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

Computación Gráfica

CONTRAST STRETCHING

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 el Thresholding o método de valor de umbral, se explicarán algunos ejercicios en el leguaje Python para dar a conocer su funcionamiento.

2. ¿Qué es Contrast Stretching?

El estiramiento de contraste (Contrast Stretching en inglés) es una técnica simple de mejora de imagen que intenta mejorar el contraste en una imagen al 'estirar' el rango de valores de intensidad que contiene para abarcar un rango deseado de valores, por ejemplo , el rango completo de valores de píxeles que el tipo de imagen en cuestión lo permite.[1]

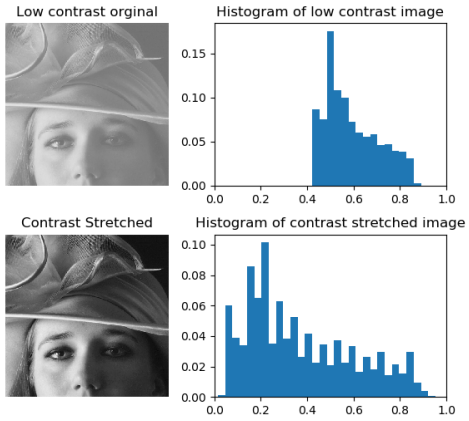

Figura 1: Contrast stretching.

3. ¿Cómo funciona Constrast Stretching?

Antes de poder realizar el estiramiento, es necesario especificar los límites de valor de píxel superior e inferior sobre los cuales se normalizará la imagen. A menudo, estos límites serán los valores mínimos y máximos de píxeles que permita el tipo de imagen en cuestión. Por ejemplo, para imágenes de graylevel de 8 bits, los límites inferior y superior pueden ser 0 y 255. Llame a los límites inferior y superior a y b respectivamente. El tipo más simple de normalización luego escanea la imagen para encontrar los valores de píxel más bajos y más altos actualmente presentes en la imagen. Llame a estos c y d . Luego, cada píxel P se escala utilizando la siguiente función:[1]

$$g[x, y] = (f[x, y] - c) * ((b - a)/(d - c)) + a \tag{1}$$

4. Contrast Stretching y outliers:

Un problema con este método es que los valores atípicos pueden reducir la efectividad de la operación. Se sugiere que a veces se podría utilizar un rango restringido de los valores de entrada según lo determinado al inspeccionar el histograma de la imagen original. Con frecuencia es ventajoso seleccionar c y d que estar en el 5to y 95avo percentiles, respectivamente, de los valores de entrada. Alternativamente, se puede comenzar en el pico del histograma y moverse hacia arriba y abajo de la lista de valores hasta que sólo un pequeño número de valores son rechazados y se dejó fuera de los límites elegidos para c y d .[2]

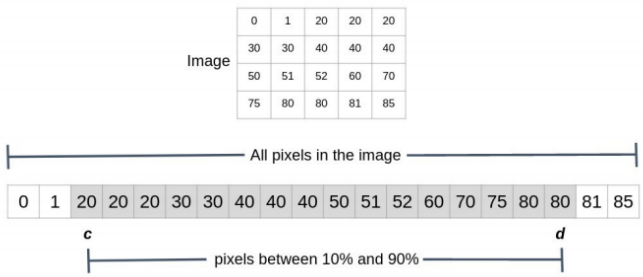

Figura 2: Explicación detallada del procedimiento con límites entre el 10 % y 90 % en una imagen de 20 pixeles.

Figura 3: Primera imagen: Imagen original,segunda imagen: Imagen con Contrast Stretching, tercera imagen: Imagen con Contrast Stretching (usando límites entre 5 % y 95 %) .

5. Aplicaciones:

Algunas de las aplicaciones del Contrast Stretching son:

- Medicina:Para la detección temprana de tumores en la sangre (Leucemia)[3] y mejorar las radiografias dentales [5].

- Mejorar la resolución de imágenes satelitales.[4]

6. Ejercicios:

- Implemente el algoritmo de Contrast Stretching y evalúe su código con la imagen de la izquierda. Como resultado debe obtener la imagen de la derecha.

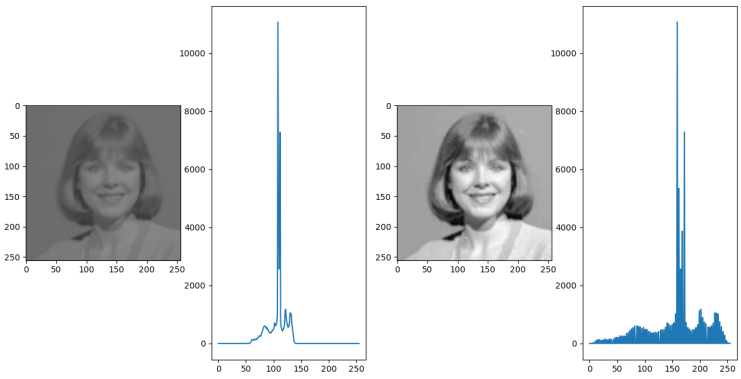

Figura 4: Resultado al aplicar Contrast Stretching a la imagen de la izquierda

Resolución:
Lo que hacemos primeramente es declarar como global nuestras variables a y b, ya que se mantendrán con los valores de 0 y 255 respectivamente, luego en nuestra función contrast, solo le pasamos el nombre de nuestra imagen. Posteriormente lo que hacemos es generar nuestro histograma de la imagen original, luego haciendo un doble for metemos los elementos de nuestra matriz, ya que sabemos que una imagen no es más que una matriz, entonces de esa forma la ordenamos y sacamos nuestro valor más pequeño y nuestro valor más grande para aplicar la fórmula vista.

Listing 1: Funcion Contrast

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import math
b=255
a=0
def contrast(name):
    fig,axs=plt.subplots(1,4)
    img1=cv2.imread(name, cv2.IMREAD_GRAYSCALE)
    axs[0].imshow(cv2.cvtColor(img1,cv2.COLOR_BGR2RGB))
    hist=cv2.calcHist([img1], [0], None, [256], [0, 256])
    axs[1].plot(hist)
    height, width=img1.shape
    l=[]
    for i in range(height):
        for j in range(width):
            l.append(img1[i][j])
    l.sort()
    for i in range(height):
        for j in range(width):
            img1[i][j]=(img1[i][j]-l[0])*((b-a)/(l[len(l)-1]-l
                [0]))+a

    hist1=cv2.calcHist([img1], [0], None, [256], [0, 256])
    axs[2].imshow(cv2.cvtColor(img1,cv2.COLOR_BGR2RGB))
    axs[3].plot(hist1)
```

- Agregue outliers a la imagen.

Figura 5: Imágen con outlier.

Resolución:
Creamos nuestra función “outlier“ la cual recibirá una imagen y el tamaño del outlier.Con dos los ciclos for anidados creara un cuadrado de color negro en la esquina superior izquierda.

Listing 2: Agregar outlier

```
        def outlier(img,size):
            for i in range(0,size):
                for j in range(0,size):
                    img[i,j]=0
            return img
```

- Agregue límites al algoritmo de Contrast Stretching para solucionar el problema del outlier.

Figura 6: Solucion

Resolución: Primero creamos una función para hallar nuestros limites(c y d)

Listing 3: Funcion Limit

```
def limit(hist, pixels, l):
    #Recibimos el histograma en una lista
    #La cantidad total de pixeles en la imagen
    #el porcentaje que aplicaremos al hallar los limites
    c=0
    d=0
    l=pixels*l/100
    #Aqui hallamos el porcentaje de pixeles que queremos
    #primero calculamos 'c'
    i=0
    while True:
        if(hist[i]>0): #Si existe el color i en la imagen, lo
            contamos
            c=c+hist[i] #Vamos acumulando todos los pixeles
            if(c>=l):   #Hasta que el valor sea igual al l% de
                 pixeles de la imagen
                c=i     #Guardamos el indice, y ese sera
                    nuestro c
                break
        i=i+1
    #La misma idea de lo anterior, pero ahora empezamos del
        final, de 255
    i=255
    while True:
        if(hist[i]>0):  #Comprobamos que el pixel existe en
            nuestra imagen
            d=d+hist[i] #Lo acumulamos
            if(d>=l):   #Y paramos cuando llega a nuestro
                limite
```

```
                d=i      #El indice donde se quedo, sera
                    nuestro d
                break
        i=i-1
    return c,d   #retornamos nuestros valores hallados
```

Ahora aplicamos nuestra funcion a nuestro gráfico de la siguiente manera

Listing 4: Funcion

```
c, d = limit(histOut, x*y,10)
a=0
b=255
for i in range(x):
    for j in range(y):
        r=(out[i][j]-c)*((b-a)/(d-c))+a
        #Agregamos la concicion de que si
        #el numero es menor a 0, que se quede en 0
        #esto para que no se salga del rango de la escala de
            grises
        if(r<0):
            res[i][j]=0
        #Agregamos la concicion de que si
        #el numero es mayor a 255, que se quede en 255
        elif(r>255):
            res[i][j]=255
        else:
            res[i][j]=r
```

El resultado Seria el siguiente

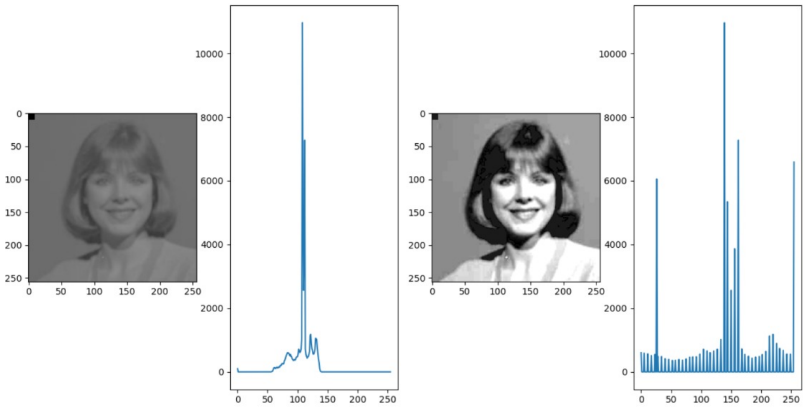

Figura 7: 10 % de limites

7. Conclusiones:

- Contrast Stretching mejora la resolución de una imagen y tiene una amplia gama de aplicaciones en la rama médico - biológica.
- Los outliers son perjudiciales para el Contrast Stretching ya que le impide funcionar correctamente.
- Contrast Stretching muestra sus resultados en el histograma estirándolo.

8. Referencias:

1. Point Operations - Contrast Stretching. (2020). Retrieved 6 May 2020, from https://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm
2. Contrast Stretching. (2020). Retrieved 6 May 2020, from https://academic.mu.edu/phys/matthysd/web226/Lab01.htm
3. Kaur, J., Choudhary, A. (2020). Comparison of Several Contrast Stretching Techniques on Acute Leukemia Images. Retrieved 6 May 2020, from https://www.semanticscholar.org/paper/Comparison-of-Several-Contrast-Stretching-on-Acute-Kaur-Choudhary/374299f41e87bb6199b0ae2fe61699865b2caa25extracted
4. Munteanu, C., Lazarescu, V. EVOLUTIONARY CONTRAST STRETCHING AND DETAIL ENHANCEMENT OF SATELLITE IMAGES. Presentation, Bucharest, Romania.
5. B Widodo, H., Soelaiman, A., Ramadhani, Y., Supriyanti, R. (2016). Calculating Contrast Stretching Variables in Order to Improve Dental Radiology Image Quality. Lecture.