



Facultad de Ingeniería  
Universidad de Buenos Aires  
Trabajos prácticos: Procesamiento de imágenes.

2<sup>er</sup> Cuatrimestre, 2019

---

Alejandro, Pablo Martín      98021      pabloale96@gmail.com

---

## 1. Introducción

En este trabajo práctico, se realizó una compilación de ejercicios propuesto en clase sobre el manejo de las imágenes. Los mismos se realizaron en código y dieron resultados. Vamos a destacar cada uno por separado y mostrar dichos resultados.

## 2. Desarrollo

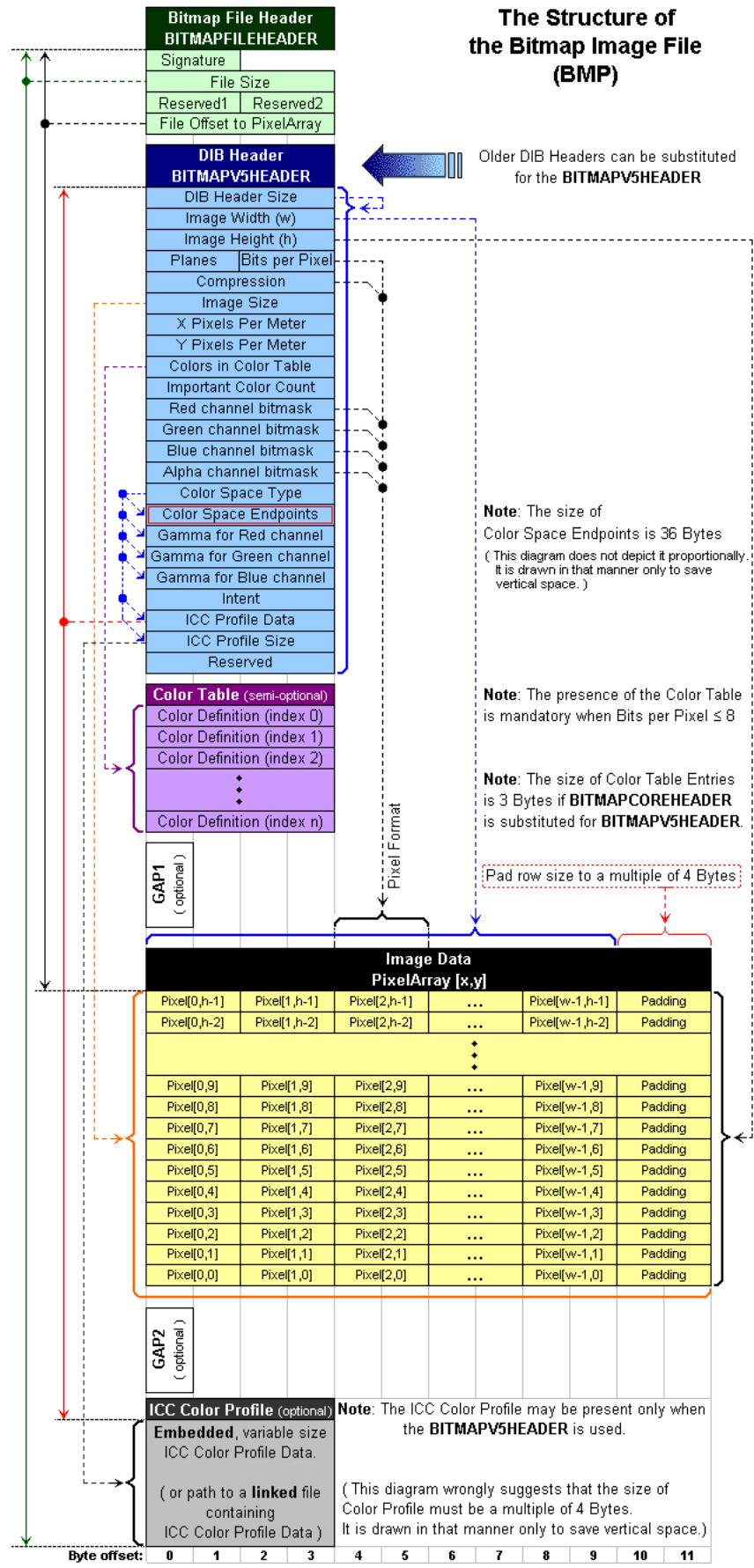
### 2.1. Manejo del Header de una Imagen

Como se sabe, hay diferentes formatos para guardar la información de una imagen, entre ellos puede ser *.jpg, .png, etc.* Para el análisis del header de la imagen, se fijo dicho formato eligiendo el formato *bmp* de *24-bytes*. En la figura (2), se observa que el formato consiste en 3 encabezados correspondiente sobre la información del archivo, la información de colores y sobre la información de los datos. Luego tiene un vector que se corresponde a los datos.

La modificación del header se hizo a través de una imagen patrón conocida como *Lenna* en formato *.bmp*. La imagen se muestra en la figura (1) tiene una altura de *512* y un ancho de *512*.



Figura 1: Imagen patrón usada en los trabajos prácticos de  $512 \times 512$

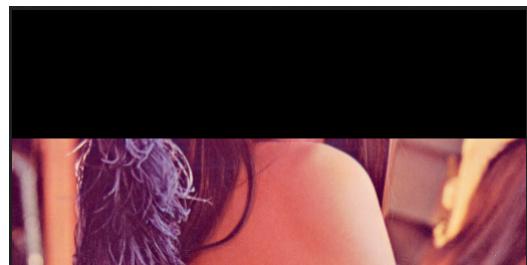
Figura 2: Encabezado de una imagen con formato  *bmp*

### 3. Modificación del Ancho y el Alto de la imagen

En este punto se agarro la imagen patrón y se modifco los valores de la altura y el ancho de la imagen observando diferentes resultados, se comenzó reduciendo y multiplicando la altura y el ancho obteniendo lo siguiente

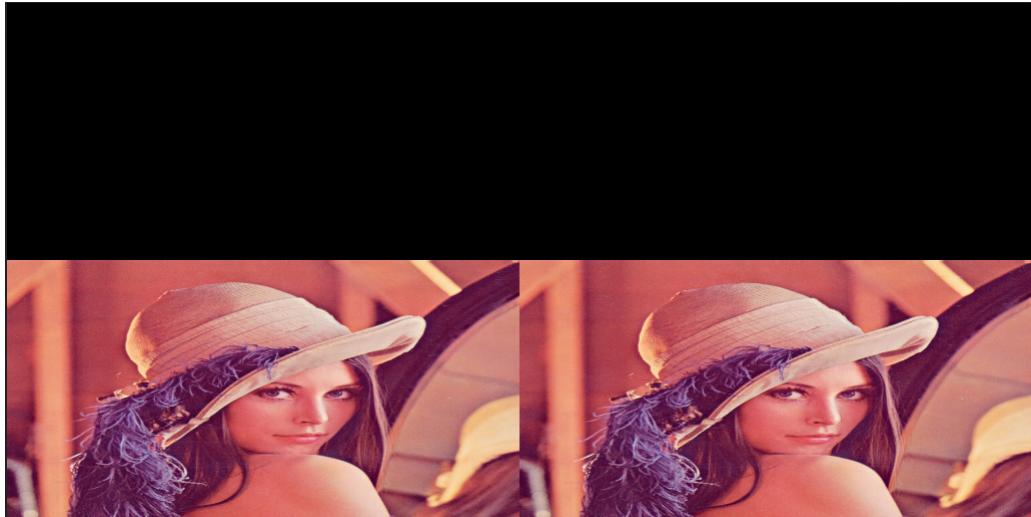


(a) Altura multiplicada por 2



(b) Altura dividida por 2

Figura 3: Resultado de cambiar la altura de la imagen.



(a) Ancho multiplicado por 2



(b) Ancho dividido por 2

Figura 4: Resultado de cambiar el ancho de la imagen.

Mirando el análisis de la modificación del header se decidió, cambiar el alto y el ancho al mismo tiempo, obteniendo lo siguiente



Figura 5: Resultado de cambiar el ancho y alto de la imagen.

Ahora si modiflico un alto o un ancho y lo vuelvo a la normalidad se estarán perdiendo datos porque cambio la imagen pierde los valores que se modificaron obteniendo

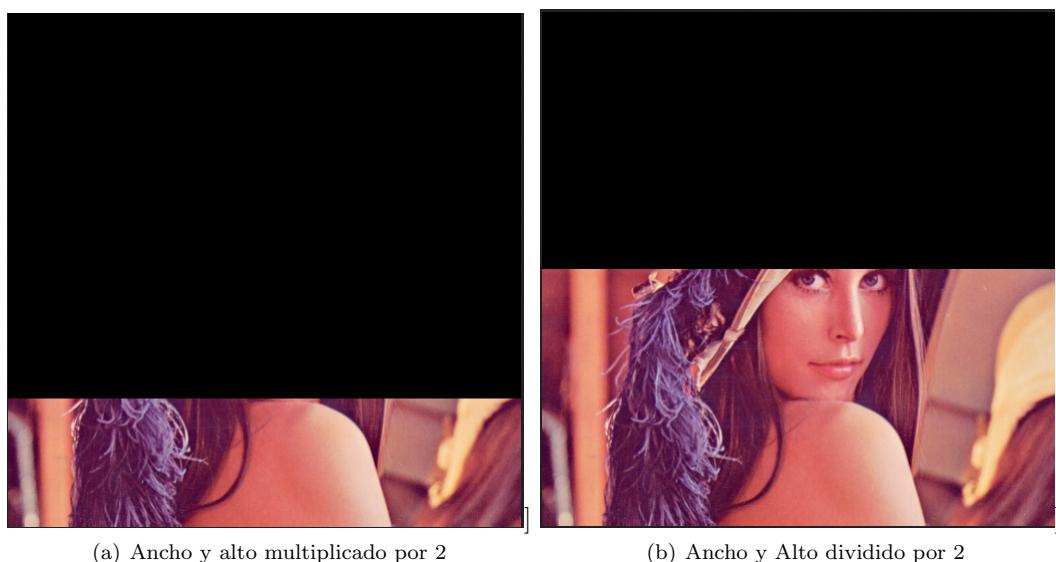
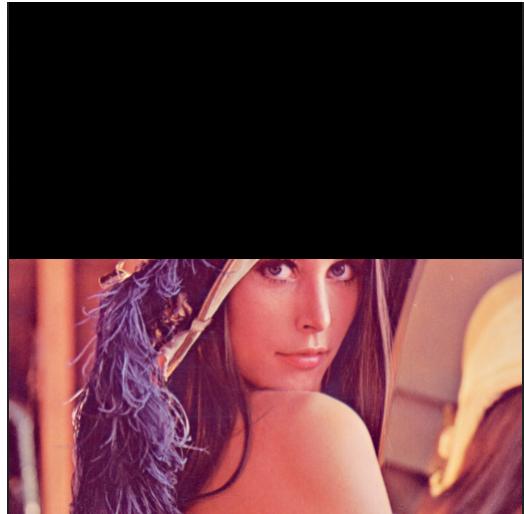
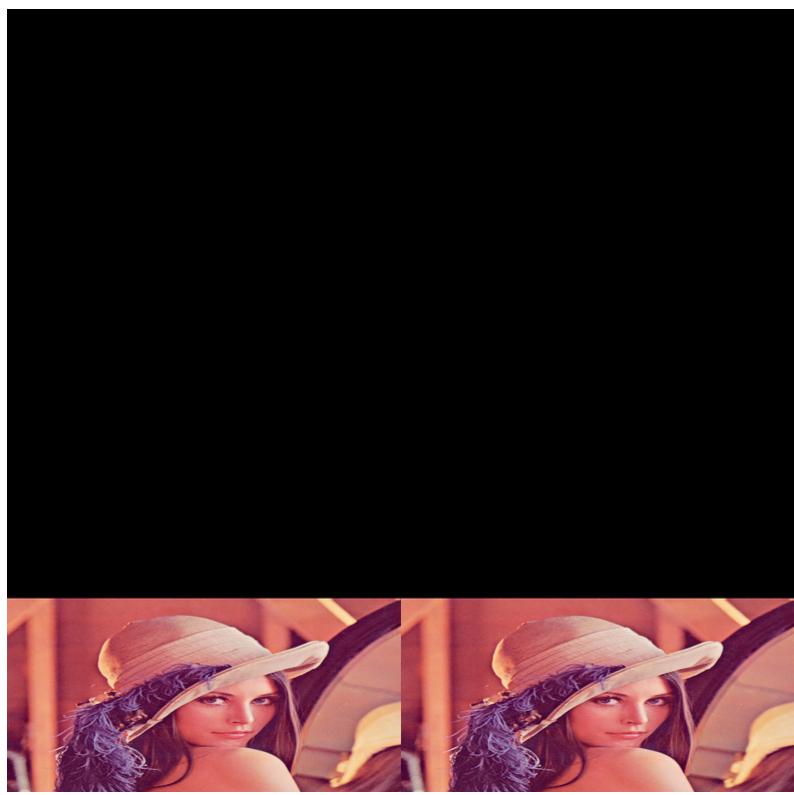


Figura 6: Resultado de cambiar el ancho o alto de la imagen y volver al tamaño original.

También se modiflico el valor del tamaño de datos de la imagen, acá se hicieron varias pruebas solo modificando el tamaño y también modificando tanto el tamaño como el alto y el ancho.



(a) Tamaño de datos dividido por 2



(b) Tamaño de datos multiplicado por 2 y alta y ancho de la imagen igual a 1024x1024

Figura 7: Resultado de cambiar el tamaño de datos de la imagen.

Por ultimo, para el manejo del header se decidió agrandar la imagen pero en lugar de no hacer nada, se decidió copiar los datos para obtener replicas de la imagen. Para eso, se tomo el vector de datos de la imagen original y se copio tantas veces que entran en la nueva imagen obteniendo el siguiente resultado

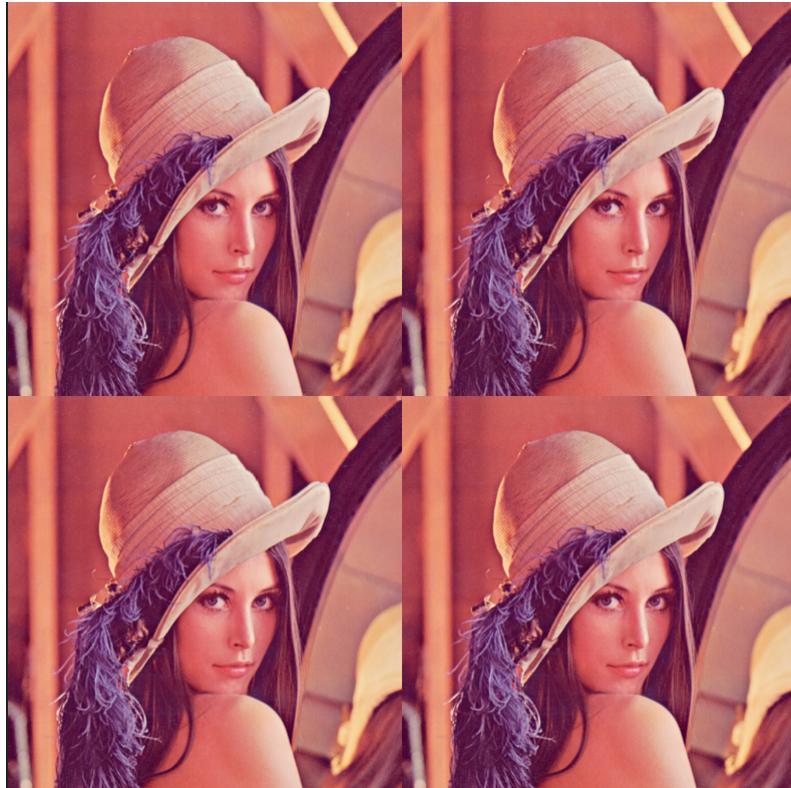


Figura 8: Imagen resultante del copiado de datos.

## 4. Algoritmo para escala de grises

En esta sección se tratan tres métodos para transformar la imagen *lenna.bmp* a una imagen en escalas de grises. Para eso se uso un promedio de los colores RGB, también se tomo el valor del gris en RGB y por ultimo se modiflico la paleta para obtener una imagen en escalas de grises.

### 4.1. Promedio de los colores RGB

Este algoritmo consiste en tomar los valores RGB de cada pixel y hacer un promedio del mismo. Esto se puede representar como (tomando que el valor de cada pixel se divide en 3 valores RGB)

$$P_i = \frac{R_i + G_i + B_i}{3} \quad (1)$$

siendo  $P_i$  el valor del nuevo pixel, por lo que se recorre la imagen haciendo calculado el valor de cada nuevo pixel obteniendo



Figura 9: Imagen resultante en escala de grises.

#### 4.2. Valor del gris

En este algoritmo, se busca el valor del gris en la escala RGB, y se vuelve a calcular cada pixel con respecto a este valor. El valor del nuevo pixel se puede calcular como

$$P_i = 0,299 \times R + 0,587 \times G + 0,144 \times B \quad (2)$$

de esta forma se escala cada pixel con el color del gris, obteniendo



Figura 10: Imagen resultante en escala de grises.

#### 4.3. Paleta

La paletizacion de la imagen consiste en definir la paleta de colores en blanco y negro tomando 256 colores. La paleta en blanco y negro se define tomando la matriz y asignando la matriz de la paleta como

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \\ \vdots & \vdots & \vdots \\ 255 & 255 & 255 \end{pmatrix} \quad (3)$$

el resultado de usar esta paletización se observa en la imagen (11). La idea es asignar a cada pixel el valor del promedio de la fila de la paleta, se observa que se termina haciendo algo parecido en el algoritmo del promedio de los colores RGB.



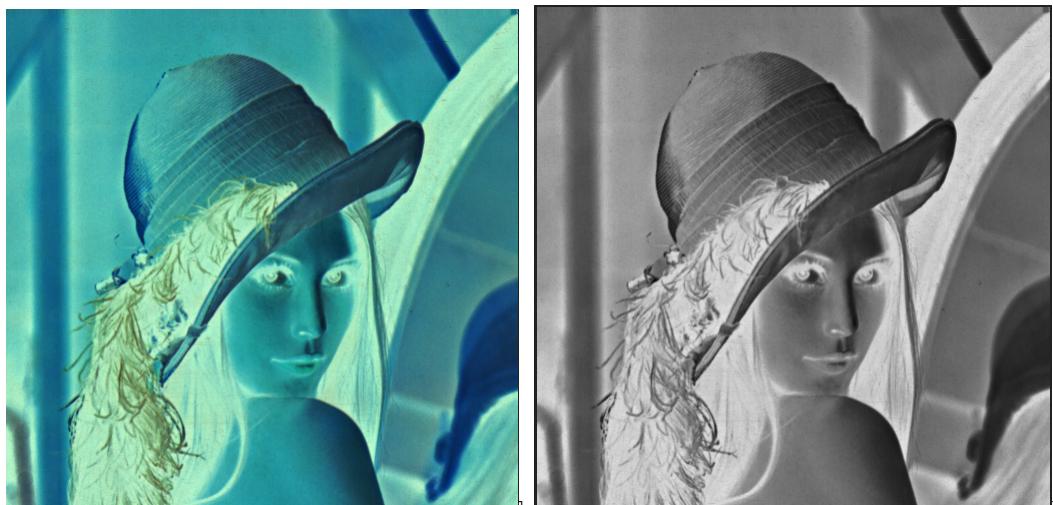
Figura 11: Imagen resultante en escala de grises.

## 5. Negativo de la imagen

El cálculo del negativo de una imagen consiste en tomar el valor valor de cada pixel de la imagen  $P_i$  y calcular

$$P_{i,nuevo} = 255 - P_i \quad (4)$$

donde 255 corresponde al valor del color máximo. Así se obtiene un negativo de la imagen que corresponde al valor máximo menos el valor en ese pixel. Los resultados son



(a) Imagen negativa con lenna a color

(b) Imagen negativa con lenna en escala de grises

Figura 12: Resultado de la imagen negativa.

## 6. Imágenes agregando un ruido

### 6.1. Ruido constante

Se agarro cada pixel de la imagen y se le agrego un ruido constante de valor 150. Matemáticamente seria

$$P_i = P_i + 150 \quad (5)$$

el resultado obtenido es una imagen mas clara, teniendo en cuenta que se aumenta el valor de cada pixel ahora si se reduci el valor se obtendria una imagen mas oscura. A continuación se observa la imagen resultante



Figura 13: Imagen resultante.

### 6.2. Ruido aleatorio con distribución Normal

El concepto es el mismo que ruido constante solo que a cada pixel le sumo un valor generado por una densidad de probabilidad normal de media 100 y desvió 50. Al realizar esto, como la media la media es mucho mas grande que el desvió provoca que en algunos pixeles la imagen se vuelva mas clara que en otros.



Figura 14: Imagen resultante.

### 6.3. Ruido Aleatorio con distribución Bernulli

Al igual que en el caso anterior, se tomo el valor de cada pixel y se lo sumo con valores generados por una densidad bernulli generando 1 o 0, que luego se multiplica por un valor de 100 obteniendo que algunos píxeles se suen con este valor y otros que no sumen nada.



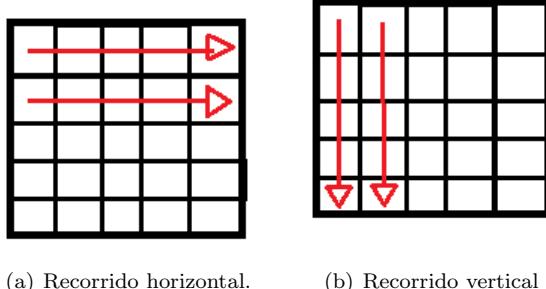
Figura 15: Imagen resultante.

## 7. Recorrido de una imagen

Existen diferentes métodos para recorrer una imagen, entre ellos los dos más conocidos son el horizontal y el vertical para los cuales se calcula la diferencia entre los pixeles para obtener el contorno de la imagen. Luego, se realizó el recorrido de la imagen con un recorrido conocido como Hilbert.

## 8. Recorrido horizontal y vertical

En la figura (??), muestra la forma de ambos recorridos. Considerando a la imagen como un matriz, cada recorrido se puede definir como variando el paso de esta matriz, esto sería que para el recorrido horizontal se mantenga fija la fila y me muevo por las columnas mientras que para el recorrido vertical mantengo fijo las columnas y me muevo por las filas.



(a) Recorrido horizontal.

(b) Recorrido vertical

Teniendo en cuenta el valor de cada pixel, se realizó los dos recorridos con el propósito de buscar un contorno de la imagen, tomando la diferencia en cada pixel y preguntando si era mayor a un cierto umbral. El mismo se mantuvo fijo para ambos recorridos, luego se realizó el histograma sobre la diferencia de los pixeles obteniendo valores muy pequeños o valores muy grandes, teniendo en cuenta que la diferencia entre pixel es muy grande cuando hay un borde.



(c) Contorno con recorrido horizontal.



(d) Contorno con recorrido vertical

Figura 16: Resultados de ambos recorridos

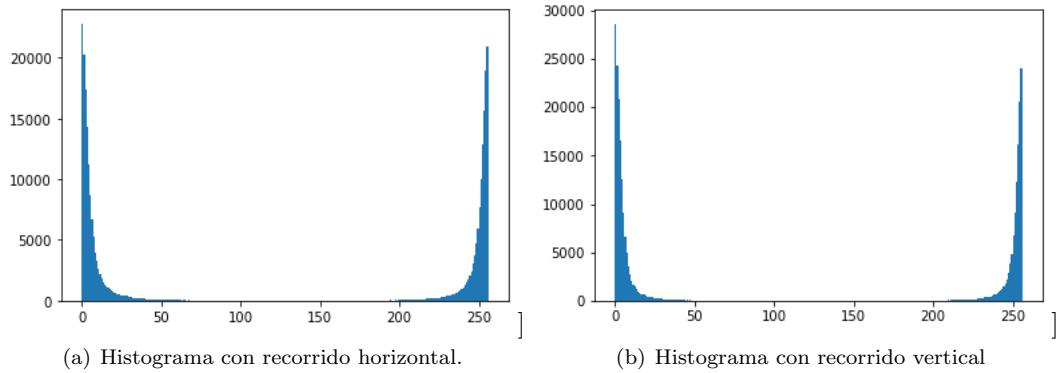


Figura 17: Histograma de ambos recorridos

Se observa que al cambiar el recorrido aparecen diferentes contornos en distintas orientaciones. Para finalizar se realizó la suma de las imágenes del contorno obteniendo, esto provoca que la imagen se vuelva más clara en algunos lugares oscuro haciendo notar más el contorno.



Figura 18: Imagen resultante.

## 9. Recorrido de Hilbert

El recorrido de Hilbert consiste en ir variando la fila o la columna a continuar a través de unas curvas conocidas como curvas de Hilbert. En la figura (19) se muestran las diferentes curvas, en donde se generan las curvas a través de la primera imagen de la figura, la cual va variando la orientación y se va uniendo para formar otros tipos de curvas.

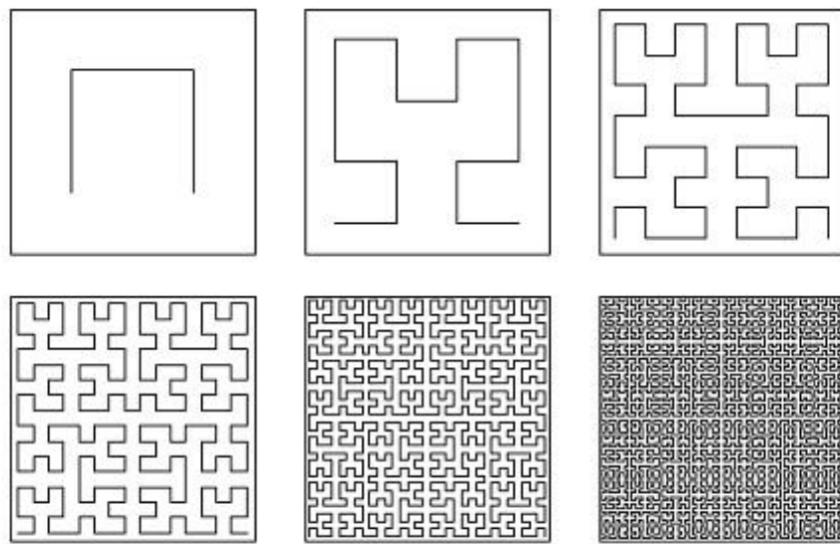


Figura 19: Curvas de Hilbert.

Entonces se tomo la imagen de lena y se realizo el recorrido. la figura (20) muestra la imagen patrón con el recorrido realizado sobre la misma imagen.

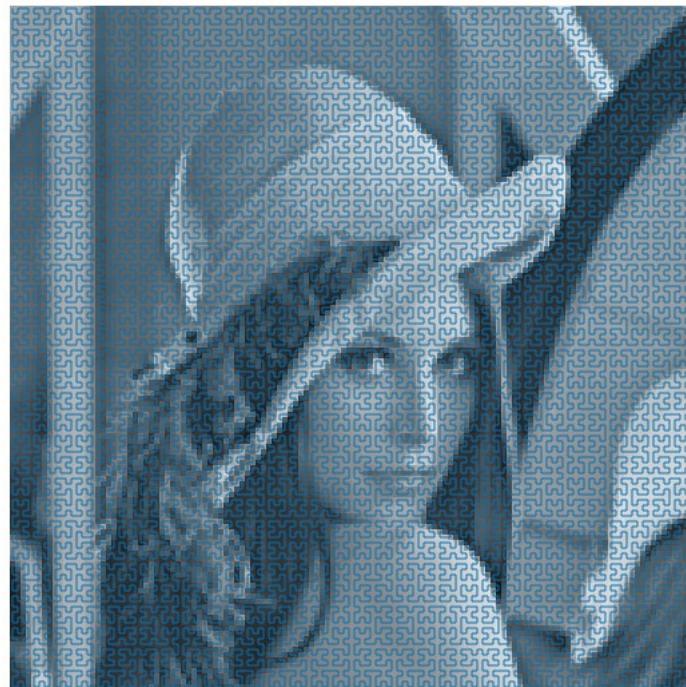


Figura 20: Recorrido de Hilbert.

## 10. Zoom

El *Zoom* de una imagen consiste en agarrar los datos de los pixel y duplicarlo en los píxeles de alrededor. La figura (21) muestra el proceso de agrandamiento de una imagen, se ve que se toma cada pixel y multiplicarlo

por una matriz de todos 1 para duplicar el pixel, luego se guarda en otra imagen de mayor tamaño o igual.

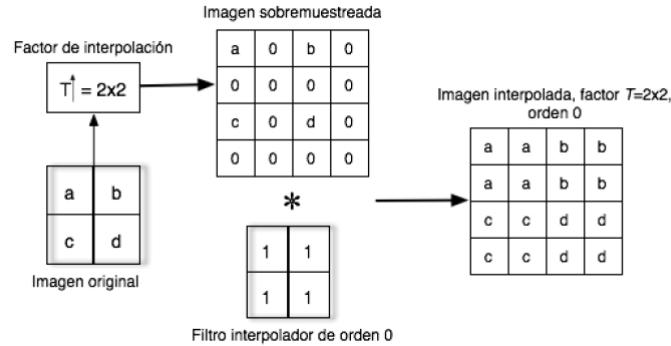


Figura 21: Recorrido de Hilbert.

Por lo tanto, se dividió la imagen en 4 partes iguales tomando un tamaño de 256x256, luego se agrando cada una de estas partes a una nueva imagen de 512x512 obteniendo



Figura 22: Zoom de la imagen original

## 11. Transformada Coseno

A partir de la imagen se decidió calcular la transformada coseno de la imagen. Luego, en la figura (23) se observa el histograma rojo correspondiente a la transformada coseno y el histograma verde correspondiente a la imagen original. Al realizar la transformada coseno se observa como la energía de la imagen se concentra en los primeros valores de los pixeles, mientras que en el histograma original los concentra sobre todos los pixeles.

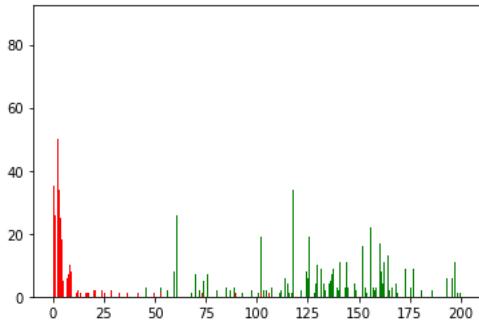


Figura 23: Transformada coseno .

Por lo tanto, se podría usar la transformada coseno para comprimir la imagen ya que me guarda la información en los primeros píxeles, permitiendo dejar de considerar los siguientes píxeles.

## 12. Ecualización del histograma

La ecualización del histograma de una imagen consiste en una transformación del histograma. Una imagen tendrá un cierto histograma que depende de los valores de los píxeles, en la ecualización del histograma se intenta hacer que el resultado sea un histograma uniforme con respecto al original. La figura (25) muestra el histograma de la figura original poco uniforme con el histograma después de la ecualización que intenta tender a una uniformidad. Por otro lado, la imagen (26) muestra la imagen resultante después de la ecualización, se puede observar un aumento en la resolución de la imagen original. Para realizar la ecualización, se uso una función de ecualización que transforma el valor de cada pixel al valor de la imagen resultante, el sistema completo se puede ver en la siguiente imagen

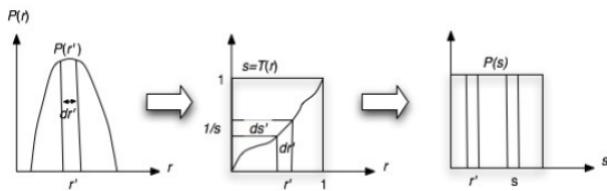


Figura 24: Sistema de ecualización del histograma.

donde se puede observar claramente que la función que transforma el histograma es la función  $T(f)$  logrando la uniformidad del mismo.

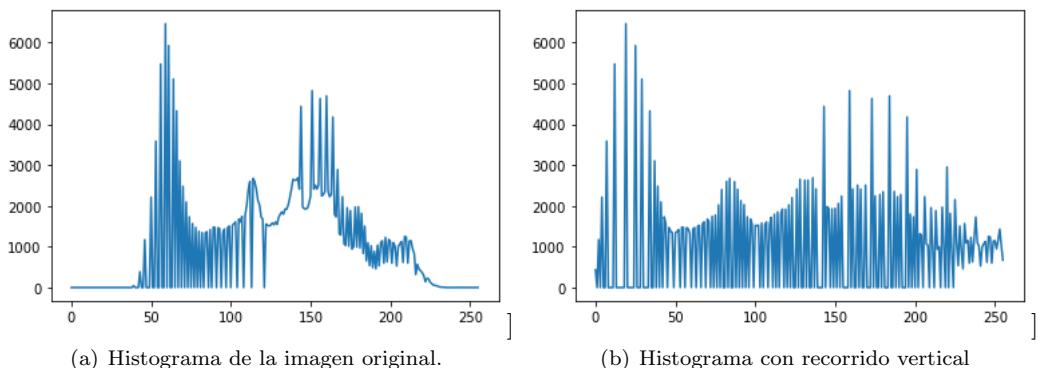


Figura 25: Histograma original y ecualizado



Figura 26: Imagen resultante histograma ecualizado.

### 13. Streching

El Streching es un método para la ecualización del histograma con la diferencia que se asigna los valores de las porciones mas que ocurren de los rangos del histograma y menos de visualización a la menor frecuencia de los valores de brillo. Este método de realce espectral reduce el contraste que se pueda presentar entre aquellas áreas muy claras o muy oscuras de una imagen con una distribución mas alargada de los extremos mínimos y máximos de un histograma, por lo que se tomara el valor máximo y mínimo de la imagen para realizar la operación de Streching, se calcula el valor de cada pixel de nuevo a traves de

$$p_i = 255 \times \frac{p_i - p_{min}}{p_{max} - p_{min}} \quad (6)$$

en donde  $p_{min}$  y  $p_{max}$  representa el valor de los pixeles máximo y mínimo de la imagen. Al realizar esto logramos aumentar la resolución como se observa a continuación



Figura 27: Stretching.

## 14. Compresión

Un problema en la codificación de datos es la compresión de los datos para ahorrar espacio en la computadora sin perder la información de los datos, para eso se usan diferentes herramientas para la codificación de los datos, en este trabajo realizaremos una compresión a través de un algoritmo no supervisado variando la cantidad de clases, y mencionaremos otro método de compresión conocido como *IFS*

### 14.1. K-means

El algoritmo de *k-means* es un algoritmo no supervisado que calcula las clases a partir de las media de cada una de ellas y busca minimizar la distancia entre los datos y las clases para determinar al clusters que pertenece. Tenes en cuenta que en una imagen, las clases serian los valores de los píxeles por ende cuando menos clases tengo mas diferencia con la imagen original tengo hasta un valor de 255 que asigna una clase a cada pixel. En la siguiente figura se observa eso, se tomaron una cantidad de clases  $K$  igual a 5, 10, 20, 40, 80, 100 y 255 para ver los cambios que ocurren al momento de aumentar la cantidad de clases, se puede observar que con un  $K = 5$  se pierde la resolución de la imagen y a medida que se aumenta la cantidad de las clases empieza a mejorar hasta un punto que no se puede notar una diferencia con la imagen original.



Figura 28: Compresión con el algoritmo k-means

Es importante destacar, que en una imagen puede ocurrir que no se usen los 255 pixeles por lo que al realizar la compresión de la misma no se observa ninguna diferencia. Por ejemplo, si uso una imagen que use solo 5 pixeles, entonces la compresión con una clase  $K = 5$  no habrá diferencia entre la imagen original porque cada pixel tendrá su clase determinado.

## 14.2. IFS

El método *IFS* consiste en una compresión fractal teniendo mejor resultados para texturas y imágenes naturales. Se usa sistemas de funciones iterativo que es un método para construir los fractales. La idea es ir generando los fractales y ir uniéndolos como se muestra en la figura (??), mostrando como se unen los cuadrados con diferentes orientaciones.

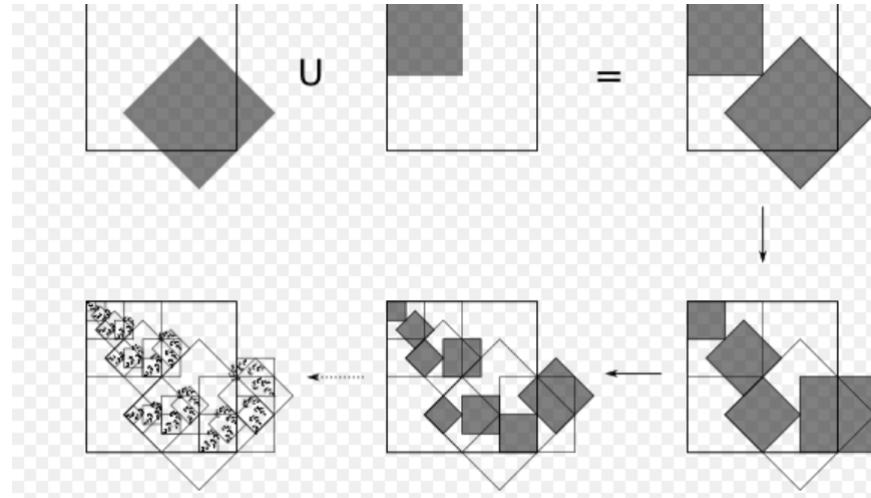


Figura 29: IFS.

Basicamente, la compresion de IFS consiste en un mapero en una dimension  $\mathbb{R}^N$  a  $\mathbb{R}^N$ . En imágenes, se puede tener imágenes binarias o imágenes ternarias obteniendo transformacion  $f_i : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  o  $f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ . Por lo que la idea del mapeo seria obtener un operador de la forma

$$H(A) = \cup_{i=1}^N f_i(A), \quad A \subset \mathbb{R} \quad (7)$$

observando que se van uniendo los fractal como se mostraba en la imagen, el valor de A permite la rotación del fractal. Generalmente, en escala de grises se usa un  $\mathbb{R}^3$  con un conjunto de fractal  $S = (x, y, u(x, y))$  siendo  $u(x, y)$  los valores en escala de grises, es la misma idea que antes tengre funciones  $f_1, \dots, f_N$  pero ahora en  $\mathbb{R}^3$ .