# Instituto Tecnológico y de Estudios Superiores de Monterrey

## Campus Querétaro


**BGE Filter**


**Isaac Mercado Silvano**
**A01020382**

# Contents

# 1 Introduction

In this project, I will implement three different image filters: blur, grayscale and edge detection; hence the name BGE. Not only will the application process the images with a given filter, but also run on a remote server where multiple clients will be able to connect and send multiple requests for a given image using Java's high concurrency objects and frameworks that will run the server-client processes and the filtering application as well.

Note that the full scope of the project is to implement different technologies regarding parallelism and concurrency; this includes CUDA, tbb, openmp and, of course,Java. As of now Java has been fully implemented, the other technologies will be included on further revisions of this project.

# 2 Multi-client Server

This project works on a multi-client server basis. This means that each client will be able to send a request and the server will process each request for every client that connects to the server itself. The following diagram shows a representation of how the server works internally.

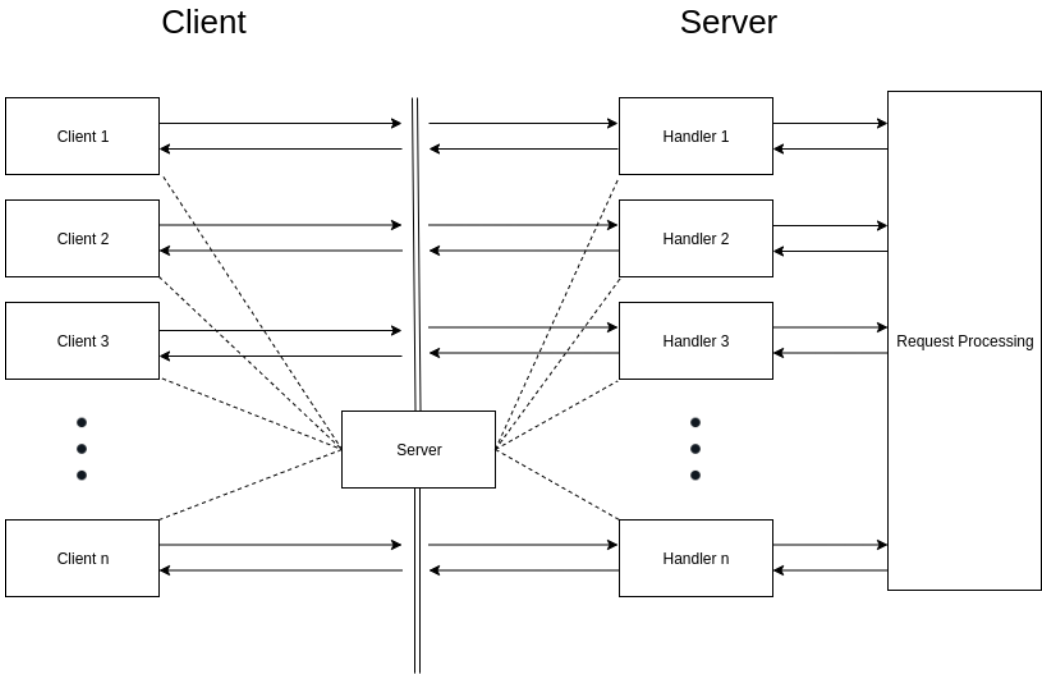



Figure 1: Multi-client server working diagram

As seen in the previous diagram, each client is able to connect, send and receive data only through a gateway. Represented by the dashed line each client will send a connection request to which the server will respond by creating a child process that will be in charge of serving the client (client handler as is the case).

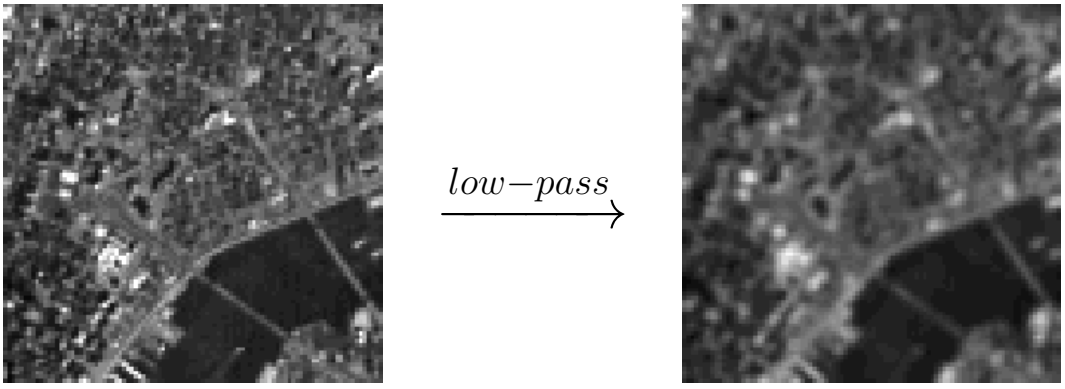
It is important to note that the server gateway will be the main thread of the server program. The main thread is in charge of only accepting any incoming connections from any client only if it connects to the corresponding port that the main thread will be constantly listening to.

Once the connection is set, the child process created by the main server thread will be able to send and receive data to and from the client process. This let's us have a direct connection to the client while keeping the main server thread less bussy as possible.

Something important to note is that the Request Processing block is represented as being accessed by every client handler. This means that each handler will run a copy of the request processing block, wait for a return value from the process, return to the handler and send the resulting data back to the corresponding client.

# 3 Image Filtering: Understanding

Image processing is something seen constantly from social media to more advanced algorithms that filter a given image to obtain significant data from it. For instance, if one has a very noisy image, one can apply a low-pass filter to the whole image to obtain a more smooth looking one. There are many other filters that can be applied to an image and its implementations are bounded by nothing less than imagination and what the problem requires.




(a) Original Image (b) Filtered Image

Figure 2: Low-pass filter application to an image

For this particular project, three filters where implemented: blur, grayscale and edge detection. One way to represent a filter being applied to an image is by thinking of an image as a matrix of numbers; each index location of the matrix represents a single pixel, each with a certain ARGB or RGBA value (depending on the system and wheter it supports the A channel):

- A: Alpha channel; represents the opacity of the image.
- R: Red channel; represents how much red is present in an image.
- G: Green channel; represents how much green is present in an image.
- B: Blue channel; represents how much blue is present in an image.

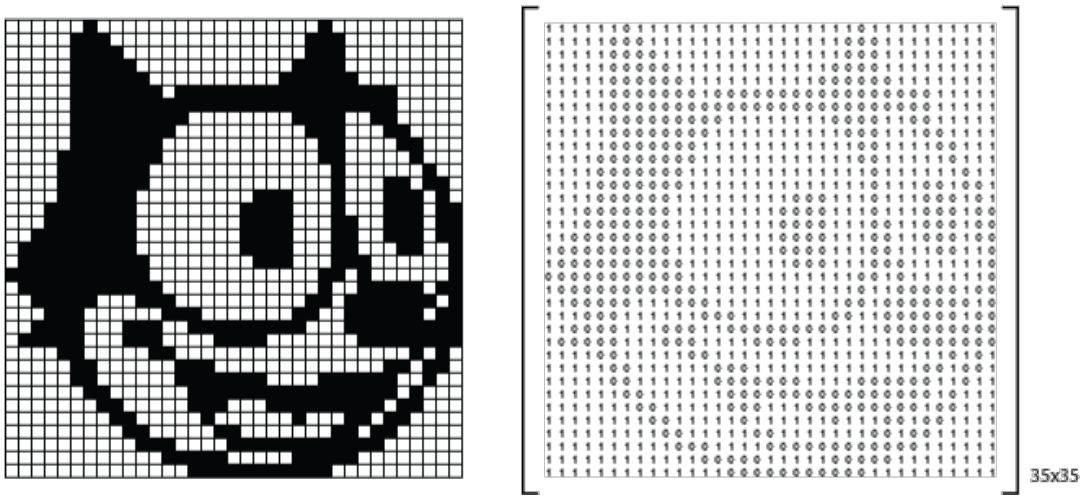



Figure 3: Image to matrix representation

As seen in the previous image, the ARGB values can be represented as integer or float values. For figure 3, the case would be an image of size 35 by 35 pixels consisting of only 1s and 0s where 1 is white and 0 is black. Such representations let us apply certain operations to the image matrix, seeing an image as an input signal where one could apply image convolution to such signal or any other applications required by the filter itself to obtain certain characteristics of an image or something different entirely.

### 3.1 Blur

### 3.2 Grayscale

### 3.3 Edge detection

## 4 Paradigm Exploration

For the pardigm exploration one must define which and how a given paradigm works for the wrapped solution from the previous sections (multiple-client server and image filtering). Through Java's high concurrency objects as well as the many interfaces they provide, mainly the Runnable implementation and the fork-join framework, which includes the RecursiveAction.

This high-concurrency objects and frameworks let's us implement the communication for multiple clients, each with a single thread to work on the

server side (child processes created by the server). This as well let's us apply the filters through the java fork-join framework which implements a pool of threads able to work on a single element (in this case an image) and return the result of their work in the corresponding position, as this method does not return a value per se, it stores the result of applying the filter of a source image to a destination path which the client handler will reference to retrieve the resulting image file.

# 5 Technology Implementation

# 6 Results

# 7 Applications

Within the previous sections we saw how to apply filters to images on a client-server basis. How does this all come together? For starters, in the field of robotics, image processing is key to many applications and with many other filters that could be applied to an image, the uses are greatly enhanced by the developer.

For a multi-client server, such applications would come in the form of a unified network of machines that could map, with the help of computer vision, a whole area just by border detection and sending that information to a server that could put the pieces together.

The implementation of this project was mostly directed toward users that would select an image manually and apply one (or all) of the filters available; subsequently the users would receive the filtered image they requested.

# 8 Conclussions

Although a simple implementation, this project could be expanded using not only Java's high concurrency objects and frameworks, but other technologies as well that tackle another kind of paradigm such as full parallelism, which greatly enhances image processing and thus increasing the response time from the server.

It is interesting how image processing works and the many applications it entitles. It is up to us, the developers, to indulge oneself in the vast world of image processing. Looking at how this may apply to my field of study I profoundly see great applications in robotics and computer vision that would greatly change the way we see the Universe that surrounds us.

# 9 Bibliography

[1] C. Saravanan (2010). Color Image to Grayscale Image Conversion. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5445596&fbclid=IwAR1EgoKwX5CnBjdJQLrgmHkPzlmUKyQvZK35JP7GaRMgt2HXUBtrC4h3BYE&tag=1

[2] Introcomputing (n..d). Grayscale Images. Retrieved from https://introcomputing.org/image-6-grayscale.html?fbclid=IwAR0E8mxXjILrPbYgxgdtetfgdf2_piGEpXAc6OQhZhpgus4MYEiat8IyA9I

[3] B. Poornima, Y. Ramadevi, T. Sridevi (2011). Threshold Based Edge Detection Algorithm. Retrieved from http://ijetch.org/papers/260-T754.pdf?fbclid=IwAR3qX0VZJMqMxe5RodFxJM7QK4rXKWj-OHE29homJzEwlR27NpOj_r80zX4

[4] You-yi Zheng, Ji-lai Rao, Lei Wu, et. al. (2010). Edge Detection Methods in Digital Image Processing. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5593576&fbclid=IwAR2rIoQ70Uq63shHmBwwlO0mY

[5] user: Packt-Pub (n.d.). Building a Java Edge Detection Application. Retrieved from https://medium.com/javarevisited/building-a-java-edge-detection-app

[6] IDL Online Help (2005). Filtering an Image. Retrieved from https://northstar-www.dartmouth.edu/doc/idl/html_6.2/Filtering_an_Imagehvr.html?fbclid=IwAR3hdgPAXU28LdKZctmNI6JI18w_dJ9X1ko09TBdH9-H_8zbflQlKeUzVC0#wp1022750

[7] (n.a) (n.d). Chapter 4. Image Filtering. Retrieved from http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter4.pdf?fbclid=IwAR1qzygXluRKIKnm0t0RTB8uhsjJHFtPJXZEJ5fvKWFzWWnwpEYd_FhXFGg

[8] Geeks for Geeks (n.d.). Introducing Threads in Socket Programming in Java. Retrieved from https://www.geeksforgeeks.org/introducing-threads-socket-progr?fbclid=IwAR3qX0VZJMqMxe5RodFxJM7QK4rXKWj-OHE29homJzEwlR27NpOj_r80zX4

[9] Geeks for Geeks (n.d). Digital Image Processing Basics. Retrieved from https://www.geeksforgeeks.org/digital-image-processing-basics/?fbclid=IwAR1DqGltCVK4AeQchqvB0rQHWLMc3ayRs_0cxDqdDD52ooNbCelDNmwuSms

[10] Daniel Shiffman (2008). Images and Pixels. Retrieved from https://processing.org/tutorials/pixels/?fbclid=IwAR2po0c9tc3QPYgagWEWcQ5MQr1Vq0extRHtNW

[11] Alvin Alexander (2016). Java exec - execute system processes with Java ProcessBuilder and Process. Retrieved from https://alvinalexander.com/java/java-exec-processbuilder-process-1