of the color channels. These values should be written in the output file in the following format:

<Maximum red > <Minimum red > <Maximum green > <Minimum green > <Maximum blue > <Minimum blue >

2.6 Mask filter

A mask is comprised by an aditional image in the ARCfmt format whose RGB values can only be 0s and 1s. To implement this utility it is necesary to read the mask file and compute the resultant image by multiplying the original image values by the mask values:

$$v_f(x,y) = v_i(x,y) * v_m(x,y) \tag{2.3}$$

Where x and y are the pixel coordinates, $v_f(x,y)$ is the resultant value of the pixel, $v_i(x,y)$ is the original value and $v_m(x,y)$ is the value corresponding to the mask. This computation should be performed in each one of the RGB channels.

2.7 Image rotation

This utility shold rotate the input image, with respect to its center, by the angle received as argument. The necessary fomulas to perform the rotation are the following:

$$x_f = \lceil cos(\theta) * x_i - sin(\theta) * y_i \rceil \tag{2.4}$$

$$y_f = \lceil sin(\theta) * x_i + cos(\theta) * y_i \rceil \tag{2.5}$$

Where x_i y y_i are the original coordinates of the pixel with respect to the center of the image, x_f and y_f are the resultant coordinates and θ is the rotation angle. Note that the resultant coordinates should be rounded to the smallest following integers.

To obtain the coordinates with respect to the center of the image is necessary to compute the center:

$$center = (width/2, height/2) \tag{2.6}$$

Afterwards, the coordinates with respect to the center can be computed as follows:

$$(x_i, y_i) = (x - width/2, y - height/2) \tag{2.7}$$

2.8 Selective gray scale conversion

This execution mode will apply a gray-sacle filter to the part of the image that is out of the circle defined by the radius received as argument. To do so, it has to be considered that the center of the circle is on the center of the image:

$$center = (width/2, height/2) \tag{2.8}$$

To delimitate the circle that contains the part of the image in which the filter is not going to be applied, it is necessary to compute the circumference of the circle:

$$(a^2 + b^2) = r^2 \tag{2.9}$$

So that, the filter will be applied on the pixels that comply the following requirement:

$$(a^2 + b^2) > r^2 \tag{2.10}$$

Being a and b the coordinates of the pixel with respect to the center of the image.

The equation to transform from RGB to gray scale is the following:

$$(r_s, g_s, b_s) = (\lfloor r_i * 0.3 \rfloor, \lfloor g_i * 0.59 \rfloor, \lfloor b_i * 0.11 \rfloor) \tag{2.11}$$

Note that, the resultant values should be rounded to the largest previous integer.

2.9 Compilation

The code should be compiled with the g++ compiler and using the following flags: -Wall -Werror -std=c++11.

On the sequential application, the compilation should be:

g++ -Wall -Werror -std=c++11 ARCfmut_seq.c -o ARCfmut_seq

On the case of the parallel application is necessary to include the flag corresponding to the OpenMP parallel framework (-fopenmp):

g++ -Wall -Werror -std=c++11 -fopenmp ARCfmut_par.c -o ARCfmut_par

Chapter 3

Evaluation and sumbission

3.1 Evaluation

The final mark of this project is obtained as follows:

- Oral presentation : 30%
- Implementation mark: 70%
  - Sequential implementation 40%
    - Funcional test 50%
    - Report 20%
    - Classification in the competition 30 %
  - Parallel implementation 60%
    - Funcional test 50%
    - Report 20%
    - Classification in the competition 30 %

The mark obtained from the clasification is the following:

- 30% fastest applications : 10
- Next 20% : 7
- Next 10% : 5
- Rest: 0

The oral presentation will consist in a presential sesion in which different questions about the developed work should be answered by the group members.

For example: a code that has passed al the functional tests (both sequential and parallel tests)(10), a not perfect document but close (7), both applications in the 20% fastest aplications and a perfect defense (10), the overall mark will be:

$$Sequentialimplementation = (((0.4 * ((0.5 * 10) + (0.2 * 0.7) + (0.3 * 7))) \quad (3.1)$$

$$Parallelimplementation = (0.6 * ((0.5 * 10) + (0.2 * 0.7) + (0.3 * 7)))) \quad (3.2)$$

$$\text{Overall mark} = (10*0.3)+((\text{Sequential implementation} + \text{Parallel implementation})*0.6) = 7.344 \quad (3.3)$$

NOTE: If the submmited code can not be compiled, the overall mark will be 0.

NOTE2: If any copy is detected, the overall mark of every member of the involved groups will be 0.

3.2 Workings of the competition platform

In order to detail the use of the competition platform and the correct submission, an auxiliar document will be provided. The platform webpage is the following: `http://tucan.arcos.inf.uc3m.es/domjudge/public/login.php`

This project should be developed by groups composed by 4 or 5 people. All the members that composes a group must belong to the same reduced group. Additionally, a member of the group should send an email to their laboratory teacher with the following information:

- Name, surname and NIAs of the group members.
- Number of the reduced group.

As an answer, the group will receive a password for the platform. This password will be the same for all the members of the group and the user name will be the NIA.

We recomend to keep this information safe to avoid potential copies or improper modifications by third parties.

3.3 Submission

The submission of the code should be done through two systems:

- The code should be uploaded to the platform and obtain a valid time result (so that, it has to pass all the tests).
- The code along with the document should be uploaded in the corresponding assignment activity on Aula Global. The code should be in compressed in a .zip file with the following name: **<group number >_g <reduced group number >.zip**. For example: 2_g89.zip. The contain of the zip file should be:
  - A folder with the sequential code, named **seq**, that contains all the necessary files to compile and run the code.
  - A folder with the parallel code, named **par**, that contains all the necessary files to compile and run the code.

  The document should be uploaded into a separate assignment activity. This document should contains, at least, the following sections:
  - Title page: should contain the title of this project, names y NIAs of the authors and the group and reduced group number.
  - Table of contents.
  - Introduction.
  - Sequential version: This section should explain the implementation and optimization of the sequential code. Do not place the code in this section.
  - Parallel version: This section should detail the implementation and optimization of the parallel code. Do not place the code in this section.
  - Conclusion.

  This report should not exceed 15 pages.

The code submitted to both the platform and Aula Global should be the same. The deadline is **November 4th** at 23:55.