UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

COMPUTACION GRAFICA

Practica 10

Alumnos:
Chayña Batallanes Josnick
Perez Rodriguez Angelo Aldo
Pucho Zevallos Kelvin Paul
Vilcapaza Flores Luis Felipe
Sihuinta Perez Luis Armando

Julio 2021

Índice

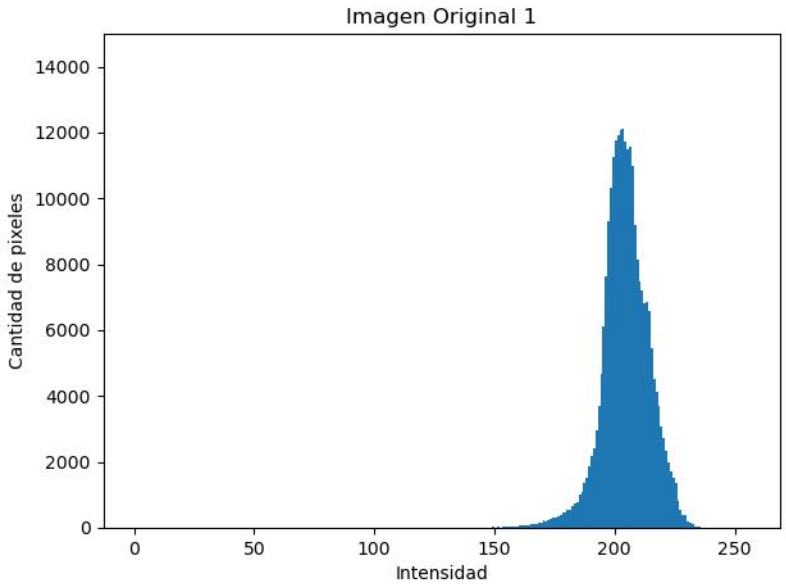
1. Algoritmo de computación gráfica Thresholding.

1.1. En los tejidos nervioso de algunos ratones, las células saludables tiene una intensidad mediana de gris, mientras que las células muertas son más densas y oscuras. Desarrolle un programa que quite las células muertas de los siguientes tejidos.

Las celulas muertas mas densas se representaran con color negro y lo demas seran de color blanco.

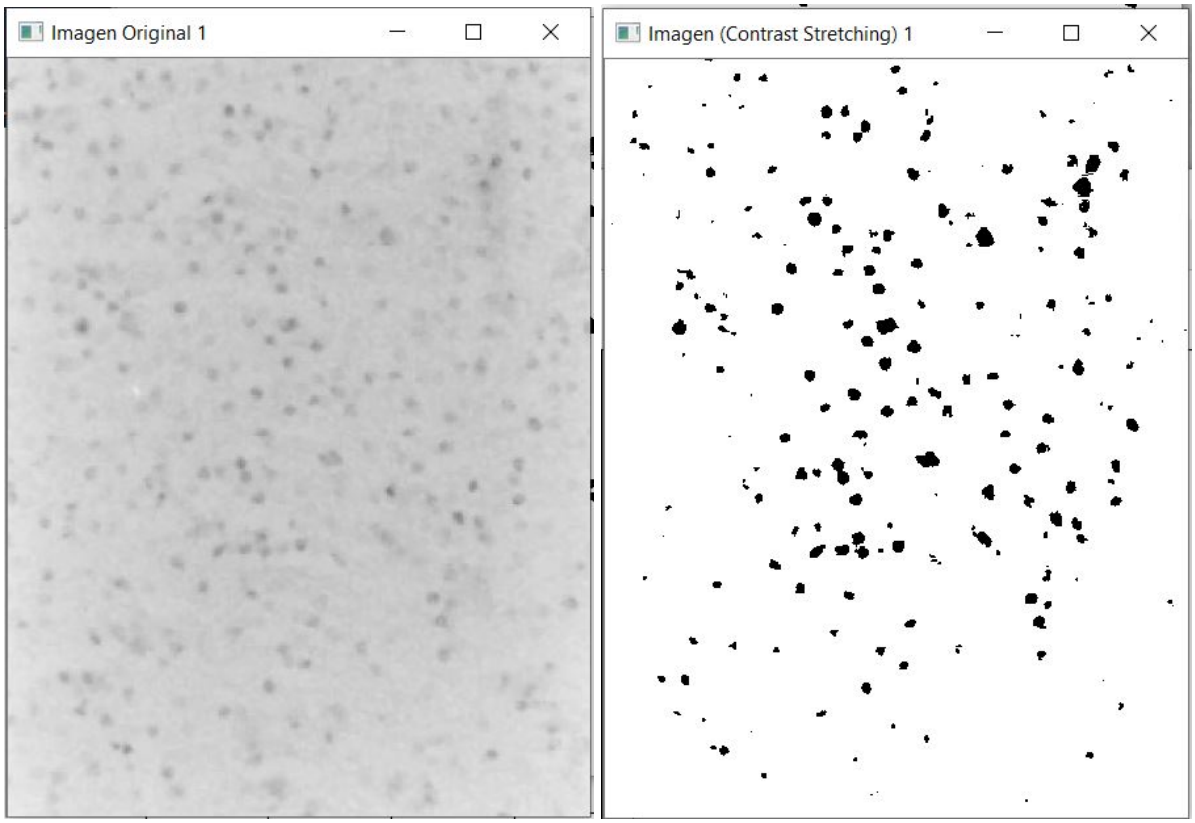
- **Thresholding del Tejido 1**

  - **Histograma**:



    Segun el histograma se uso un threshold de 184.

  - **Resultados**:

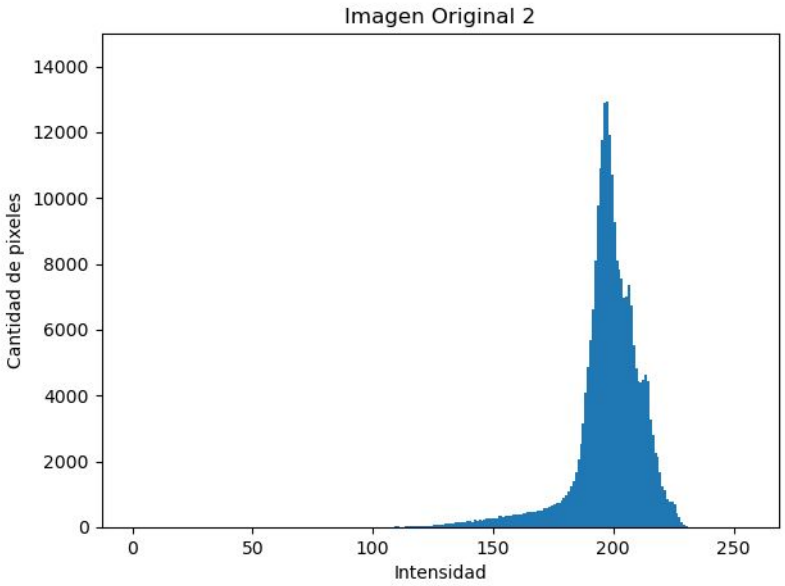


(a) Imagen Real (b) Imagen segmentada

Figura 1: Thresholding

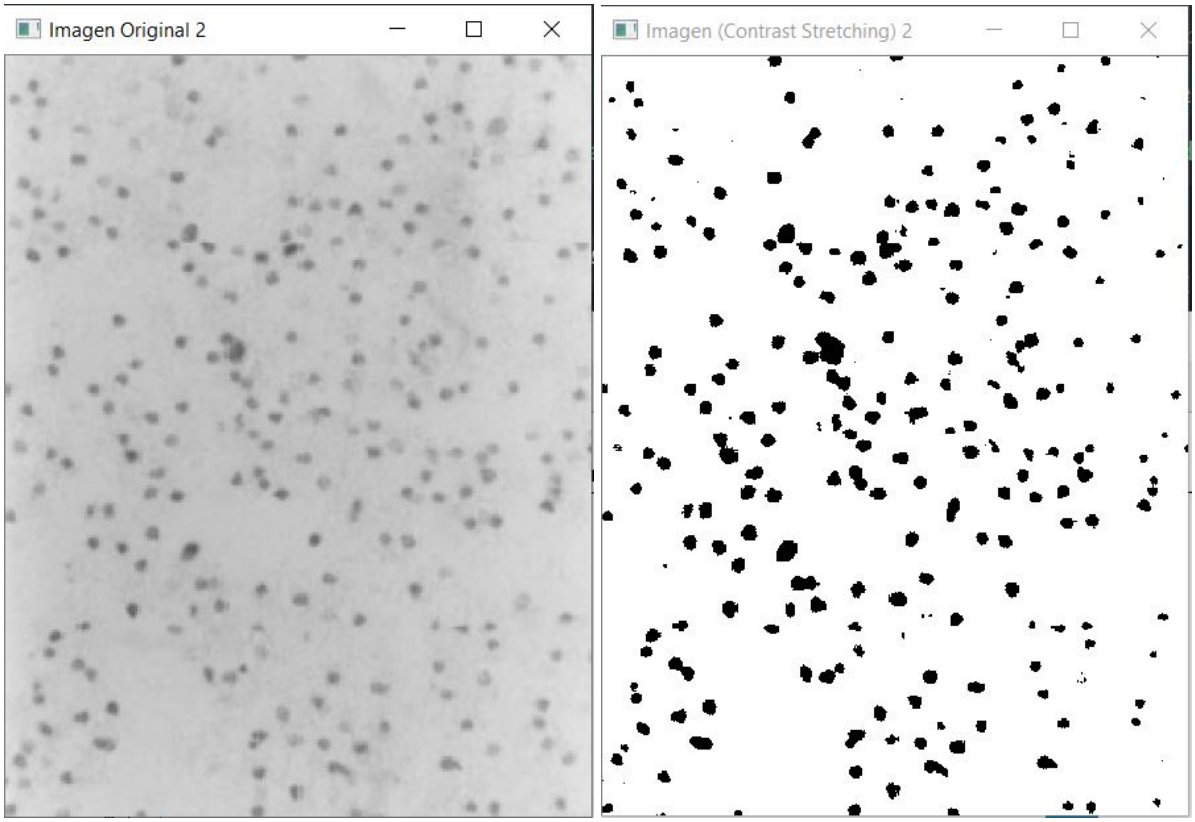
- **Thresholding del Tejido 2**

  - **Histograma**:



    Segun el histograma se uso un threshold de 175.

  - **Resultados**:



(a) Imagen Real (b) Imagen segmentada

Figura 2: Thresholding

- **Codigo para ambas imagenes**:

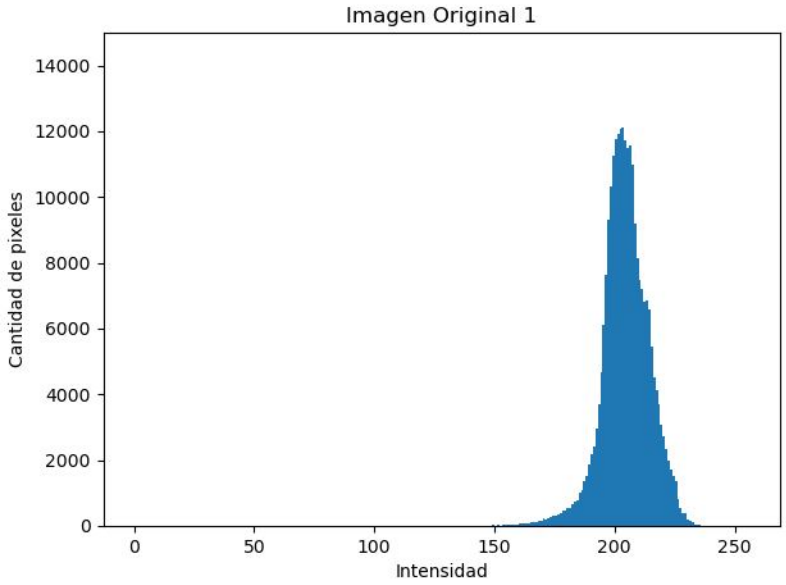
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

def getThresholding(img,threshold):
    modifiedImg = img
    print(img)
    for i in range(len(img)):
        for j in range(len(img[i])):
            if(img[i][j] < threshold):
                modifiedImg[i][j] = np.uint8(0)
            else:
                modifiedImg[i][j] = np.uint8(255)
    return modifiedImg

# read an image
listImg = ["tejido1.JPG","tejido2.JPG"]
threshold = [178,175]
for i in range(len(listImg)):
    imgReal = cv2.imread(listImg[i],cv2.IMREAD_GRAYSCALE)
    img = imgReal

    #tablas para los histogramas

    f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(15, 5))
    ax1.hist(img.ravel(),256,[0,256])
    title_ = "Imagen Original " + str(i+1)
    ax1.set_title(title_)
    ax1.set_xlabel('Intensidad')
    ax1.set_ylabel('Cantidad de pixeles')
    ax1.set_ylim(0,15000)
    # display the merged images
    cv2.imshow(title_, img)

    #thresholding

    img = getThresholding(img,threshold[i])
    print(img)
    ax2.hist(img.ravel(),256,[0,256])
    title_ = "Imagen (Contrast Stretching) " + str(i+1)
    ax2.set_title(title_)
    ax2.set_xlabel('Intensidad')
    ax2.set_ylabel('Cantidad de pixeles')
    cv2.imshow(title_,img)

plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

1.2. De la imágen anterior, implemente un programa que quite las células saludables.

Las celulas saludables se representaran con color negro y lo demas seran de color blanco.

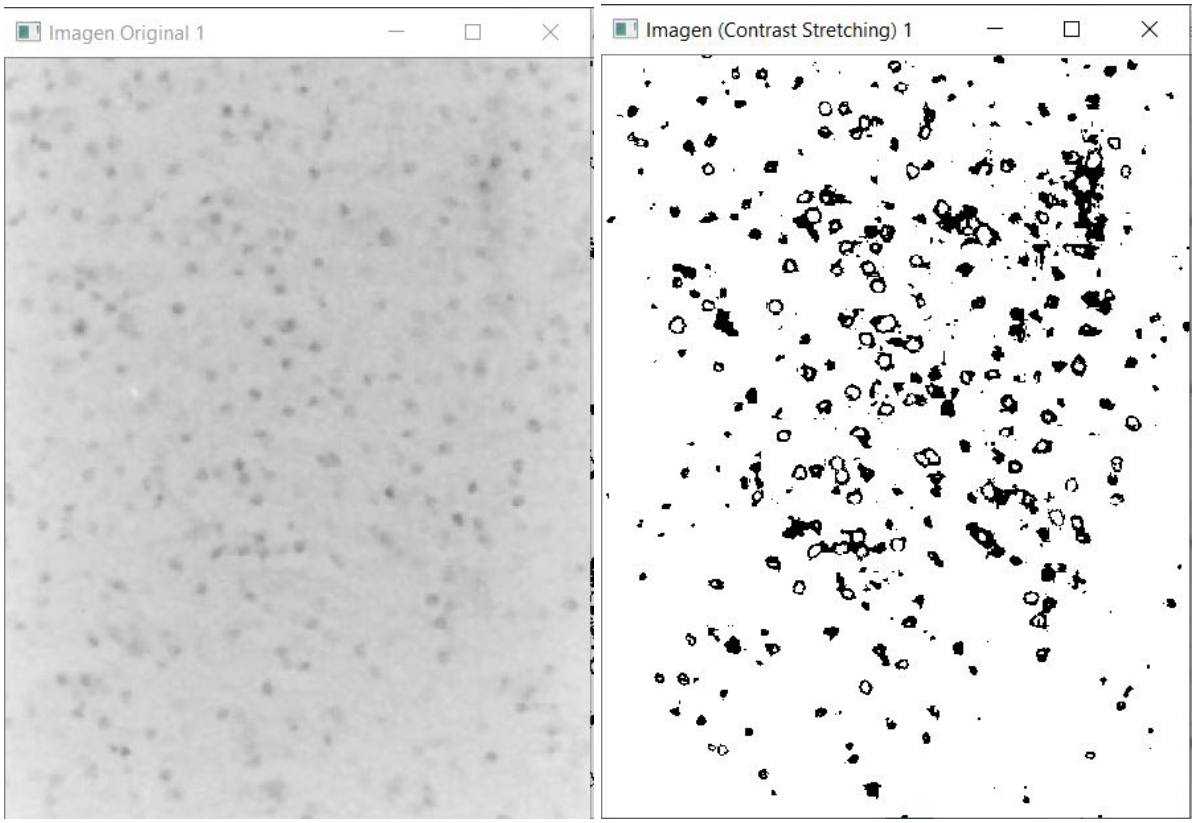
- **Thresholding del Tejido 1**

  - **Histograma**:



    Segun el histograma se uso un threshold de rangp de 178 a 193.

  - **Resultados**:

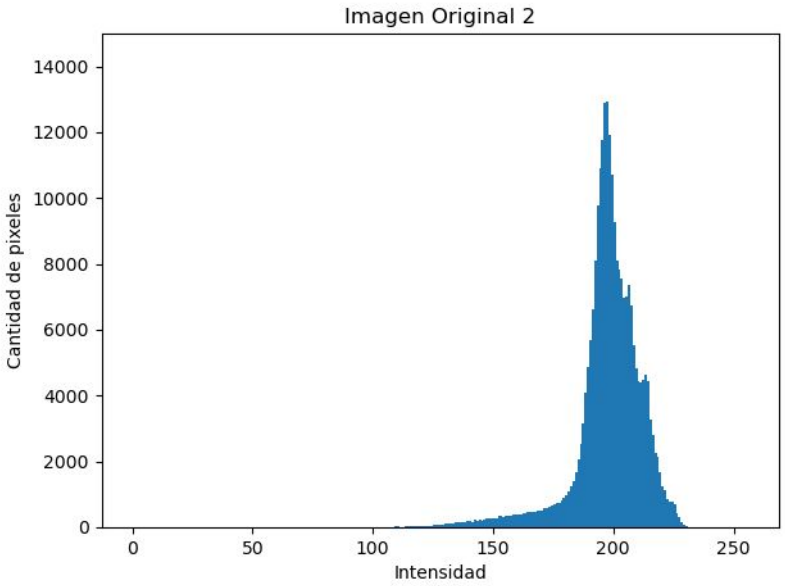


(a) Imagen Real (b) Imagen segmentada

Figura 3: Thresholding

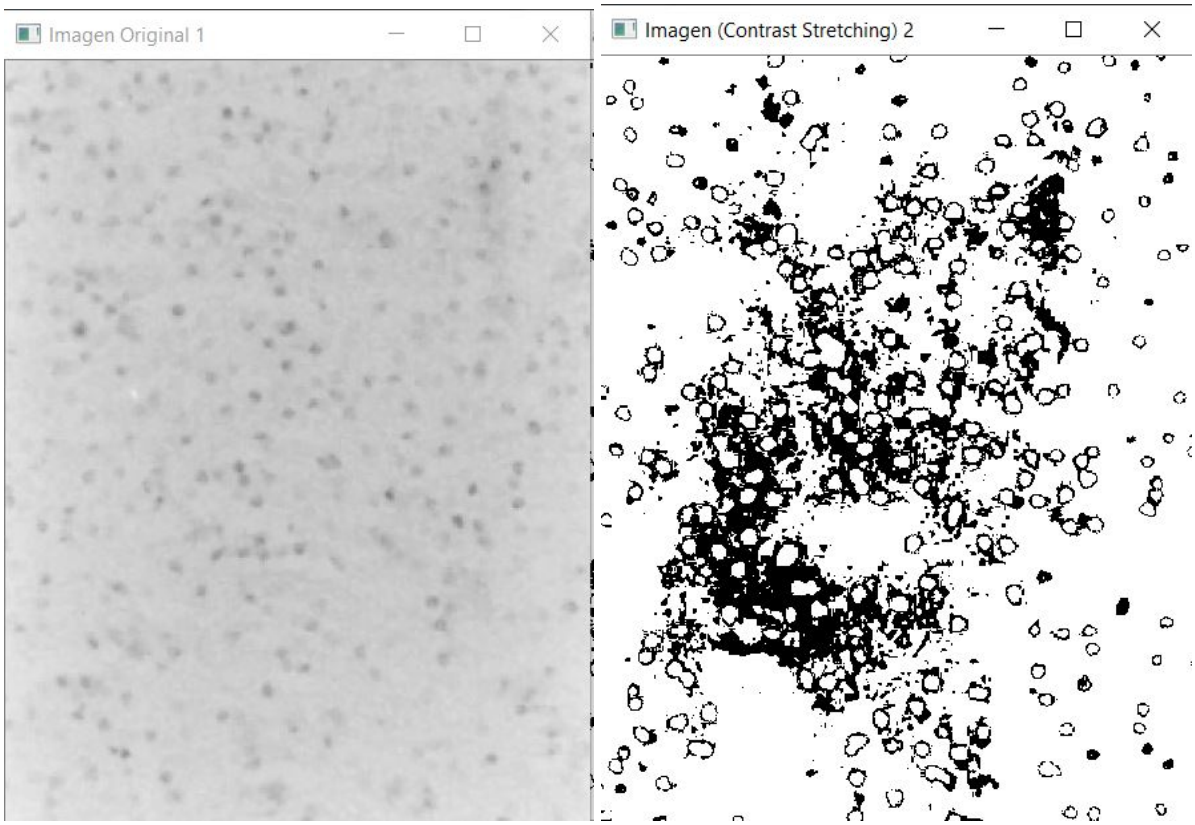
- **Thresholding del Tejido 2**

  - **Histograma**:



    Segun el histograma se uso un threshold de 175.

  - **Resultados**:



(a) Imagen Real (b) Imagen segmentada

Figura 4: Thresholding

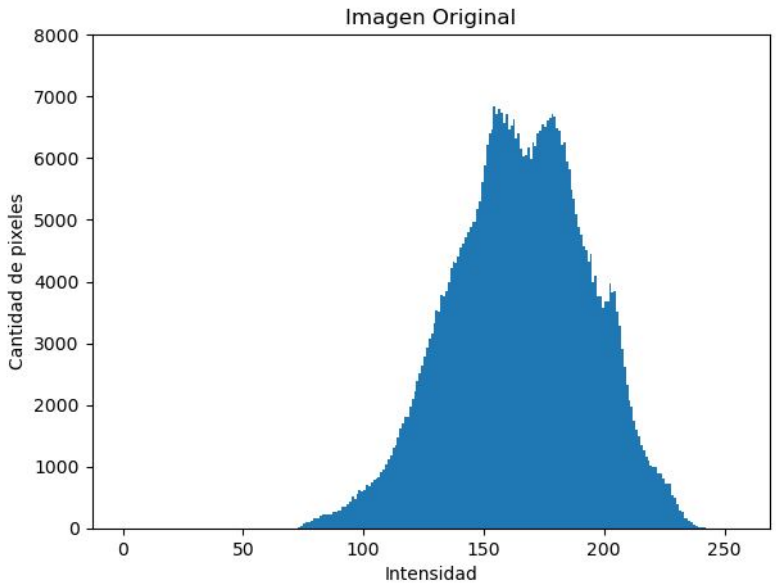
- **Codigo para ambas imagenes**:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

def getThresholding(img):
    modifiedImg = img
    print(img)
    for i in range(len(img)):
        for j in range(len(img[i])):
            if(img[i][j] > 175 and img[i][j] < 193):
                modifiedImg[i][j] = np.uint8(0)
            else:
                modifiedImg[i][j] = np.uint8(255)
    return modifiedImg

# read an image
listImg = ["tejido1.JPG","tejido2.JPG"]
# threshold = [182,182]
for i in range(len(listImg)):
    imgReal = cv2.imread(listImg[i],cv2.IMREAD_GRAYSCALE)

    img = imgReal

    #tablas para los histogramas

    f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(15, 5))
    ax1.hist(img.ravel(),256,[0,256])
    title_ = "Imagen Original " + str(i+1)
    ax1.set_title(title_)
    ax1.set_xlabel('Intensidad')
    ax1.set_ylabel('Cantidad de pixeles')
    ax1.set_ylim(0,15000)
    # display the merged images
    cv2.imshow(title_, img)


    #thresholding

    # img = getThresholding(img,threshold[i])
    img = getThresholding(img)
    print(img)
    ax2.hist(img.ravel(),256,[0,256])
    title_ = "Imagen (Contrast Stretching) " + str(i+1)
    ax1.set_title(title_)
    ax2.set_xlabel('Intensidad')
    ax2.set_ylabel('Cantidad de pixeles')
    cv2.imshow(title_,img)


plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

1.3. Desarrolle un programa que segmente las cosechas de trigo (campos amarillos) en la imagen satelital (ver Figura 2).

Para los campos amarillos se segmento de color negro y lo demas de color blanco

- **Histograma**
  Segun el histograma se uso un threshold de rangp de 180 a 210.



- **Codigo**

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

def getThresholding(img):
    modifiedImg = img
    print(img)
    for i in range(len(img)):
        for j in range(len(img[i])):
            if(img[i][j] >= 180 and img[i][j] <= 210):#185
                modifiedImg[i][j] = np.uint8(0)
            else:
                modifiedImg[i][j] = np.uint8(255)
    return modifiedImg

# read an image
imgColor = cv2.imread("campo.JPG",1) #8 bit por escala de grises
imgReal = cv2.imread("campo.JPG",cv2.IMREAD_GRAYSCALE) #8 bit por
    escala de grises

img = imgReal


#=================================================================
```

```
# tablas para los histogramas
#=============================================================================
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(15, 5))
ax1.hist(img.ravel(),256,[0,256])
ax1.set_title('Imagen Original')
ax1.set_xlabel('Intensidad')
ax1.set_ylabel('Cantidad de pixeles')
ax1.set_ylim(0,8000)
# display the merged images
cv2.imshow('Image Original', imgColor)

#thresholding
#de acuerdo al histograma

u_img = getThresholding(img)
print(u_img)
ax2.hist(u_img.ravel(),256,[0,256])
ax2.set_title('Imagen (Contrast Stretching)')
ax2.set_xlabel('Intensidad')
ax2.set_ylabel('Cantidad de pixeles')
cv2.imshow('Imagen (Constrast Stretching)',u_img)

plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Resultados**

Figura 5: Imagen Real

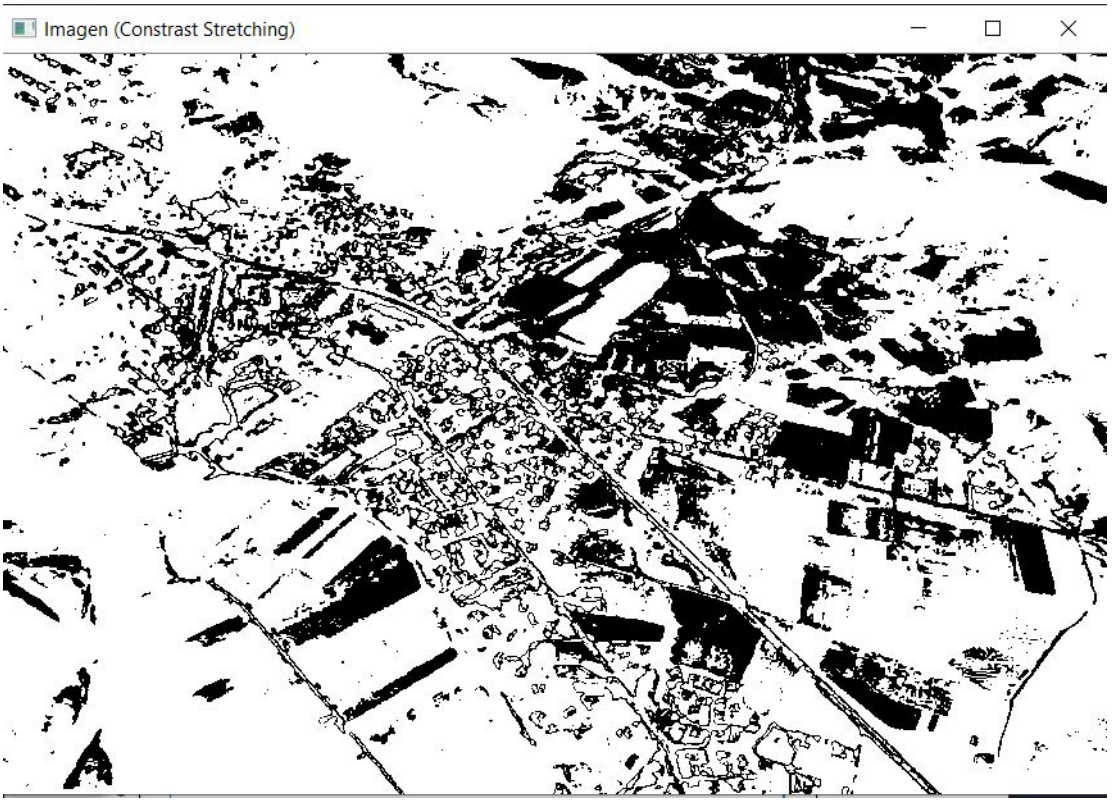


Figura 6: Imagen Segmentada

2. Algoritmo de computación gráfica Contrast Stretching.

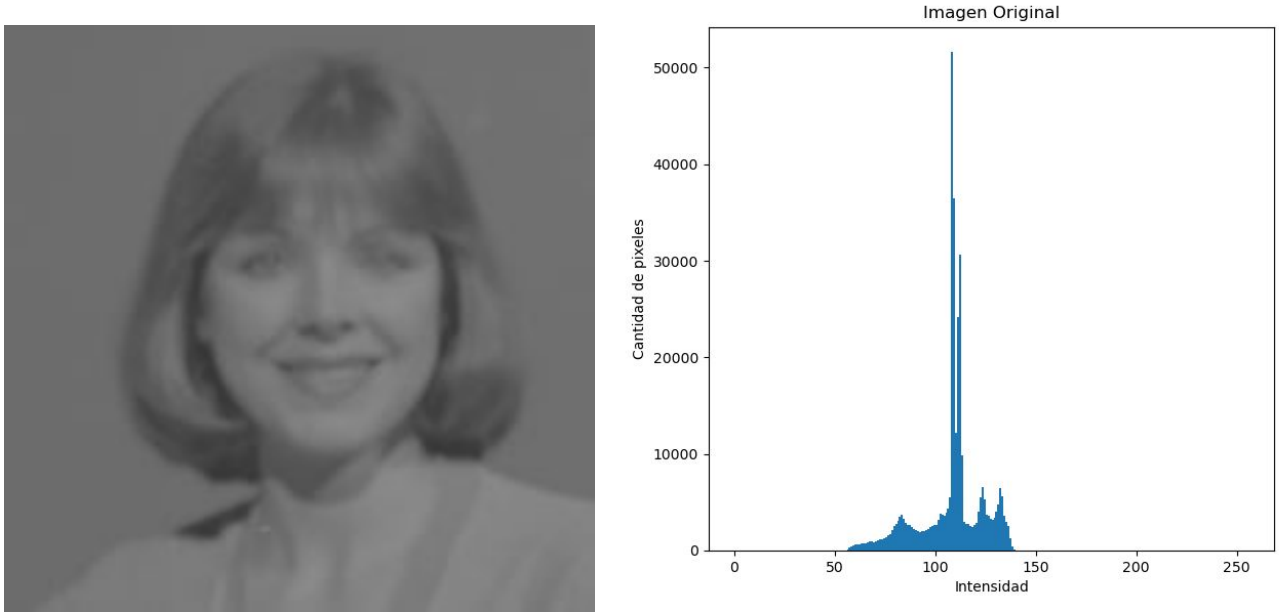
2.1. Implemente el algoritmo de Contrast Stretching y evalúe su código con la imagen de la izquierda. Como resultado debe obtener la imagen de la derecha.

- **Codigo:**

```
1
2  import cv2
3  import numpy as np
4  from matplotlib import pyplot as plt
5
6  def EquationContrastStretching(pixel,a,b,c,d):
7      g = ((pixel - c) * ((b-a)/(d-c)))+a
8      return g
9
10 def PixelOperations(img,c,d):
11     a = 0
12     b = 255
13     for row in range(len(img)):
14         for pixel in range(len(img[row])):
15             img[row][pixel] = EquationContrastStretching(img[row][
                   pixel],a,b,c,d)
16     img = np.array(img, dtype=np.uint8)
17     return img
```

```
18
19  imgReal = cv2.imread("../imagen5.JPG", cv2.IMREAD_GRAYSCALE) #8 bit
        por escala de grises
20  imgReal = cv2.resize(imgReal,(600,600))
21  img = imgReal
22
23  #tablas para los histogramas
24  f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(15, 5))
25  ax1.hist(img.ravel(),256,[0,256])
26  ax1.set_title('Imagen Original')
27  ax1.set_xlabel('Intensidad')
28  ax1.set_ylabel('Cantidad de pixeles')
29  cv2.imshow('Imagen real',img)
30
31  # por escala de grises en 8 bit entonces
32  arrayImg = img.ravel()
33  sortArray = np.sort(arrayImg)
34  print(sortArray)
35  # 0% y el 100 %
36  c = sortArray[0]
37  d = sortArray[int(len(sortArray))-1]
38  img = PixelOperations(img.astype(int),int(c),int(d))
39
40  ax2.hist(img.ravel(),256,[0,256])
41  ax2.set_title('Imagen (Contrast Stretching)')
42  ax2.set_xlabel('Intensidad')
43  ax2.set_ylabel('Cantidad de pixeles')
44  cv2.imshow('Imagen (Constrast Stretching)',img)
45
46  plt.show()
47  cv2.waitKey(0)
48  cv2.destroyAllWindows()
```

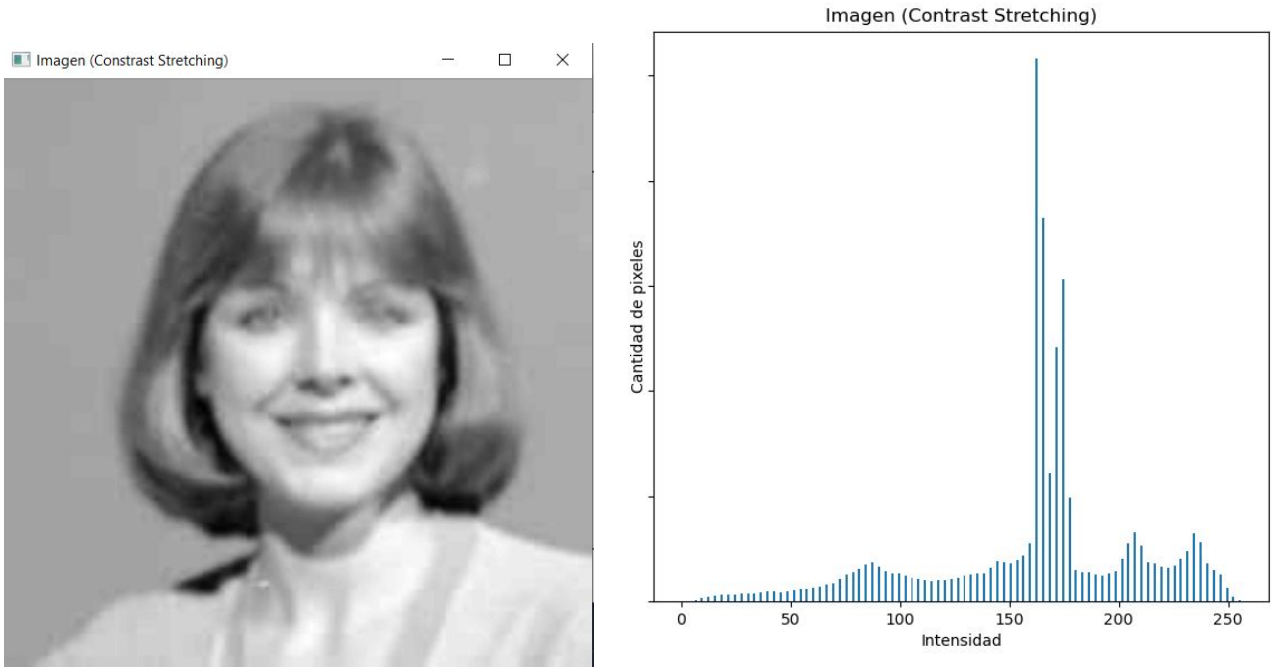
- **Resultados:**



(a) Imagen Real

(b) Histograma

Figura 7: Imgen Real e Histograma



(a) Imagen despues de aplicar contrast streching.

(b) Histograma

Figura 8: Resultados

2.2. Agregue outliers a la imagen.

- **Codigo corto**
  Para agregar outliers a la imagen

```
    for i in range(20):
        for j in range(20):
            img[i][j] = 0
```

- **Resultados:**

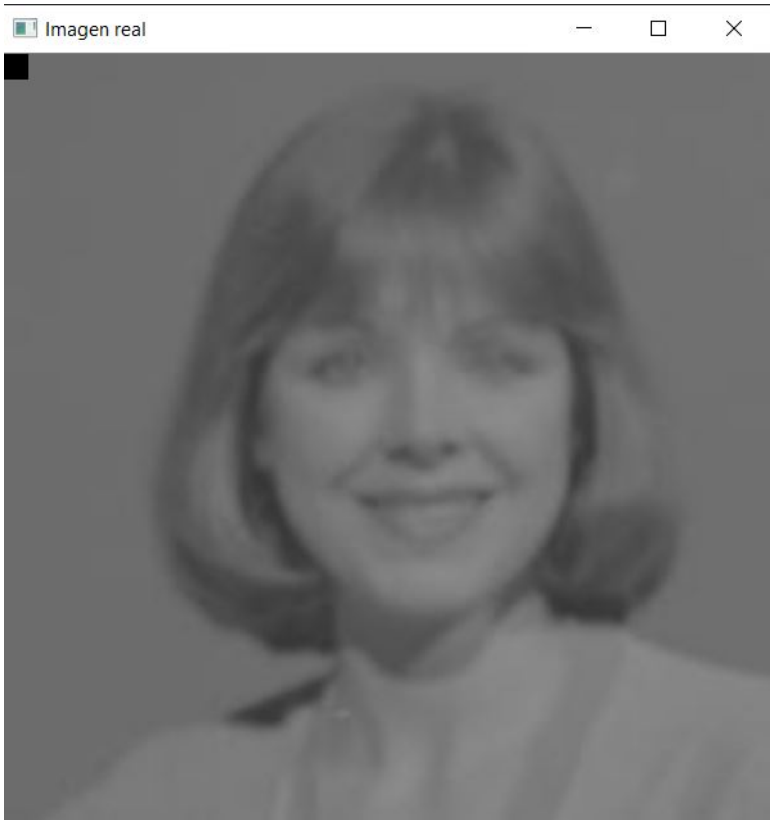


Figura 9: Imagen Real con outliers

2.3. Agregue limites al algoritmo de Contrast Stretching para solucionar el problema del outlier.

Para este ejercicio se establecio un limite del 10 % de los del total de pixeles que tendra la variable "c", y el 90 % que tendra la variable "d"

- **Codigo**

```

import cv2
import numpy as np
from matplotlib import pyplot as plt
```

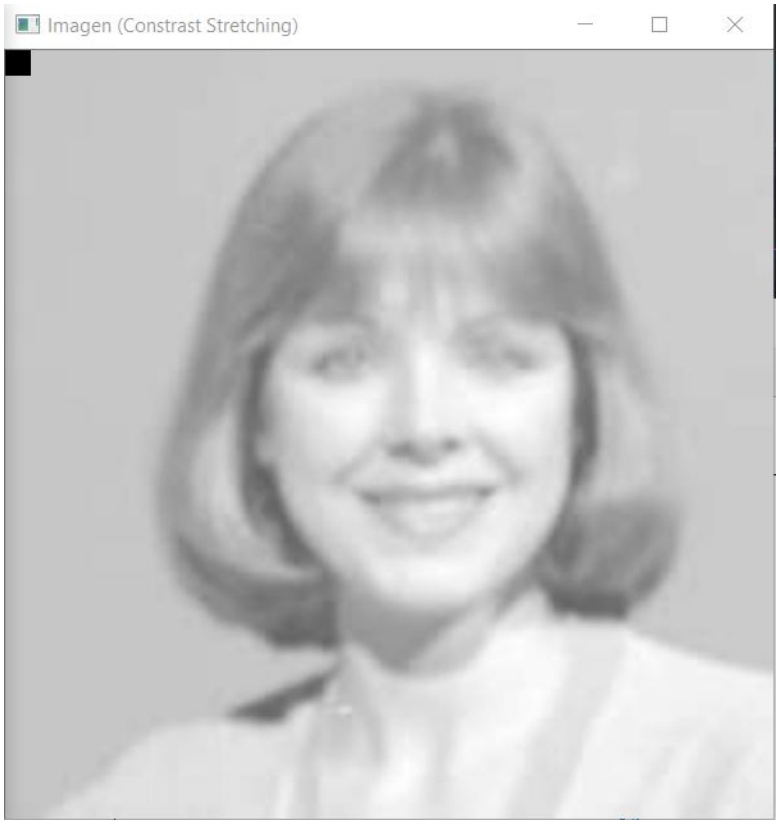
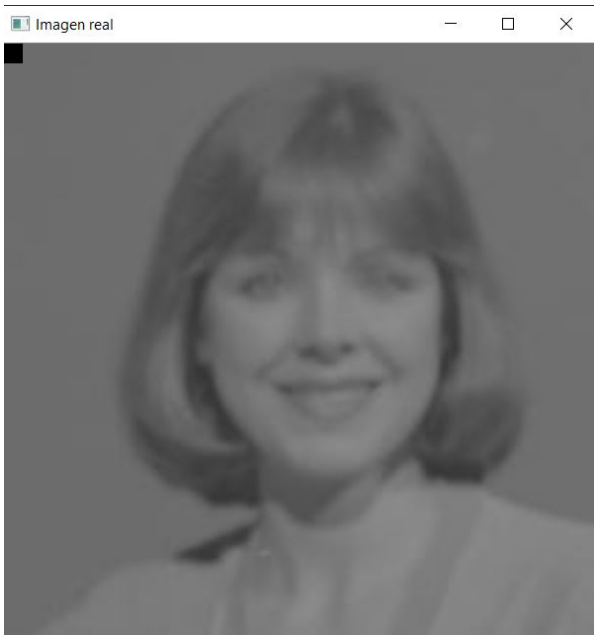


Figura 10: Imagen despues de aplicar contrast streching

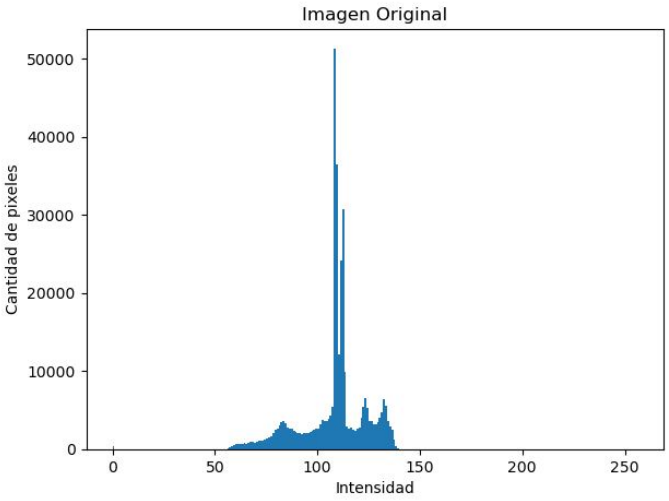
```
5
6  def PixelOperations(img,c,d):
7      a = 0
8      b = 255
9      for row in range(len(img)):
10         for pixel in range(len(img[row])):
11             if(0<=img[row][pixel]  and img[row][pixel]  <= c):
12                 img[row][pixel] = (a/c)*img[row][pixel]
13             elif(c<img[row][pixel] and img[row][pixel] <= d):
14                 img[row][pixel] = ((img[row][pixel] - c) * ((b-a)/(d-
                       c)))+a
15             else:
16                  img[row][pixel] = ((img[row][pixel] - d) * ((255-b)
                        /(255-d)))+b
17     img = np.array(img, dtype=np.uint8)
18     return img
19
20 imgReal = cv2.imread("dama.JPG", cv2.IMREAD_GRAYSCALE) #8 bit por
       escala de grises
21 imgReal = cv2.resize(imgReal,(600,600))
22 img = imgReal
23
24 #tablas para los histogramas
25 f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(15, 5))
26
27 for i in range(20):
28     for j in range(20):
```

```
            img[i][j] = 0
    ax1.hist(img.ravel(),256,[0,256])
    ax1.set_title('Imagen Original')
    ax1.set_xlabel('Intensidad')
    ax1.set_ylabel('Cantidad de pixeles')
    cv2.imshow('Imagen real',img)

    # por escala de grises en 8 bit entonces
    arrayImg = img.ravel()
    sortArray = np.sort(arrayImg)
    print(sortArray)

    # 10% y el 90 %
    min_ = int(len(sortArray) * 0.1)
    max_ = int(len(sortArray) * 0.9)
    c = sortArray[min_]
    d = sortArray[max_]

    img = PixelOperations(img.astype(int),int(c),int(d))

    ax2.hist(img.ravel(),256,[0,256])
    ax2.set_title('Imagen (Contrast Stretching)')
    ax2.set_xlabel('Intensidad')
    ax2.set_ylabel('Cantidad de pixeles')
    cv2.imshow('Imagen (Constrast Stretching)',img)

    plt.show()
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

- **Resultados:**

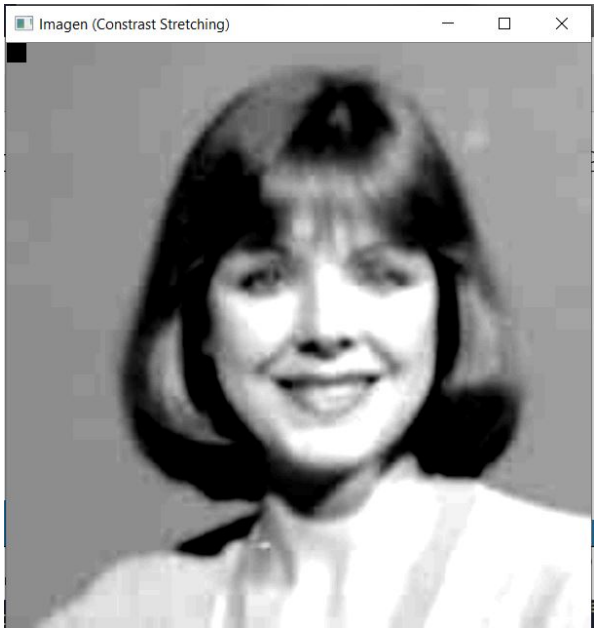


(a) Imagen con outliers

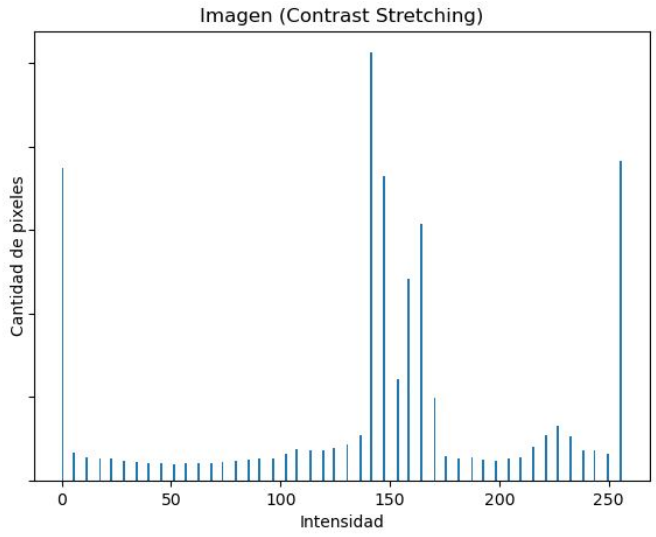


(b) Histograma

Figura 11: Imagen de entrada



(a) Imagen con outliers despues de aplicar Contrast stretching



(b) Histograma

Figura 12: Contrast stretching with outliers y limites.

3. Link de los códigos en github

https://github.com/kpzaolod6000/Graphics-Computing/tree/main/parcial_2/lab_10