Universidad de Guanajuato

División de Ciencias e Ingenierías

Campus León

Fundamentos de Procesamiento Digital de Imágenes

Grupo:A

Tarea 6 - Histogram Matching y CLAHE
Unidad 1

Fecha de entrega: 29 de septiembre de 2022

Alumno:

Miguel Ángel Hernández Tapia

Profesor:

Dr. Arturo González Vega

Índice

Fundamentos de Procesamiento Digital de Imágenes
Tarea 6 - Histogram Matching y CLAHE

Miguel Ángel Hernández Tapia

29 de septiembre de 2022

Resumen

En este reporte se introducen los algoritmos de coincidencias de histogramas (Histogram Matching) y la ecualización adaptativa del histograma (AHE) con la rama de contraste limitado (CLAHE). La idea es compaginar el histograma de una imagen con uno definido aparte y con este nuevo histograma recrear la imagen que cumpla con la distribución de intensidades.

1. Introducción

La Coincidencia de histogramas o Histogram Matching, es una transformación de una imagen obteniendo su histograma propio y se ajusta a la forma de un histograma externo. La idea es cambiar las intensidades de una imagen haciéndola coincidir con las de un histograma definido aparte. Esto podría servir para llevar las tonalidades de una imagen a unos rangos de color específicos o reducirlos.

Supongamos una distribución de intensidades de una imagen $p_r(r)$ que puede ser calculado como el histograma normalizado

$$p_r(r_j) = n_j/n$$

donde n_j es la frecuencia de la intensidad r_j y n el total de píxeles de la imagen. Sea $S(r)$ el histograma acumulado como

$$S(r_k) = \sum_{j=0}^{k} p_r(r_j)$$

donde j va desde 0 hasta $k = 255$ (las intensidades en la escala de grises). Sea $G(z)$ otro histograma acumulado

$$G(z_k) = \sum_{j=0}^{k} p_z(z_j)$$

Como la idea es mapear cada valor de r a z, entonces podemos igualar las dos distribuciones de probabilidades como

$$S(r_k) = G(z_k)$$

para obtener los valores z solo despejamos la función con su inverso, tal que

$$z = G^{-1}(S(r))$$

La ecualización podría saturar de brillo debido a que en el histograma no se específica la reducción de los valores máximos o mínimos de intensidad. La ecualización adaptativa del histograma (AHE) es una técnica de procesamiento de imágenes por computadora utilizada para mejorar el contraste en las imágenes. Se diferencia de la ecualización del histograma ordinario en el sentido de que el método adaptativo calcula varios histogramas, cada uno correspondiente a una sección distinta de la imagen, y los utiliza para redistribuir los valores de ligereza de la imagen. Por lo tanto, es adecuado para mejorar el contraste local y mejorar las definiciones de bordes en cada región de una imagen.

Una variante de la ecualización adaptativa del histograma llamada ecualización del histograma adaptativo limitado por contraste (CLAHE) evita esto al limitar la amplificación de algunas intensidades.[1] [2] [3]

2. Metodología

Este trabajo se desarrollo en Google Colab. En este ambiente se puede compilar el lenguaje Python con una basta cantidad de librerías ya incluidas sin necesidad de instalarlas. Se utilizaron las librerías cv2 (OpenCV), numpy y matplotlib.pyplot. La primera sirve para manejo de imágenes como objetos que tiene diversos métodos y funciones definidas. Numpy es una librería optimizada para operaciones de arreglos (vectores, matrices, etc.), muy útil para reducir y optimizar las operaciones con sus métodos. Por último, matplotlib.pyplot es una función para graficar, funciona muy similar al plot de Matlab.

Los pasos son:

1. Definir una función que se llame $Hisma$, que reciba una matriz con la imagen $Im1$(valores entre 0 y 255) y un vector $h2$ que tiene 256 elementos, el primer elemento se refiere al valor de intensidad 0 y el elemento 256 se refiere a la intensidad 255, suponga que el histograma está normalizado. Primero tiene que identificar los 0's del histograma y cambiarlos por $1/tamIm1$, donde $tamI1$ es el tamaño en pixeles de la imagen. La función deberá aplicar la transformación que lleve de $I1$ a $I2$, donde el histograma de $I2$ sea muy parecido a $h2$, la imagen $I2$ deberá tener valores entre 0 y 255. La función regresará la imagen $I2$ y el vector $invH2$, que es la transformación que lleva Ieq a $I2$.

2. Cargar los valores de $'HistLuna.mat'$ y guardarlos en $h2Luna$ y probar la función con la imagen $Im1$ del archivo $'Fig0310(a).tif'$.

3. Obtener los histogramas de $Im1$ y compararlos con $h2Luna$.

4. Comparar con la transformación de directa de Python $equalizeHist$ (nombrada Ipy) con la imagen $Im1$ y $h2Luna$.

5. Ahora calcular 3 modos de $createCLAHE$.

   a) El default de Matlab (ClipLimit = 2.55, tileGridSize=(8,8)).

   b) Con tileGridSize=(25,25)

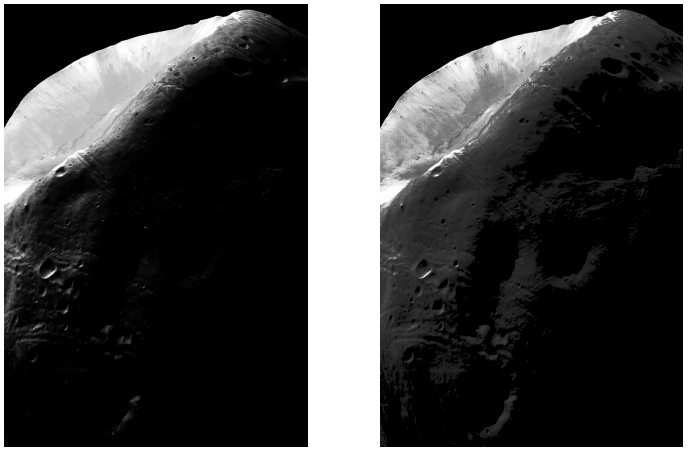
   c) Por último ClipLimit = 12.75 con tileGridSize=(25,25). Comparar los resultados.

3. Resultados

1.- El código de la función es

```
def Hisma(Im1, h2):
  # HISTOGRAMAS
  np.place(h2, h2==0, 1 / Im1.max()) # cambiando ceros a 1/pixeles
  h1 = np.histogram(img.flatten(), 256)[0]
  h1 = h1 / h1.max()
  H1 = h1.cumsum()
  H1 = H1 / H1.max()
  H2 = h2.cumsum()

  x = np.linspace(0.0, 255.0, 256)
  Ieq = H1[Im1]

  y = np.linspace(min(H2), 1.0, 256) # INVERSO DE H2
  inv_H2 = np.interp(y, H2, y)
  inv_H2 = inv_H2 / inv_H2.max()

  # CALCULO DE I2
  Ieq = ((Ieq - Ieq.min()) * (1/(Ieq.max() - Ieq.min()) * 255)).astype('uint8')
  I2 = inv_H2[Ieq]
  I2 = ((I2 - I2.min()) * (1/(I2.max() - I2.min()) * 255)).astype('uint8')

  return I2, inv_H2, H2
```

2.- La imagen $I2$ se muestra en la Figura 1.

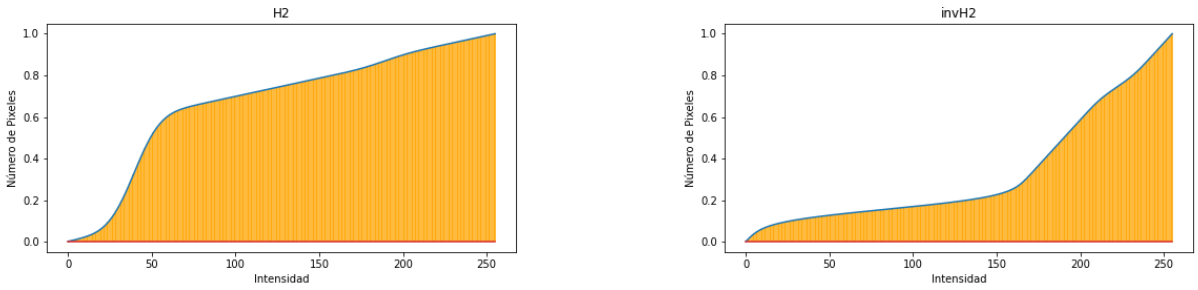


(a) Imagen Original.

(b) Imagen Transformada I2.

Figura 1: Transformación con la función Hisma (Comparación).

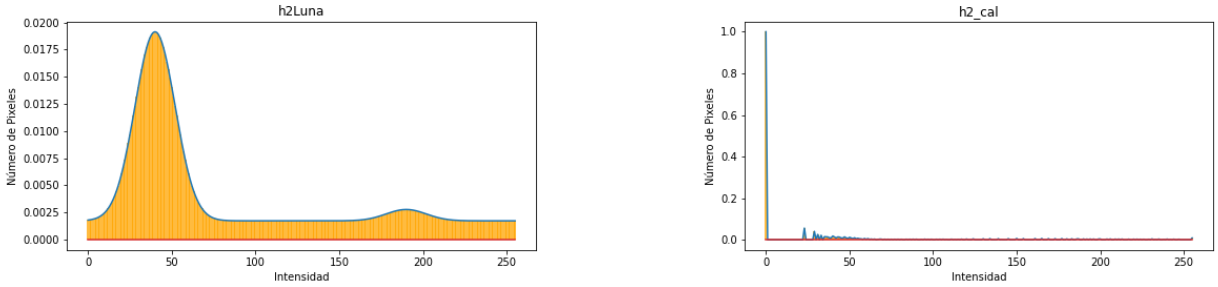
y el histograma acumulado y el inverso se muestran en la Figura 2.



(a) Histograma Acumulado de I2. (b) Histograma Acumulado Inverso de I2.

Figura 2: Histograma Acumulado e Inverso (Comparación).

3.- El histograma de $I2$ ($h2$) y el histograma aplicado a $I2$ ($h2Luna$). Estas se muestran en la Figura 3.



(a) Histograma de h2Luna. (b) Histograma de I2.

Figura 3: Histogramas de $I2$ y el aplicado en Hisma (Comparación).

El histograma Acumulado de $h2Luna$ y el calculado de $I2$ se muestran en la Figura 4.

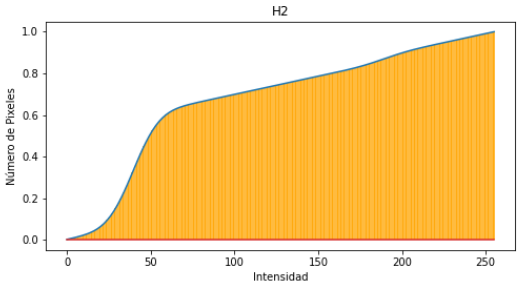
4.- La imagen resultante con la función directa de OpenCV se muestra en la Figura 5.

Los histogramas de la imagen transformada con OpenCV se muestran en la Figura 6.

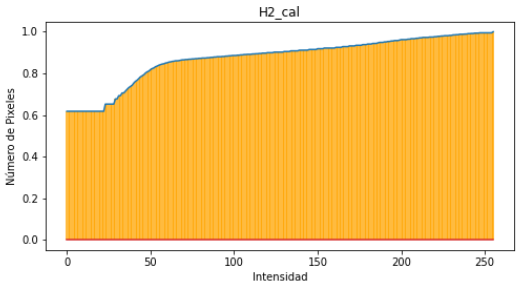
5.- Los resultados con las configuraciones mostradas en la metodología se muestra en la Figura 7.

4. Discusiones

1.- La función Hisma calcula los histogramas y los normaliza para poder trabajar. Se ecualiza la imagen $Im1$ con la que se compara para llegar a $I2$. El calculo es rápido y requiere de cambiar la

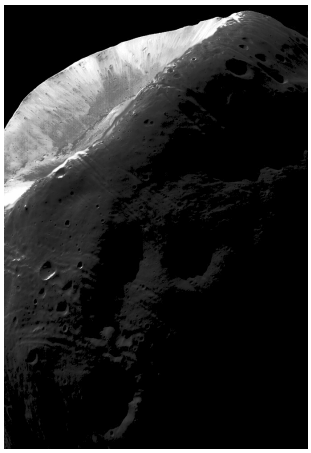


(a) Histograma Acumulado de h2Luna.

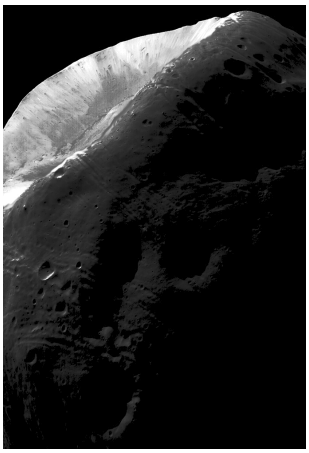


(b) Histograma Acumulado Calculado de I2.

Figura 4: Comparación de Histogramas Acumulados.



(a) Imagen Transformada I2.



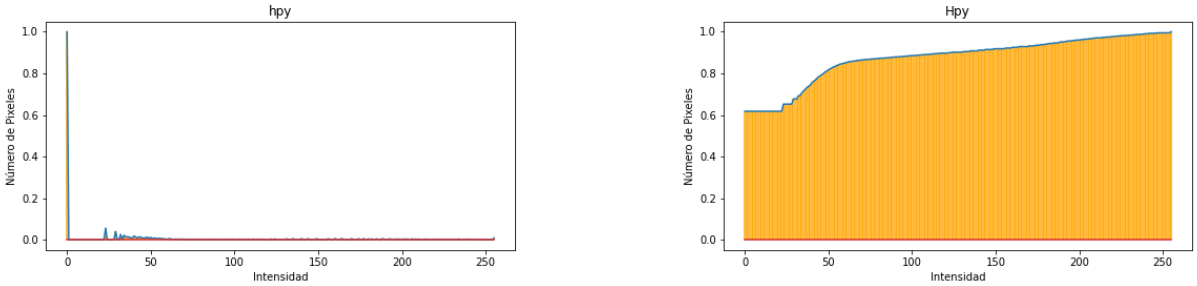
(b) Imagen Transformada con OpenCV Ipy.

Figura 5: Transformación Hisma vs OpenCV (Comparación).

escala de Ieq e $I2$ y cambiar a enteros para poder ser transformado e imprimir la imagen.

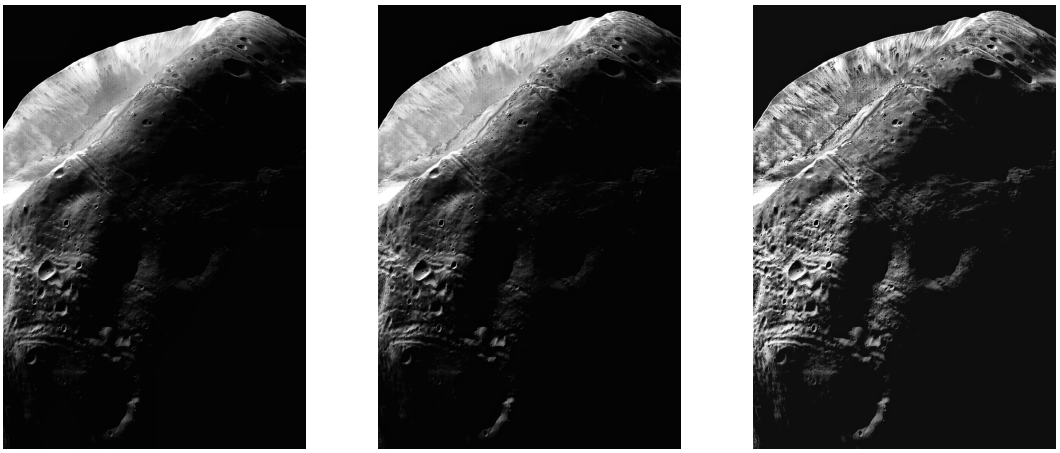
2.- El resultado de la imagen, es resaltar valores de grises oscuros para reducir los negros y también resaltar los grises claros pero reduciendo los blancos. Como se muestra en los histogramas, se distinguen estos comportamientos al crecer rápido en $H2$ después de los tonos más oscuros y ligeramente antes de los más claros, siendo uniforme en la parte central. Notamos también que las 2 gráficas son simétricas entre sí con el eje $y = x$, por tanto, son inversas entre sí.

3.- Al observar el histograma de $h2Luna$, el que aplicamos a la imagen, notamos un crecimiento en valores después de 0, un decrecimiento antes del centro, un comportamiento continuo, una leve subida antes de la tonalidad 200 y decrece antes del tono más claro. Se espera que el histograma acumulado de $I2$, tenga la forma del histograma aplicado $h2Luna$, y como observamos, los valores



(a) Histograma de Ipy. (b) Histograma Acumulado Ipy.

Figura 6: Comparación de Histogramas de Ipy.



(a) CLAHE default Matlab. (b) CLAHE size (25,25). (c) CLAHE ClipLimit 12.75.

Figura 7: Transformaciones CLAHE (Comparación).

de tonalidad 0 siguen siendo los predominantes pero el histograma tiene un aumento de las tonalidades cerca de la tonalidad 50 y luego se mantiene constante y se alcanza a notar un resalte de las tonalidades cerca de 200. Por lo que, se concluye que sí se realizó la transformación esperada.

4.-Con la ecualización de OpenCV y el vector inverso llegamos una imagen muy similar a $I2$. En la Figura 5 se comparan los resultados de los dos procesos y prácticamente no hay diferencias. Los histogramas mostrados en la Figura 6, también son similares a los mostrados anteriormente. Donde más se aprecia una diferencia, en en el histograma (b)6 con su análogo (b)4, donde en la segunda la curva se ve más suave y la primera con más relieves.

5.- Notamos que cada imagen es diferente. La primera no cambia mucho a la imagen original y resalta un poco los colores claros. El segundo es parecido al primero ero con algunos bordes más

distinguibles. La tercera es diferente a las demás, con menos tonalidades de 0 y de 255, es decir, con más grises, y con aún más bordes y su separación. Podemos concluir que cambia más rápido el resultado, cambiando el valor de *ClipLimit* comparado con el de *tileGridSize*. Para llegar a esta conclusión de forma más precisa, faltaría comprobar el cambio de *tileGridSize* a un rango pequeño y 5-10 con un *ClipLimit* grande como el usado en la tercera configuración o mayor, por ejemplo, entre 12-15.

5. Conclusión

Las transformaciones de la función *Hisma* y del directo de OpenCV dan prácticamente el mismo resultado. Donde se observan diferencias es en sus histogramas acumulados, la curva de la función *Hisma* es más suave y continua. Se comprobó que los histogramas son inversos entre sí, por lo cual, la operación fue correcta.

La forma de los histogramas pareciera no ser muy similar a *h2Luna* pero se notan los aumentos en las tonalidades antes de 50 y 200, así como la reducción de 0 y 255 y la ausencia de grises intermedios. Por lo que, es obtuvieron los resultados esperados con el algoritmo de Histogram Matching.

Para la última parte de la práctica, se comparan los resultados del algoritmo CLAHE. Los cuales tienen su mayor cambio cuando el valor de *ClipLimit* aumentó. No se concluye que sea lo que cause el mayor cambio, por lo cual, faltaría probar con un *ClipLimit* grande y un *tileGridSize* pequeño. Esto para haber tenido todas las permutaciones posibles entre estos dos parámetros. Se concluye también, que el algoritmo CLAHE es mejor para notar todas las componentes de una imagen, evita excesos de valores claros y excesos de valores oscuros. Esto permite apreciar mejor una imagen en general.

A. Código

El código diseñado para esta práctica se encuentra en un repositorio de Github para su visualización y con la opción de abrir en Google Colab.

Referencias

[1] Chung, B. W. (2017). *Getting Started with Processing and OpenCV*, pages 1–37. Apress, Berkeley, CA.

[2] Solem, J. (2012). *Programming Computer Vision with Python: Tools and algorithms for analyzing images*, pages 73–273. O'Reilly Media.

[3] Woods, G. . (2018). *Digital Image Processing*, pages 1–40. Pearson, New York, NY.