# 

PrOgramming Languages

IMAGES FILTERS USING CUDA

# ITESM Campus Queretaro

# José Andrés Cortez Villao - A01209780

Contenido

[Introduction 2](#_Toc530467739)

[Motivation 3](#_Toc530467740)

[Objectives 4](#_Toc530467741)

[Results 4](#_Toc530467742)

[Programming Paradigm 4](#_Toc530467743)

[Image Results after being filtered 5](#_Toc530467744)

[Speed Up Analysis 5](#_Toc530467745)

[Code Explanation 5](#_Toc530467746)

[Problems Faced 5](#_Toc530467747)

[Conclusion 6](#_Toc530467748)

[Areas of improvement 6](#_Toc530467749)

[Set-up instructions 6](#_Toc530467750)

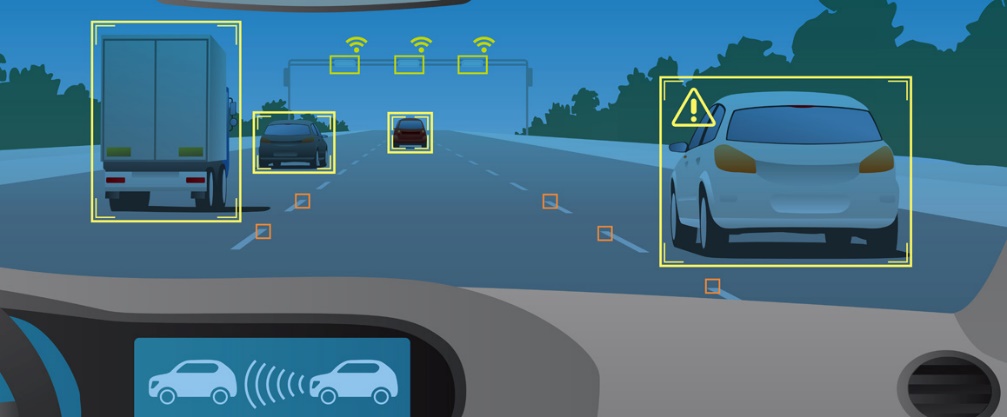
[Points to take in consideration 6](#_Toc530467751)

[Steps for run the program 6](#_Toc530467752)

[References 6](#_Toc530467753)

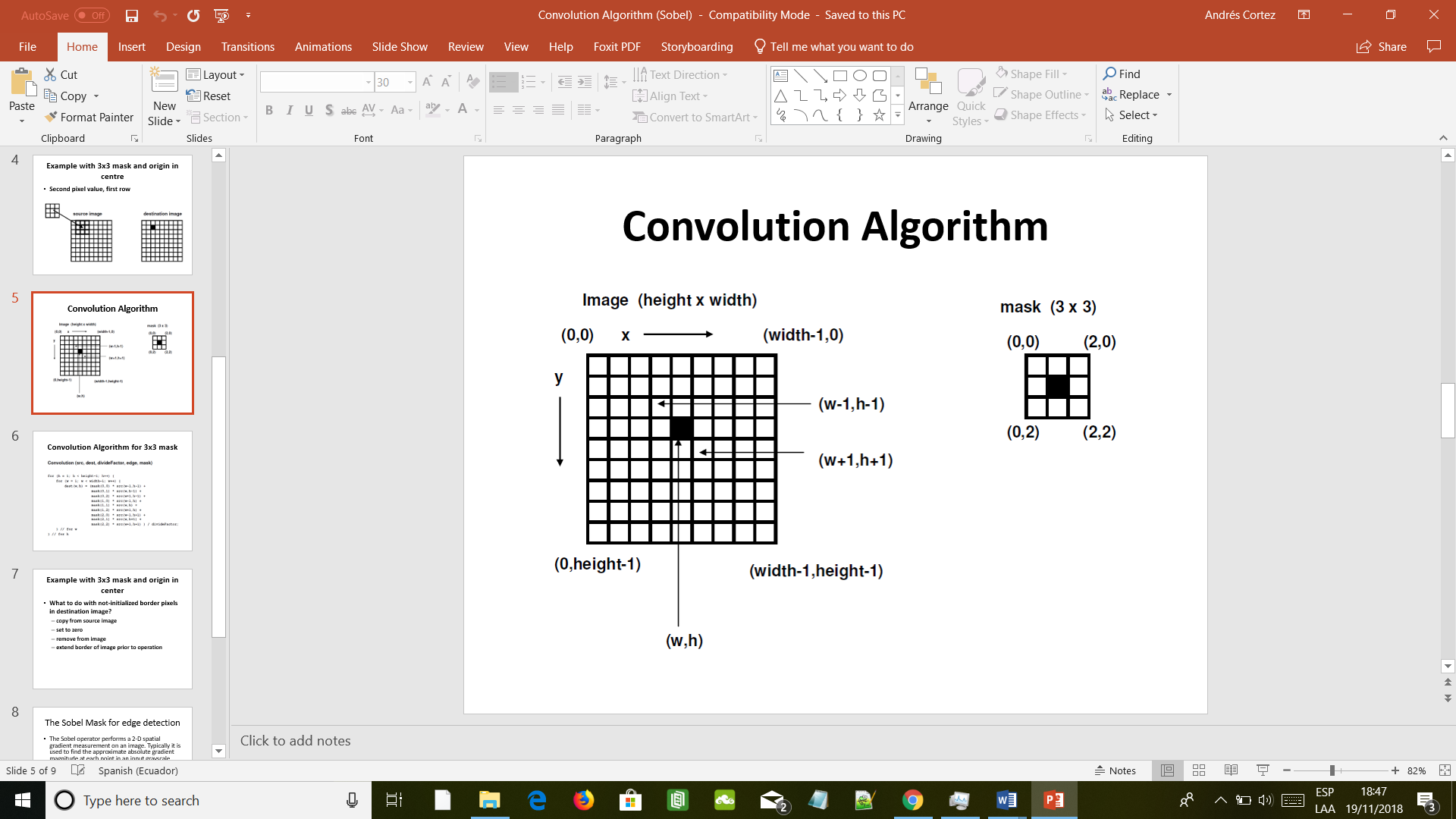
# Introduction

Image processing is the use of computer algorithms to transform the initial image in a new desire one. The implementation of image processing has different utilities depending on the field of use. The automobile industry is taking great advantage of this algorithms to improve the vehicles and enhance their technology, image processing is being used to detect objects and follow lines for creating semi-autonomous vehicles as seen in **Fig 1**, is expected that in a couple of years the vehicles will be fully autonomous thanks to image processing.



**Fig 1: Image processing used by cars.**

Sobel operator is an algorithm that is used to find the edges of an image, the operator performs a 2-D spatial gradient measurement on an image. Typically, it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image. The Sobel edge detector uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows). A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time(Kanoupolous, 1988).



**Fig 2: Sobel mask convolution algorithm**

To implement the Sobel operator, the input image must be in gray scale. To convert an RGB image to gray scale an average of the three colors must be compute for each pixel of the image, so to find the edges of an image there are two steps to be done first the gray scale and then the Sobel operator.



**Fig 3: Sobel operator implementation**

# Motivation

Robotic Vision is one of the main parts of a robot, the correct implementation of it is crucial for the correct functioning of a robot. Nowadays is being widely used in the car industry and it is amazing the incredible growth that is happening in this sector. There is also being used for medical purposes to enhance the results that they received from the scan machines, thus increasing the diagnosis accuracy being able to save more lives. The use of robotic vision can be used in almost any sector like the construction one, or even there are prototypes that wants to implement the image processing for rescuing purposes.

# Objectives

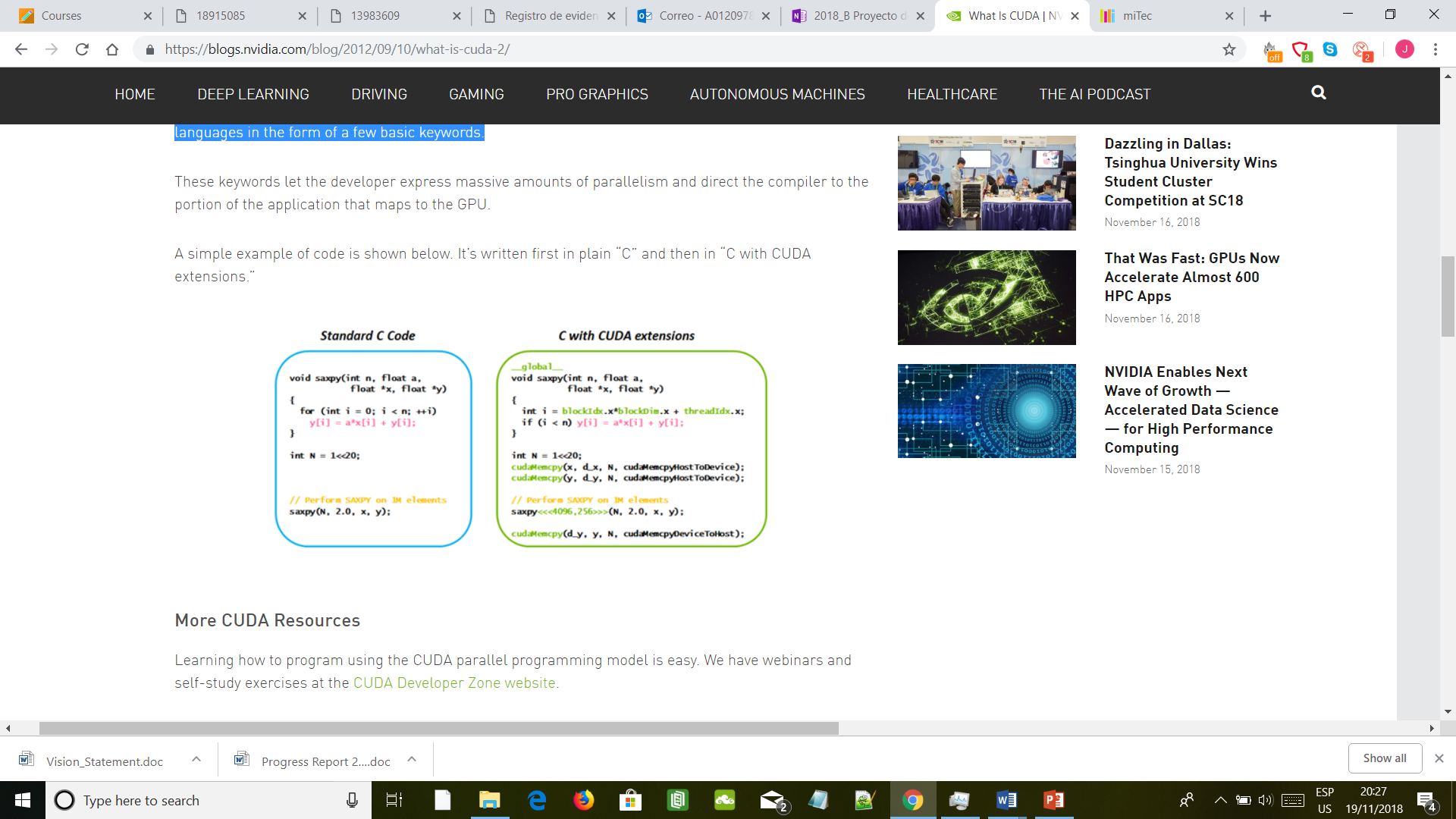
* Implement the Sobel Operator using CUDA
* Compare the Times of a sequential implementation of Sobel with a CUDA implementation

# Results

## Programming Paradigm

For the implementation of the Sobel operator I decided to use CUDA instead of the others programming languages and paradigms that we saw due to the huge amount of Threads and Blocks that I can use, this will allow me to reduce the time considerably compared to other tools that we saw. CUDA is an excellent option when we talk about image processing and it uses the GPU with his multiple cores to modify each pixel of the image in parallel (depending on the GPU). I also can used other tools like Java Threads or Fork Join but they cannot match CUDA speed.

CUDA is a parallel computing platform and programming model that makes using a GPU for general purpose computing simple and elegant. The developer still programs in the familiar C, C++, Fortran, or an ever-expanding list of supported languages, and incorporates extensions of these languages in the form of a few basic keywords (NVIDIA,2012).



**FIG 4: Example of C code and CUDA code**

## Image Results after being filtered

I tested the program with three different images of different sizes to obtain different times result that can be compared in time and execution.

Lenna.png

boca.png

House.png

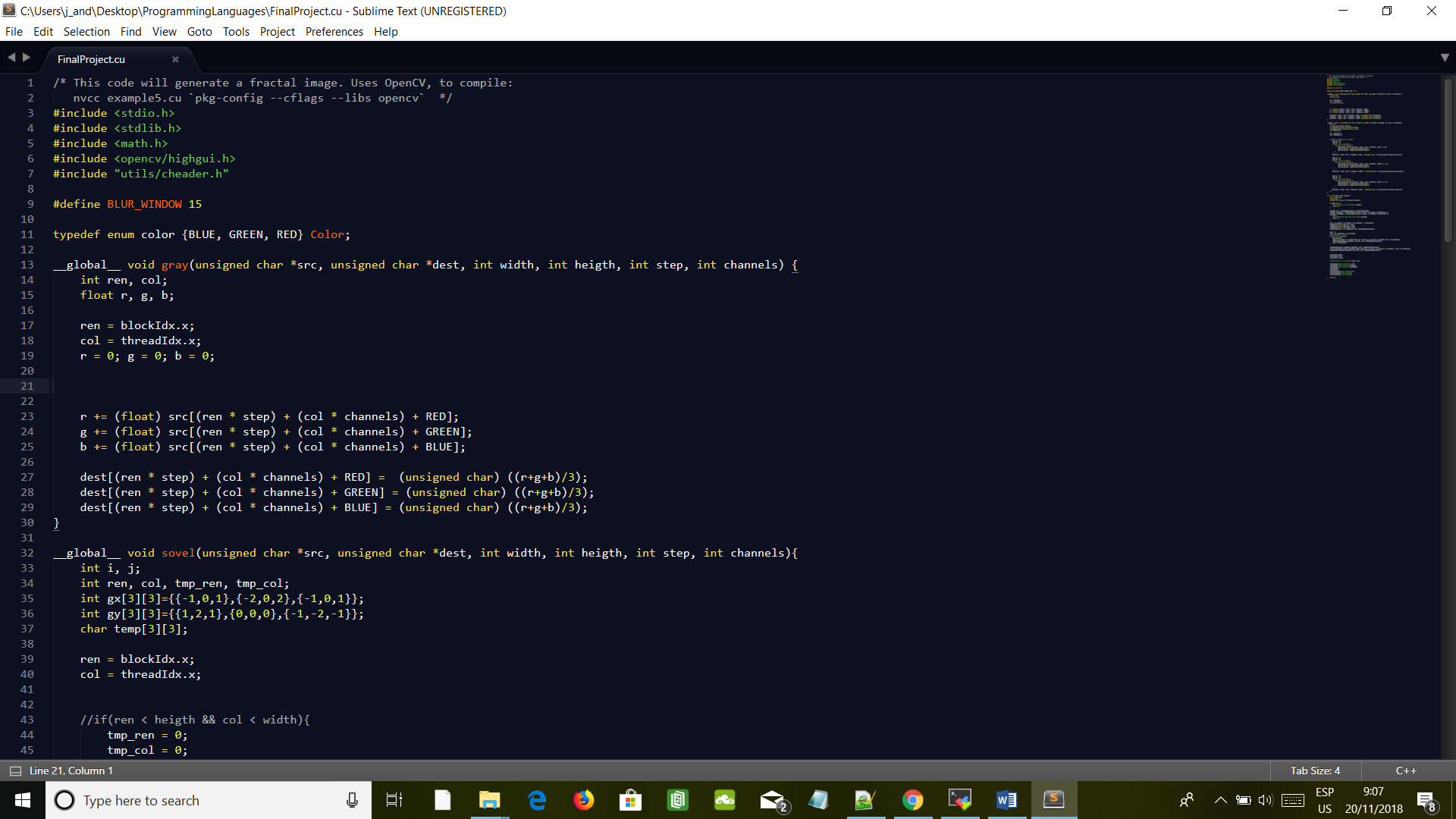
## Speed Up Analysis

|  |  |  |
| --- | --- | --- |
| Image | Parallel | Sequential |
| Lenna.png |  |  |
| RollingStones.png |  |  |
| House.png |  |  |

# Code Explanation

### Gray Function

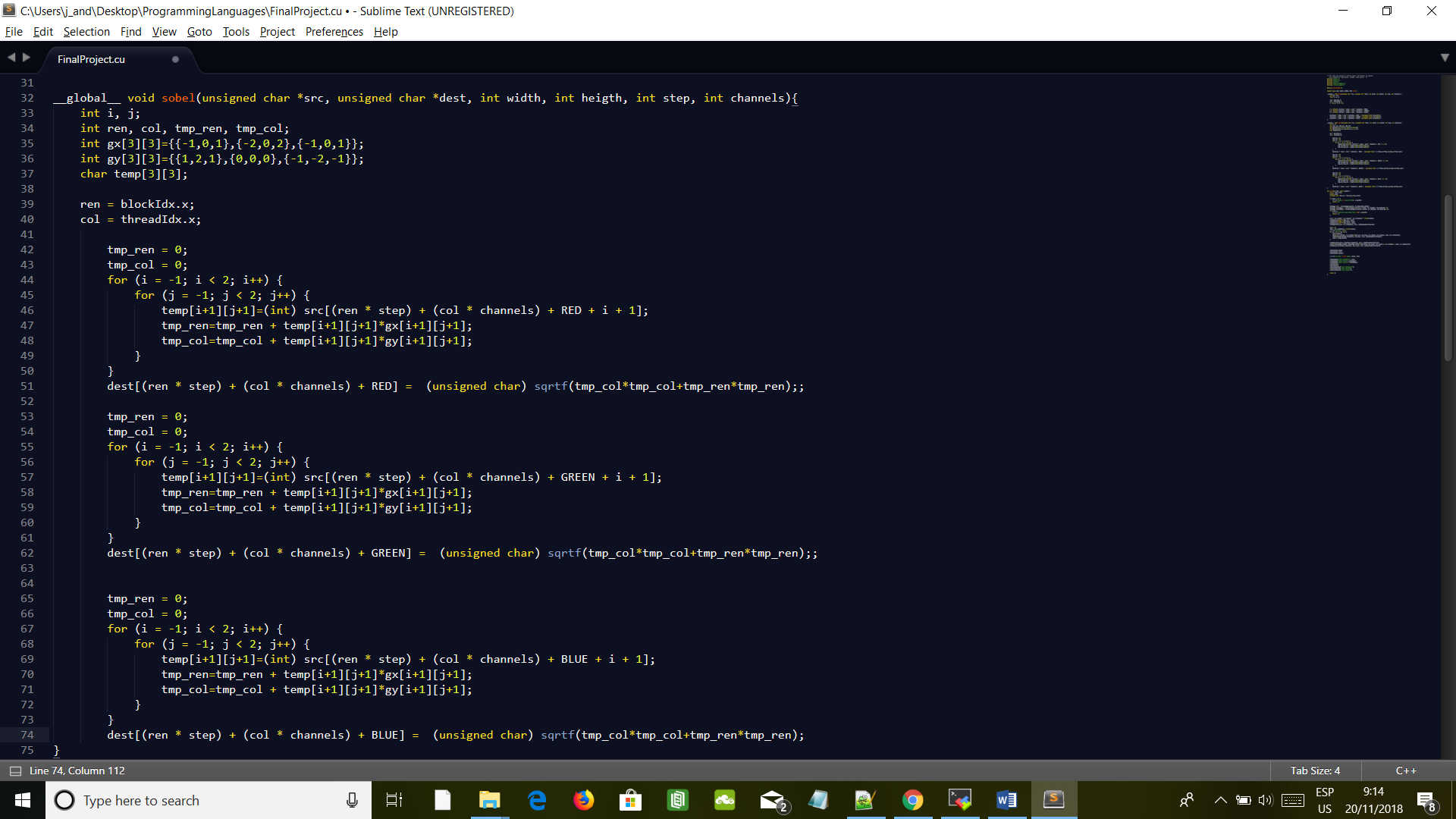
Converts the received image \*src to a gray scale image \*dest by obtaining the average of each color of the pixel of the image. Each Row is assigned for a Thread and the column with a Block, so it works in a parallel way for each pixel. The receiving image array and the destination is a one-dimensional array being the reason why step and channels variables are used to convert a two-dimensional array into one



**Fig x: Fragment of the code with the Gray function**

### Sobel Function

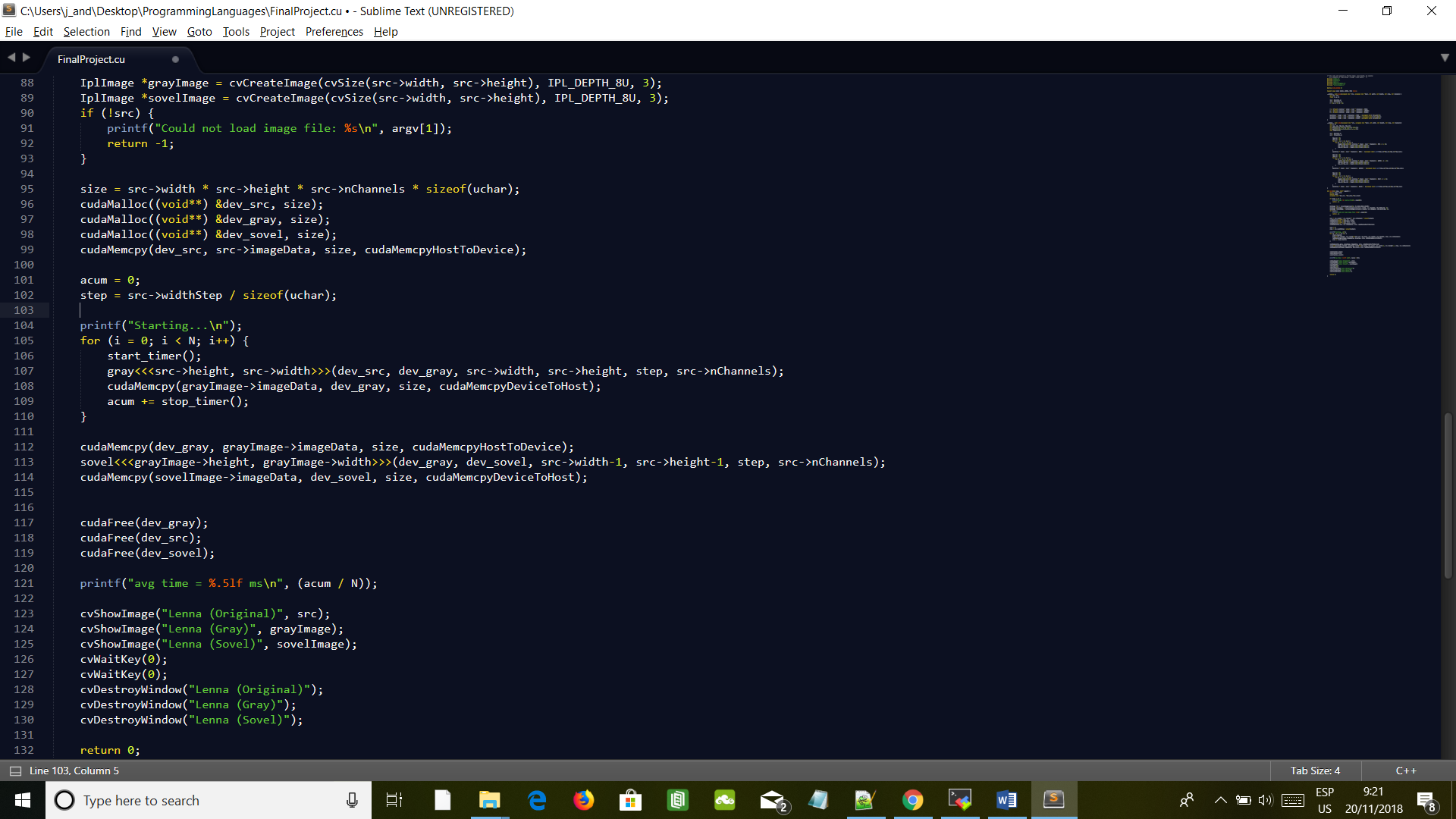
As explained previously the Sobel Operator realize a convolution to a grid of the matrix surrounding each pixel, this function applies this convolution to each color of the image linearizing this process ending with a liner array. The function is done in parallel as the Gray function.



**Fig x: Fragment of the code with the Sobel Function**

### Function Synchronization

In order to avoid race condition due to accesing information of the array by both functions at the same time it was necessary to call the gray function first and then wait until it finishes to call the Sobel function as we can see in the code.



**Fig x: Fragment of the code with the synchronization step**

# Problems Faced

I faced several problems when implementing the Sobel algorithm:

* In order to do the Sobel Operator, I must input a gray scale image, so I have to create another function that convert the original image to gray scale and pass it to the Sobel function.
* Because I am using CUDA to convert the images, the threads will be working on different pixels at the same so if I run the two functions simultaneously they will encounter a race condition trying to access the same pixel at the same time, to correct this first I do the gray scale function and then I realized the Sobel operator.
* The Sobel operator works with a two-dimensional matrix doing the convolution, but the opencv return a cubic matrix (a two-dimensional matrix for each color), so the solution was to implement the Sobel operator for each color.

# Conclusion

As we can see in the results of implementing with CUDA against a sequential implementation the Speed up is high, this mean that is way better to work with CUDA in some applications like image processing because it reduces the time of execution considerably. I think that the only problem with CUDA is that not an easy language to work with, in more complex cases like gaming, I think the programming is extremely difficult, also you have to be completely aware of the memory usage in the kernel and in the GPU. For programming in CUDA there is an important key that is working with pointers that are a real pain for most of the younger programmers, eventhough of the complexity of programming in CUDA I strongly believe that is the future of programming because of the use of thousands of Threads that will make a computer much faster.

# Areas of improvement

* CUDA can process this algorithm at a rate that allow us to work with video in real time, so the next step is to apply the Sobel algorithm using video.
* Find a way to do the Gray scale conversion while doing the Sobel in order to save time.
* Improve the User interface o it became more user friendly
* Add new Filters that will improve our previous filters

# Set-up instructions

### Points to take in consideration

To run the program, OpenCV libraries and CUDA are necessary to compile the programs.

In the folder of the project there are all the libraries necessaries to run the program (OpenCV is not included) and several images for testing, images can be download and try with the corresponding path.

### Steps for run the program

1. Compile the program with the following line: nvcc FinalProject.cu `pkg-config --cflags --libs opencv`
2. After compiling a .out program will generate if the name is not indicated a file *a.out* will be created.
3. Finally run the program with the following line: ./a.out Lenna.png (./a.out *imageName)*

# Git hub Repository

https://github.com/AndresCortez/Sobel-and-Gray-filters-using-Cuda-.git

# References

Kanopoulos, N., Vasanthavada, N., & Baker, R. L. (1988). Design of an image edge detection filter using the Sobel operator. IEEE Journal of solid-state circuits, 23(2), 358-367.

Mark Ebersole (2012). What is CUDA? November 20, 2018, from NVIDIA Web Site: <https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/>