# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

# FACULTAD DE INGENIERÍA


**ALUMNO:** UGALDE VELASCO ARMANDO

***ESTRUCTURAS DE DATOS Y ALGORITMOS II***

**GRUPO** 1

## PROYECTO FINAL: ECUALIZACIÓN DEL HISTOGRAMA

**SEMESTRE** 2021-1

# INTRODUCCIÓN

Se desarrolló un programa que ecualiza el histograma de una imagen digital en escala de grises, llevando a cabo ajustes en los niveles de contraste para así mejorar su visibilidad. Cabe mencionar que el programa admite imágenes de más de un canal, sin embargo, como se mencionó, éstas deben encontrarse en escala de grises para que el algoritmo trabaje correctamente. Si esta condición no se cumple, es posible que se obtenga un resultado inesperado.

# DESARROLLO

El programa se dividió en dos directorios principales: **util**, que contiene una serie de funciones utilizadas en diversas partes del programa, e **image_equalization**, que contiene la lógica relacionada a la ecualización del histograma. En esta última carpeta, se encuentra presentes dos subdirectorios: **sequential** y **parallel**, que contienen las funciones necesarias para llevar a cabo la ecualización en forma secuencial y paralela, respectivamente. Además, esta carpeta también contiene el archivo **equalize_image**, que lleva a cabo la ecualización de la imagen utilizando los dos enfoques mencionados.

A continuación, se profundizará en los archivos contenidos en cada carpeta:

## DIRECTORIO UTIL

### filenames.c

***get_filename:*** Retorna el nombre del archivo contenido en un path, excluyendo su extensión.

***generate_new_filename:*** Concatena las cadenas original y suffix proporcionadas.

### generate_array.c

***generate_empty_unsigned_char:*** Retorna un arreglo de unsigned char alojado con el tamaño especificado. Contiene el valor cero en todos sus elementos.

***generate_empty_unsigned_char:*** Retorna un arreglo de long alojado con el tamaño especificado. Contiene el valor cero en todos sus elementos.

## generate_csv.c

***generate_csv:*** Genera el archivo csv correspondiente a una imagen, dados sus histogramas.

## get_equalized_value.c

***get_equalized_value:*** Retorna el valor ecualizado de la distribución acumulada utilizando los valores proporcionados.

## get_first_nonzero.c

***get_first_nonzero:*** retorna el primer valor diferente de cero de un arreglo.

### DIRECTORIO IMAGE_EQUALIZATION

## equalize_image.c

***equalize_image***: Abre la imagen indicada por el path proporcionado. De ser imposible la tarea anterior, retorna e imprime un mensaje de error. Si no, obtiene los datos pertinentes de la imagen y los imprime, además de ecualizar las imágenes de forma paralela y secuencial, y obtener sus métricas e información necesarias.

### SUBDIRECTORIO SEQUENTIAL

## equalize_image_sequential.c

***equalize_image_sequential:*** Ecualiza una imagen de forma secuencial y obtiene el tiempo de ejecución de la tarea anterior. Además, genera el archivo csv correspondiente.

## cumulative_distribution_sequential.c

***get_cumulative_distribution_sequential:*** Obtiene la distribución acumulada dado el histograma de una imagen, de forma secuencial.

***get_equalized_cumulative_sequential:*** Ecualiza una distribución acumulada acorde a la fórmula proporcionada de forma secuencial.

## generate_image_sequential.c

***generate_image_sequential:*** Genera un arreglo con los nuevos valores de color de una imagen de forma secuencial, dada su distribución ecualizada.

## histogram_sequential.c

***generate_histogram_sequential***: Genera el histograma de una imagen de forma secuencial.

### SUBDIRECTORIO PARALLEL

## equalize_image_parallel.c

***equalize_image_parallel:*** Ecualiza una imagen de forma paralela y obtiene el tiempo de ejecución de la tarea anterior. Además, genera el archivo csv correspondiente.

## cumulative_distribution_parallel.c

***get_equalized_cumulative_parallel:*** Ecualiza una distribución acumulada acorde a la fórmula proporcionada de forma paralela.

## generate_image_parallel.c

***generate_image_parallel:*** Genera un arreglo con los nuevos valores de color de una imagen de forma paralela, dada su distribución ecualizada.

## histogram_parallel.c

***generate_histogram_parallel:*** Genera el histograma de una imagen de forma paralela.

Además, es pertinente mencionar que se desarrolló un programa en **Python** para generar las gráficas de los histogramas, a partir de los archivos **csv**.

# RESULTADOS

A continuación, se muestra la ecualización del histograma de algunas imágenes:

## IMAGEN 1

**Imagen original**

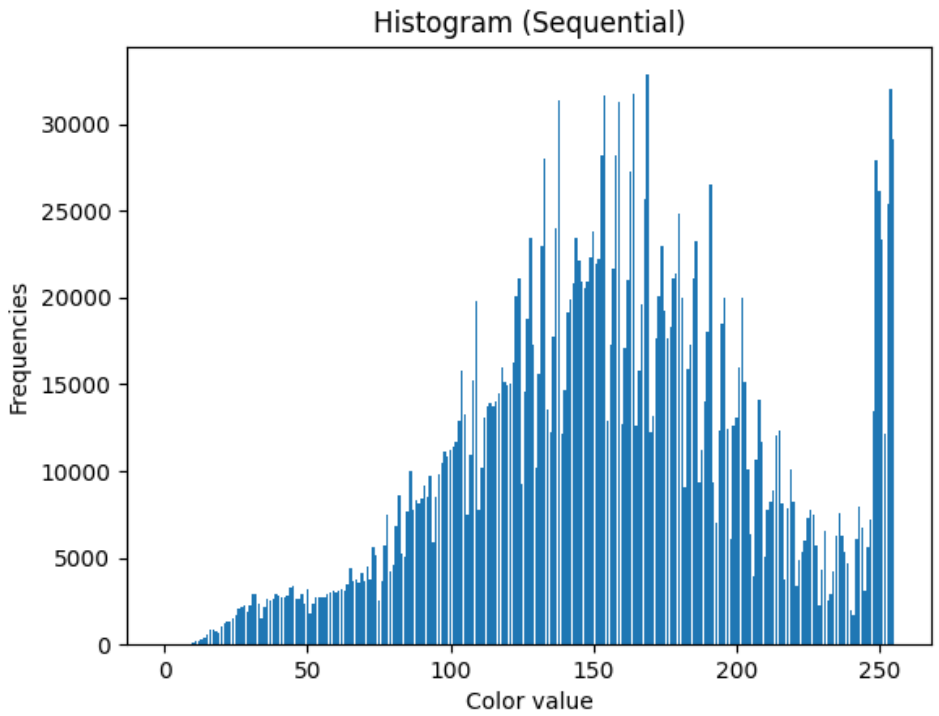
**Imagen ecualizada de forma secuencial**

**Imagen ecualizada de forma paralela**

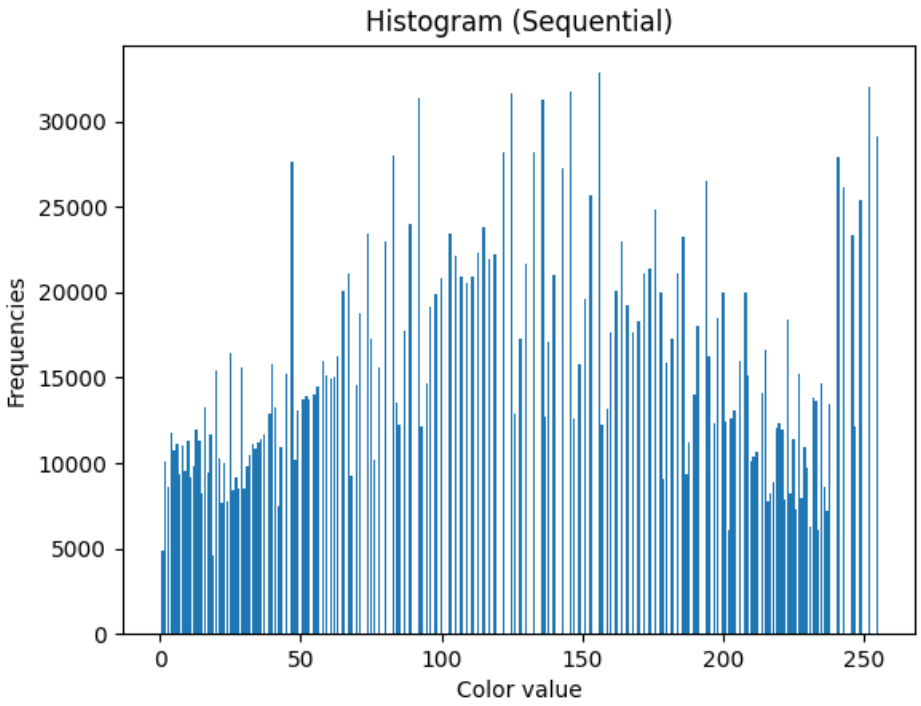
## Ejecución del programa y métricas obtenidas

```
~/OneDrive/Documentos/Escuela/EDA 2/histogram-equalization> ./main.exe imaGray_2.jpg
Nombre de la imagen: imaGray_2
Tiempo de carga: 0.076000
Ancho: 2000
Alto: 1333
Numero de canales: 1
Tamano: 2666000

Tiempo de generacion de csv (secuencial): 0.001000
Tiempo de generacion de imagen (secuencial): 0.401000

Tiempo de generacion de csv (paralelo): 0.000000
Tiempo de generacion de imagen (paralelo): 0.401000

Numero de procesadores: 8
Tiempo secuencial: 0.023000
Tiempo paralelo: 0.008000
Speedup: 2.875037
Eficiencia: 0.359380
Overhead: 0.005125
```

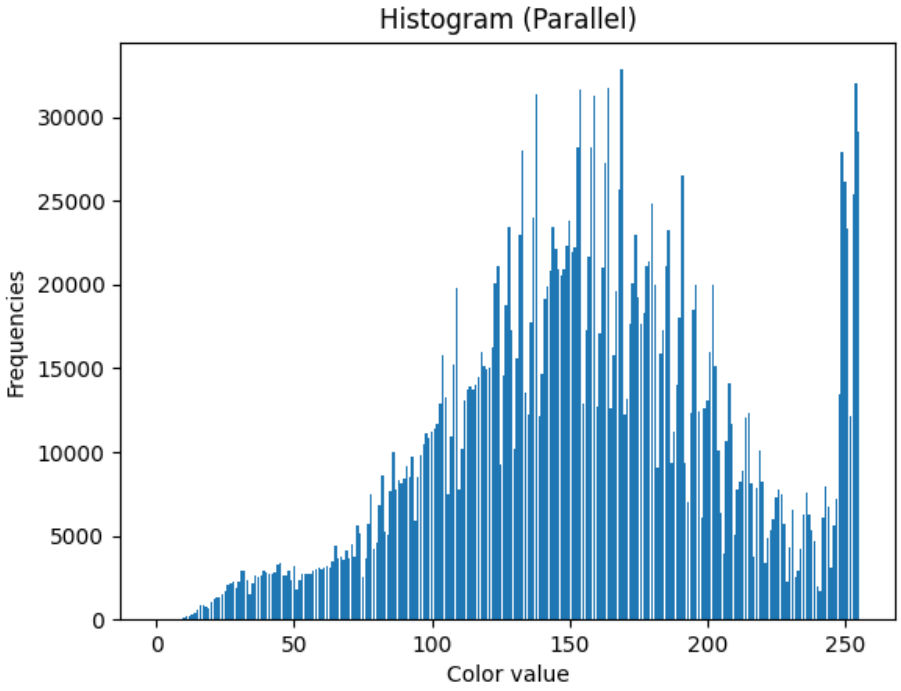
## Histograma original generado de forma secuencial



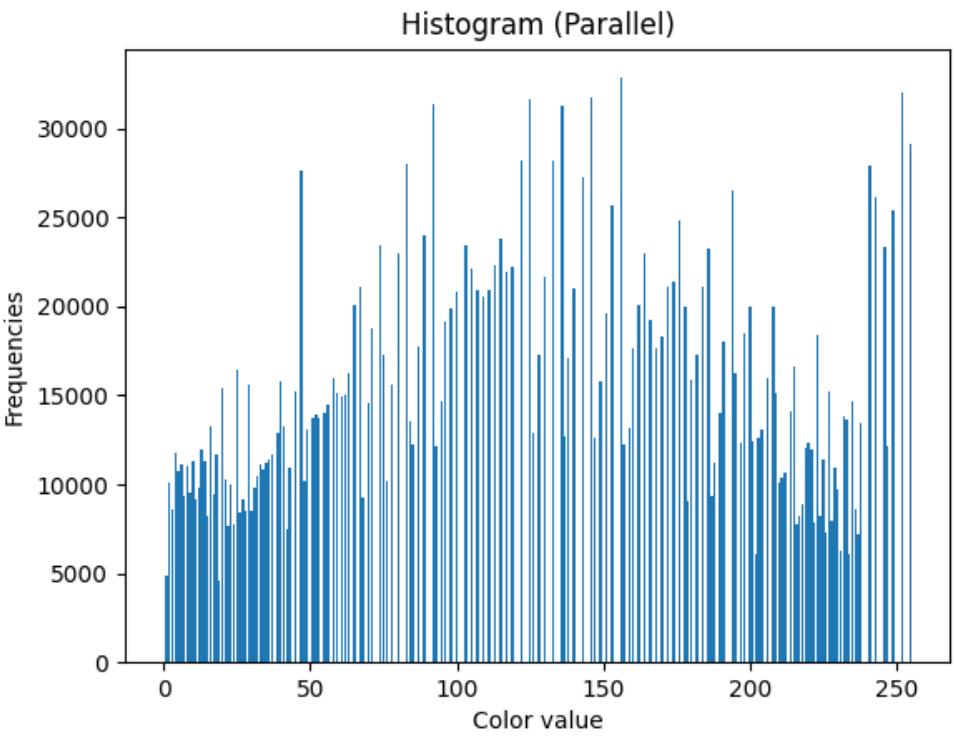
**Histograma ecualizado generado de forma secuencial**




**Histograma original generado de forma paralela**



**Histograma ecualizado generado de forma paralela**




**Histograma de imagen original generado por aplicación**

**Histograma de imagen ecualizada de forma secuencial generado por aplicación**

**Histograma de imagen ecualizada de forma paralela generado por aplicación**

## IMAGEN 2

**Imagen original**

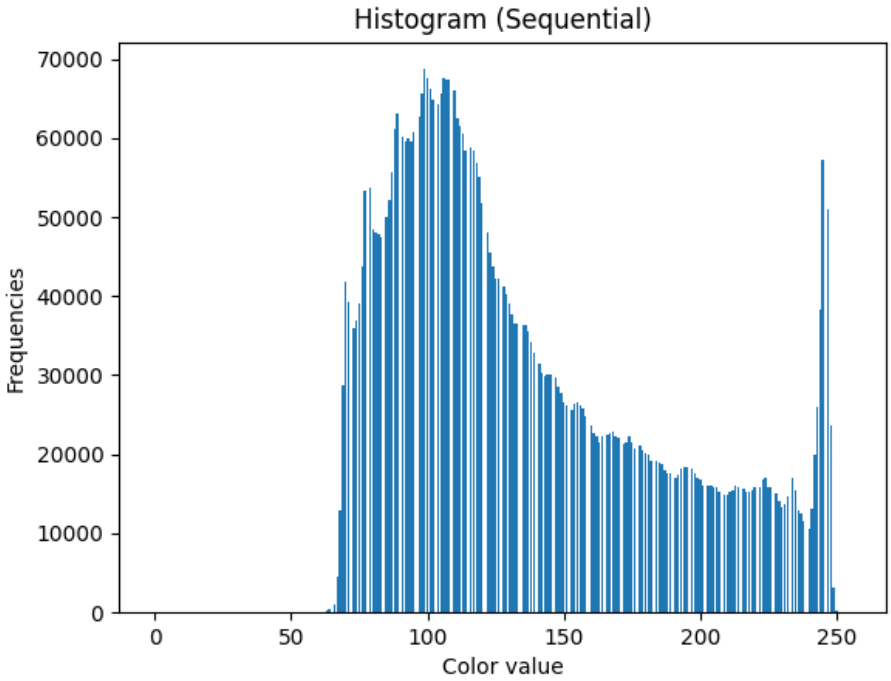
**Imagen ecualizada de forma secuencial**

## Imagen ecualizada de forma paralela

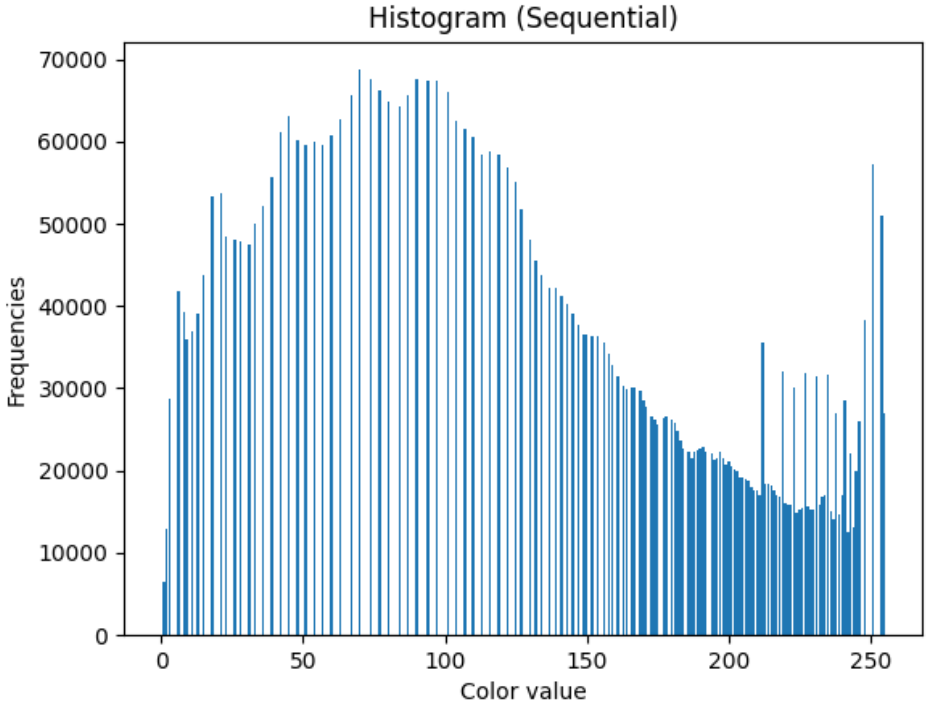
## Ejecución del programa y métricas obtenidas

```
./main.exe img_1.png
Nombre de la imagen: img_1
Tiempo de carga: 0.171000
Ancho: 2736
Alto: 1824
Numero de canales: 3
Tamano: 4990464

Tiempo de generacion de csv (secuencial): 0.001000
Tiempo de generacion de imagen (secuencial): 0.551000

Tiempo de generacion de csv (paralelo): 0.000000
Tiempo de generacion de imagen (paralelo): 0.551000

Numero de procesadores: 8
Tiempo secuencial: 0.062000
Tiempo paralelo: 0.017000
Speedup: 3.647070
Eficiencia: 0.455884
Overhead: 0.009250
```

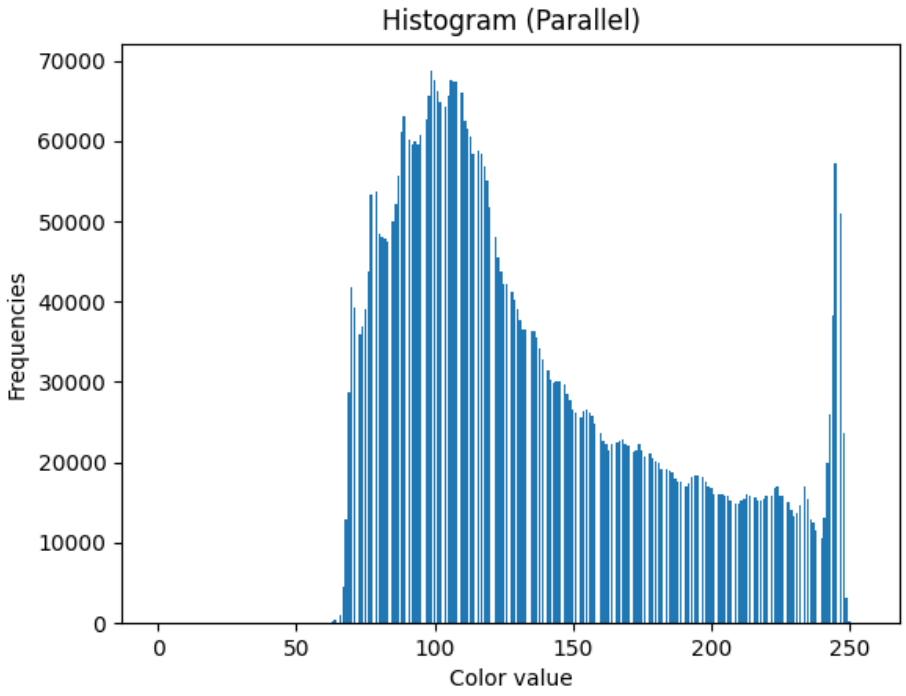
**Histograma original generado de forma secuencial**



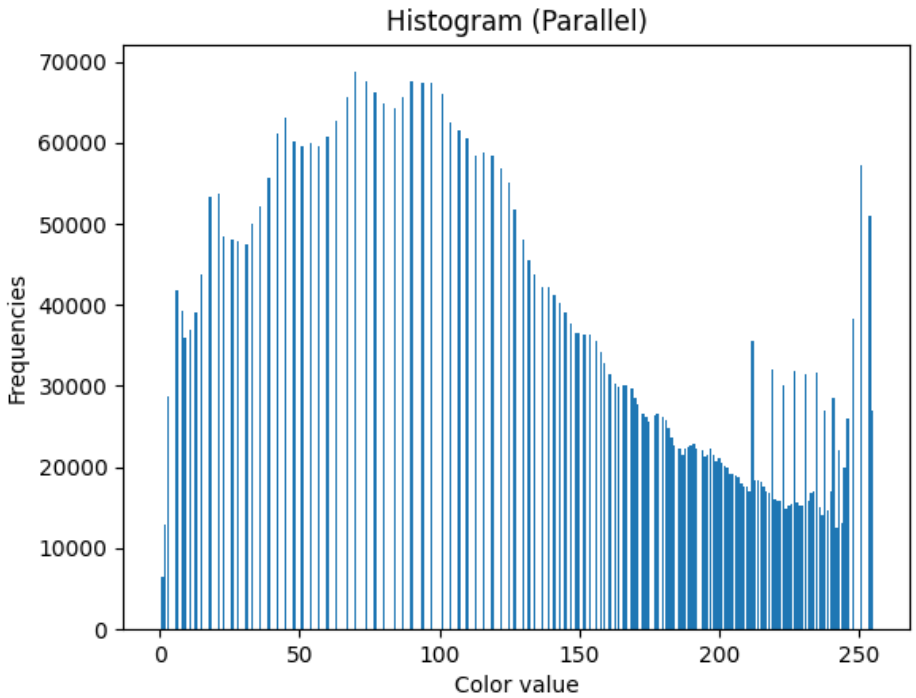

**Histograma ecualizado generado de forma secuencial**



**Histograma original generado de forma paralela**




**Histograma ecualizado generado de forma paralela**



**Histograma de imagen original generado por aplicación**

**Histograma de imagen ecualizada de forma secuencial generado por aplicación**

**Histograma de imagen ecualizada de forma paralela generado por aplicación**

## IMAGEN 3

**Imagen original**

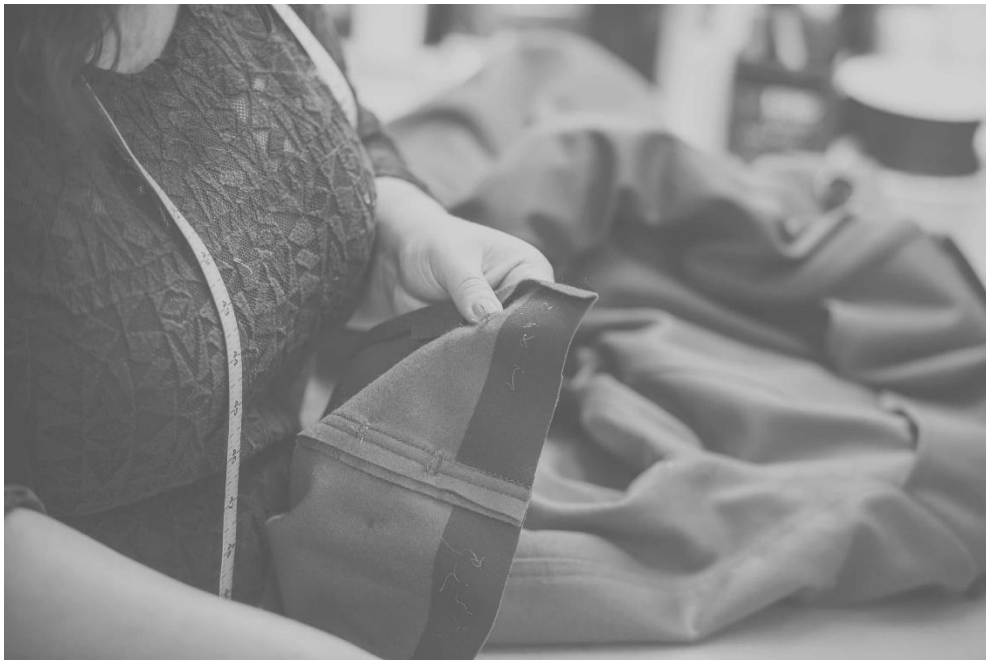


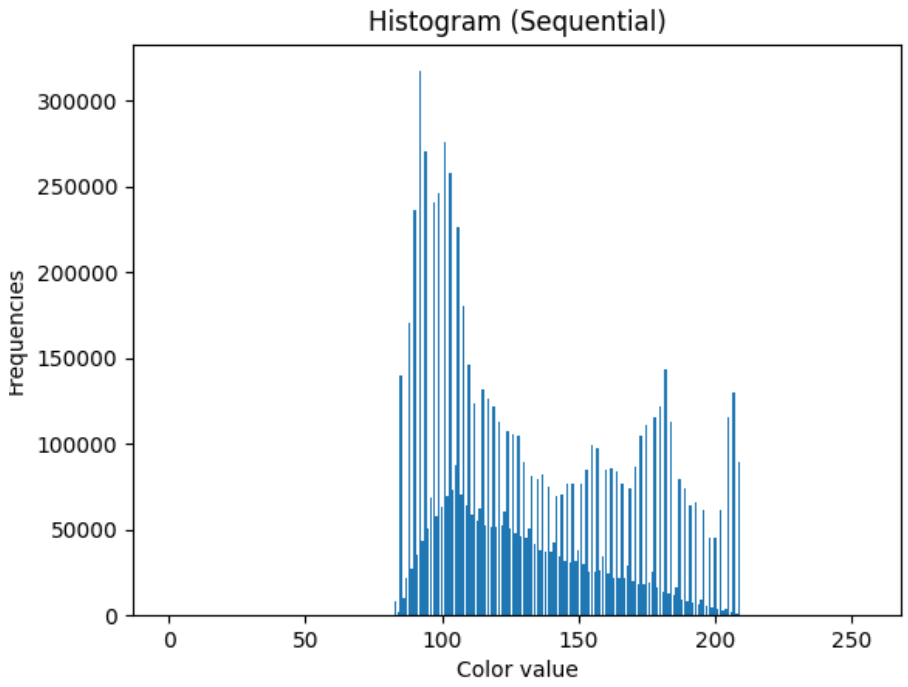
**Imagen ecualizada de forma secuencial**

**Imagen ecualizada de forma paralela**

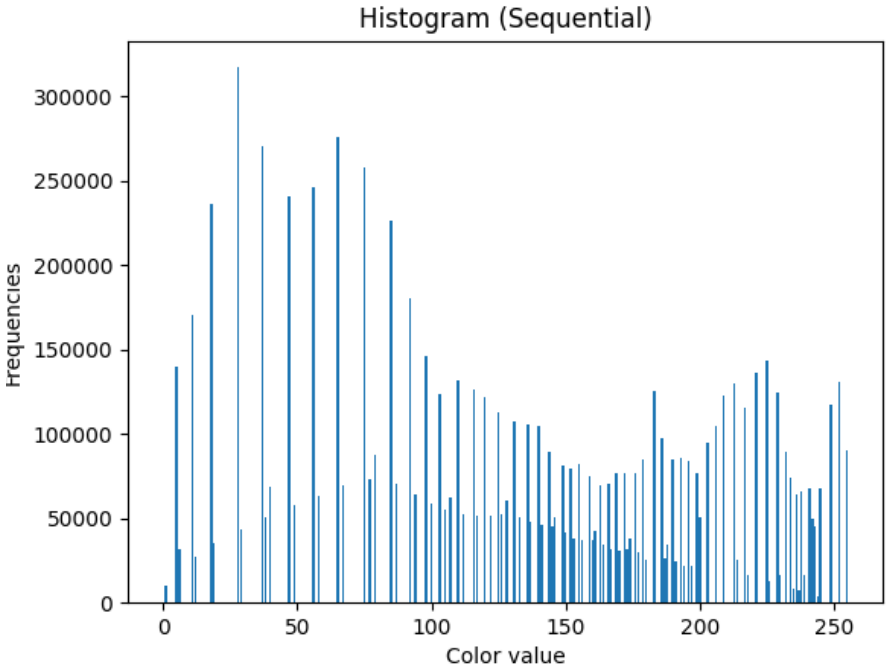
## Ejecución del programa y métricas obtenidas

```
~/OneDrive/Documentos/Escuela/EDA 2/histogram-equalization> ./main.exe imaRGB_1.jpg
Nombre de la imagen: imaRGB_1
Tiempo de carga: 0.267000
Ancho: 3680
Alto: 2456
Numero de canales: 3
Tamano: 9038080

Tiempo de generacion de csv (secuencial): 0.001000
Tiempo de generacion de imagen (secuencial): 0.697000

Tiempo de generacion de csv (paralelo): 0.001000
Tiempo de generacion de imagen (paralelo): 0.696000

Numero de procesadores: 8
Tiempo secuencial: 0.113000
Tiempo paralelo: 0.029000
Speedup: 3.896551
Eficiencia: 0.487069
Overhead: 0.014875
```

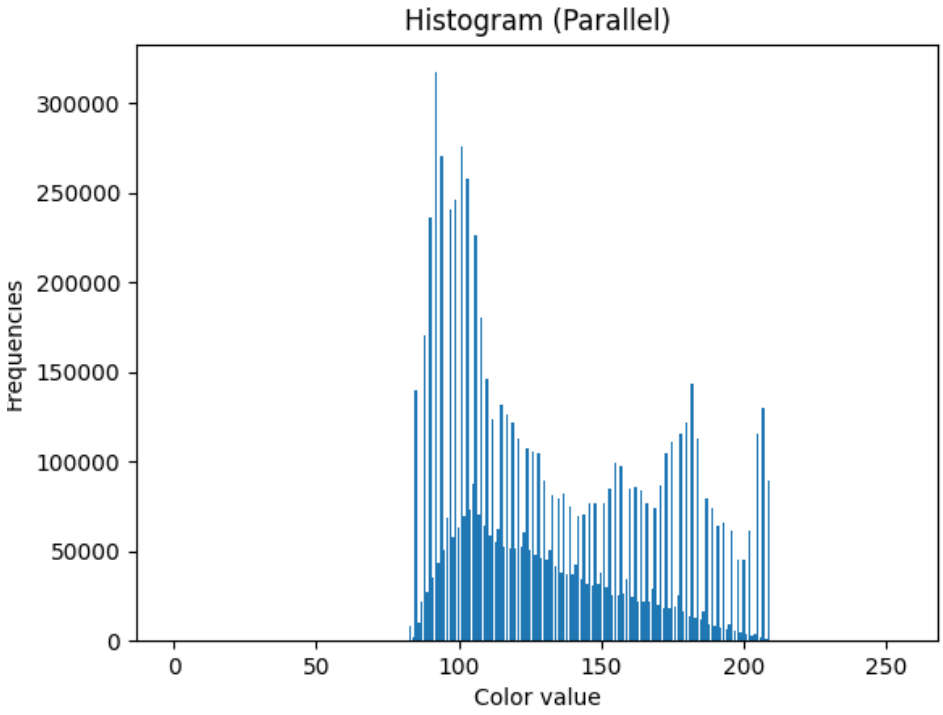
## Histograma original generado de forma secuencial



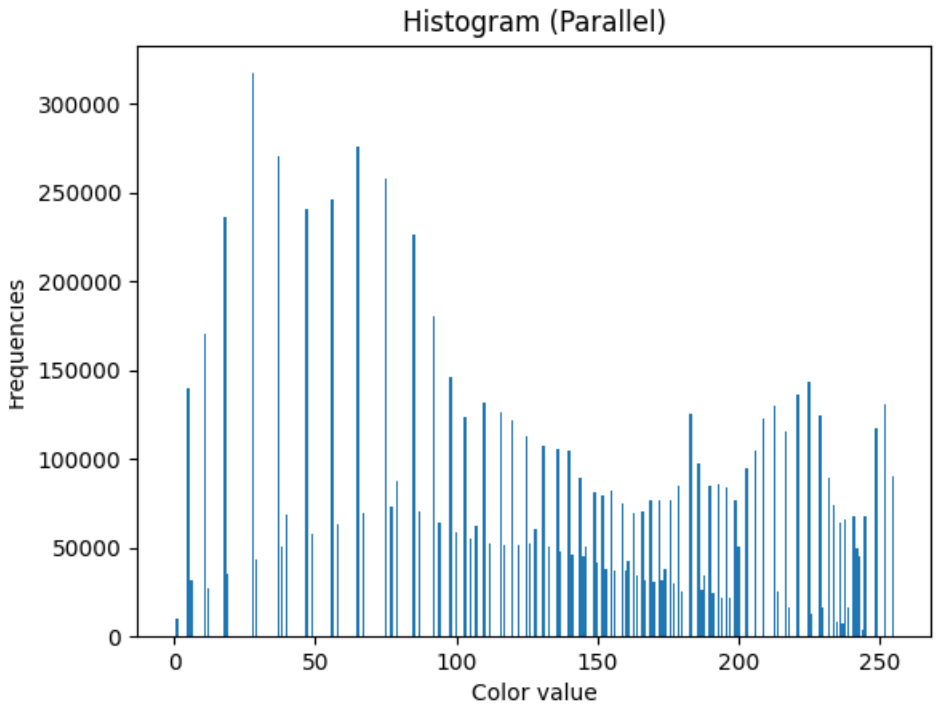
**Histograma ecualizado generado de forma secuencial**



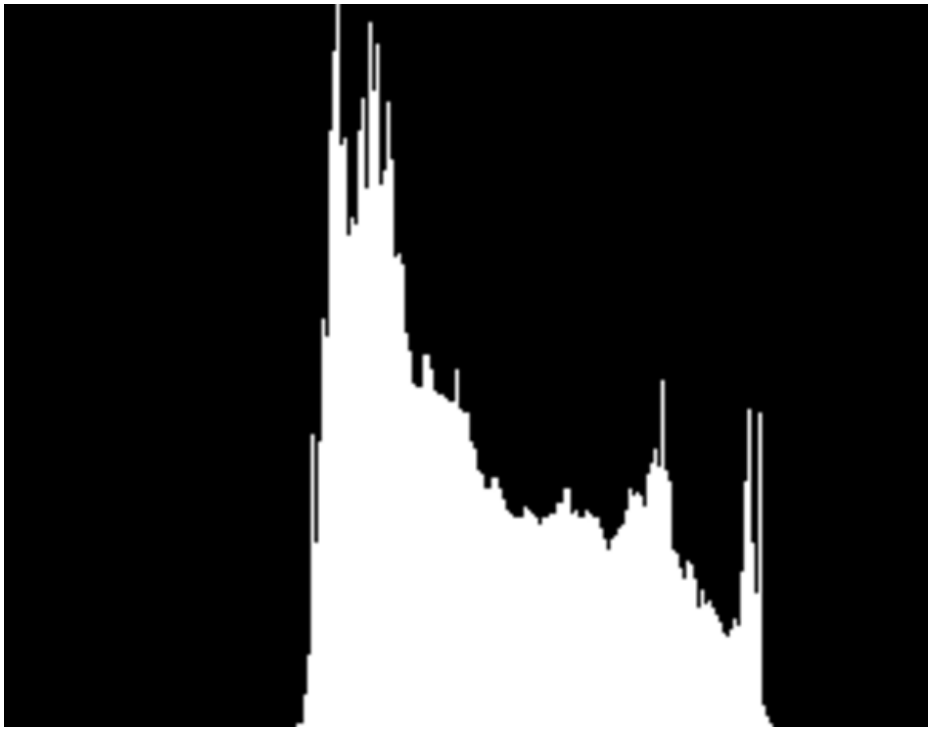

**Histograma original generado de forma paralela**



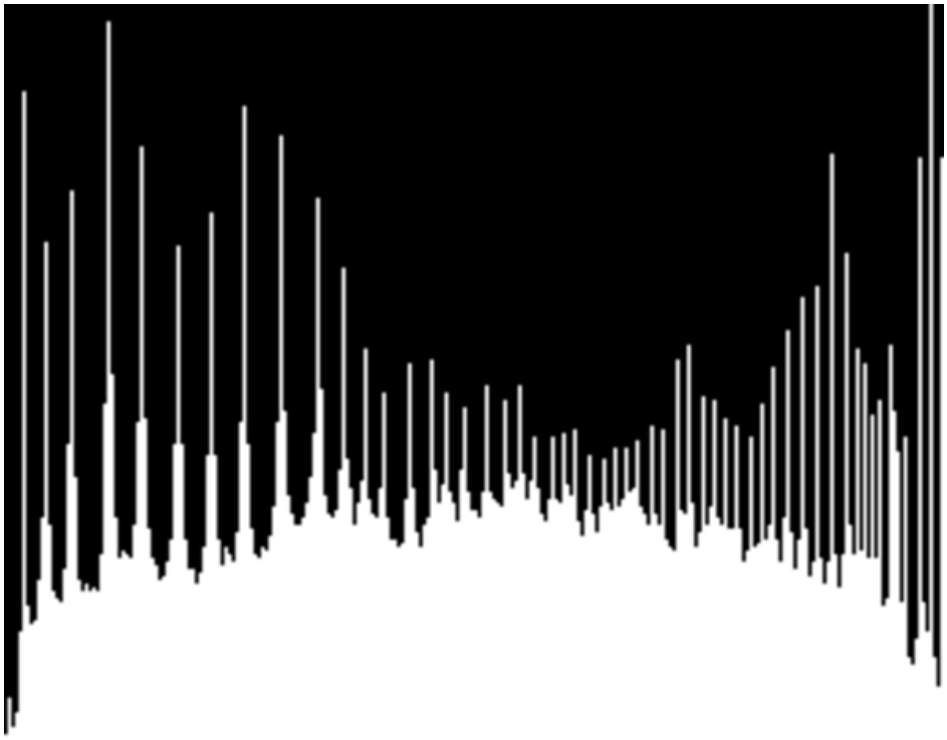
**Histograma ecualizado generado de forma paralela**



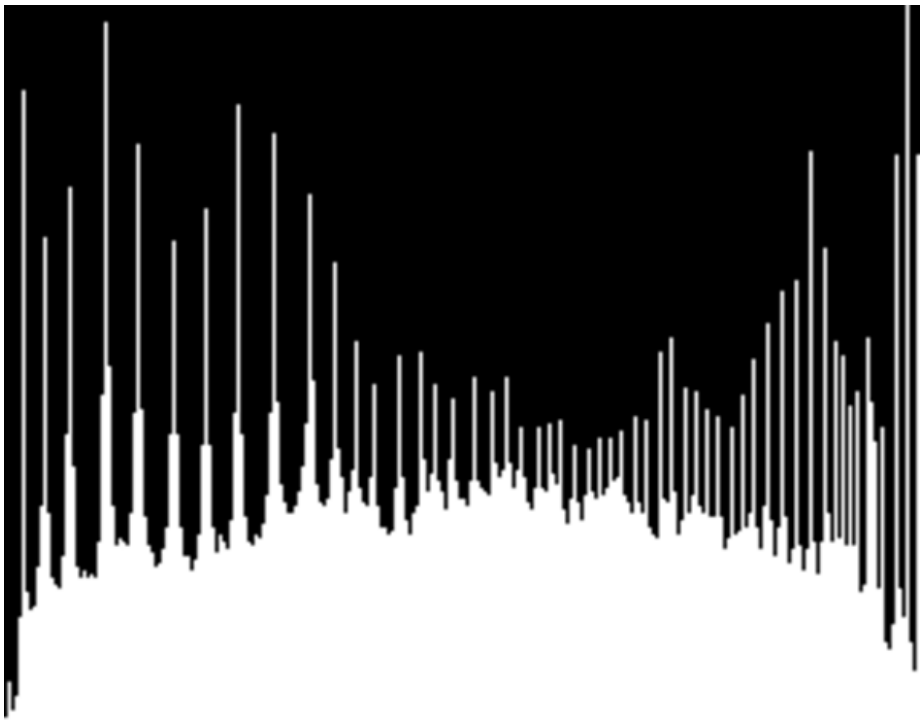

**Histograma de imagen original generado por aplicación**



**Histograma de imagen ecualizada de forma secuencial generado por aplicación**



**Histograma de imagen ecualizada de forma paralela generado por aplicación**



# CONCLUSIONES

Al paralelizar el algoritmo de ecualización del histograma se obtuvo un rendimiento considerablemente **mejor** que el del algoritmo implementado en forma secuencial. Sin embargo, lo anterior únicamente se cumplió para imágenes considerablemente grandes. En los casos de imágenes relativamente pequeñas, se obtuvo un peor rendimiento, debido probablemente al **overhead** causado por el manejo de los hilos.

Dicho lo anterior, una posible mejora a implementar es la decisión del algoritmo a utilizar en tiempo de ejecución, dependiendo del tamaño de la imagen. Por ejemplo, si la imagen tiene una dimensión menor a 600 pixeles, se determina utilizar el algoritmo implementado de forma secuencial.

El funcionamiento del algoritmo se encuentra únicamente garantizado para imágenes en escala de grises, por lo tanto, otra mejora a considerar consiste en extender su funcionamiento para cualquier tipo de imagen.