

MANUAL DE USUARIO

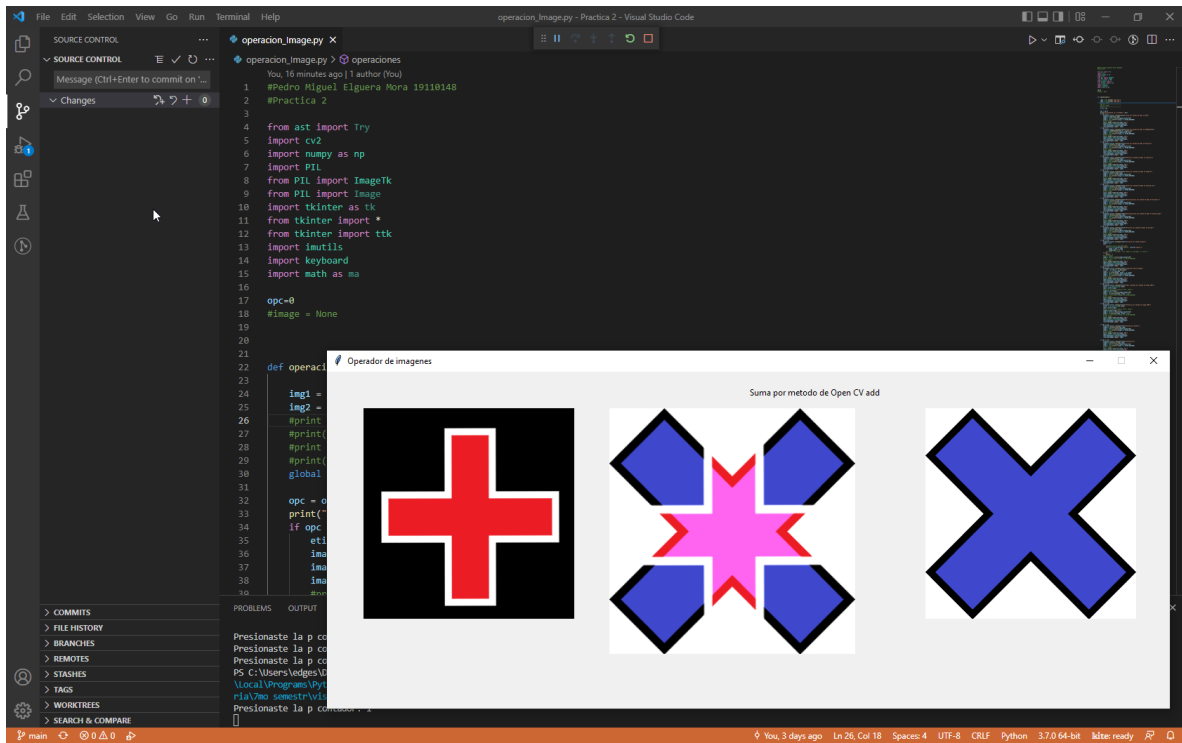
PRACTICA 2

PEDRO MIGUEL ELGUERA MORA 19110148

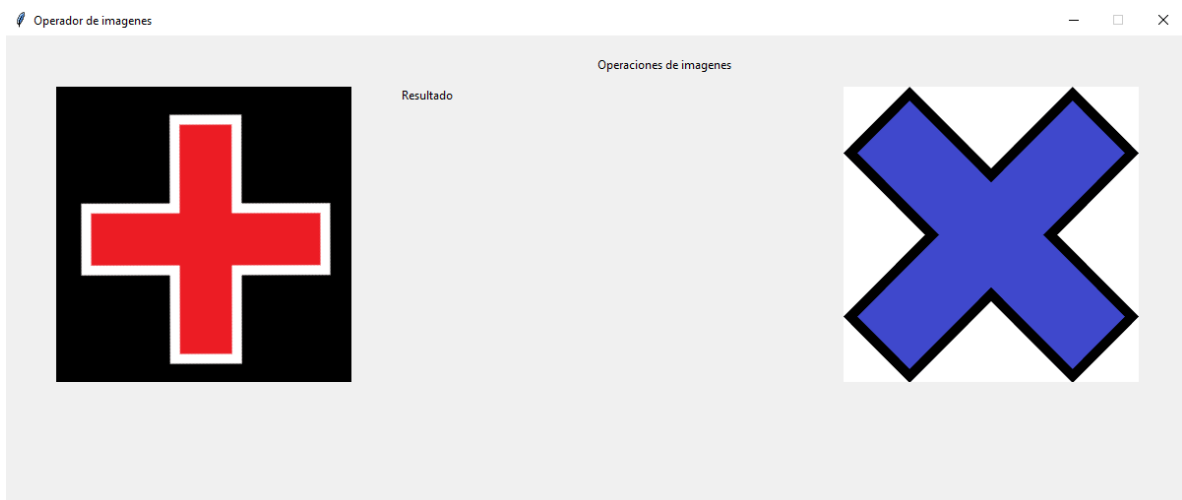
CETI COLOMOS VISION ARTIFICIAL 7E1

MANUAL DE USUARIO

EVIDENCIA



APP



Esta es la vista principal de la aplicación.

Entre cada operación en las imágenes es necesario ir presionando la letra "P" e ira cambiando una a una, cambiara el título de la etiqueta.



La app al llegar al final de las operaciones reiniciara el contador y regresara a la primera operación.

Cerrar la app



GIT:

<https://github.com/PedroElgueraCeti/Practica-2-VisionArtificial.git>

CODE:

```
#Pedro Miguel Elguera Mora 19110148
#Practica 2

from ast import Try
import cv2
import numpy as np
import PIL
from PIL import ImageTk
from PIL import Image
import tkinter as tk
from tkinter import *
from tkinter import ttk
import imutils
import keyboard
import math as ma

opc=0
#image = None

def operaciones():

    img1 = cv2.imread('img1.png')
    img2 = cv2.imread('img2.png')
    #print (img1)
    #print("-----")
    #print (img2)
    #print("-----")
    global opc

    opc = opc+1
    print("Presionaste la p contador:",opc)
    if opc == 1:
        etiqueta_titular.configure(text="Suma por metodo de Open CV add")
        image=cv2.add(img1,img2)
        image = imutils.resize(image,height=350)
        image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        #print(image)
        im=PIL.Image.fromarray(image,'RGB')
        img3r=ImageTk.PhotoImage(image=im)
        resultadoImagen.config(image=img3r)
        resultadoImagen.image = img3r
    if opc == 2:
        etiqueta_titular.configure(text="Suma por metodo de Open CV addWeighted")
        image=cv2.addWeighted(img1,0.7,img2,0.3,0)
        image = imutils.resize(image,height=350)
        image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
```

```

        #print(image)
        im=PIL.Image.fromarray(image,'RGB')
        img3r=ImageTk.PhotoImage(image=im)
        resultadoImagen.config(image=img3r)
        resultadoImagen.image = img3r
    if opc == 3:
        etiqueta_titular.configure(text="Suma por metodo de Open CV Bitwise")
        image=cv2.bitwise_or(img1,img2)
        image = imutils.resize(image,height=350)
        image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        #print(image)
        im=PIL.Image.fromarray(image,'RGB')
        img3r=ImageTk.PhotoImage(image=im)
        resultadoImagen.config(image=img3r)
        resultadoImagen.image = img3r
    if opc == 4:
        etiqueta_titular.configure(text="Resta por metodo de Open CV subtract")
        image=cv2.subtract(img1,img2)
        image = imutils.resize(image,height=350)
        image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        #print(image)
        im=PIL.Image.fromarray(image,'RGB')
        img3r=ImageTk.PhotoImage(image=im)
        resultadoImagen.config(image=img3r)
        resultadoImagen.image = img3r
    if opc == 5:
        etiqueta_titular.configure(text="Resta por metodo de Open CV absdiff")
        image=cv2.absdiff(img1,img2)
        image = imutils.resize(image,height=350)
        image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        #print(image)
        im=PIL.Image.fromarray(image,'RGB')
        img3r=ImageTk.PhotoImage(image=im)
        resultadoImagen.config(image=img3r)
        resultadoImagen.image = img3r
    if opc == 6:
        etiqueta_titular.configure(text="Resta por metodo de Open CV bitwise_xor")
        image=cv2.bitwise_xor(img1,img2)
        image = imutils.resize(image,height=350)
        image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        #print(image)
        im=PIL.Image.fromarray(image,'RGB')
        img3r=ImageTk.PhotoImage(image=im)
        resultadoImagen.config(image=img3r)
        resultadoImagen.image = img3r
    if opc == 7:
        etiqueta_titular.configure(text="Multiplicacion por metodo de Open CV multiply ")
        image=cv2.multiply(img1,img2)
        image = imutils.resize(image,height=350)
        image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        #print(image)
        im=PIL.Image.fromarray(image,'RGB')
        img3r=ImageTk.PhotoImage(image=im)
        resultadoImagen.config(image=img3r)
        resultadoImagen.image = img3r
    if opc == 8:
        etiqueta_titular.configure(text="Multiplicacion por metodo de Open CV bitwise_and")
        image=cv2.bitwise_and(img1,img2)
        image = imutils.resize(image,height=350)
        image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        #print(image)
        im=PIL.Image.fromarray(image,'RGB')
        img3r=ImageTk.PhotoImage(image=im)
        resultadoImagen.config(image=img3r)
        resultadoImagen.image = img3r
    if opc == 9:
        etiqueta_titular.configure(text="Division por metodo de Open CV divide")
        image=cv2.divide(img1,img2)
        image = imutils.resize(image,height=350)
        image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        #print(image)
        im=PIL.Image.fromarray(image,'RGB')
        img3r=ImageTk.PhotoImage(image=im)
        resultadoImagen.config(image=img3r)
        resultadoImagen.image = img3r
    if opc == 10:
        etiqueta_titular.configure(text="Division por metodo manual")
        image = None
    try:

```

```

        #image=np.nan_to_num(img1/img2)
        with np.errstate(divide='ignore', invalid='ignore'):
            image = img1 // img2
            image[image2 == 0] = 0
            #resimage = int(img1) / int (img2) if int(img2) != 0 else 0
    except:
        print(" ")
    #print (image)
    image = imutils.resize(image,height=350)
    #image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r
if opc == 11:
    etiqueta_titular.configure(text="Logarimo natural Numpy")
    c = 255 / np.log(1 + np.max(img2))
    image = c * (np.log(img2 + 1))
    image = np.array(image, dtype = np.uint8)
    image = imutils.resize(image,height=350)
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r
if opc == 12:
    etiqueta_titular.configure(text="Raiz cuadrada por metodo de numpy SQRT")
    vector = np.vectorize(np.float)
    img1l=vector(img2)
    #img1l=np.array(list(map(np.float, img1)))
    image=np.sqrt(img1l)
    image = imutils.resize(image,height=350)
    image = np.asarray(image, dtype = int)
    #image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r
if opc == 13:
    etiqueta_titular.configure(text="Potencia por metodo de numpy SQRT")
    vector = np.vectorize(np.float)
    img1l=vector(img2)
    #img1l=np.array(list(map(np.float, img1)))
    image=np.power(img1l,1.1)
    image = imutils.resize(image,height=350)
    image = np.asarray(image, dtype = int)
    #image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r
if opc == 14:
    etiqueta_titular.configure(text="Conjuncion Python")
    image=cv2.bitwise_and(img1,img2)
    image = imutils.resize(image,height=350)
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r
if opc == 15:
    etiqueta_titular.configure(text="Disyuncion")
    image=cv2.bitwise_or(img1,img2)
    image = imutils.resize(image,height=350)
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r
if opc == 16:

```

```

etiqueta_titular.configure(text="Negacion")
image= 255-img1
image = imutils.resize(image,height=350)
image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
#print(image)
im=PIL.Image.fromarray(image,'RGB')
img3r=ImageTk.PhotoImage(image=im)
resultadoImagen.config(image=img3r)
resultadoImagen.image = img3r

if opc == 17:
    etiqueta_titular.configure(text="Translacion")
    ancho = img1.shape[1] #columnas
    alto = img1.shape[0] #fila
    M = np.float32([[1,0,100],[0,1,150]])
    image = cv2.warpAffine(img1,M,(ancho,alto))
    image = imutils.resize(image,height=350)
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r

if opc == 18:
    etiqueta_titular.configure(text="Escalado")
    image = imutils.resize(img1,height=400)
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r

if opc == 19:
    etiqueta_titular.configure(text="Rotacion")
    ancho = img1.shape[1] #columnas
    alto = img1.shape[0] # filas
    # Rotación
    M = cv2.getRotationMatrix2D((ancho//2,alto//2),15,1)
    image = cv2.warpAffine(img1,M,(ancho,alto))
    image = imutils.resize(image,height=350)
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r

if opc == 20:
    etiqueta_titular.configure(text="Translacion a Fin")
    rows,cols = img1.shape[:2]
    pts1 = np.float32([[50,50],[200,50],[50,200]])
    pts2 = np.float32([[10,100],[200,50],[100,250]])
    M = cv2.getAffineTransform(pts1,pts2)
    image = cv2.warpAffine(img1,M,(rows,cols))

    image = imutils.resize(image,height=350)
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r

if opc == 21:
    etiqueta_titular.configure(text="Transpuesta")
    image = cv2.transpose(img1)
    image = imutils.resize(image,height=350)
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    #print(image)
    im=PIL.Image.fromarray(image,'RGB')
    img3r=ImageTk.PhotoImage(image=im)
    resultadoImagen.config(image=img3r)
    resultadoImagen.image = img3r

if opc >= 22:
    opc=0

```

```
raiz = Tk()
raiz.title("Operador de imagenes") #Cambiar el nombre de la ventana
raiz.geometry("1200x480") #Configurar tamaño
raiz.resizable(0,0)
etiqueta_titular = ttk.Label(text="Operaciones de imagenes")
img1vi=PhotoImage(file="img1.png")
widget=Label(raiz,text="Imagen 1",image=img1vi).place(x=50,y=50)
img2vi=PhotoImage(file="img2.png")
widget2=Label(raiz,text="imagen 2",image=img2vi).place(x=850,y=50)
etiqueta_titular.place(x=600, y=20)
resultadoImagen = Label(raiz,text="Resultado")
resultadoImagen.place(x=400,y=50)
keyboard.on_press_key("p", lambda _:operaciones())

raiz.mainloop()
```