FACULTAD DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

Computación Gráfica

Ecualización de histograma

GRUPO 3:
Renzo Vicente Castro
Andy Ñaca Rodríguez
Eduardo Sánchez Hincho
Luis Villanueva Flores

Docente:
Vicente Machaca Arceda

AREQUIPA
2020

1. Introducción:

En el presente documento se hablará sobre la ecualización de histograma, también se explicarán algunos ejercicios en el lenguaje Python para dar a conocer su funcionamiento.

2. ¿Qué es la Ecualización de histograma?

La ecualización de histograma es una técnica de procesamiento de imágenes por computadora utilizada para mejorar el contraste de las imágenes. Esto se logra mediante la distribución efectiva de los valores de intensidad más frecuentes, es decir, estirando el rango de intensidad de la imagen. Este método generalmente aumenta el contraste global de las imágenes cuando sus datos utilizables están representados por valores de contraste cercanos. Esto permite que las áreas de menor contraste local obtengan un mayor contraste.[1]

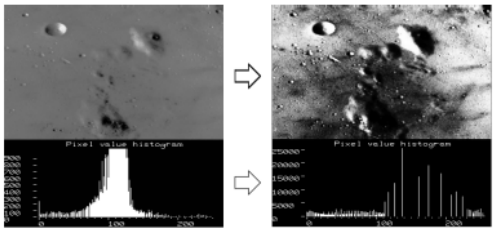


Figura 1: Ecualización de histograma.

3. ¿Cómo funciona la ecualización de histograma?

Dicho método se basa en aplicar la siguiente formula:[2]

$$g[x,y] = s_n; s_n = floor(L-1) * ([\sum_{i=0}^{n}]P_n) \quad (1)$$

Donde:
P_n=número de píxeles con intensidad n/número total de píxeles
L=Longitud de la intensidad del píxel
En teoría, la aplicación de esta operación debería transformar el histograma en otro con una forma perfectamente uniforme sobre todos los niveles de gris. Sin embargo, en la práctica esto no se va a poder conseguir pues se estaría trabajando con funciones de distribución discretas en lugar de continuas. En la transformación, todos los píxeles de un mismo nivel de gris se transformarán a otro nivel de gris, y el histograma se distribuirá en todo el rango disponible separando en lo posible las ocupaciones de cada nivel.[2]

4. Otros tipos de Ecualización de histograma:

4.1. Ecualización de histograma adaptativo

La ecualización de histograma adaptativo difiere de la ecualización de 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 usa para redistribuir los valores de claridad 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.[1]

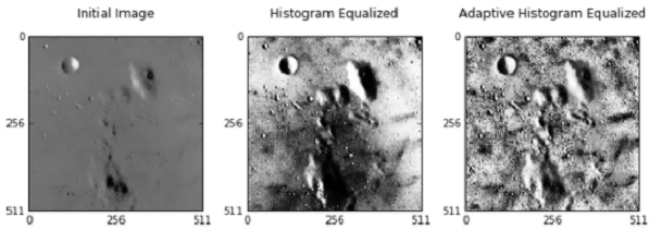


Figura 2: Comparación entre la ecualización de histograma ordinaria con la adaptativa.

4.2. Ecualización adaptativa limitada contrastante

Dicho método difiere de la ecualización de histograma adaptativo en su limitación de contraste.Se desarrolló para evitar la amplificación excesiva de ruido que puede provocar la ecualización de histograma adaptativo.[1]

Figura 3: Comparación entre la ecualización de histograma ordinaria y la adaptativa limitada contrastante.

5. Aplicaciones:

Algunas de las aplicaciones de la Ecualización de histograma son:

- Medicina, por ejemplo, en las radiografías digitales en las que los colores logrados son una paleta de blancos y negros, los diferentes tipos de colores le dan al médico una idea del tipo de densidad que está observando. Por lo tanto, es probable que las estructuras blancas indiquen hueso o agua y las estructuras negras representan aire. Cuando las patologías están presentes en una imagen, tratar de delimitar el área de la lesión u objeto de interés puede ser un desafío, porque las diferentes estructuras generalmente se superponen una sobre la otra. Por ejemplo, en el caso del cofre, el corazón, los pulmones y los vasos sanguíneos están tan juntos que el contraste es crítico para lograr un diagnóstico preciso.[3]
- Reconocimiento facial, el uso de este reconocimiento biométrico tendrá un gran uso en dispositivos actuales y futuros.[4]

6. Ejercicios:

- Implemente el algoritmo de ecualización de histograma y evalué sus resultados con estas imágenes.

Figura 4: Imagen de referencia.

Resolución:

Tenemos dos funciones, la primera nos devolverá una lista con los Sn que obtengamos de nuestro procedimiento, lo que hacemos ahí es recorrer nuestro histograma y a partir de la fórmula que tenemos calcular los nuevos valores e insertarlos en una lista que es lo que retornará nuestra función. Se muestra a continuación la primera función:

Listing 1: Función equali

```
1 import cv2
2 import numpy as np
3 from matplotlib import pyplot as plt
4 import math
5
6 fig,axs=plt.subplots(1,4)
```

```
7  def equali(name):
8      img1=cv2.imread(name, cv2.IMREAD_GRAYSCALE)
9      axs[0].imshow(cv2.cvtColor(img1,cv2.COLOR_BGR2RGB))
10     hist=cv2.calcHist([img1], [0], None, [256], [0, 256])
11     axs[1].plot(hist)
12     height, width=img1.shape
13     s=[]
14     L=256
15     p_n=0
16     for i in range(L):
17         p_n=p_n+ (hist[i]/(height*width))
18         s.append(math.floor((L-1)*(p_n)))
19     return s
```

Lo que hacemos en la segunda función es recorrer nuestra imagen y verificar que intensidad existe en cada pixel y reemplazarlo por los nuevos valores que calculamos en la función de arriba, de esa forma aplanamos el histograma. Se muestra la segunda función a continucación:

Listing 2: Función histog

```
1  def histog(name):
2      img1=cv2.imread(name, cv2.IMREAD_GRAYSCALE)
3      height, width=img1.shape
4      s=equali(name)
5      for i in range(height):
6          for j in range(width):
7              c=img1[i][j]
8              img1[i][j]=s[c]
9
10     axs[3].imshow(cv2.cvtColor(img1,cv2.COLOR_BGR2RGB))
11     hist=cv2.calcHist([img1], [0], None, [256], [0, 256])
12     axs[2].plot(hist)
```

Se muestra a continuación los resultados para las dos imágenes solicitadas y se puede observar también como es que cambia el histograma para ambas, viendo la imagen original con su histograma y la resultante también con su histograma:

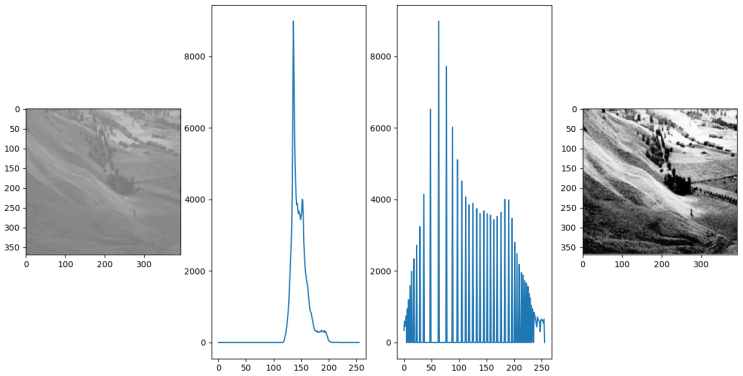


Figura 5: Resultados imagen 1.

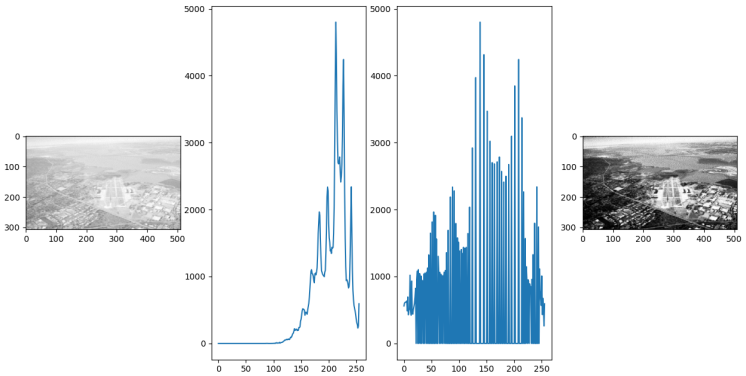


Figura 6: Resultados imagen 2.

- En la siguiente figura se muestra una iglesia antes y después de aplicar ecualización de histograma. Usted, debe obtener una subimagen así como se ve en la otra figura y luego procesar s_n (n es la intensidad de un píxel) con los datos de la sub imagen y obtener una imagen mejorada con la formula g[x, y] = s_f[x,y]

Figura 7: Imagen de referencia.

Resolución: En la función histogram generamos un vector con todos los valores nuevos hallados en base a la fórmula matemática de la ecualización de histograma. Posteriormente creamos la función equalization que realizará el reemplazo de los valores de cada píxel de la imagen este procedimiento se hará almacenando en una variable la intensidad del píxel actual que posteriormente servirá como índice de cada uno de los elementos del vector creado que reemplazarán a los actuales píxeles para mejorar el contraste de dicha imagen.

Listing 3: Funciones histogram y equalization

```
1   import cv2 as cv
2   import numpy as np
3   from matplotlib import pyplot as plt
4   import math
5   def histogram(inp,h):
6       s=0
7       acum=0
8       tamima=inp.size
9       tam=len(h)
10      L=256
11      V=[]
12      for i in range(tam):
13          acum=acum+(h[i]/tamima)
14          s=math.floor((L-1)*acum)
15          V.append(s)
16      return V
17  def equalization(inp,h):
18      nh=histogram(inp,h)
19      f,c=inp.shape
20      for i in range(f):
21          for j in range(c):
22              col=inp[i][j]
23              inp[i][j]=nh[col]
24      cv.imshow('res',inp)
```

Figura 8: Resultado.

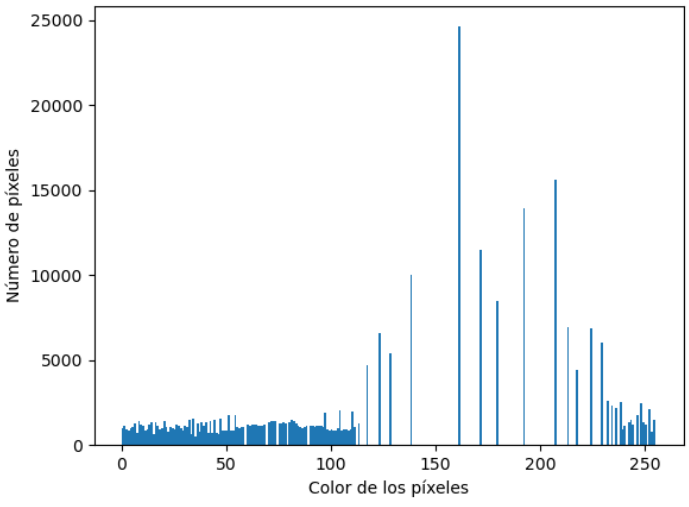


Figura 9: Histograma del resultado.

Figura 10: Imagen de referencia.

Resolución: Primero crearemos una sub imagen a partir de la imagen original, lo hacemos de la siguiente manera.

Listing 4: SubImagen

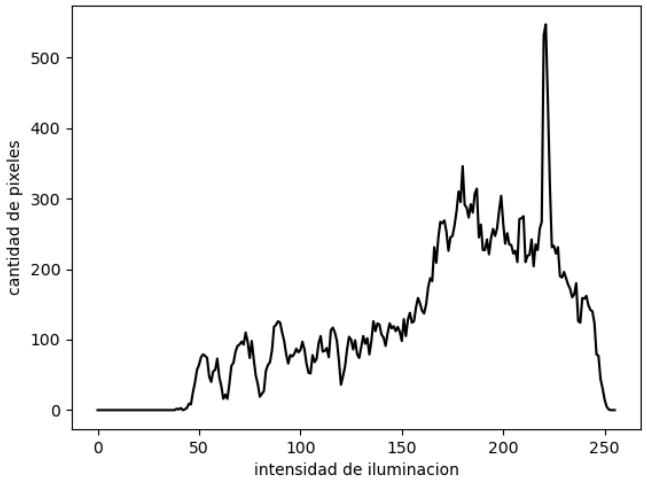
```
1  img = cv2.imread('hist10_1.jpg', cv2.IMREAD_GRAYSCALE)
2  height, width = img.shape
3  #Sacaremos una tercera parte de la imagen
4  sh = height//3
5  sw = width//3
6  simg = np.zeros((sh,sw),np.uint8)
7
8  for i in range(sh):
9      for j in range(sw):
10         simg[i][j] = img[i+sh][j+sw]
11 cv2.imshow('SubImagen',simg)
```

Obtendremos esta imagen

Sacamos su histograma y el resultado es el siguiente

Listing 5: Histograma SubImagen

```
histS = cv2.calcHist([simg], [0], None, [256], [0, 256])
```



Hallamos los promedios de pixeles en la SubImagen y los guardamos en una lista

Listing 6: Promedios

```
#Total de pixeles en la imagen
t=sh*sw
l=256
p=[]
for i in range(l):
    p.append(histS[i]/t)
```

Ahora añadimos los S aplicando la formula, y los guardamos en una lista.

Listing 7: Formula

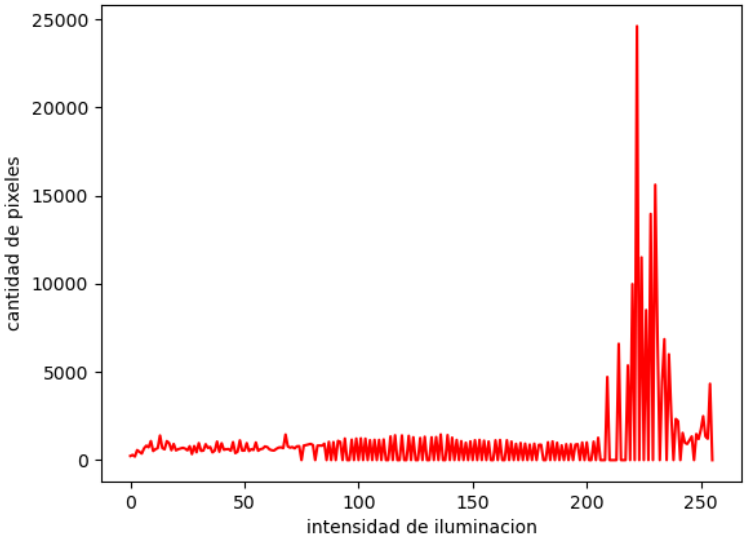
```
s=[]
for i in range(l):
    sp=0
    for j in range(i+1):
        sp=sp+p[j]
    s.append(math.floor((l-1)*sp))
```

Con los valores obtenidos, reemplazamos los pixeles de la imagen original para obtener el resultado final

Listing 8: Resultado

```
for i in range(height):
    for j in range(width):
        res[i][j]=s[img[i][j]]
```

El resultado de la Ecualización de histograma local es el siguiente



7. Conclusiones:

- La ecualización del histograma nos permite mejorar ampliamente el contraste de las imágenes.
- La ecualización del histograma es una técnica sencilla de procesamiento de imágenes que mejoran la calidad de imágenes especialmente las de blanco y negro.

8. Referencias:

1. Histogram Equalization. (2017). Retrieved 10 May 2020, from https://towardsdatascience.com/histogram-equalization-5d1013626e64largo https://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm
2. Ecualización del histograma. (2020). Retrieved 10 May 2020, from https://es.wikipedia.org/wiki/Ecualizac
3. (2020). Retrieved 10 May 2020, from https://www.nxp.com/docs/en/application-note/AN4318.pdf
4. Sistema de reconocimiento facial. (2020). Retrieved 10 May 2020, from https://es.wikipedia.org/wiki/Sistema$_d e_r econocimiento_f acial$