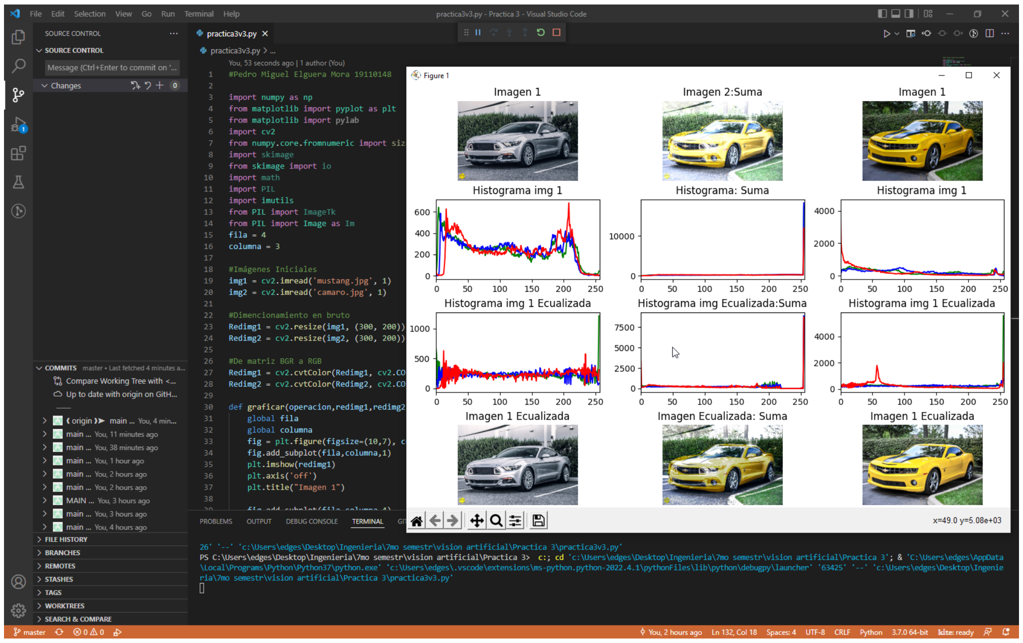
MANUAL DE USUARIO PRACTICA 3

PEDRO MIGUEL ELGUERA MORA 19110148

CETI COLOMOS VISION ARTIFICIAL 7E1

MANUAL DE USUARIO

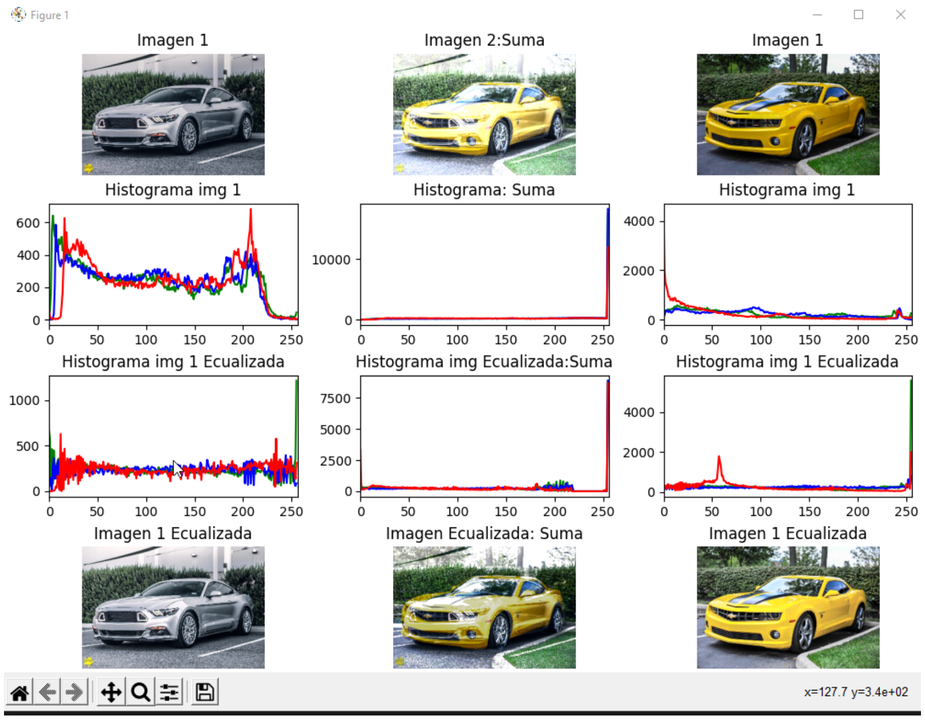
EVIDENCIA



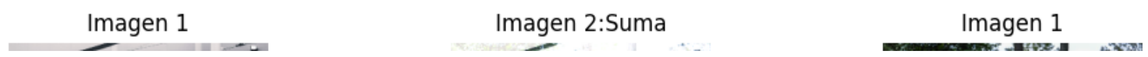
```python
#Pedro Miguel Elguera Mora 19110148

import numpy as np
from matplotlib import pyplot as plt
from matplotlib import pylab
import cv2
from numpy.core.fromnumeric import siz
import skimage
from skimage import io
import math
import PIL
import imutils
from PIL import ImageTk
from PIL import Image as Im
fila = 4
columna = 3

#Imágenes Iniciales
img1 = cv2.imread('mustang.jpg', 1)
img2 = cv2.imread('camaro.jpg', 1)

#Dimencionamiento en bruto
Redimg1 = cv2.resize(img1, (300, 200))
Redimg2 = cv2.resize(img2, (300, 200))

#De matriz BGR a RGB
Redimg1 = cv2.cvtColor(Redimg1, cv2.CO
Redimg2 = cv2.cvtColor(Redimg2, cv2.CO

def graficar(operacion,redimg1,redimg2
    global fila
    global columna
    fig = plt.figure(figsize=(10,7), c
    fig.add_subplot(fila,columna,1)
    plt.imshow(redimg1)
    plt.axis('off')
    plt.title("Imagen 1")
```

```
26' '--' 'c:\Users\edges\Desktop\Ingenieria\7mo semestr\vision artificial\Practica 3\practica3v3.py'
PS C:\Users\edges\Desktop\Ingenieria\7mo semestr\vision artificial\Practica 3>  c:; cd 'c:\Users\edges\Desktop\Ingenieria\7mo semestr\vision artificial\Practica 3'; & 'C:\Users\edges\AppData
\Local\Programs\Python\Python37\python.exe' 'c:\Users\edges\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '63425' '--' 'c:\Users\edges\Desktop\Ingenie
ria\7mo semestr\vision artificial\Practica 3\practica3v3.py'
```

APP



Esta es la vista principal de la aplicación.
Entre cada operación en las imágenes es cerrar la ventana e ira cambiando una a una, cambiara el título de la etiqueta y la operación realizada.



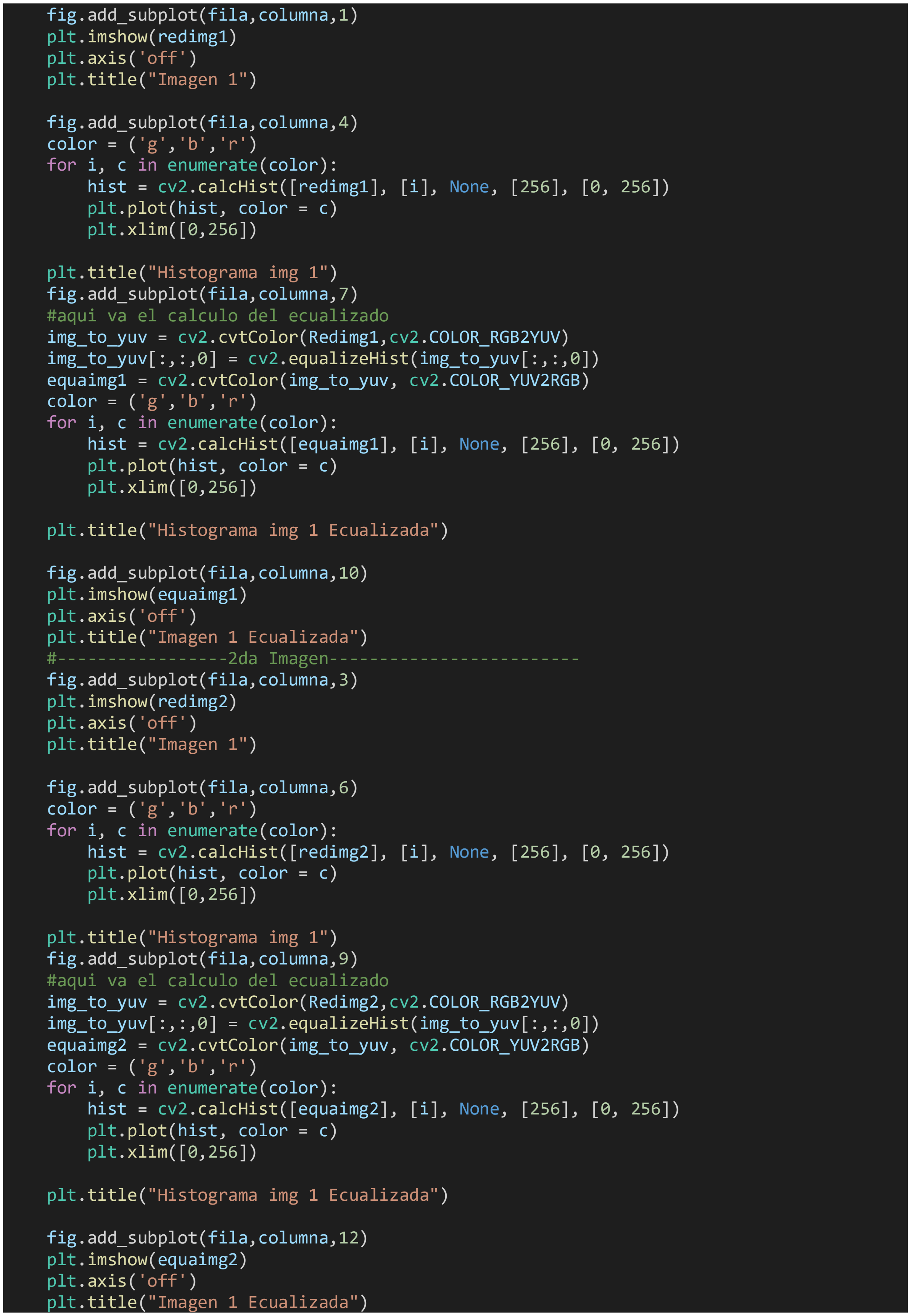
Cerrar el app

Git:

https://github.com/PedroElgueraCeti/Practica-3-VisionArtificial.git

Code:

```
#Pedro Miguel Elguera Mora 19110148

import numpy as np
from matplotlib import pyplot as plt
from matplotlib import pylab
import cv2
from numpy.core.fromnumeric import size #Opencv
import skimage
from skimage import io
import math
import PIL
import imutils
from PIL import ImageTk
from PIL import Image as Im
fila = 4
columna = 3

#Imágenes Iniciales
img1 = cv2.imread('mustang.jpg', 1)
img2 = cv2.imread('camaro.jpg', 1)

#Dimencionamiento en bruto
Redimg1 = cv2.resize(img1, (300, 200))
Redimg2 = cv2.resize(img2, (300, 200))

#De matriz BGR a RGB
Redimg1 = cv2.cvtColor(Redimg1, cv2.COLOR_BGR2RGB)
Redimg2 = cv2.cvtColor(Redimg2, cv2.COLOR_BGR2RGB)

def graficar(operacion,redimg1,redimg2,redimgop):
    global fila
    global columna
    fig = plt.figure(figsize=(10,7), constrained_layout=True)
```



```python
    fig.add_subplot(fila,columna,1)
    plt.imshow(redimg1)
    plt.axis('off')
    plt.title("Imagen 1")

    fig.add_subplot(fila,columna,4)
    color = ('g','b','r')
    for i, c in enumerate(color):
        hist = cv2.calcHist([redimg1], [i], None, [256], [0, 256])
        plt.plot(hist, color = c)
        plt.xlim([0,256])

    plt.title("Histograma img 1")
    fig.add_subplot(fila,columna,7)
    #aqui va el calculo del ecualizado
    img_to_yuv = cv2.cvtColor(Redimg1,cv2.COLOR_RGB2YUV)
    img_to_yuv[:,:,0] = cv2.equalizeHist(img_to_yuv[:,:,0])
    equaimg1 = cv2.cvtColor(img_to_yuv, cv2.COLOR_YUV2RGB)
    color = ('g','b','r')
    for i, c in enumerate(color):
        hist = cv2.calcHist([equaimg1], [i], None, [256], [0, 256])
        plt.plot(hist, color = c)
        plt.xlim([0,256])

    plt.title("Histograma img 1 Ecualizada")

    fig.add_subplot(fila,columna,10)
    plt.imshow(equaimg1)
    plt.axis('off')
    plt.title("Imagen 1 Ecualizada")
    #----------------2da Imagen------------------------
    fig.add_subplot(fila,columna,3)
    plt.imshow(redimg2)
    plt.axis('off')
    plt.title("Imagen 1")

    fig.add_subplot(fila,columna,6)
    color = ('g','b','r')
    for i, c in enumerate(color):
        hist = cv2.calcHist([redimg2], [i], None, [256], [0, 256])
        plt.plot(hist, color = c)
        plt.xlim([0,256])

    plt.title("Histograma img 1")
    fig.add_subplot(fila,columna,9)
    #aqui va el calculo del ecualizado
    img_to_yuv = cv2.cvtColor(Redimg2,cv2.COLOR_RGB2YUV)
    img_to_yuv[:,:,0] = cv2.equalizeHist(img_to_yuv[:,:,0])
    equaimg2 = cv2.cvtColor(img_to_yuv, cv2.COLOR_YUV2RGB)
    color = ('g','b','r')
    for i, c in enumerate(color):
        hist = cv2.calcHist([equaimg2], [i], None, [256], [0, 256])
        plt.plot(hist, color = c)
        plt.xlim([0,256])

    plt.title("Histograma img 1 Ecualizada")

    fig.add_subplot(fila,columna,12)
    plt.imshow(equaimg2)
    plt.axis('off')
    plt.title("Imagen 1 Ecualizada")
    #--------------------Operacion------------------
    fig.add_subplot(fila,columna,2)
    plt.imshow(redimgop)
    plt.axis('off')
    plt.title("Imagen 2:"+operacion)

    fig.add_subplot(fila,columna,5)
    color = ('g','b','r')
    for i, c in enumerate(color):
        hist = cv2.calcHist([redimgop], [i], None, [256], [0, 256])
        plt.plot(hist, color = c)
        plt.xlim([0,256])

    plt.title("Histograma: "+operacion)
    fig.add_subplot(fila,columna,8)
    #aqui va el calculo del ecualizado
    img_to_yuv = cv2.cvtColor(redimgop,cv2.COLOR_RGB2YUV)
    img_to_yuv[:,:,0] = cv2.equalizeHist(img_to_yuv[:,:,0])
    equaimgOp = cv2.cvtColor(img_to_yuv, cv2.COLOR_YUV2RGB)
    color = ('g','b','r')
    for i, c in enumerate(color):
        hist = cv2.calcHist([equaimgOp], [i], None, [256], [0, 256])
        plt.plot(hist, color = c)
        plt.xlim([0,256])

    plt.title("Histograma img Ecualizada:"+operacion)

    fig.add_subplot(fila,columna,11)
    plt.imshow(equaimgOp)
    plt.axis('off')
    plt.title("Imagen Ecualizada: "+operacion)
    plt.show()

operacion="Suma"
Redimgop=cv2.add(Redimg1,Redimg2)
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
operacion="Resta"
Redimgop=cv2.subtract(Redimg1,Redimg2)
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
operacion="Multiplicacion"
Redimgop=cv2.multiply(Redimg1,Redimg2)
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
operacion="Division"
Redimgop=cv2.divide(Redimg1,Redimg2)
graficar(operacion,Redimg1,Redimg2,Redimgop)
operacion="Raiz cuadrada"
Redimgop=Redimg1
Redimgop=np.sqrt(Redimgop)
Redimgop=np.asarray(Redimgop, dtype = int)
cv2.imwrite("resultSQRT.jpg",Redimgop)
Redimgop = cv2.imread('resultSQRT.jpg', 1)
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
operacion="Potencia"
Redimgop=Redimg1
Redimgop=np.power(Redimgop,2)
Redimgop=np.asarray(Redimgop, dtype = int)
cv2.imwrite("resultPower.jpg",Redimgop)
```

```python
Redimgop = cv2.imread('resultPower.jpg', 1)
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
operacion="Conjuncion"
Redimgop=cv2.bitwise_and(Redimg1,Redimg2)
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
operacion="Disyuncion"
Redimgop=cv2.bitwise_or(Redimg1,Redimg2)
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
operacion="Negacion"
Redimgop=Redimg1
Redimgop=image= 255-Redimgop
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
operacion="Translacion"
ancho = Redimg1.shape[1] #columnas
alto = Redimg1.shape[0] #fila
M = np.float32([[1,0,2],[0,1,2]])
Redimgop = cv2.warpAffine(img1,M,(ancho,alto))
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
operacion="Escalado"
Redimgop= imutils.resize(Redimg1,height=400)
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
operacion="Rotacion"
ancho = Redimg1.shape[1] #columnas
alto = Redimg1.shape[0] #fila
M = cv2.getRotationMatrix2D((ancho//2,alto//2),15,1)
Redimgop = cv2.warpAffine(img1,M,(ancho,alto))
graficar(operacion,Redimg1,Redimg2,Redimgop)
plt.close()
```