UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ingeniería, Arquitectura y Diseño

LABORATORIO DE SISTEMAS EMBEBIDOS

Práctica No. 8

Procesamiento de imágenes digitales empleando Python y OpenCV

7mo Semestre

Ingeniería en Electrónica - Grupo 371

Integrantes

Muñiz Hernández Oscar Javier - 358851

Guzman Gudiño Said Raul - 361198

Cruz Bautista Dante Danilo-362118

Docente

Dr. Everardo Inzunza Gonzalez

Ensenada, Baja California a 23 de noviembre de 2022

Objetivo

Realizar operaciones básicas y de filtrado con imágenes digitales mediante Python y OpenCV.

Material

1 Computadora Raspberry Pi 2 o equivalente

1 Monitor HDMI

1 Teclado y mouse

1 Dongle WiFi

1 Fuente de poder para Raspberry Pi (5V DC @ 2A)

1 Conexión a red ethernet (Nodo y cable de red con conectores RJ-45) 1 Memoria MicroSD clase 10 o similar con capacidad mínima de 8 GB

A) PROCEDIMIENTO

1. Actualizar la Raspberry.

Con el fin de poder encontrar librerías en los repositorios, es necesario tener actualizada la Raspberry por lo cual desde la terminal se ingresan los siguientes comandos:

$sudo apt-get update

$sudo apt-get upgrade

2. Instalar OpenCV.

Para poder trabajar con imágenes en Python, es necesario utilizar adicionalmente la herramienta OpenCV, el primer paso para poder utilizarlo es instalar la librería numpy:

$sudo apt-get install python-numpy

Ahora para instalar opencv2 es necesario realizar las siguientes instrucciones:

$ sudo apt-get install python-opencv

$ sudo apt-get install python-scipy

$ sudo apt-get install ipython

$ sudo apt-get install libgl1-mesa-dri

$ sudo apt-get install libopencv-dev python-opencv

Es posible que ahora salga una enorme lista con librerías y paquetes que faltan y en consecuencia no se hayan instalado los paquetes del segundo comando, no se preocupen, favor de ejecutar el siguiente comando:

$ sudo apt-get -f install

Una vez instalado, escribimos de nuevo:

sudo apt-get install libopencv-dev python-opencv

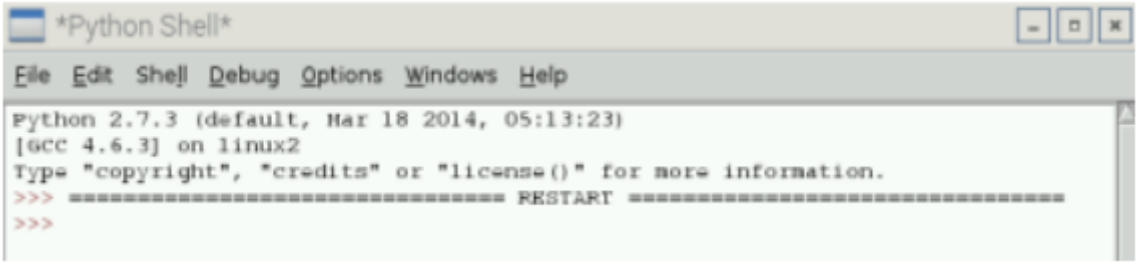
Finalmente se debe reiniciar la Raspberry:

$ sudo reboot

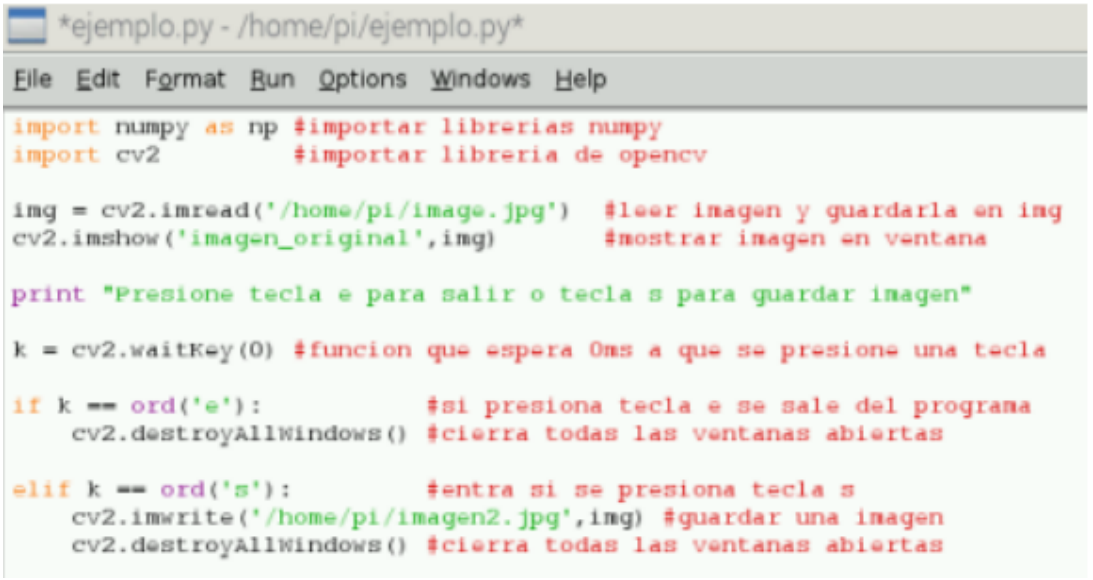
Una vez que se haya reiniciado, desde la terminal se ejecuta python:

$ idle

Y aparecerá el Shell de python:



Para realizar una prueba es necesario crear un nuevo programa, seleccione File y de clic en New window, posteriormente escriba el siguiente código y guárdelo con algún nombre con extensión ".py", en este caso se le dió el nombre de ejemplo.py:



Hay que poner especial atención en la línea del código:

img = cv2.imread('/home/pi/image.jpg')

en ella se indica entre apóstrofes la ruta de archivo donde se encuentra la foto original que se abrirá, la imagen usada en este ejemplo es la siguiente:

se presiona F5 para correr el programa y como resultado aparecerá la imagen principal, si se presiona la tecla "s" se guardará la foto original con el nombre de "imagen2.jpg"

image.jpg

imagen2.jpg

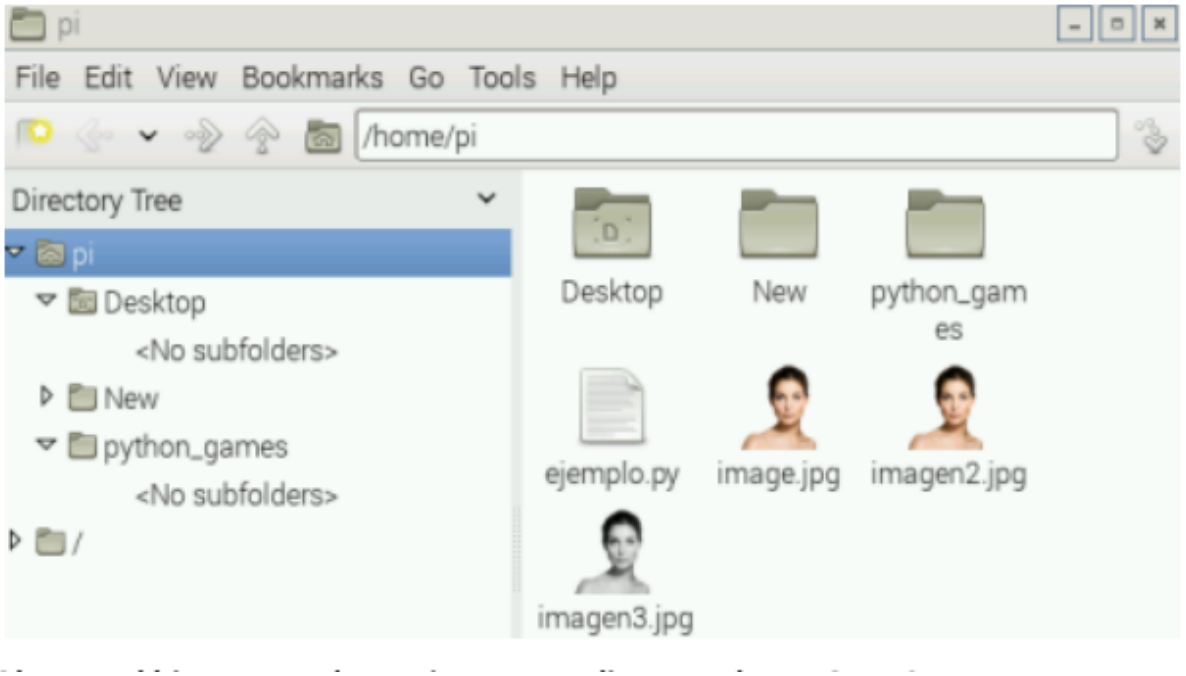
Ahora se modificará el código para que la imagen de salida sea en escala de grises, el código es el siguiente:

ejemplo.py - /home/pi/ejemplo.py

File Edit Format Run Options Windows Help

```
import numpy as np #importar librerias numpy
import cv2         #importar libreria de opencv

img = cv2.imread('/home/pi/image.jpg',0)  #leer imagen y guardarla en img
cv2.imshow('imagen_escala_de_grises',img)       #mostrar imagen en ventana

print "Presione tecla e para salir o tecla s para guardar imagen"

k = cv2.waitKey(0) #funcion que espera 0ms a que se presione una tecla

if k == ord('e'):           #si presiona tecla e se sale del programa
    cv2.destroyAllWindows() #cierra todas las ventanas abiertas

elif k == ord('s'):         #entra si se presiona tecla s
    cv2.imwrite('/home/pi/imagen3.jpg',img) #guardar una imagen
    cv2.destroyAllWindows() #cierra todas las ventanas abiertas
```

Y el resultado es el siguiente:



3. Obtener el histograma de una imagen mediante python y OpenCV

Para obtener el histograma de una imagen en escala de grises es necesario instalar la siguiente librería:

$ sudo apt-get install python-matplotlib

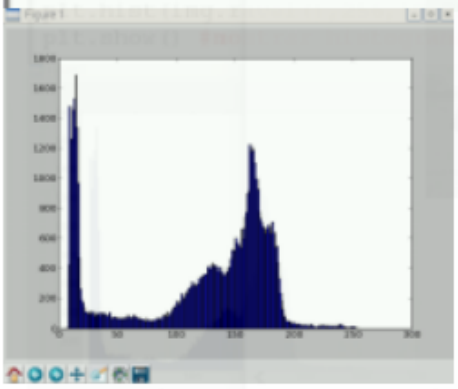
La imagen con la que se trabajará en este segundo programa será la de cameraman.tif que es la siguiente:

posteriormente se escribe el código en donde se obtiene el histograma de la imagen:

histograma.py - /home/pi/histograma.py

File Edit Format Run Options Windows Help

```
import cv2 #importar librerias de opencv
import numpy as np #importar librerias de numpy
from matplotlib import pyplot as plt #importar librerias de pyplot

img = cv2.imread('/home/pi/cameraman.tif',0) # leer imagen y guardarla en img
plt.hist(img.ravel(),256,[0,256]); #obtener histograma
plt.show() #mostrar histograma
```



4. Realización de un proceso de filtrado

OpenCV también tiene la capacidad de realizar filtros, por ejemplo el filtro promedio, el cual es logrado mediante la convolución de la imagen con un filtro de caja normalizada. El siguiente código realiza dicho filtro:

filtro.py - /home/pi/filtro.py

File Edit Format Run Options Windows Help

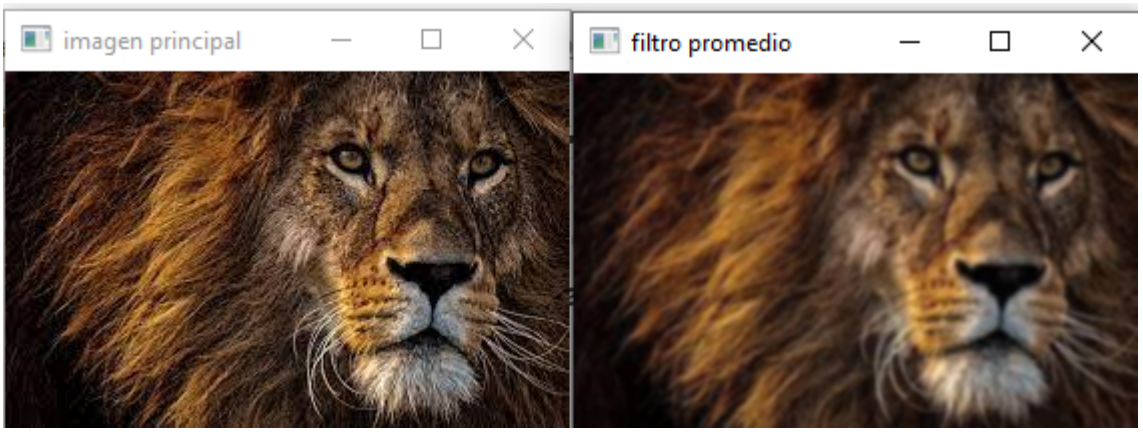
```
import cv2 #importar librerias de opencv
import numpy as np #importar librerias de numpy
from matplotlib import pyplot as plt #importar librerias de pyplot

img = cv2.imread('/home/pi/opencv-logo-white.png') #leer imagen y guardarla en img

blur = cv2.blur(img,(5,5)) #realiza la convolucion de la imagen original con un filtro 5x5
plt.subplot(121),plt.imshow(img),plt.title('Original') #muestra imagen original
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Blurred') #muestra imagen filtrada
plt.xticks([]), plt.yticks([])
plt.show()
```

Y el resultado es el siguiente:

5. Tarea: Investigar 5 filtros adicionales de procesamiento de imágenes y realizar un ejemplo en Python de cada uno de estos filtros.

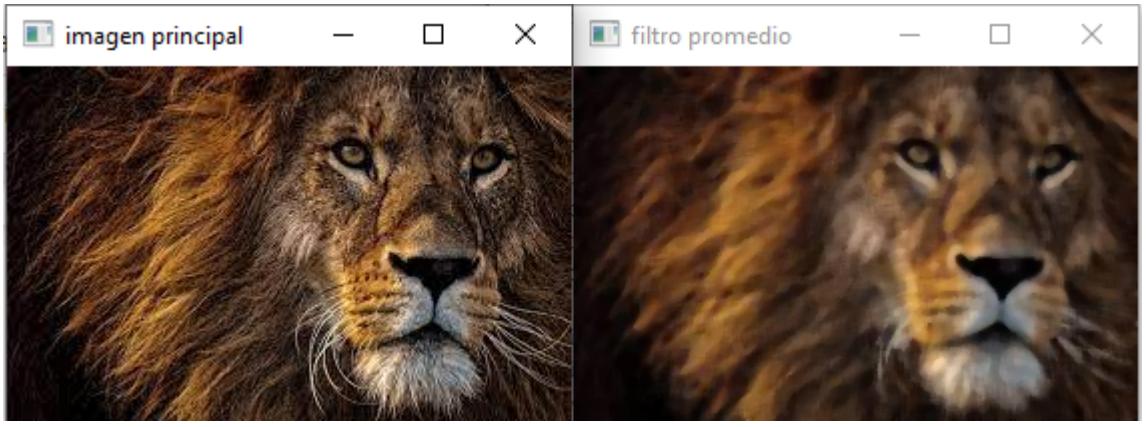
Filtro Gaussiano

```
import cv2
import numpy as np

img= cv2.imread('descarga.jpg')
#Filtro promedio
f1= cv2.blur(img,(3,3))
#Filtro GaussianBlur
f2= cv2.GaussianBlur (img,(5,5),0)

cv2.imshow('filtro promedio',f2)
cv2.imshow('imagen principal',img)
cv2.waitKey(0)
cv2.destroyALLwindow ()
```



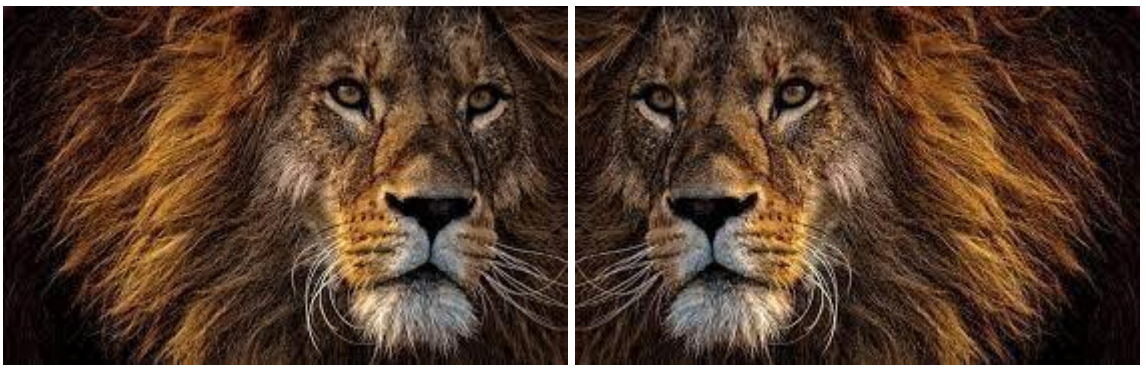
Filtro mediana

```
import cv2
import numpy as np

img= cv2.imread('descarga.jpg')
#Filtro promedio
f1= cv2.blur(img,(3,3))
#Filtro GaussianBlur
f2= cv2.GaussianBlur (img,(5,5),0)
#Filtro mediana
f3= cv2.medianBlur (img,5)

cv2.imshow('filtro promedio',f3)
cv2.imshow('imagen principal',img)
cv2.waitKey(0)
cv2.destroyALLwindow ()
```



Filtro de reflejo

```
import numpy as np
import imageio
NOMBRE_IMAGEN = "descarga.jpg"


def leer_imagen(ruta):
    return np.array(imageio.imread(ruta), dtype='int').tolist()


def guardar_imagen(ruta, matriz):
    return imageio.imwrite(ruta, np.array(matriz, dtype="uint8"))


def reflejo_horizontal(nombre_imagen):
    matriz = leer_imagen(nombre_imagen)
    ancho = len(matriz[0])
    alto = len(matriz)
    for y in range(alto):
        for x in range(int(ancho/2)):
            indice_opuesto = ancho - x - 1
            opuesto = matriz[y][indice_opuesto]
            actual = matriz[y][x]
            matriz[y][indice_opuesto] = actual
            matriz[y][x] = opuesto
    return matriz


guardar_imagen("reflejo.jpg", reflejo_horizontal(NOMBRE_IMAGEN))
```



Filtro sepia

```
import numpy as np
import imageio
NOMBRE_IMAGEN = "descarga.jpg"


def leer_imagen(ruta):
    return np.array(imageio.imread(ruta), dtype='int').tolist()


def guardar_imagen(ruta, matriz):
    return imageio.imwrite(ruta, np.array(matriz, dtype="uint8"))


def sepia(nombre_imagen):
    matriz = leer_imagen(nombre_imagen)
    ancho = len(matriz[0])
    alto = len(matriz)
    for y in range(alto):
        for x in range(ancho):
            pixel = matriz[y][x]
            original_rojo = pixel[0]
            original_verde = pixel[1]
            original_azul = pixel[2]
            sepia_rojo = round(0.393*original_rojo +
                               0.769*original_verde+0.189*original_azul)
            sepia_verde = round(0.349*original_rojo +
                                0.686*original_verde+0.168*original_azul)
            sepia_azul = round(0.272*original_rojo +
                               0.534*original_verde+0.131*original_azul)
            pixel_sepia = [sepia_rojo, sepia_verde, sepia_azul]
            for indice in range(len(pixel_sepia)):
                if pixel_sepia[indice] < 0:
                    pixel_sepia[indice] = 0
                elif pixel_sepia[indice] > 255:
                    pixel_sepia[indice] = 255
            matriz[y][x] = pixel_sepia
    return matriz


guardar_imagen("travel_sepia.bmp", sepia(NOMBRE_IMAGEN))
```

Filtro Canny

```
import cv2
import numpy as np

img= cv2.imread('descarga.jpg')

imgCanny = cv2.Canny(img,200,200)

cv2.imshow('filtro Canny',imgCanny)
cv2.imshow('imagen principal',img)
cv2.waitKey(0)
cv2.destroyALLwindow ()
```