PROYECTO FINAL

Ecualización de Histogramas en Imágenes

ESTRUCTURA DE DATOS Y ALGORITMOS II

Profesor: Jesús Cruz Navarro

Grupo: 1

Hernandez Diaz Sebastian

INTRODUCCION

En este proyecto lo que se planea realizar es que dada una imagen en formato jpg se ecualice su histograma, lo primero que se debe de hacer para lograr esto es obtener el histograma de la imagen original y trabajar con este mismo, para lograr la ecualización se siguen una serie de pasos, para que como resultado final se pueda crear una nueva imagen mediante ese histograma ecualizado, tambien al hacer este proceso se debe de guardar la nueva imagen, y un archivo de tipo csv, en el cual se tendrá como primera columna los valores del nivel de gris que van desde 0 hasta 255, en la segunda columna se guardaran los valores del histograma de la imagen original y por último en la tercera columna se guardaran los valores del histograma de la imagen ecualizada.

Este programa se debe de ejecutar en 2 formas, una será la secuencial y la otra debe de ser de forma paralela, utilizando OMP como se vio en las practicas, por lo cual al finalizar el programa debe de haber una imagen y un archivo csv para la forma secuencial y estos mismos archivos para la forma en paralelo.

Para el manejo con las imágenes se proporcionó la librería STB, de esta forma se puede trabajar de mejor forma con las imágenes, el código debe de funcionar con imágenes de 1 y 3 canales.

Al finalizar el programa se deberán de poder visualizar los datos de la imagen, tales como la altura, la anchura, los canales y el tamaño, los tiempos de ejecución, ya sea en serie o en paralelo, el número de procesadores, las métricas vistas en clase como lo son el speedup, la eficiencia y el overhead, por último, debe de mostrar el tiempo que tarda en cargar la imagen, el tiempo que tarda en crear la nueva imagen y en crear el archivo CSV.

DESARROLLO

Para poder realizar este programa se necesita primeramente importar las librerías necesarias para el manejo de las imágenes, en este caso la que fue proporcionada, tanto para la versión de 3 y 1 canal se realiza lo mismo solo en un par de pasos esto cambia, se mencionaran los cambios en cuanto se llegue al punto en que esto ocurre.

El primer paso que se debe de hacer es cargar la imagen, la ruta de esta imagen es pasada como parámetro al ejecutar en consola, se usa stbi_load, de la biblioteca proporcionada, se le manda como parámetro lo obtenido en la consola, y un par de argumentos más en los cuales se almacenaran el ancho, alto y los canales de la imagen, esta funciona nos retorna un apuntador de unsigned char los cuales serán los valores de los pixeles de la imagen, mediante un arreglo que será el que tendrá los valores del histograma lo primero que se debe de hacer es llenarlo de ceros para evitar errores, si la imagen es de 3 canales lo que se debe de hacer es crear un arreglo mediante malloc de un tamaño que se obtiene multiplicando la altura por la anchura por los canales, después de llenar el arreglo de ceros, si la imagen es de un canal, mediante un for que recorrerá todos los pixeles, el valor del histograma aumentara en 1 en la posición que se encuentre en i del arreglo de la imagen.

```
for (int i = 0; i < imatam; i++)
{
    histo[srcIma[i]]++;
}
```

En esta imagen se ve de mejor manera, se buscará en el arreglo de la imagen a i, dependiendo del valor que este contenga se le sumara 1 en el histograma, si la imagen es de 3 canales, lo que se hace es lo siguiente:

```
for (int i = 0; i < imatam; i++)
{
    unsigned char r = srcIma[i * channels + 0];
    imaRed[i * channels + 0] = r;
    imaRed[i * channels + 1] = 0;
    imaRed[i * channels + 2] = 0;
    histo[srcIma[i * channels + 0]]++;
}
```

Esto se hace para separar 1 canal de los 3 que tiene por eso se necesita hacer de forma diferente, mediante la forma que se muestra ahí para poder obtener el histograma en este caso de 1 canal para poder trabajarlo.

El siguiente paso una vez obtenido el histograma lo que se debe de hacer es obtener la Cumulative Distributive Function, esta es una función de distribución acumulada, al ser un arreglo tambien se debe de llenar de ceros, una vez se termina esto lo que

se hace es asignarle el primer valor del histograma al primer valor de este arreglo que es cdf, esto porque al ser acumulativo se debe de empezar desde 1 y la forma en que se hará es mediante un for, el cual utiliza a cdf en su posición de i menos 1, por lo cual se debe de empezar desde cero, el cuarto paso es obtener el cdfmin, este valor es el primer menor valor diferente de cero en el arreglo cdf.

El siguiente paso a seguir es obtener un eqCdf, para obtener este arreglo se aplica una formula, se requiere del cdf original y el cdfmin que se obtuvo con anterioridad, la formula se aplica a cada valor del cdf y así obteniendo un nuevo valor que se colocara en el arreglo eqCdf, con este nuevo arreglo se necesita obtener un nuevo arreglo para la imagen a crear, si la imagen es de 1 canal lo que se hace es lo siguiente:

```
for (int i = 0; i < imatam; i++)
{
    eqImage[i] = (unsigned char)eqCdf[srcIma[i]];
}
```

Para el nuevo valor de la imagen se obtiene el valor en i de la imagen original, dado este valor se busca en el eqCdf para poder colocarlo en la nueva imagen, si la imagen es de 3 canales se hace el mismo procedimiento solo que en srcIma se debe de multiplicar a i por los canales.

Al terminar con esto se debe de obtener el histograma de esta nueva imagen, para ello se llena un arreglo con ceros, y como se vio con anterioridad mediante un bucle se busca en eqImage un valor en i y este valor se le sumara 1 en el arreglo que contiene al histograma de la nueva imagen, de la siguiente manera.

```
for (int i = 0; i < imatam; i++)
{
    eqHisto[eqImage[i]]++;
}
```

Se usa a i para obtener un valor en eqImage, y dependiendo de este valor se busca en eqHisto para sumarle 1.

El siguiente paso es la creación de la imagen, lo que se debe de hacer en este caso es hacer uso de la función stbi_write_jpg de la biblioteca proporcionada, a esta función se le debe de pasar como parámetro el nombre de la nueva imagen, el ancho, el alto, los canales, un arreglo con el cual formará la imagen y una calidad que en este caso será de 100 para evitar la compresión, por último, debemos liberar la memoria de este arreglo con la función stbi_image_free.

El último paso a seguir en este proyecto es la creación del archivo csv para ello lo primero que se hace es abrir el archivo mediante fopen, dando como parámetros el nombre del archivo, y la forma de escritura, con fprintf se escribe el encabezado donde se pondrá la columna de los valores de gris, el valor del histograma original

y el valor del histograma de la imagen ecualizada, esto con un bucle for que ira recopilando estos valores, finalmente se debe de cerrar el documento.

Para las métricas lo que se hizo fue tomar los tiempos necesarios para poder aplicar las fórmulas y así obtener estas mismas.

En la versión en paralelo ocurre de forma muy similar, lo primero que se debe de hacer es declarar las variables compartidas fuera de la región en paralelo ya que al declararlas dentro podría ocasionar algún error, después de esto se puede reducir código al llenar de ceros los arreglos, esto con la directiva for, pero se debe de tener cuidado porque debemos de esperar a que todos los hilos terminen y así poder avanzar más con el código, tambien con la clausula reduction se puede reducir código además de que esta clausula ayuda a que las ejecuciones siempre sean iguales, tambien se debe de tener en cuenta que en algunas ocasiones el uso de los hilos no es muy recomendado por lo cual se debe de usar la directiva single para no tener que cerrar y abrir las regiones en paralelo ya que es algo que tarda más tiempo.

RESULTADOS

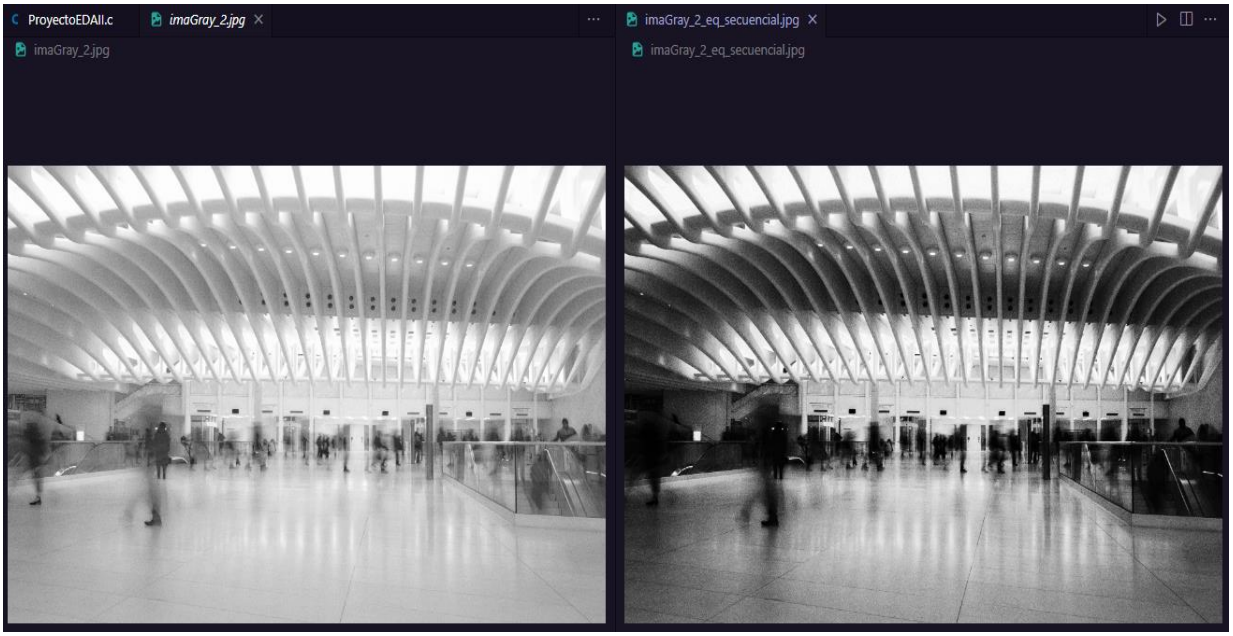
La primera imagen de prueba es la siguiente:

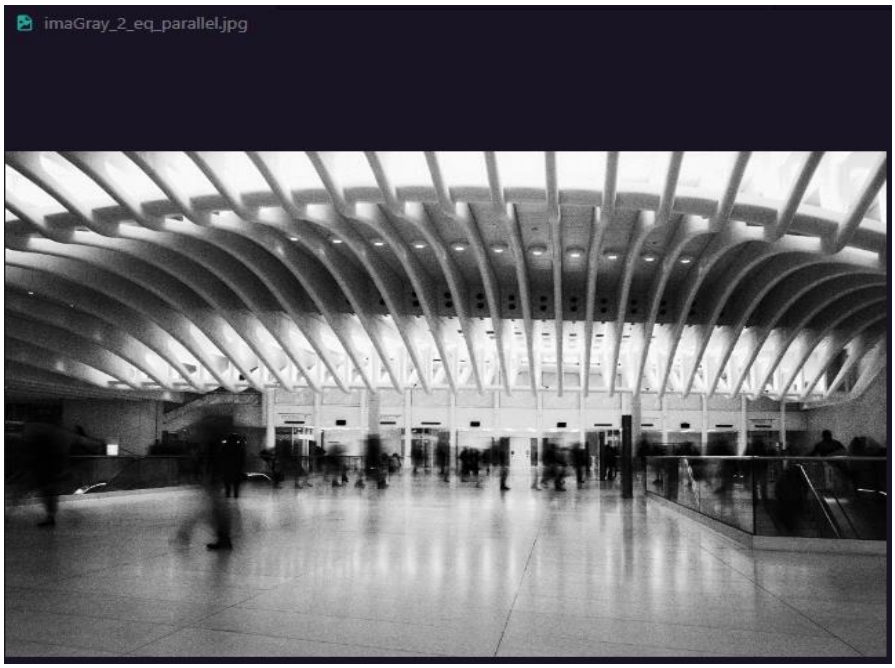
```
sebas@LAPTOP-SM8Q1ICG:/mnt/c/Users/herna/Desktop/ProyectoEDAII$ gcc ProyectoEDAII.c -o ProyectoEDAII -fopenmp -lm
sebas@LAPTOP-SM8Q1ICG:/mnt/c/Users/herna/Desktop/ProyectoEDAII$ ./ProyectoEDAII imaGray_2.jpg
```

Primero tenemos la compilación y la ejecución en donde se puede ver que se manda el nombre de la imagen como parámetro, el resultado en consola es el siguiente:

```
sebas@LAPTOP-SM8Q1ICG:/mnt/c/Users/herna/Desktop/ProyectoEDAII$ ./ProyectoEDAII imaGray_2.jpg
        Resolucion de la imagen
Ancho: 2000
Alto: 1333
Canales: 1
Tamaño: 2666000

Imagen cargada correctamente: 2000x1333 srcIma con 1 canales.

Tiempo en secuencial: 0.031480

Imagen cargada correctamente: 2000x1333 srcIma con 1 canales.

Tiempo en paralelo: 0.017817

Numero de procesadores: 2

El speedup es: 1.766886
La eficiencia es: 0.883443
Tiempo de Over Head: 0.002077

        Otros tiempos Secuencial
Tiempo de carga de imagen: 0.154695
Tiempo de generacion de imagen: 0.762873
Tiempo de generacion de archivo csv: 0.007707

        Otros tiempos Parallel
Tiempo de carga de imagen: 0.338834
Tiempo de generacion de imagen: 1.453551
Tiempo de generacion de archivo csv: 0.005401
```

En este caso se pueden ver los datos de la imagen, el tiempo en secuencial, en paralelo, el cual es más rápido y las métricas además de otros tiempos.



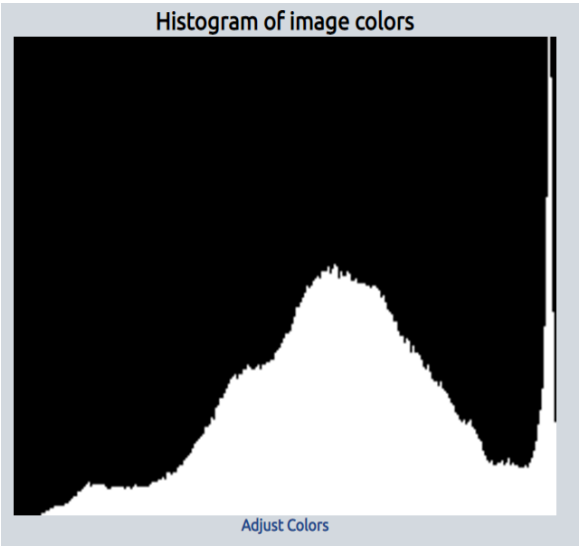


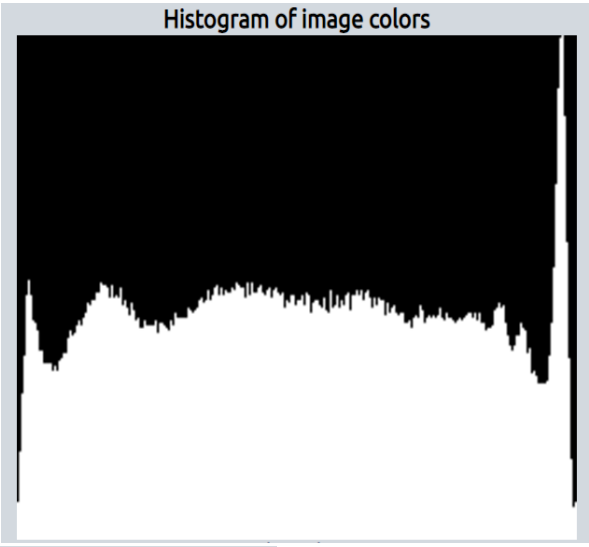
Primero tenemos a la imagen original, después se muestra la imagen en secuencial y, por último, en paralelo, tambien se puede ver que ambas imágenes mantienen el nombre original más un sufijo proporcionado.

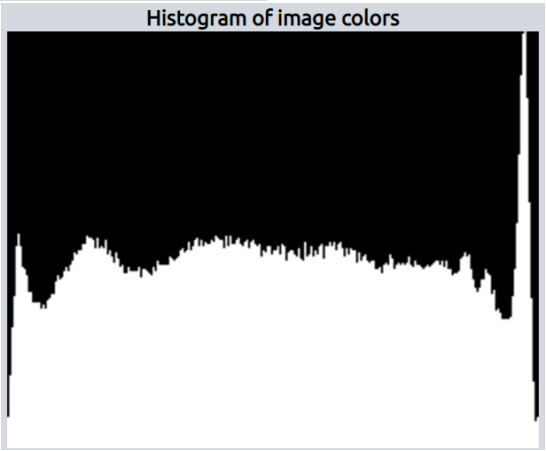
Las métricas antes mencionadas, se puede ver que el speedup es de 1.766886, lo cual significa que la mejora entre las dos ejecuciones fue buena.

La eficiencia fue de 0.883443 es una eficiencia buena, dado el costo de la ejecución.

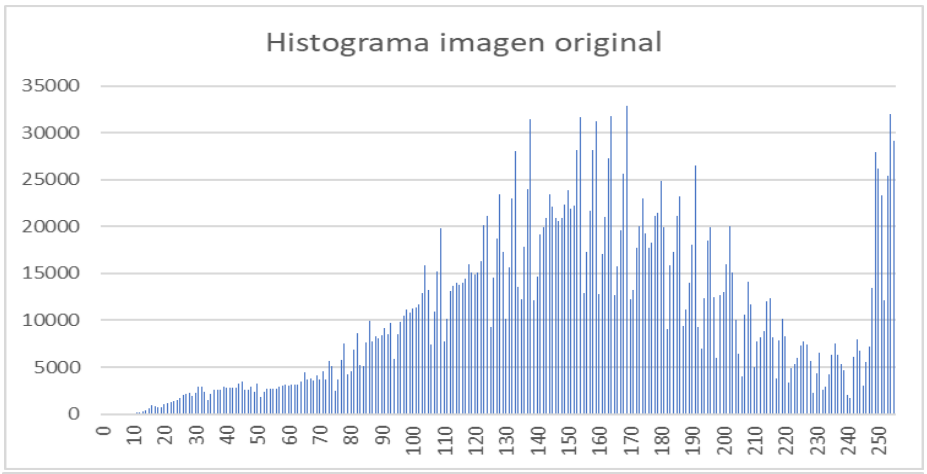
El tiempo de overhead fue de 0.002077 como se vio con anterioridad este tiempo al ser pequeño es muy bueno ya que es el tiempo que se toma al sincronizar datos y demás por lo que entre más pequeño sea es mejor.

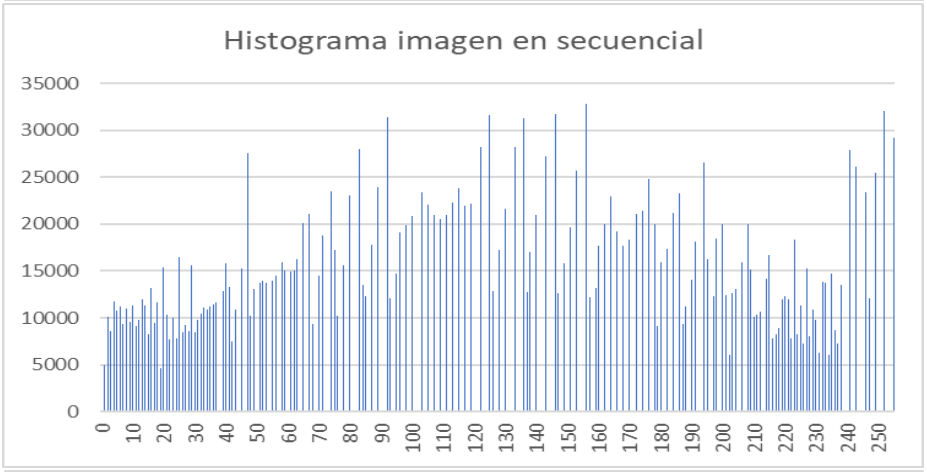


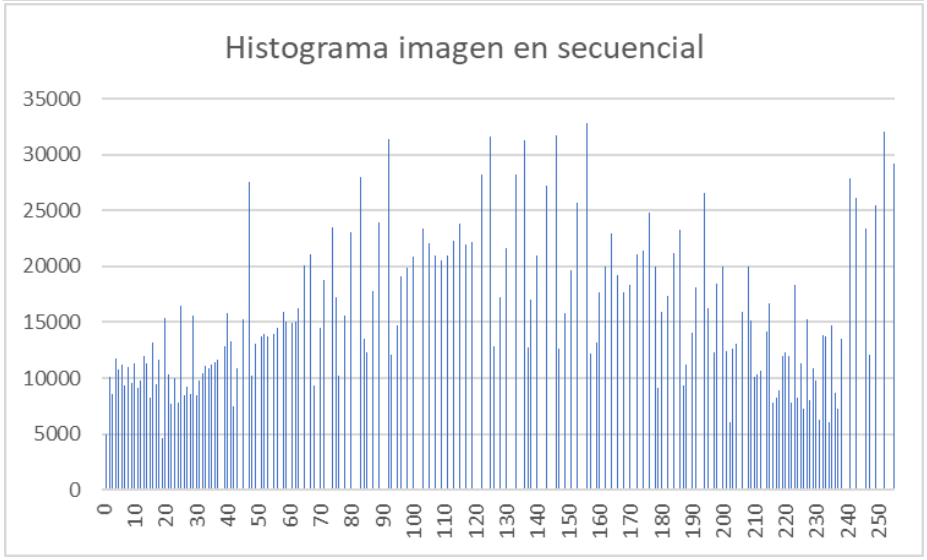




Estos son los histogramas de las imágenes, la primera es de la imagen original, la segunda es de la imagen en secuencial y, por último, se muestra el histograma de la imagen en paralelo, se puede ver como el de las imágenes secuencial y paralelo son los mismos, además de que la forma del histograma de las imágenes es como “estirar” el histograma original de esta forma sus colores se ven de mejor manera en la imagen.





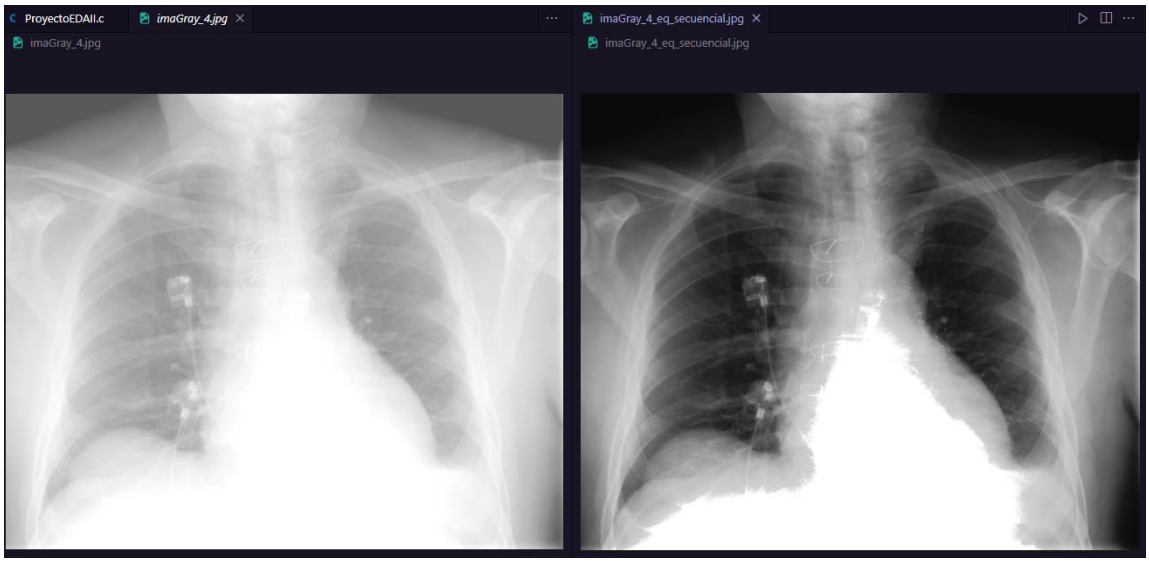


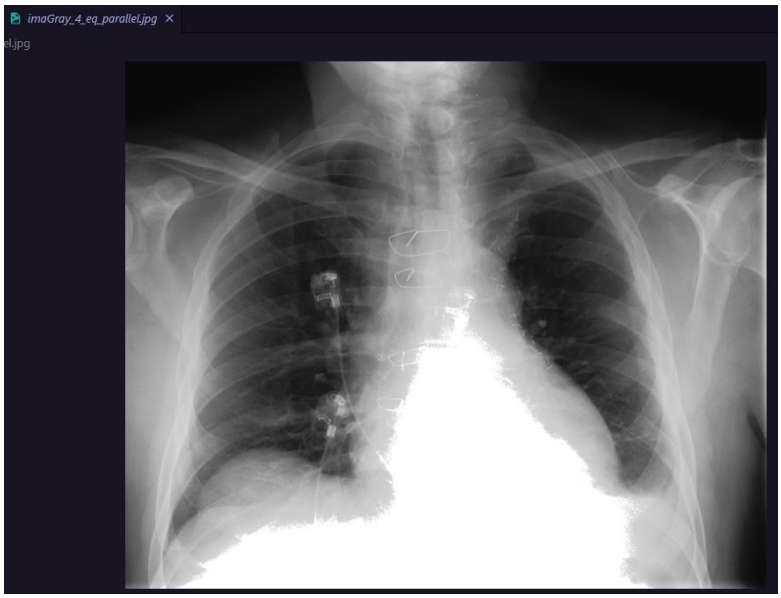
Estas son las gráficas de la imagen, en sus 3 formas, se puede ver que son iguales a las que son dadas con el programa en línea.

```
sebas@LAPTOP-SM8Q1ICG:/mnt/c/Users/herna/Desktop/ProyectoEDAII$ ./ProyectoEDAII imaGray_4.jpg
        Resolucion de la imagen
Ancho: 2828
Alto: 2320
Canales: 1
Tamaño: 6560960

Imagen cargada correctamente: 2828x2320 srcIma con 1 canales.

Tiempo en secuencial: 0.072203

Imagen cargada correctamente: 2828x2320 srcIma con 1 canales.

Tiempo en paralelo: 0.050104

Numero de procesadores: 2

El speedup es: 1.441060
La eficiencia es: 0.720530
Tiempo de Over Head: 0.014003

        Otros tiempos Secuencial
Tiempo de carga de imagen: 0.275316
Tiempo de generacion de imagen: 1.212853
Tiempo de generacion de archivo csv: 0.006322

        Otros tiempos Parallel
Tiempo de carga de imagen: 0.265119
Tiempo de generacion de imagen: 1.260046
Tiempo de generacion de archivo csv: 0.005444
```

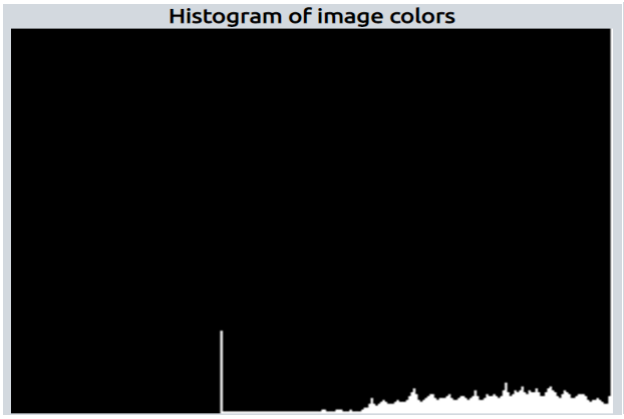
Este es el resultado de la segunda imagen de 1 canal, se pueden ver los datos arrojados por el programa, tambien se ve que el algoritmo en paralelo es más rápido que el secuencial.

Las métricas arrojadas tambien se pueden ver, donde se aprecia que el speedup es de 1.441060 donde se puede ver que hubo una mejora entre los algoritmos, tambien la eficiencia fue alta además de que el tiempo de overhead fue muy pequeño lo cual indica que el tiempo que tomaron distintas actividades no fue muy alto, esto es bueno para el código en paralelo.

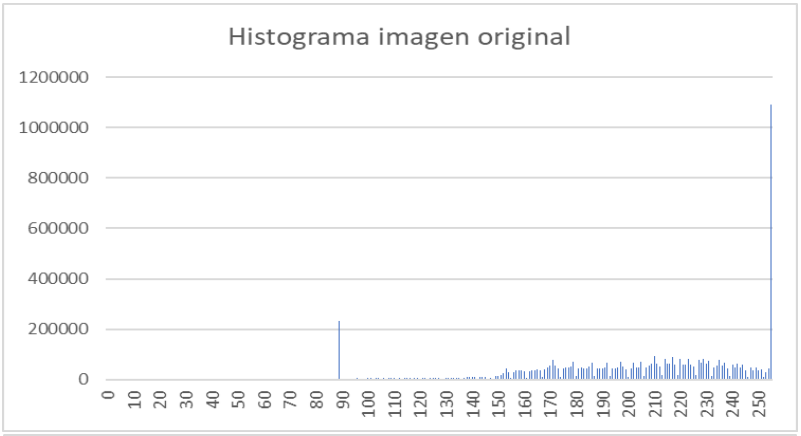


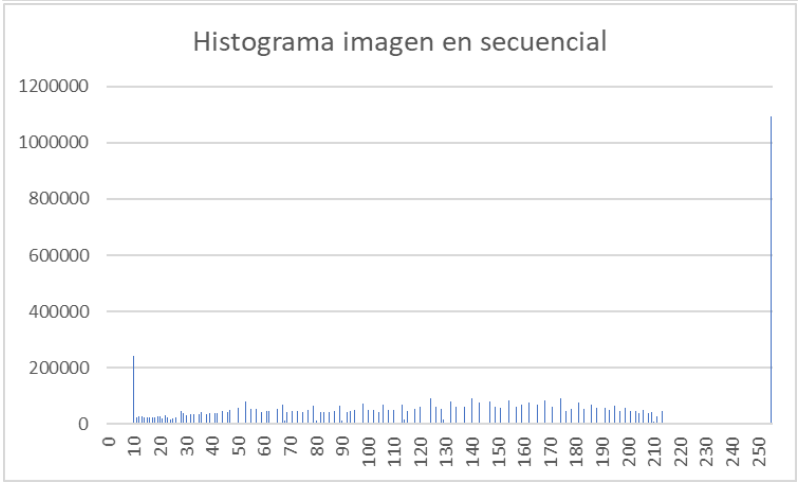


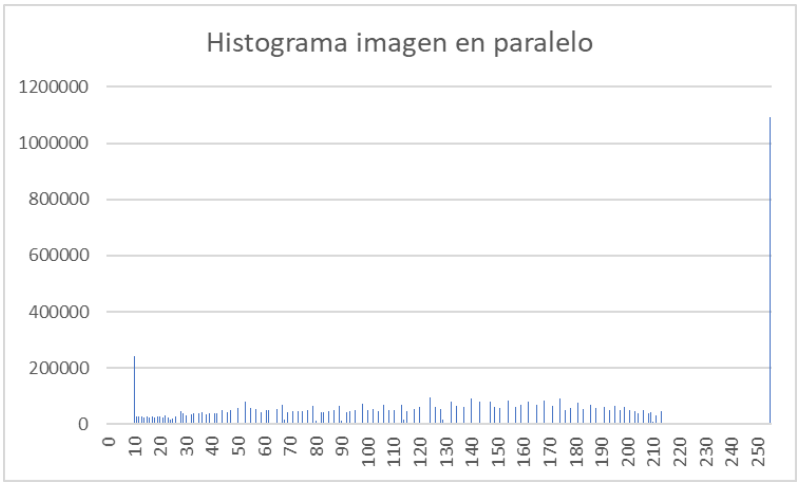
Estas son las imágenes creadas por el código, primero tenemos la imagen original, a su derecha esta la imagen generada en secuencial como se puede ver en su nombre y, por último, tenemos a la imagen en paralelo.



Estos son los histogramas de las imágenes, la primera es de la imagen original, la segunda es de la imagen en secuencial y, por último, se muestra el histograma de la imagen en paralelo, se puede ver como el de las imágenes secuencial y paralelo son los mismos, además de que la forma del histograma de las imágenes es como “estirar” el histograma original de esta forma sus colores se ven de mejor manera en la imagen, se puede ver de manera más plana el histograma dando así una mejora en su vista.





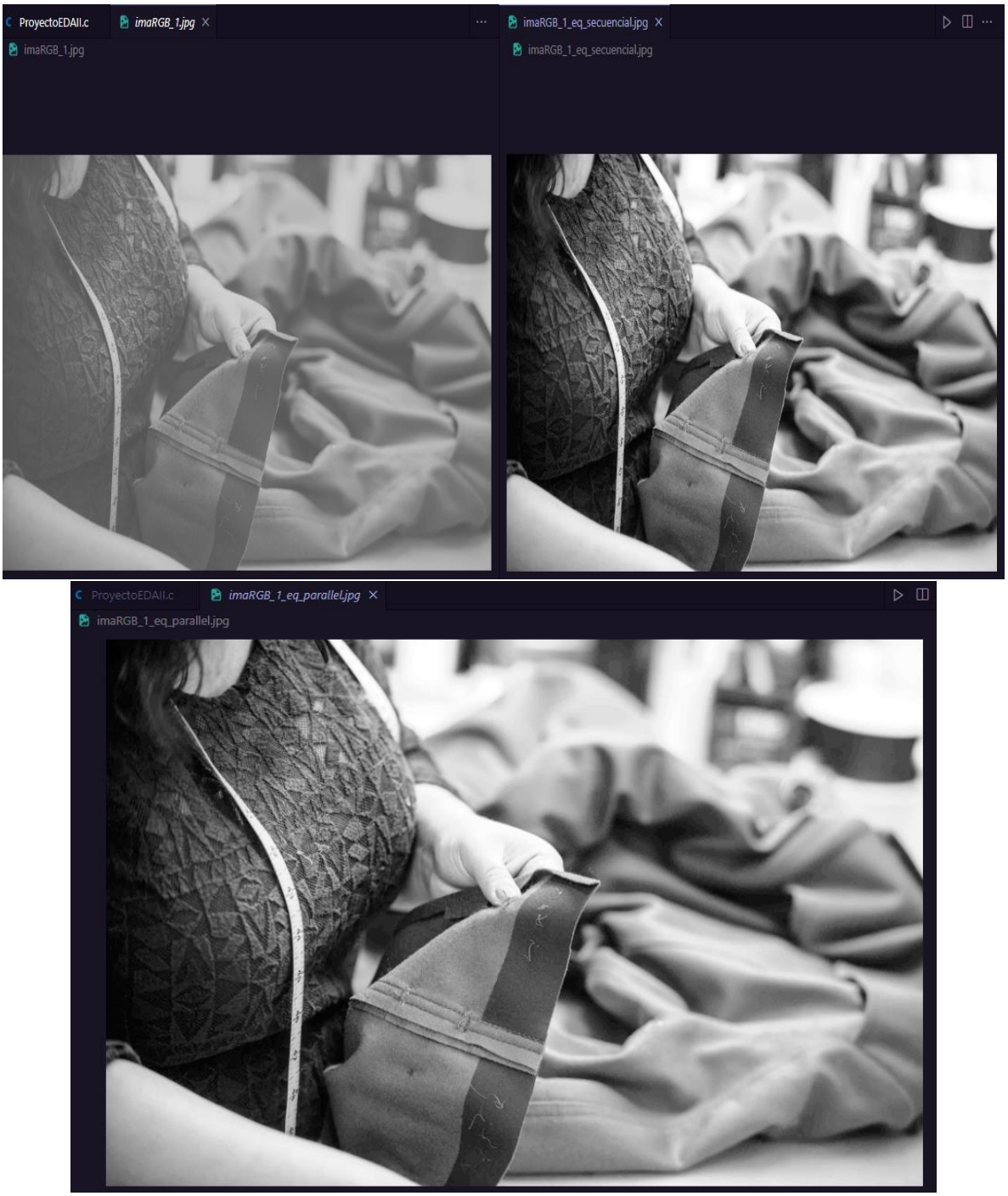


Estas son las gráficas generadas con Excel, se puede ver que son iguales a las del programa y que tanto en secuencial como en paralelo son iguales.

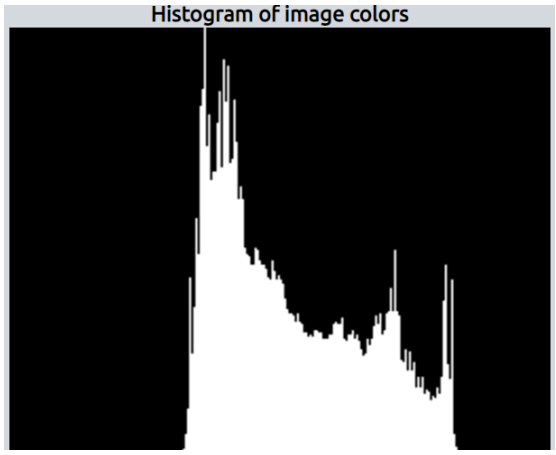
```
sebas@LAPTOP-SM8Q1ICG:/mnt/c/Users/herna/Desktop/ProyectoEDAII$ ./ProyectoEDAII imaRGB_1.jpg
        Resolucion de la imagen
Ancho: 3680
Alto: 2456
Canales: 3
Tamaño: 9038080

Imagen cargada correctamente: 3680x2456 srcIma con 3 canales.

Tiempo en secuencial: 0.220387

Imagen cargada correctamente: 3680x2456 srcIma con 3 canales.

Tiempo en paralelo: 0.174468

Numero de procesadores: 2

El speedup es: 1.263193
La eficiencia es: 0.631596
Tiempo de Over Head: 0.064275

        Otros tiempos Secuencial
Tiempo de carga de imagen: 0.960753
Tiempo de generacion de imagen: 1.161283
Tiempo de generacion de archivo csv: 0.007034

        Otros tiempos Parallel
Tiempo de carga de imagen: 0.681943
Tiempo de generacion de imagen: 1.245198
Tiempo de generacion de archivo csv: 0.005561
```

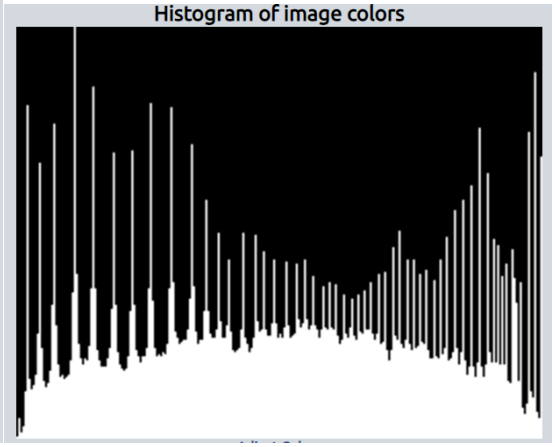
Este es el tercer ejemplo, en este caso es con una imagen RGB de 3 canales, se puede ver su resolución, y el tiempo que tardo en cada algoritmo dando como resultado que el algoritmo en paralelo es más rápido que el secuencial como era esperado.

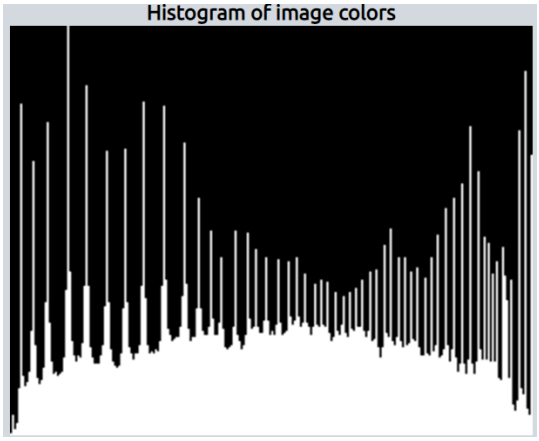
Las métricas arrojadas nos demuestran esto tambien al mostrar un speedup alto, tambien que es eficiente al tener un valor de 0.631596, y, por último, el tiempo de overhead el cual fue muy pequeño mostrando así que entre sincronización y demás no se perdió mucho tiempo.



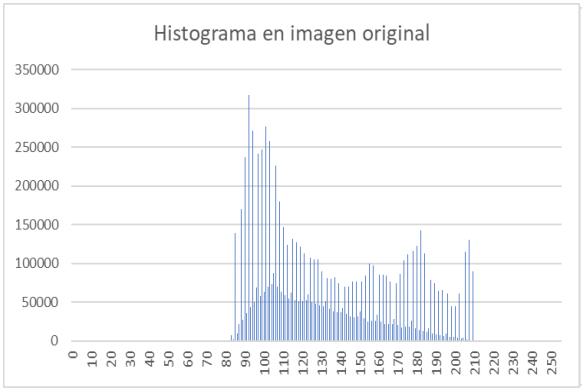
Este es el resultado de la imagen, primero se muestra la imagen original y posteriormente su resultado en secuencial y paralelo, ambas imágenes conservan su nombre de la imagen original y se les agrego un sufijo, como se espera tanto la imagen en secuencial y en paralelo son las mismas, esto se puede ver de mejor manera en el histograma.

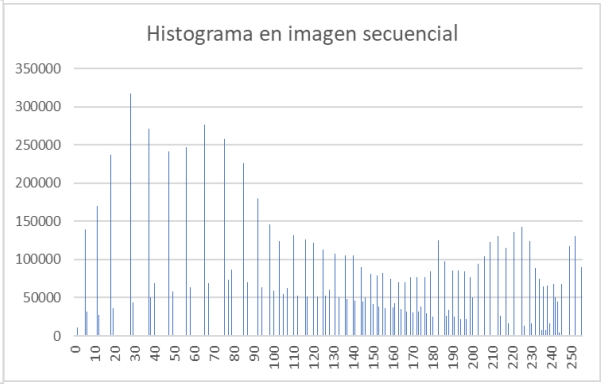


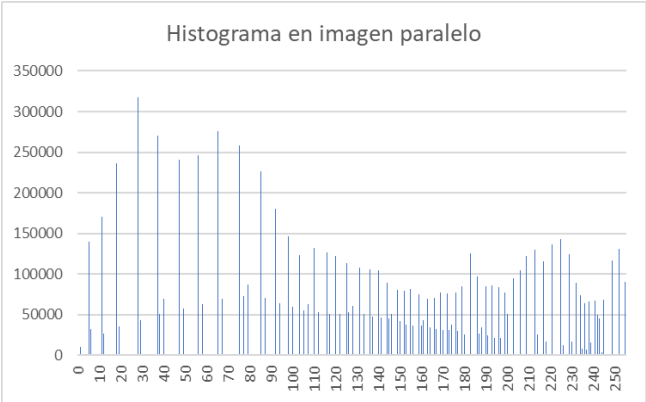




Estos son los histogramas obtenidos de la página proporcionada, se puede ver que tanto el histograma en forma secuencial que es la segunda imagen como el histograma en la tercera imagen que es de la imagen en paralelo son iguales.





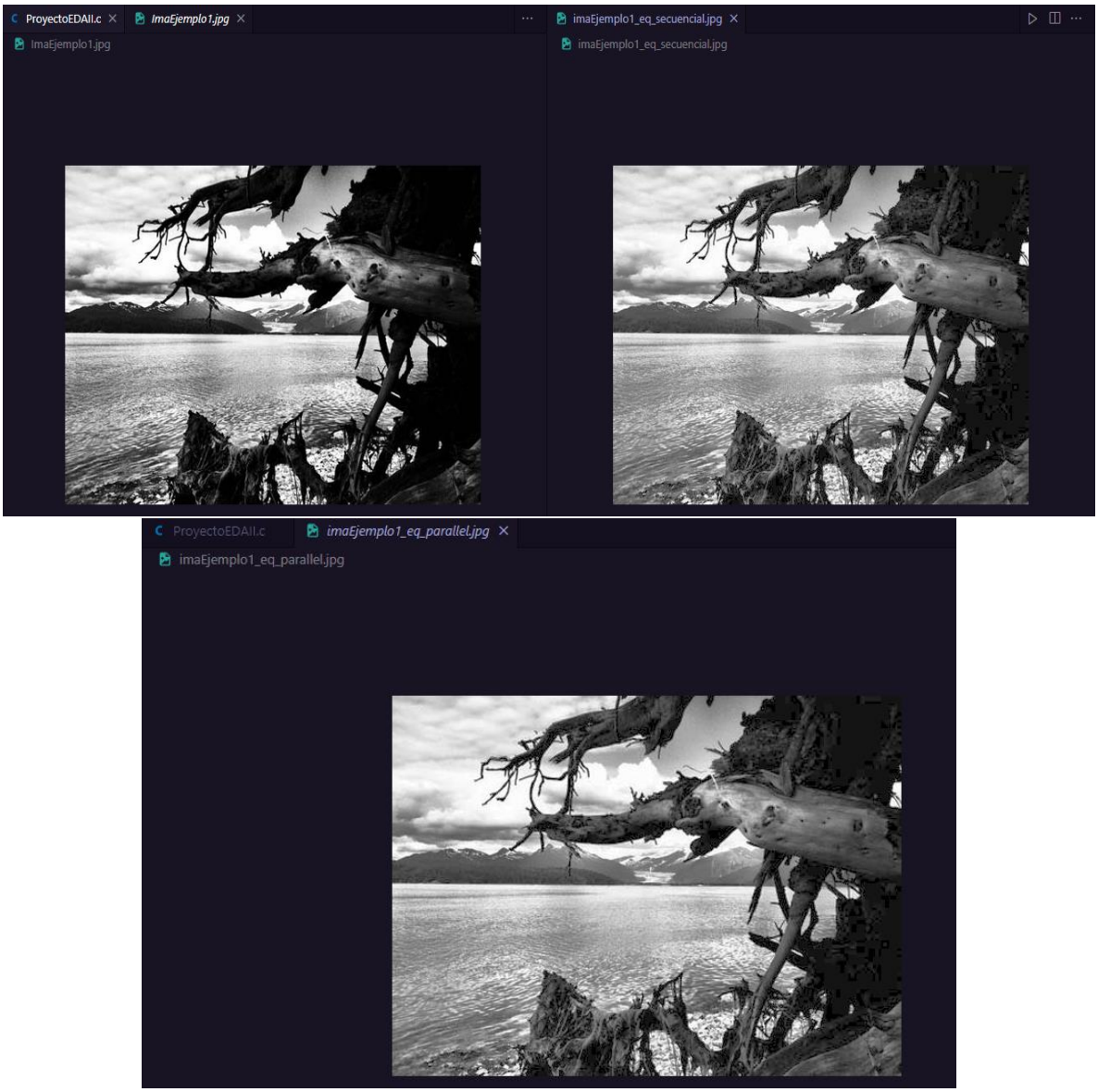


Estas son las gráficas realizadas en Excel con los archivos csv de las imágenes, los cuales se puede observar que son muy similares ya que las variaciones son por la aplicación.

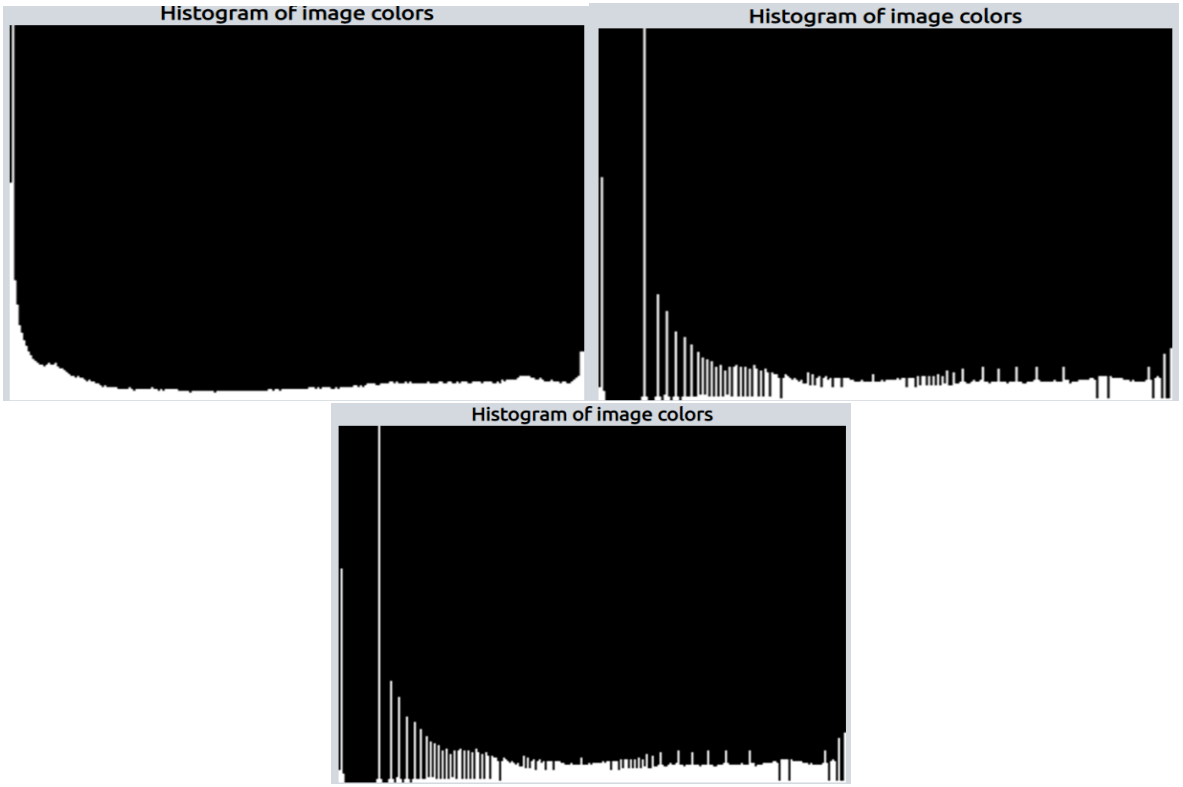
IMÁGENES DE EJEMPLO ESCOGIDAS POR EL ALUMNO

```
sebas@LAPTOP-SM8Q1ICG:/mnt/c/Users/herna/Desktop/ProyectoEDAII$ ./ProyectoEDAII imaEjemplo1.jpg
        Resolucion de la imagen
Ancho: 500
Alto: 373
Canales: 1
Tamaño: 186500

Imagen cargada correctamente: 500x373 srcIma con 1 canales.

Tiempo en secuencial: 0.003553

Imagen cargada correctamente: 500x373 srcIma con 1 canales.

Tiempo en paralelo: 0.002118

Numero de procesadores: 2

El speedup es: 1.678032
La eficiencia es: 0.839016
Tiempo de Over Head: 0.000341

        Otros tiempos Secuencial
Tiempo de carga de imagen: 0.104698
Tiempo de generacion de imagen: 0.060537
Tiempo de generacion de archivo csv: 0.018960

        Otros tiempos Parallel
Tiempo de carga de imagen: 0.008649
Tiempo de generacion de imagen: 0.063948
Tiempo de generacion de archivo csv: 0.007690
```

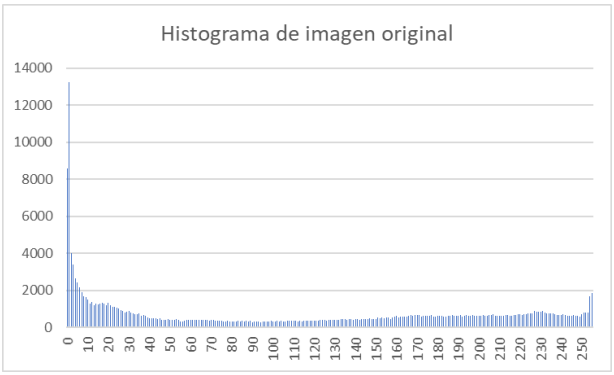
Esta es la ejecución de una imagen que yo seleccione, es de 1 canal, se pueden ver las métricas, tambien que el tiempo en secuencial fue mayor al paralelo como se espera, el speedup fue alto lo cual es bueno para el programa, tambien la eficiencia fue alto lo cual demuestra la comparación entre ambos métodos de la ejecución, y el tiempo de overhead fue muy pequeño lo cual es bueno ya que no tomo mucho tiempo en ciertas tareas como la sincronización y demás.

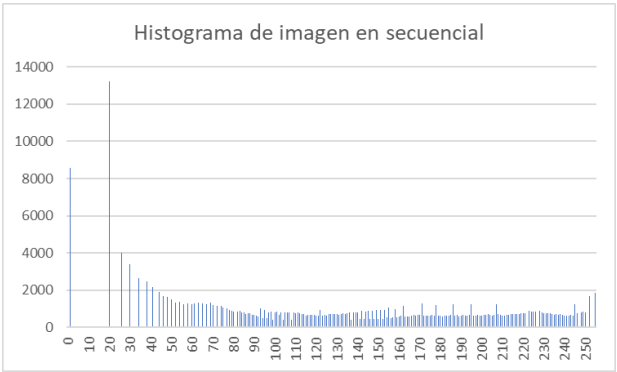


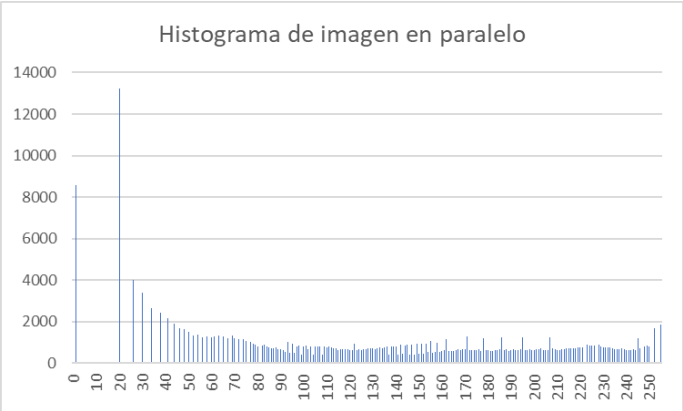
Este es el resultado de las imágenes, se puede observar una mejora ya que las partes oscuras de la imagen original se aprecian de mejor manera en las imágenes ecualizadas.



Este es el resultado de los histogramas dados por la página proporcionada, el primer histograma es el de la imagen original y los otros 2 son de la imagen tanto en secuencial como en paralelo respectivamente.



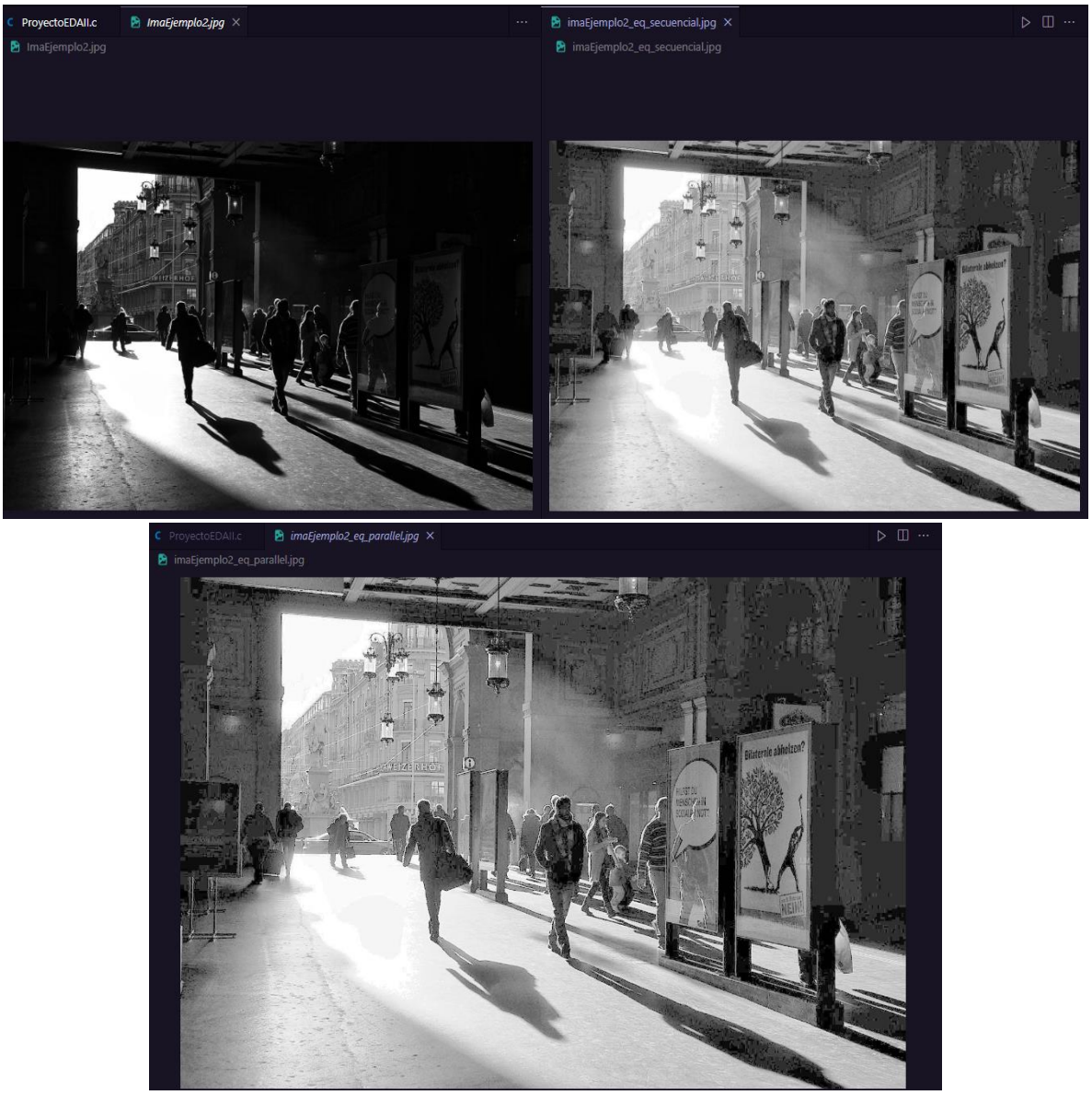




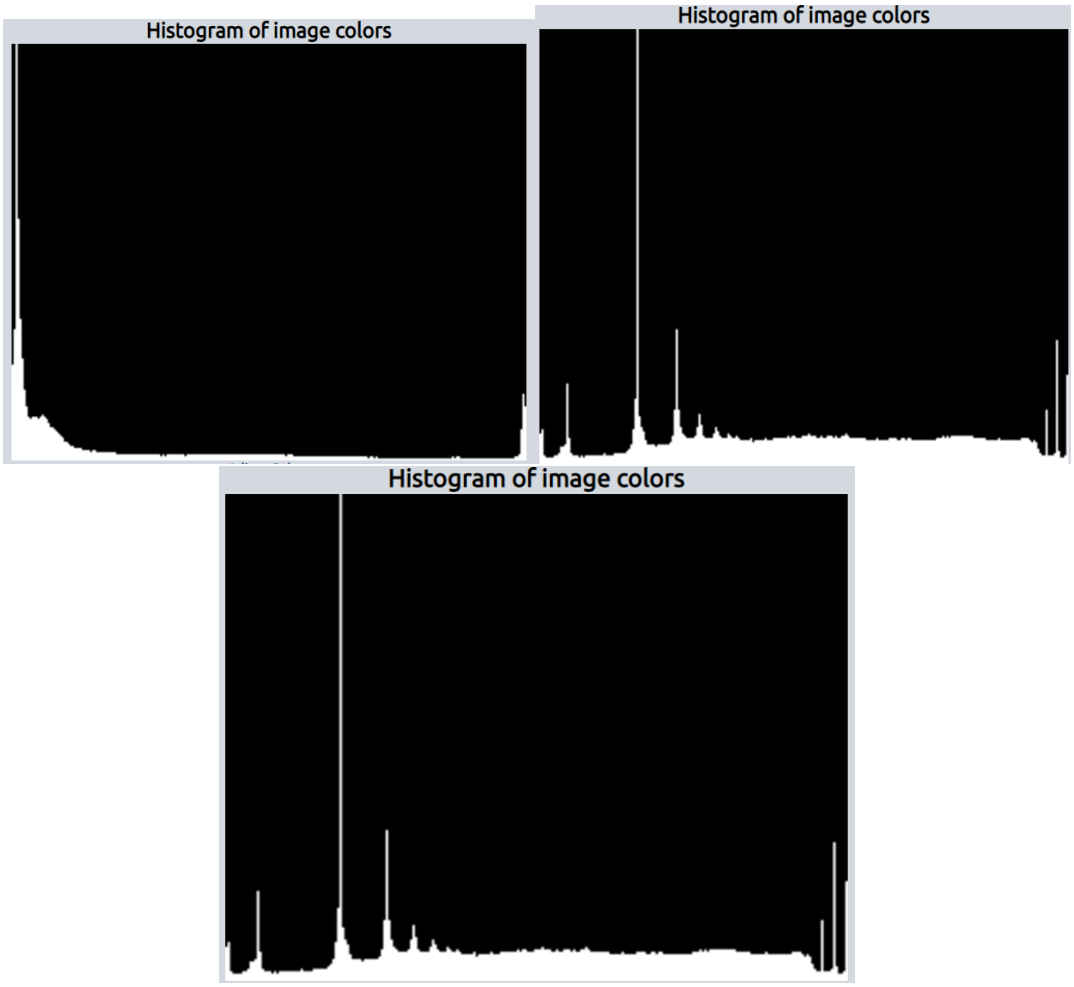
Estas son las gráficas de la imagen hechas en Excel, se puede ver que son iguales a las proporcionadas por la página.

```
sebas@LAPTOP-SM8Q1ICG:/mnt/c/Users/herna/Desktop/ProyectoEDAII$ ./ProyectoEDAII imaEjemplo2.jpg
        Resolucion de la imagen
Ancho: 1542
Alto: 1024
Canales: 3
Tamaño: 1579008

Imagen cargada correctamente: 1542x1024 srcIma con 3 canales.

Tiempo en secuencial: 0.045018

Imagen cargada correctamente: 1542x1024 srcIma con 3 canales.

Tiempo en paralelo: 0.024431

Numero de procesadores: 2

El speedup es: 1.842675
La eficiencia es: 0.921337
Tiempo de Over Head: 0.001922

        Otros tiempos Secuencial
Tiempo de carga de imagen: 0.190229
Tiempo de generacion de imagen: 0.362443
Tiempo de generacion de archivo csv: 0.008375

        Otros tiempos Parallel
Tiempo de carga de imagen: 0.174785
Tiempo de generacion de imagen: 0.311980
Tiempo de generacion de archivo csv: 0.006218
```

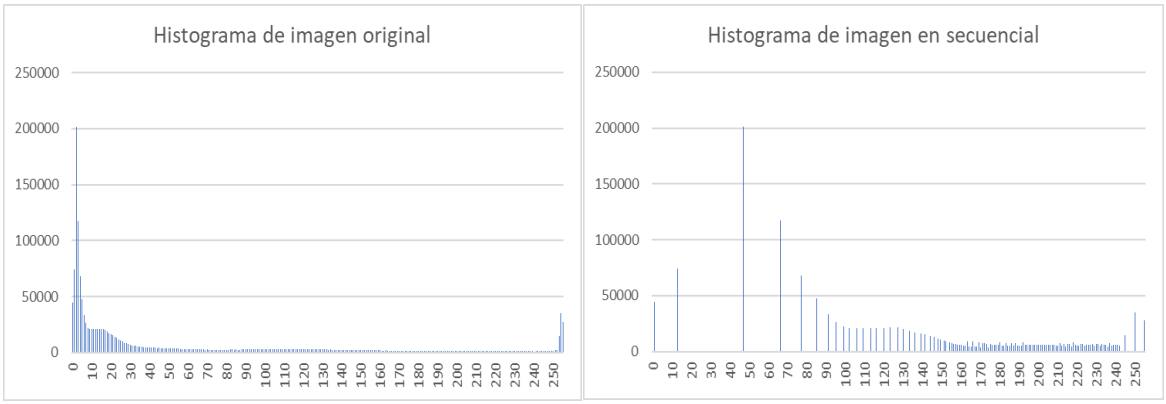
Este es el segundo ejemplo de las imágenes escogidas por mí, se puede ver que es de 3 canales con los cuales se trabajó la imagen, se puede ver que el algoritmo de forma paralela fue más rápido lo cual es lo esperado, tambien se ven las métricas, donde se ve un speedup bueno dando a indicar que el algoritmo es más rápido, esto junto con la eficiencia la cual tambien es alta y con el tiempo de overhead al ser pequeño indica que se perdió poco tiempo en distintas actividades lo cual favorece al algoritmo en paralelo.

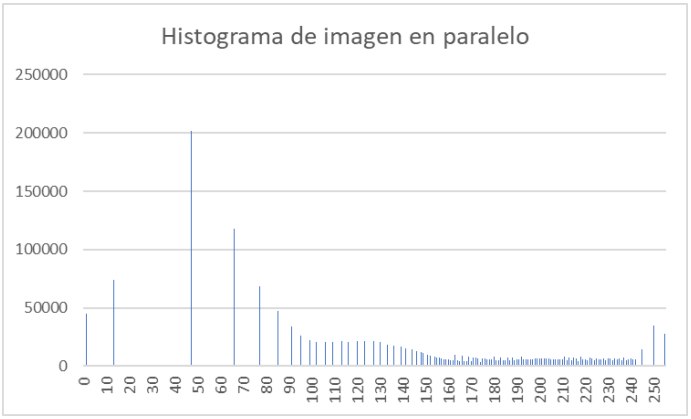


Estas son las imágenes resultantes, primero tenemos la imagen a ecualizar la cual se ve muy oscura por lo cual al ecualizarla esto se arregla dejando ver de mejor forma todo el contenido de la imagen, a la derecha de la imagen original se tiene la imagen en secuencial y debajo de estas 2 se tiene la imagen en paralelo.



Este es el resultado de los histogramas obtenidos mediante la página proporcionada, la primera imagen es el histograma de la imagen original, las demás son del algoritmo en secuencial y paralelo respectivamente.

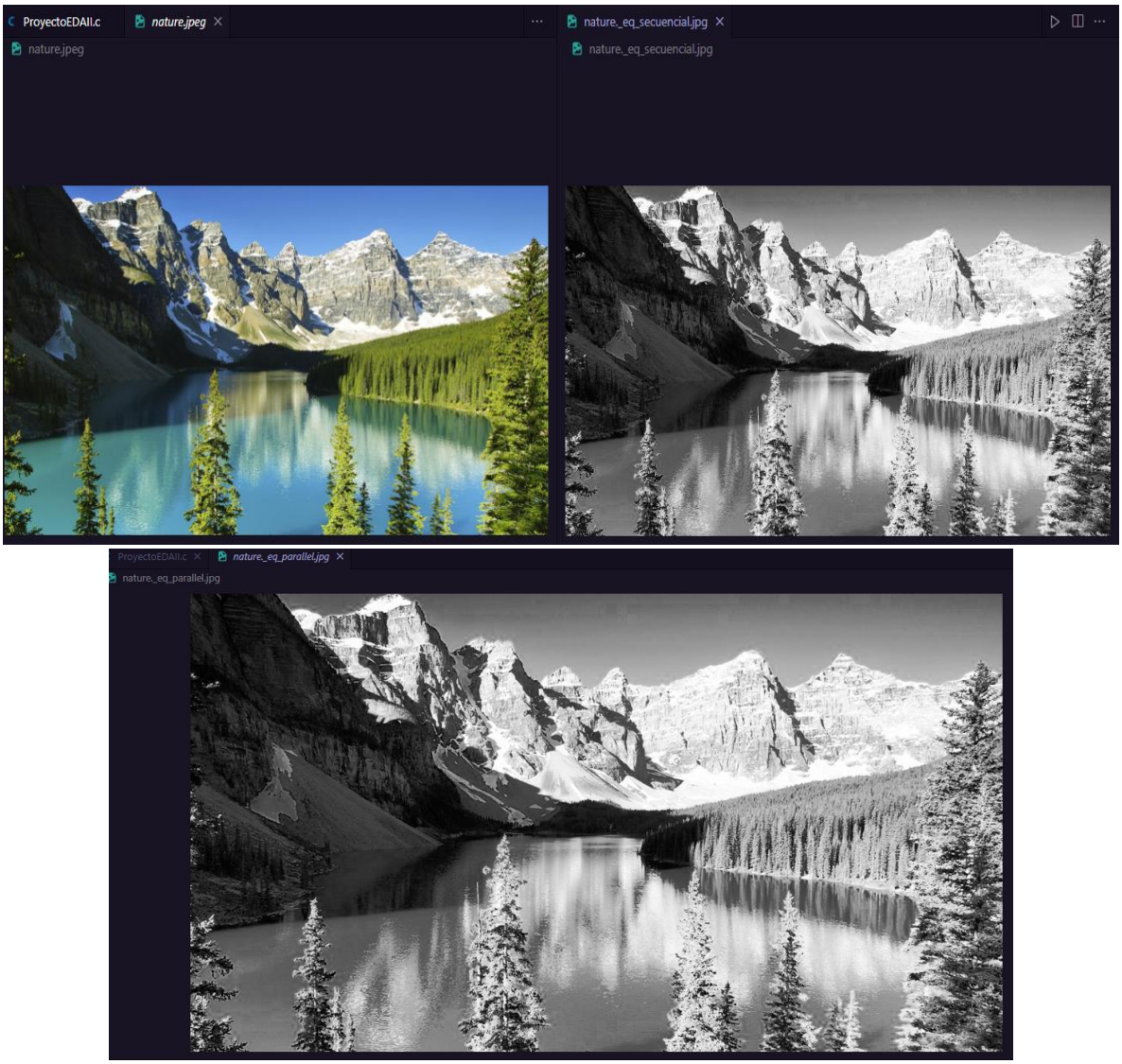




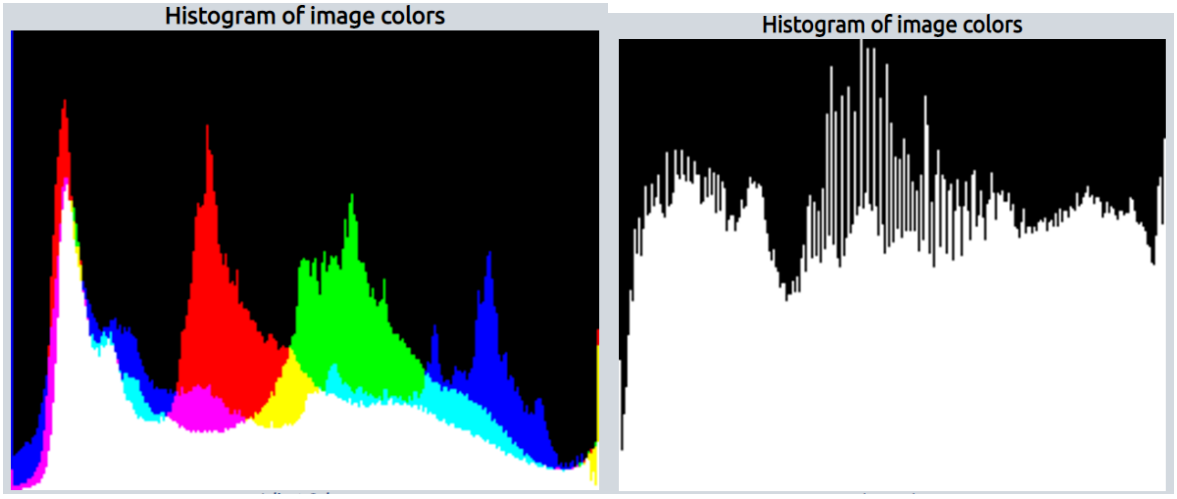
Estas son las gráficas obtenidas mediante los archivos csv generados por los algoritmos, la primera grafica es de la imagen original, la segunda y tercera de los algoritmos en secuencial y paralelo.

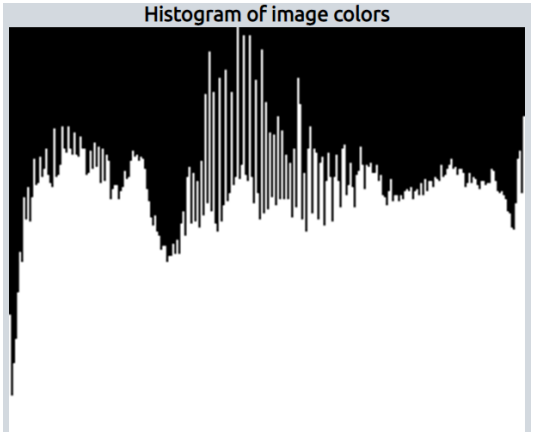
```
sebas@LAPTOP-SM8Q1ICG:/mnt/c/Users/herna/Desktop/ProyectoEDAII$ ./ProyectoEDAII nature.jpeg
        Resolucion de la imagen
Ancho: 1280
Alto: 720
Canales: 3
Tamaño: 921600

Imagen cargada correctamente: 1280x720 srcIma con 3 canales.

Tiempo en secuencial: 0.022247

Imagen cargada correctamente: 1280x720 srcIma con 3 canales.

Tiempo en paralelo: 0.016940

Numero de procesadores: 2

El speedup es: 1.313298
La eficiencia es: 0.656649
Tiempo de Over Head: 0.005816

        Otros tiempos Secuencial
Tiempo de carga de imagen: 0.099075
Tiempo de generacion de imagen: 0.225757
Tiempo de generacion de archivo csv: 0.015388

        Otros tiempos Parallel
Tiempo de carga de imagen: 0.080244
Tiempo de generacion de imagen: 0.313466
Tiempo de generacion de archivo csv: 0.008171
```

Este es el resultado de la ejecución en este caso con una imagen a color de 3 canales, se pueden ver las métricas donde se indica tanto en el speedup como en la eficiencia que el algoritmo en paralelo mejora los tiempos con respecto al secuencial, tambien en el tiempo de overhead se nota que el tiempo que tomo en otras actividades no fue mucho.

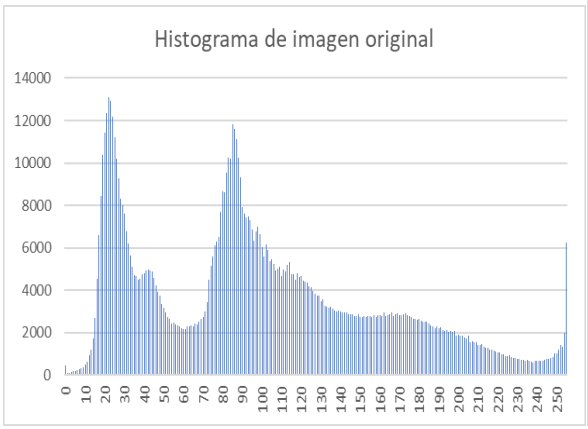


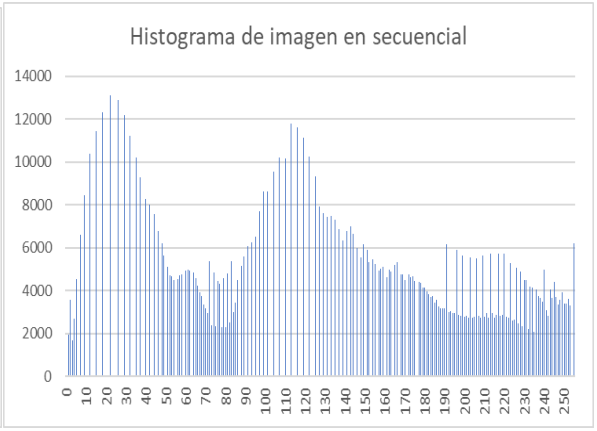
Este es el resultado después de ejecutar el programa, se puede ver que la imagen guardada es en colores de gris.

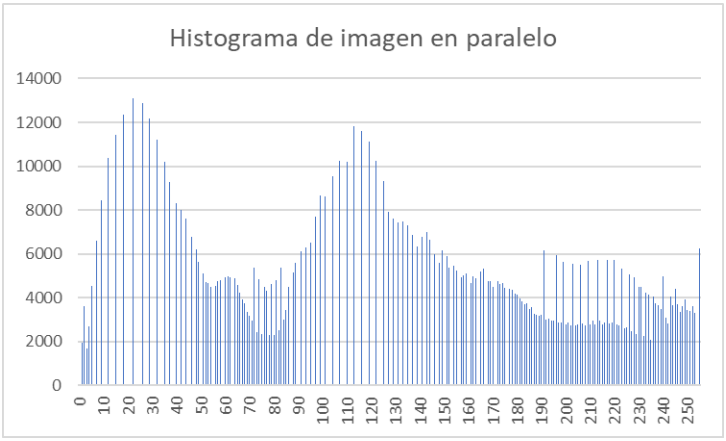




Estos son los histogramas obtenidos mediante la página dada, en el caso de la imagen original se puede ver que son 3 uno por cada color, en este caso el histograma que se toma es el de color rojo y con ese se trabaja.







En estas graficas se ve de mejor manera como en la primera imagen que es la del histograma de la imagen origina si se compara con la obtenida en la página se puede ver que el histograma usado es el rojo, tambien se ve de mejor manera que al ecualizar este histograma se “estira” para así poder obtener una mejor imagen.

CONCLUSIONES

En este proyecto se aplicaron muchas cosas, pero principalmente fueron 2 que sería el manejo de archivos y los algoritmos en paralelo, en especial este segundo fue el que mas se dificulto porque como se vio en clase muchas veces se trabaja de distinta forma por cada ejecución, esto fue algo que me ocurrió pues cuando ejecutaba el programa me generaba una imagen, pero por el manejo de hilos esta siempre era diferencia y no se parecía a la versión en secuencial más que un par de veces, pero esto se resolvió utilizando las clausulas correctas en donde son necesarias, esto haciendo que los hilos se sincronizaran en distintos puntos del programa para hacerlo de manera correcta, la directiva reduction fue de gran ayuda porque gracias a esta pude realizar muchos pasos en el cálculos de los histogramas y que no salieran diferentes en cada ejecución, la parte de escritura de los archivos csv fue muy sencilla gracias a los ejemplos proporcionados, en este curso vimos los archivos en el lenguaje Python, en c es bastante similar pero la lógica ya se tenía, lo cual es lo mas importante porque no importa en que lenguaje se implemente la teoría el como se manejan los archivos es muy similar sino que igual, en este caso eso fue de gran ayuda y con la sintaxis simplemente era poner atención a los ejemplos y poder aplicarlos en nuestro código.

En los pasos para poder ejecutar el código de manera correcta fueron algo sencillos pero en el caso del calculo del cdf ecualizado fue algo complicado aplicar la formula, cuando intente hacer toda la formula en una sola línea de código como es mostrada en la información que se recibió siempre obtenía valores de cero al aplicar la función round, es por esto que decidí aplicar la formula separándola, de esta forma funciono de mejor manera y el código obtenía los resultados correctos, el uso de la librería fue de gran ayuda porque esto nos facilito todo el manejo con la imagen, no nos teníamos que preocupar por como obtener el histograma de una imagen y de ahí trabajarlo ni tampoco saber como dado un histograma obtener una imagen, el uso de esta librería fue sencillo y solo se uso en esas ocasiones donde podría llegar a dificultarse el proyecto, la versión con 3 canales fue tambien algo sencilla pues solo en algunas ocasiones se necesitaba cambiar el código ya que se debía de tomar en cuenta los canales.

En conclusión, este proyecto nos ayudo a reforzar mas los conocimientos que se vieron en clase, tambien nos ayudo a mejorar la lógica con hilos, comprenderlos de mejor manera y aprender a usar de mejor manera todas las directivas que nos ofrece OMP con la programación paralela, es un buen proyecto y nos enseña algo muy interesante como lo es el procesamiento de imágenes.