Instituto Politécnico Nacional
Escuela Superior de Cómputo

Asignatura
Image Analysis

Alumno
Carapia González José Ricardo
García Medina Saúl
Guadarrama Hidalgo Luis Jorge

Título
Práctica 1 reporte. Ajuste de brillo

Entrega
Día 25/04/2021

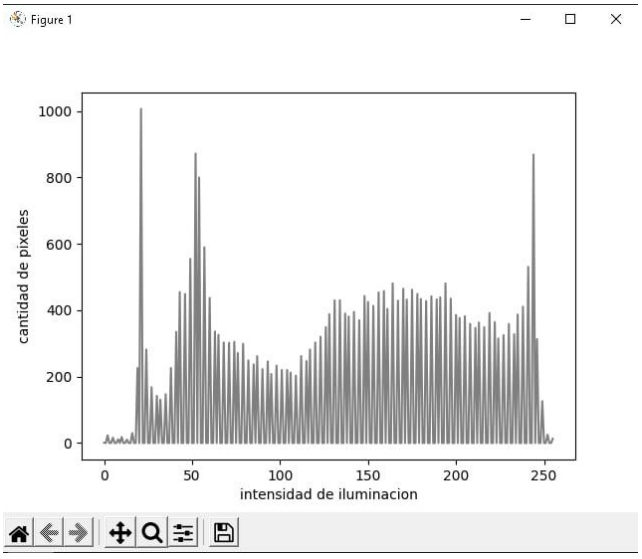
1. *Expansión del Histograma*

```
def expansion(self, ruta, nuevoMin, nuevoMax):
    img = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE)
    tamanioImg = img.shape #(ancho, alto)
    print(tamanioImg)
    pixeles = [] #lista
    for x in range(tamanioImg[0]): #(min, max)
        for y in range(tamanioImg[1]):
            pixeles.append(img.item(x, y))
    npPixeles = np.array(pixeles)
    infoImg = np.unique(npPixeles) #arreglo
    max = np.amax(infoImg)
    min = np.amin(infoImg)
    recta = Mates()
    for x in range(tamanioImg[0]):
        for y in range(tamanioImg[1]):
            img.itemset((x, y), recta.ecRecta(min, max, nuevoMin, nuevoMax, img.item(x, y)))
    cv2.imshow("Histograma", img)
    hist = cv2.calcHist([img], [0], None, [256], [0, 256])
    plt.plot(hist, color='gray' )

    plt.xlabel('intensidad de iluminacion')
    plt.ylabel('cantidad de pixeles')
    plt.show()
    cv2.destroyAllWindows()
```

```
def ecRecta(self, x1, x2, y1, y2, intensidadPixel):
    #Recordemos que una recta es y = mx + b
    m = (y2-y1)/(x2-x1)
    b = y1 - (m * x1)

    return m * intensidadPixel + b
```

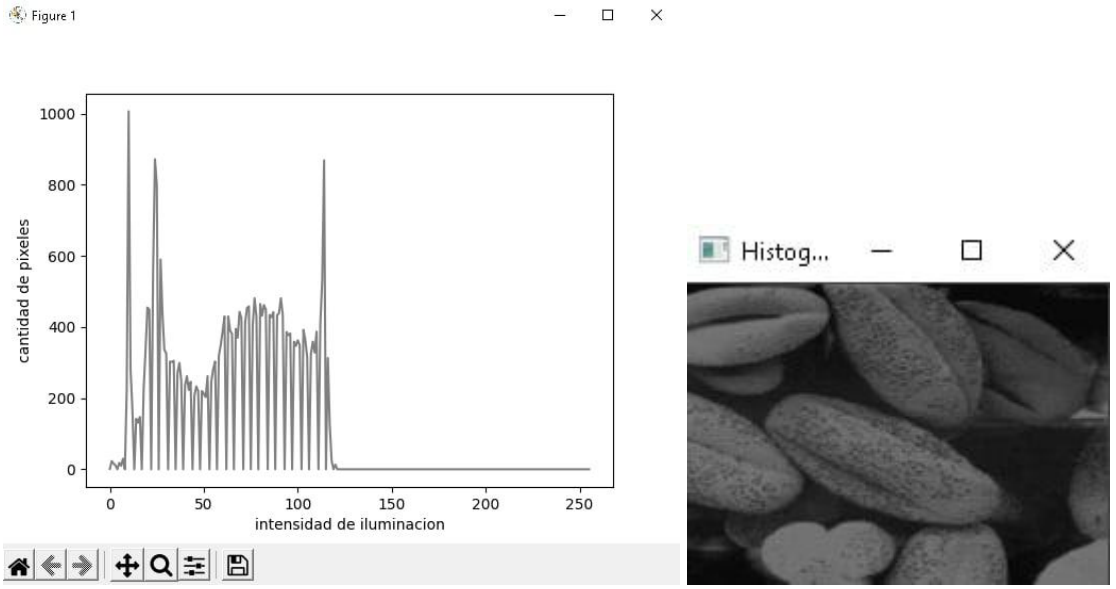


2. Contracción del HIstograma

```python
def contraccion(self, ruta, nuevoMin, nuevoMax):
    img = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE)
    tamanioImg = img.shape
    pixeles = []
    for x in range(tamanioImg[0]):
        for y in range(tamanioImg[1]):
            pixeles.append(img.item(x, y))
    npPixeles = np.array(pixeles)
    infoImg = np.unique(npPixeles)#arreglo
    max = np.amax(infoImg)
    min = np.amin(infoImg)
    contraccion = Mates()
    for x in range(tamanioImg[0]):
        for y in range(tamanioImg[1]):
            img.itemset((x, y), contraccion.contraer(nuevoMax, nuevoMin, max, min, img.item(x, y)))
    cv2.imshow("Histograma", img)
    hist = cv2.calcHist([img], [0], None, [256], [0, 256])
    plt.plot(hist, color='gray' )

    plt.xlabel('intensidad de iluminacion')
    plt.ylabel('cantidad de pixeles')
    plt.show()
    cv2.destroyAllWindows()
```

```python
def contraer(self, cmax, cmin, rmax, rmin, rk):
    return ((cmax - cmin)/ (rmax - rmin)) * (rk - rmin) + cmin
```

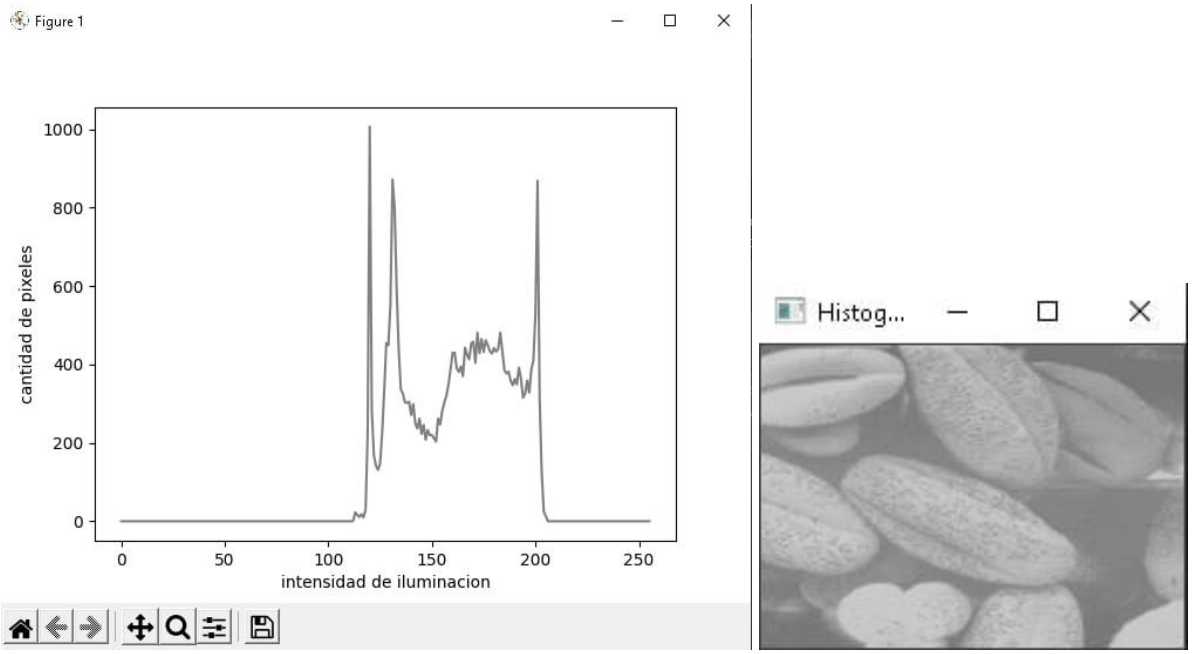


3. Desplazamiento del Histograma

```python
def desplazar(self, desplazamiento, pixel):
    return pixel + desplazamiento
```

```python
def desplazamiento(self, ruta, des):
    img = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE)
    tamanioImg = img.shape
    desplazamiento = Mates()
    for x in range(tamanioImg[0]):
        for y in range(tamanioImg[1]):
            img.itemset((x, y), desplazamiento.desplazar(des, img.item(x, y)))
    cv2.imshow("Histograma", img)
    hist = cv2.calcHist([img], [0], None, [256], [0, 256])
    plt.plot(hist, color='gray' )

    plt.xlabel('intensidad de iluminacion')
    plt.ylabel('cantidad de pixeles')
    plt.show()
    cv2.destroyAllWindows()
```



4. *Ecualización exponencial*

```python
def ecExp(self, ruta, alfa):
    img = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE)
    tamanioImg = img.shape
    totalElementos = img.size
    pixeles = []
    for x in range(tamanioImg[0]):
        for y in range(tamanioImg[1]):
            pixeles.append(img.item(x, y))
    npPixeles = np.array(pixeles)
    infoImg, indices, frecuencias = np.unique(npPixeles,return_inverse=True, return_counts=True)
    probabilidad = []
    min = np.amin(infoImg)
    for i in range(0, len(infoImg)):
        probabilidad.append(frecuencias[i]/totalElementos)
    ecu = Mates()
    pG = []
    for i in range(0, len(probabilidad) - 1):
        if(i == 0):
            pG.append(probabilidad[i])
        else:
            pG.append(probabilidad[i] + pG[i-1])
    cont = 0
```

```python
for x in range(tamanioImg[0]):
    for y in range(tamanioImg[1]):
        img.itemset((x, y), infoImg[indices[cont]])
        cont = cont + 1
cv2.imshow("Histograma", img)
hist = cv2.calcHist([img], [0], None, [256], [0, 256])
plt.plot(hist, color='gray' )
plt.xlabel('intensidad de iluminacion')
plt.ylabel('cantidad de pixeles')
plt.show()
cv2.destroyAllWindows()
```

```python
def ecualizacionExp(self, rmin, alfa, probabilidades, rj):
    return round(rmin - ((1/alfa) * log(1 - probabilidades)))
```

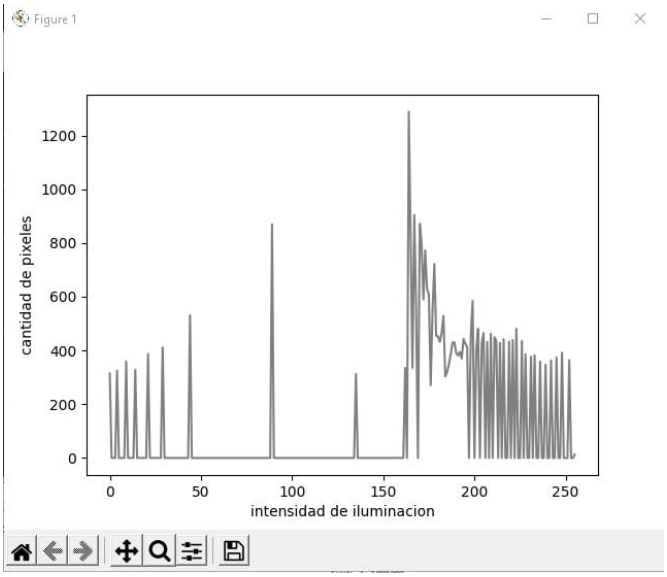
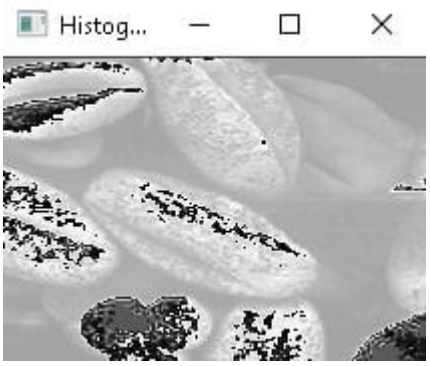


Figure 1
cantidad de pixeles
intensidad de iluminacion
0
50
100
150
200
250
0
200
400
600
800
1000
1200


Histog...