Software Concorrente e Fiável
Mestrado em Engenharia Informática
Instituto Superior de Engenharia do Porto

Digital Image Processing and Analysis in Multicore Systems

Introduction

The goal of this project is to implement digital image processing techniques using a sequential, multithreaded and threadpool-based approach, to study how to maximize the eventual performance gain and analyze the results.

Highlight Forest Fires

Forest fires are sometimes captured by satellite images. The goal of the first part of the project is to filter satellite images with fires to highlight the fire. For each input image, one must create a new image based on the original one where visible fire pixels are replaced by red pixels and the remaining pixels are replaced with their grayscale equivalent. In figure 1 one can find a satellite image of a forest fire that took place in Siberia, Russia in September 2021 and was captured by a Sentinel satellite [1]. In figure 2 areas with perceived fire have been replaced by red pixels and the remaining areas are now in grayscale.

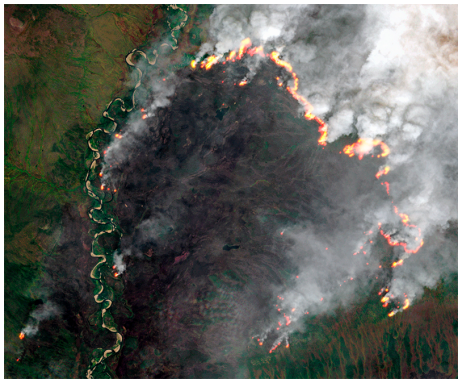

Figure 1: Original photo

Grayscale pixels can be obtained by replacing their red, green and blue values by their average. Thus, given (r)ed, (g)reen and (b)lue values for a pixel, replacing each of these values by:

$$avg = \frac{(r + g + b)}{3}$$

will result in a grayscale equivalent pixel.

[1] https://sentinel.esa.int/web/sentinel/home

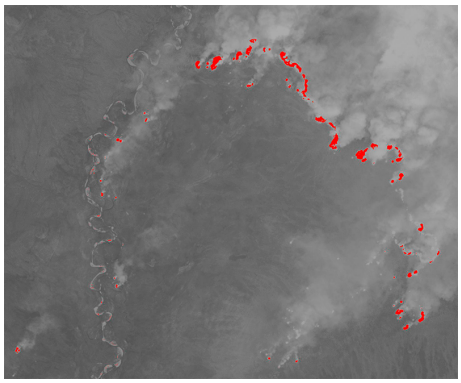

Figure 2: Fire pixels translated to red pixels and remaining ones in grayscale

The detection of the fire pixels is more tricky and we will opt to follow a simple but error prone approach. To the interested student, there are many research works out there on how to do this with high precision but since that analysis is out of the scope of this project we will just follow some simple rules described next to know if a pixel is a fire pixel:

- The red pixel is higher than the average values of the red, green and blue values of the pixel multiplied by a threshold:

$$r > \frac{(r+g+b)}{3} * threshold$$

- The green value is bellow some value v_g
- The blue value is bellow some value v_b

If the three rules are true then, the pixel is translated into a red pixel in the output image, meaning that $r = 255$, $g = 0$ and $b = 0$.

Example 1. *Given a threshold of* 1.35, $v_g = 100$ *and* $v_b = 200$ *then, a pixel* p *with red value* p_r*, green value* p_g *and blue value* p_b *is considered a fire pixel if:*

$$p_r > \frac{(p_r+p_g+p_b)}{3} * 1.35 \text{ and } p_g < 100 \text{ and } p_b < 200$$

Thus, a pixel with (r, g, b) *values of* $(250, 99, 150)$ *is converted into a red pixel.*

Starter code is provided and includes a sequential implementation of this algorithm. Feel free to change the way how to detect fire pixels. Several images with different sizes and resolutions are also available for testing purposes.

Clean Image

In the second part of the project the goal is to process digital images removing objects and people from the image and creating a cleaner version. Here the input consists of an arbitrary number of images that will result in a new image without any permanent objects and people. In figure 3 one can find an example of such application. To solve this problem we need several images of the same place and compare them to remove everything that is not part of the scenario. To create an image O from sources $S_1, \ldots, S_n$ we compute each pixel (x, y) of O as follows:

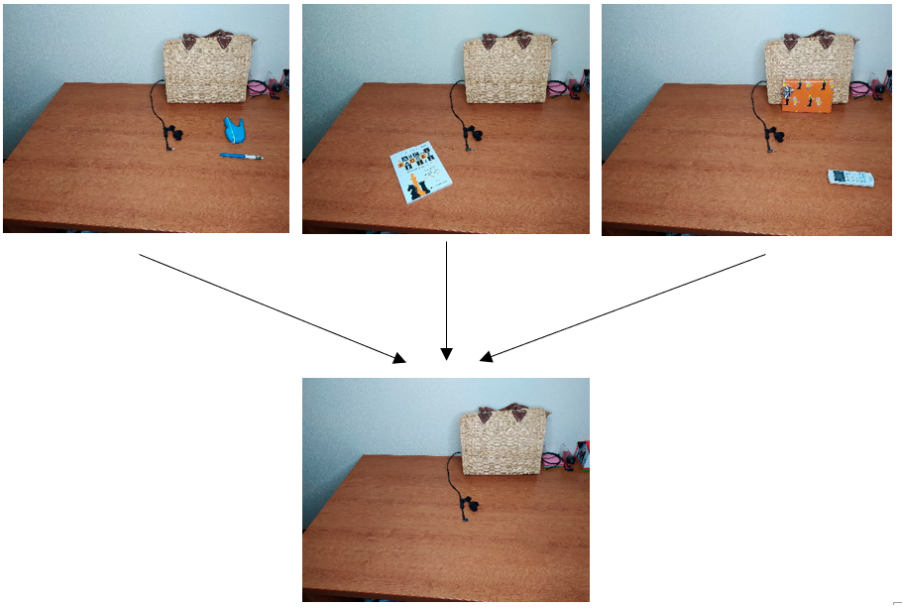

Figure 3: Cleaning an image

- Computing the average of values for all the correspondent pixels in source images.
- Calculate the distance of the RGB values of each pixel in source images to the average.
- Choose the closest one.

More formally, given a set of images $\{S_1, \ldots, S_n\}$ such that for each S_i, the pixel in (x, y) is represented by $S_i(x, y) = (r_i, g_i, b_i)$ then, the average value of red r_a is given by

$$r_a = \frac{(r_1 + \ldots + r_n)}{n}$$

the average value of green g_a is given by

$$g_a = \frac{(g_1 + \ldots + g_n)}{n}$$

the average value of blue b_a is given by

$$b_a = \frac{(b_1 + \ldots + b_n)}{n}$$

So, the average of all of them is $\mathcal{A} = (r_a, g_a, b_a)$ and the distance d_i of a given $S_i(x, y) = (r_i, g_i, b_i)$ to $\mathcal{A}$ is given by:

$$d_i = \sqrt{(r_a - r_i)^2 + (g_a - g_i)^2 + (b_a - b_i)^2}$$

Then, from the set of distances $\mathcal{D} = \{d_1, \ldots, d_n\}$ get the minimal d_k such that $\forall d_i \in \mathcal{D}, d_k \leq d_i$ and use the pixel (r_k, g_k, b_k) for the output image. An example is presented next.

Example 2. *For an image with 3 sources* S_1,S_2 *and* S_3*, with the value of pixel* (x, y) *for each of them being* $S_1(x, y) = (1, 2, 3)$*,* $S_2(x, y) = (1, 4, 3)$ *and* $S_3(x, y) = (23, 43, 9)$*, then*

$$r_a = \frac{(1 + 1 + 23)}{3} = 8$$

$$g_a = \frac{(2 + 4 + 43)}{3} = 16$$

$$b_a = \frac{(3 + 3 + 9)}{3} = 5$$

Thus, the average is $(8, 16, 5)$ *and for* $(1, 2, 3)$ *we have:*

$$\sqrt{(8-1)^2 + (16-2)^2 + (5-3)^2} = 15$$

For $(1, 4, 3)$*:*

$$\sqrt{(8-1)^2 + (16-4)^2 + (5-3)^2} = 14$$

And finally, for $(23, 43, 9)$*:*

$$\sqrt{(8-23)^2 + (16-43)^2 + (5-9)^2} = 31$$

Thus, the choosen pixel for the target image is $(1, 4, 3)$

Sequential implementation

The first task is to create a sequential implementation of the filters. A simple API is provided to translate between images and matrices of Color objects and vice versa along with reading and writing images from and to the filesystem. There are also references that can help to understand such implementations for example the book Digital Image Processing 4th edition [1] and the book Computer Science: An Interdisciplinary Approach [2] and the web site of the former book [2].

Multithreaded implementation

The second part of this project is to create a multithreaded implementation of the filters were the images are divided and split by different threads that should optimize the application of the filters relying on multicore architectures. Figure 4 exemplifies this process. Note that it is just an example and defining the number of threads and the amount of work for each one of them is part of the projects' objectives.

Threadpool-based implementation

The third part of this project is to use threadpools to solve the problem. Meaning that thread creation and teardown should be of the responsibility of a threadpool (as presented in figure 5 - image from Wikipedia page [3]). The amount of work to be feeded to the threadpool should be carefully analyzed.

[2] `https://introcs.cs.princeton.edu/java/31datatype/`

[3] `https://en.wikipedia.org/wiki/Thread_pool/`

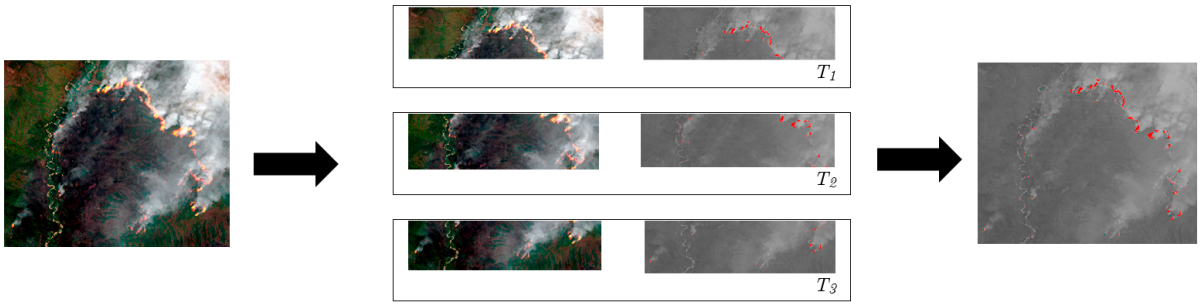

Figure 4: Image processing split by threads T_1, T_2 and T_3

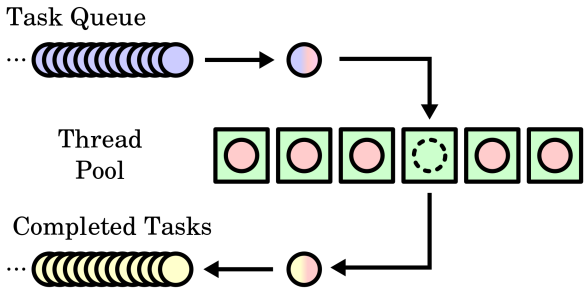

Figure 5: Example of threadpool

Concurrency

This is a problem that fits in the domain of parallel programming but, if at some point, concurrency is necessary its use should be explained carefully.

Generation of results

The last part of this project consists in measuring and analyzing the results of the different approaches. The developed application should measure time of execution of the application of the filters for images of different size or given different number of images as input. All the data should be gathered in automatically generated tables/charts and discuss them in the report.

Starter Code

Starter code is provided. It includes both a simple API to the images and an example of the application of one of the filters. The API is briefly described next:

Color[][] loadImage(String filename) : given the path to an image file in **filename** returns an array **Color[][]** of pixels of that image.

void writeImage(Color[][] image,String filename) : given an array of **Color** objects and a path to a file in **filename**, writes the array to that **filename** creating an image in the filesystem.

Color[][][] copyImage(Color[][][] image) : creates and returns a copy of an array of **Color** objects. This is useful when it is necessary to work over a copy of the image.

The file **ApplyFilters.java** uses the simple API described before to create an object **Filter** that reads an image from the filesystem, applies the *Highlight Fire Filter* to that image and writes the result back to the filesystem.

Report

The report should include the following content:

- Cover with title and the identification of the class, and authors (number and full name)
- Introduction
- Sequential Implementations
- Multithreaded implementations
- Threadpool-based implementations.
- Performance comparison and discussion
- Conclusions

Note that in each chapter the implementation should be described by text and add essential snippets of code (not all the code).

Grading

Project is to be solved individually or in groups of maximum, two (2) elements, but evaluation is individual upon the work performed by each student. Grading criteria is described next.

Description	Percentage (%)
Implementation of image filters	15
Multithread-based processing	25
Threadpool-based processing	25
Generation of tables/charts	15
Final report	20

Scheduling

The final deadline for the submission of this project is 13^{th} of June of 2022 and the presentations will take place on the following days. There will be partial presentations on the weeks of 18^{th} of April and 23^{rd} of May as described in the following table.

Description	Deadline	Presentation date
Implementation of image filters and Multithread-based processing	18^{th} of April	same week
Threadpool-based processing	23^{rd} of May	same week
Analysis of results and discussion	13^{th} of June	TBD

Code of honor

In "Código de boas Práticas de Conduta" from 27th of October of 2020, it is stated that students must submit a declaration as described in the Article 8 for each submitted project. This declaration is a signed commitment of the originality of the submitted work. Failure to submit this declaration will remove the work from evaluation. Submitting a work that does not comply with the signed declaration will have legal consequences.

References

[1] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Pearson, 2018.

[2] Robert Sedgewick and Kevin Wayne. *Computer Science: An Interdisciplinary Approach*. Addison-Wesley Professional, 1st edition, 2016.