5/6/2018

Practical 1

Applying Threads on a mean mask to blur an image in C#

Rohan Pretorius Louis J Scheepers Stephan Gunter

26198738 26368676 26603764

Lecturer: Henry Foulds

Index:

[1. Introduction to the Report: 1](#_Toc513473652)

[2. Introduction to Threading: 2](#_Toc513473653)

[2.1 Hyper-Threading: 2](#_Toc513473654)

[2.2 Multiple Cores: 2](#_Toc513473655)

[3. Threading: 3](#_Toc513473656)

[3.1 What is threading? 3](#_Toc513473657)

[3.2 Synchronous Threads 3](#_Toc513473658)

[3.3 Asynchronous Threads 4](#_Toc513473659)

[4. Masks and Filters 4](#_Toc513473660)

[4.1 What are masks and filters? 4](#_Toc513473661)

[4.2 Blurring 4](#_Toc513473662)

[4.3How do you use masks to blur an image? 5](#_Toc513473663)

[4.3.1 Mean Filter 5](#_Toc513473664)

[4.3.2 Weighted average filter 5](#_Toc513473665)

[5. Project description: 5](#_Toc513473666)

[6. Hypotheses and results: 6](#_Toc513473667)

[6.1 Hypothesis 6](#_Toc513473668)

[6.2 Results 6](#_Toc513473669)

[7. First phase of the Project (Applying the 3x3 “mask”): 6](#_Toc513473670)

[8. Second phase of the Project (Implementing Threads): 7](#_Toc513473671)

[9. Third phase of the Project (Splitting the “Bitmap”): 7](#_Toc513473672)

[10. Fourth and Final phase of the Project (Applying each part of the split “Bitmap” to a single Thread): 8](#_Toc513473673)

[11. Final Project Results: 8](#_Toc513473674)

[12. Bibliography: 10](#_Toc513473675)

[12.1. For Coding: 11](#_Toc513473676)

# 1. Introduction to the Report:

In this report we will discuss threads, their history, how they work, and how synchronous and asynchronous threads differ from each other.

Next, we will discuss the concept and implementation of masks in terms of traversing an image file in order to make it blur, and how the image is stored as a bitmap.

After discussing the history and concepts of threading we need to understand the problem we have been given we will then proceed to discuss how we intend to solve it.

This problem can be summarized as: Developing a program in either Java or C# which uses multi-threading to blur or distort images.

We agreed that the program should satisfy the following requirements:

* The program should be developed in C#.
* The program should allow a user to upload an image file to be blurred, otherwise a default image will be used if no input image is given.
* The program will make use of the mean filter implementation in order to blur the image.
* The image should be loaded in a picture box and stored as a bitmap
* The user should have the ability to choose the amount of threads to be used each time they wish to blur or distort an image.
* The allowable range of threads to choose from should be 1 - 64, this option can be given as a prompt or a text input field below the image.
* A timer should be present to show the time it took each thread to complete.
* Thereafter the program should execute the algorithm to blur the image and provide the blurred image as output to the user possibly displaying the time it took to execute said algorithm.

We will also discuss how we split the bitmap into an array in order to send different parts of the image to each of the amount of threads the users have chosen. How we constructed and analysed a graph depicting the results of the masking algorithm, to see how each number of threads will influence the programs runtime.

We then use this analysis to come to a conclusion and understand how increasing the amount of threads will influence our hypothesis.

# 2. Introduction to Threading:

## 2.1 Hyper-Threading:

In the early days of computers, the CPU (Central Processing Unit) contained only one core per Processor unit. With only one core per unit running interactive and real-time applications where either impractical or impossible.

This problem was addressed by the creation use of threads inside a single core. The idea behind this approach was to let each thread that was created by the CPU’s core manage a single program. With this implementation, the CPU was able to run multiple programs at the same time letting each thread run for a certain amount of time before it was either paused or completed.

Later in 2002 Intel designed the first Hyper-Threading CPU that was made available to the public. This CPU was called the Pentium 4 HT. The idea behind Hyper-Threading was that it allowed a single CPU core to appear as two different logical CPUs to the operating system. This fooled the operating system into thinking that there were two physical CPU’s installed in the hardware. The main problem however was that the CPU was still assigned a single set of resources.

While the CPU is pretending to have more cores than it should, it speeds-up the program execution using its own logical cores. By using Hyper-Threading, the CPU shares the physical execution resources among its cores. This speed things up significantly by allowing the other cores to use the resources when the executing core has completed its task or is stalled/waiting fir I/O. (Howtogeek.com, 2018)

## 2.2 Multiple Cores:

Originally, the first generation of CPUs contained only one logical processing unit. This means that it had one single core to process threads or job queries, simultaneous or real-time processing was not possible at the time. Later as CPU’s become more efficient, manufactures began to implement multiple cores on a single CPU chip.

This design allowed for the creation of multiple central processing units to be implemented on a single chip, thus improving the performance and keeping the original size of a one-socketed chip. The using hyper-threading with this design meant that a chip can have two central processing units on a single chip that is hyper threaded to mimic the appearance of 4 physical cores to the operating system. (Howtogeek.com, 2018)

# 3. Threading:

## 3.1 What is threading?

According to techopedia.com a thread is the smallest unit of processing that can be performed by an operating system. In modern operating systems, a thread exists within a process. This means that a single process can contain multiple threads (techopedia.com, 2018)

A thread can be described as a series of executing instructions that process code, with a program counter and system register which stores the current assigned variables on a stack, this stack contains the execution history. Furthermore, a thread can be called a “lightweight process” or a “sub process”. The implementation of threads allows for improved application performance through parallel execution. Threads is a software approach for improving performance on the OS (Operating System) by reducing the overhead required by an executing process (Tutorialspoint.com, 2018).

## 3.2 Synchronous Threads

Synchronization of threads prevents threads from using data or values that other threads are using at any given moment. The thread has to wait for those resources to become available before it is able to use them otherwise a variables value might be changed by a thread while another thread is using that variable in a computation and assigns it a new value. This will cause inconclusive results and may even render some resources or data values corrupt.

Synchronization techniques prevent the above-mentioned scenario from playing out, by organizing each thread so that they become in sync. Each thread will know when another thread is busy, with a resource it desires to use, and will allow them to finish and only use that resource when it becomes available.

Synchronization techniques do not only deal with resource allocation but also the following

• Deadlocks

• Starvation

• Priority Inversion

• Busy waiting for access to a critical section

(Tutorialspoint.com, 2018)

## 3.3 Asynchronous Threads

Asynchronous threads are threads that run independently from other operations such as other threads.

In an applications code the application starts running in the main method. The main method calls other operations to run sequentially one after the other. Each operation waits until the previous operation is finished. The applications execution ends when the main method returns. (Ahmed, 2018)

In asynchronous programming a method is called and runs in the background, the calling thread is not blocked. After calling the method the execution flow immediately flows back to the calling thread and performs other tasks. What this means is that when thread1() (where thread1() can be any thread) is called, it immediately goes back to the main method to execute thread2(), where thread2() can be any other thread, and so forth instead of waiting for thread1() to complete its operation before executing thread2(). (Ahmed, 2018)

By implementing asynchronous programming and threads a programs time it takes to execute is significantly reduced. This will speed up the programs operations significantly since there is no need to wait for the previous operation or thread to complete.

# 4. Masks and Filters

## 4.1 What are masks and filters?

Masks are in fact filters this also means that masking is the same as filtering. Masks are used on an image for multiple reasons, such as edge detection and sharpness or masks can be used to blur an image or to reduce noise. (Tutorialspoint. 2018) Many other types of masks can be applied to an image but we will mainly focus on blurring the image for the purposes of this practical assignment.

## 4.2 Blurring

Blurring is when an image has some or all of its sharpness or small details removed. We can also define blurring as when we can no longer perceive the shapes inside an image correctly. Colours seem to be edgeless and smoothly transition into the colour next to it (Tutorialspoint. 2018).

## 4.3How do you use masks to blur an image?

There are multiple ways of achieving a blurred image using filters. Some of them are briefly discussed below:

### 4.3.1 Mean Filter

A mean filter also known as an average filter has the following properties: It must be odd ordered, the sum of all elements should be 1, and all the elements should be the same. (Tutorialspoint. 2018)

The filter operates in a box for example 3x3 and then uses the elements inside the box to blur the image. Each time you increase the size of the box the amount of blur applied also increases.

How a 3x3 mean filter would look:

1/9 1/9 1/9

1/9 1/9 1/9

1/9 1/9 1/9

(Tutorialspoint. 2018)

### 4.3.2 Weighted average filter

In a weighted average filter more weight is given to the centre value. This means that the centre will contribute more than the rest of the values. This means we can actually control the blur. A weighted average filter has the following properties: it must be odd numbered, the sum of all the elements should be 1, and the weight of the centre element should be more than all of the other elements. (Tutorialspoint. 2018)

An example of how a weighted average filter might look

1 1 1

1 3 1

1 1 1

(Tutorialspoint. 2018)

# 5. Project description:

The specifications of this practical are as follows: “Group assignment for 3 members: Write a program (language of your choice that support threads) that will open an image and apply a mask (mean filter, e.g. 3x3) on the image and show the result. Allow the user to choose the number of threads (e.g. 1 to 64). Write a report on the project and your results. (PS: 4 to 5 members much more work and experimentation is required).” This is the instruction given on efundi.nwu.ac.za

# 6. Hypotheses and results:

## 6.1 Hypothesis

We hypothesized that there would be some exponential scaling for the first 4 threads that we create and there after a slight increase for the remaining threads.

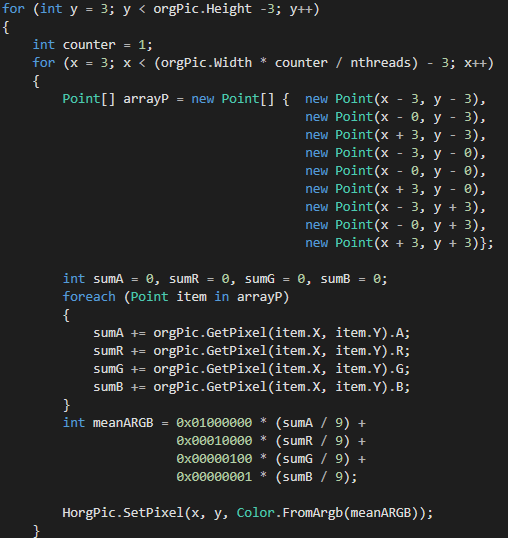
## 6.2 Results

There were improvements up until 6 threads (these results were prior to using the Intel i5 M 430 laptop CPU and 4GB of RAM). After the amount of 6 to 26 threads were reached the average run time remained the same but there were slight variations. From about 27 threads onwards there is not a notable improvement in the execution run time of the application. We believe this is because of the algorithm structure or because of the CPU. The reason for this deduction, is that the time to create the thread, and the time for each thread to complete the masking remains mainly the same. Our optimal execution time was reached on exactly 4 threads as can be seen on the image provided below. (Image)

# 7. First phase of the Project (Applying the 3x3 “mask”):

The first problem we addressed in our project was applying a 3x3 mask on the image loaded into the picturebox. First, we made a new Bitmap of the image and saved it as “orgPic”. Then we proceeded to determine the points of every pixel surrounding those pixels (if there is not a pixel we made those values equal to NULL) and the value of that specific pixel in that area. In the following step of the masking process, we get the sum of each of the colours of pixel “A” for the ambient light (the brightness of the LED, then the RGB for the Red, Green and Blue).

After we determined the surrounding pixels we add them to a total sum as demonstrated in figure 1.1 in the foreach function. Once this is done we need to get the average of each pixel to make a blurring mask, this is done in meanARGB. Once this is completed the new pixel is added to a bitmap named “HorgPic”. This Bitmap will be used to build up the “Masked” image then displayed in picturebox2 seen in figure 1.2.

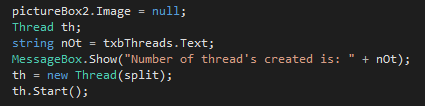
 

**Figure 1.1 Figure 1.2**

# 8. Second phase of the Project (Implementing Threads):

For the first implementation of threading, we implemented Synchronize threading this isn’t the best way of doing threading but we first just want to see how threads could be used with a C# program to execute a method and if it works.

The code below shows us implementing the “System.Threading” namespace to enable us to create threads in our code. The second figure demonstrates how we create a thread and then assign the split method to that thread which blurs and divide our bitmap.

**Figure 2.1 Figure 2.2**

# 9. Third phase of the Project (Splitting the “Bitmap”):

For the third step we had to split the bitmap in order to send different parts of it to individual threads for execution. This was done by splitting the bitmap along its width in vertical sections and then passing each section of to a static method that could then be called by the thread.



**Figure 3.1**

# 10. Fourth and Final phase of the Project (Applying each part of the split “Bitmap” to a single Thread):

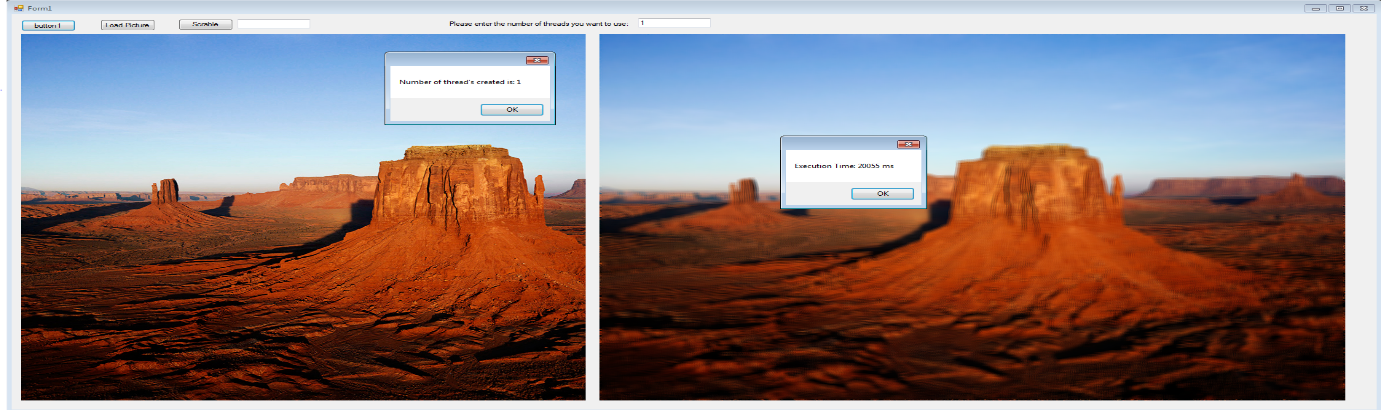
The final step in our project we wanted to implement the execution of threads in an Asynchronized fashion. This would allow us to increase the performance of the programming exponentially. We achieved this by creating a method “myThreadCreation” that creates the threads [Thread th1 = new Thread(multiSplit1)]. Which was sitting in a while list for each number of threads that the user specified the method will create that number of threads e.g. (6 Threads specified would result in th1, th2, th3…, th”n”).

These threads will each then receive an object of the split bitmap named multiSplit1 ..., multiSplit”n”, which was created by the main method Split to start with. The way we split the bit was already explained (in section 9), the way we passed the parts to the threads was by creating objects of each of the split bitmap sections and then passing it as an arrangement to the Thread to be handled.

This resulting in an Asynchronized way of doing threading through the use of the individual th”n”.Strart() and th”n”.Join() method.

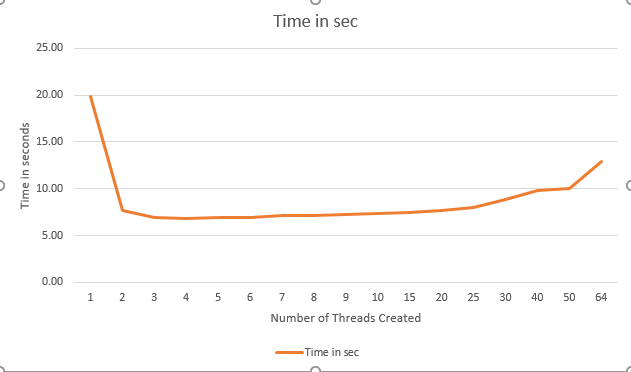
# 11. Final Project Results:

This is the result of our project demonstrating the start and end screens. To begin the user can load a picture into the program by clicking on the “Load Picture” button. This will bring-up a new explorer windows from where the user can select his picture. Next the user can specify the number of threads he want to use in either textbox. The first textbox is for Synchronized threading and the second for Asynchronized threading. There are two different buttons that will execute each function. The first button “button1” will run the program in an Asynchronized fashion and the “Scrable“button for executing the program in a Synchronized fashion like demonstrated below in figure 4.1.



**Figure 4.1**

The final time results of each thread running time are shown in figure 4.2.



**Figure 4.1**

# 12. Bibliography:

Techopedia.com. (2018). What is a thread in an operating system? – Definition from Techopedia. [Online] Available at: <https://www.techopedia.com/definition/27857/thread-operating-systems> [Accessed 6 May 2018].

Gramoli, V. (2015). More than you ever wanted to know about synchronization: Synchrobench, measuring the impact of the synchronization on concurrent algorithms. Pp.1-10

Howtogeek.com. (2018). CPU Basics: Multiple CPU’s, Cores, and Hyper threading explained. [Online] Available at: <https://www.howtogeek.com/194756/cpu-basics-multiple-cpus-cores-and-hyper-threading-explained/>. [Accessed 7 May 2018].

Stokes, J. (2018). Introduction to Multithreading, Superthreading and Hyperthreading. [Online] Ars Technica. Available at: <https://arstechnica.com/features/2002/10/hyperthreading/>. [Accessed 7 May 2018].

GeeksforGeeks. (2018). Operating System | Thread – GeeksforGeeks. [Online] Available at: <https://www.geeksforgeeks.org/operarting-system-thread/>

[Accessed 7 May 2018].

Tutorialspoint.com. (2018). Operating System – Multi-threading. [Online] Available at: <https://www.tutorialspoint.com/operating_system/os_multi_threading.htm>. [Accessed 7 May 2018].

Tutorialspoint.com. (2018). Concept of Mask. [Online] Available at: <https://www.tutorialspoint.com/dip/concept_of_mask.htm>. [Accessed 7 May 2018].

Tutorialspoint.com. (2018). Concept of Blurring. [Online] Available at: <https://www.tutorialspoint.com/dip/concept_of_blurring.htm>. [Accessed 7 May 2018].

Kanjilal, J. (2018). How to work with threads in C#. [Online] InfoWorld. Available at: <https://www.infoworld.com/article/3035134/application-development/how-to-work-with-threads-in-c.html> [Accessed 7 May 2018].

Ahmed, B. (2018). Asynchronous programming and threading in C#(.NET4.5) – CodeProject. [Online] Codeproject.com. Available at: [www.codeproject.com/Articles/996857/Asynchronous-programming-and-threading-in-Csharp-N](http://www.codeproject.com/Articles/996857/Asynchronous-programming-and-threading-in-Csharp-N) [Accessed 7 May 2018].

## 12.1. For Coding:

<https://docs.microsoft.com/en-us/dotnet/standard/threading/creating-threads-and-passing-data-at-start-time>

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/threading/