CENTRO DE ENSEÑANZA TECNICA INDUSTRIAL

Alumno: Yasser Asaf Hernandez Garcia

Materia: Visión Artificial

Registro: 19110208

Grado y Grupo: 7E1

PRACTICA 2

PRACTICA #2

Para esta practica utilizaremos diversos métodos para realizar algunas operaciones aritméticas como lo son suma, restas, divisiones, etc. Entre otras funciones que realizaremos con nuestras imágenes, como lo son la conjunción, la disyunción, traslación a fin, entre otras.

Esto se realiza con el fin de ver que es lo que sucede con las dos imágenes al aplicar estas operaciones, en algunas veremos como los colores cambian, algunas se vuelven completamente oscuras entre otras cosas que les suceden a las imágenes.

A continuación se muestra el código que se utilizo para poder realizar estas operaciones.

```
#Yasser Asaf Hernandez Garcia 19110208
#Vision Artificial
import numpy as np
from matplotlib import pyplot as plt
import cv2 #OpenCV
from PIL import Image
import imutils
import math as ma

#Primero obtendremos las imagenes orriginales, a escala de grises y a color.
img1 = cv2.imread('foto.png',1)
img2 = cv2.imread('foto2.png',1)
imgi = cv2.imread('foto.png',0)
imgd = cv2.imread('foto2.png',0)

#Ahora escalaremos las imagenes para que sean del mismo tamaño y poder trabajar con ellas sin ningun problema
rimg1 = imutils.resize(img1,height=250)
rimg2 = imutils.resize(img2,height=250)
imagi = imutils.resize(imgi,height=250)
imagd = imutils.resize(imgd,height=250)

#Se convertiran las imagenes en matrices
np_arrayi = np.array(imagi)
np_arrayd = np.array(imagd)
arrayi = np.array(rimg1)
arrayd = np.array(rimg2)

#SUMA#
#Metodo 1
suma = rimg1+rimg2
#Metodo 2
#suma = np.add(rimg1, rimg2)
#Metodo 3
#img = cv2.imread('Izquierda.PNG',1)
#rimg = imutils.resize(img,height=250)
#suma = rimg
#suma += rimg2
```

```
#RESTA #
#Metodo 1
resta = rimg1-rimg2
#Metodo 2
#resta = np.subtract(rimg1, rimg2)
#Metodo 3
#img = cv2.imread('Izquierda.PNG',1)
#rimg = imutils.resize(img,height=250)
#resta = rimg
#resta -= rimg2

#DIVISION#
with np.errstate(divide='ignore'):
     #Metodo 1
     division=rimg1/rimg2
     #Metodo 2
     #division = np.divide(rimg1, rimg2)
     #Metodo 3
     #division=rimg1//rimg2

#MULTIPLICACION#
#Metodo 1
multiplicacion = rimg1*rimg2
#Metodo 2
#multiplicacion = np.multiply(rimg1, rimg2)
#Metodo 3
#img = cv2.imread('Izquierda.PNG',1)
#rimg = imutils.resize(img,height=250)
#multiplicacion = rimg
#multiplicacion *= rimg2

#LOGARITMO#
with np.errstate(divide='ignore'):
     logaritmo_ni = np.log(np_arrayi)
     imglogi = logaritmo_ni.astype(np.uint8)
     logaritmo_nd = np.log(np_arrayd)
     imglogd = logaritmo_nd.astype(np.uint8)
```

```
#RAIZ#
#imgraizi = rimg1**(1/2)
#imgraizd = rimg2**(1/2)
#Metodo 2
raizi = np_arrayi**(1/2)
imgraizi = raizi.astype(np.uint8)
raizd = np_arrayd**(1/2)
imgraizd = raizd.astype(np.uint8)
#Metodo 3
'''raizi=np.sqrt(np_arrayi)
imgraizi = raizi.astype(np.uint8)
raizd=np.sqrt(np_arrayd)
imgraizd = raizd.astype(np.uint8)'''

#DERIVADA#
derivadai=np.diff(np_arrayi)
imgderi = derivadai.astype(np.uint8)
derivadad=np.diff(np_arrayd)
imgderd = derivadad.astype(np.uint8)

#POTENCIA#
#Metodo 1
potenciai = rimg1**3
potenciad = rimg2**3
#Metodo 2
#potenciai = rimg1*rimg1*rimg1
#potenciad = rimg2*rimg2*rimg2
#Metodo 3
#n=3
#potenciad=1
#for i in range(n):
#    potenciad = potenciad*rimg1
#n=3
#potenciad=1
#for i in range(n):
#    potenciad = potenciad*rimg2
```

```
#CONJUNCION#
#Metodo 1
conjuncion = rimg1 & rimg2
#Metodo 2
#conjuncion = cv2.bitwise_and (rimg1, rimg2)
#Metodo 3
'''conjuncion = np.array(rimg1)
col = arrayi.shape[1] #columnas
fil = arrayi.shape[0] # filas
rgb = 3
for i in range(fil):
    for j in range(col):
        for k in range (rgb):
            a = arrayi[i][j][k]
            b = arrayd[i][j][k]
            if (a <= 128):
                a=0
            else:
                a=255
            if (b <= 128):
                b=0
            else:
                b=255
            if (a != b):
                conjuncion[i][j][k]=0
            else:
                conjuncion[i][j][k]=255'''
#Metodo 4
'''conjuncion = np.array(rimg1)
fil = arrayd.shape[0] # filas
col = arrayi.shape[1] #columnas
rgb = 3
for i in range(fil):
    for j in range(col):
        for k in range (rgb):
            a = arrayi[i][j][k]
            b = arrayd[i][j][k]
            if (a <= 128):
                a=0
            else:
                a=1
            else:
                a=1
            if (b <= 128):
                b=0
            else:
                b=1
            con = np.logical_and (a, b)
            if (con == 1):
                conjuncion[i][j][k] = 255
            else:
                conjuncion[i][j][k] = 0'''

#DISYUNCION#
#Metodo 1
disyuncion = rimg1 | rimg2
#Metodo 2
#disyuncion = cv2.bitwise_or (rimg1, rimg2)
#Metodo 3
'''disyuncion = np.array(rimg1)
col = arrayi.shape[1] #columnas
fil = arrayd.shape[0] # filas
rgb = 3
for i in range(fil):
    for j in range(col):
        for k in range (rgb):
            a = arrayi[i][j][k]
            b = arrayd[i][j][k]
            if (a <= 128):
                a=0
            else:
                a=255
            if (b <= 128):
                b=0
            else:
                b=255
            if (a == 255 or b == 255):
                disyuncion[i][j][k] = 255
            else:
                disyuncion[i][j][k] = 0'''
```

```
                disyuncion[i][j][k]   0
#Metodo 4
'''disyuncion = np.array(rimg1)
fil = arrayd.shape[0] # filas
col = arrayi.shape[1] #columnas
rgb = 3
for i in range(fil):
    for j in range(col):
        for k in range (rgb):
            a = arrayi[i][j][k]
            b = arrayd[i][j][k]
            if (a <= 128):
                a=0
            else:
                a=1
            if (b <= 128):
                b=0
            else:
                b=1
            dis = np.logical_or (a, b)
            if (dis == 1):
                disyuncion[i][j][k] = 255
            else:
                disyuncion[i][j][k] = 0'''

#NEGACION#
#Metodo 1
negacioni = ~rimg1
negaciond = ~rimg2
#Metodo 2
#negacioni = cv2.bitwise_not (rimg1)
#negaciond = cv2.bitwise_not (rimg2)
#Metodo 3
'''negacioni = np.array(rimg1)
fil = arrayi.shape[0] # filas
col = arrayi.shape[1] #columnas
rgb = 3
for i in range(fil):
    for j in range(col):
        for k in range (rgb):
            a = arrayi[i][j][k]
```

```
            a = arrayi[i][j][k]
            if (a <= 128):
                a=0
            else:
                a=1
            negi = np.logical_not (a)
            if (negi == 1):
                negacioni[i][j][k] = 255
            else:
                negacioni[i][j][k] = 0
negaciond = np.array(rimg2)
fil = arrayd.shape[0] # filas
col = arrayd.shape[1] #columnas
rgb = 3
for i in range(fil):
    for j in range(col):
        for k in range (rgb):
            negaciond[i][j][k] = np.logical_not (arrayd[i][j][k])
            b = arrayd[i][j][k]
            if (b <= 128):
                b=0
            else:
                b=1
            negd = np.logical_not (b)
            if (negd == 1):
                negaciond[i][j][k] = 255
            else:
                negaciond[i][j][k] = 0'''

#TRASLACION#
ancho = rimg1.shape[1] #columnas
alto = rimg1.shape[0] # filas
M = np.float32([[1,0,100],[0,1,150]])
traslacioni = cv2.warpAffine(rimg1,M,(ancho,alto))
ancho = rimg2.shape[1] #columnas
alto = rimg2.shape[0] # filas
M = np.float32([[1,0,50],[0,1,50]])
traslaciond = cv2.warpAffine(rimg2,M,(ancho,alto))

#ESCALADO#
#Metodo 1
escaladoi = cv2.resize(rimg1, (0,0), fx=0.5, fy=0.5)
escaladod = cv2.resize(rimg2, (0,0), fx=0.5, fy=0.5)
#Metodo 2
#escaladoi = imutils.resize(rimg1,height=100)
#escaladod = imutils.resize(rimg2,height=100)

#ROTACION#
ancho = rimg1.shape[1] #columnas
alto = rimg1.shape[0] # filas
M = cv2.getRotationMatrix2D((ancho//2,alto//2),15,1)
rotacioni = cv2.warpAffine(rimg1,M,(ancho,alto))
ancho = rimg2.shape[1] #columnas
alto = rimg2.shape[0] # filas
M = cv2.getRotationMatrix2D((ancho//2,alto//2),40,1)
rotaciond = cv2.warpAffine(rimg2,M,(ancho,alto))

#TRASLACION A FIN#
rows,cols,ch = rimg1.shape
pts1 = np.float32([[100,400],[400,100],[100,100]])
pts2 = np.float32([[50,300],[400,200],[80,150]])
M = cv2.getAffineTransform(pts1,pts2)
trasai = cv2.warpAffine(rimg1,M,(cols,rows))
rows,cols,ch = rimg2.shape
pts1 = np.float32([[100,400],[400,100],[100,100]])
pts2 = np.float32([[50,300],[400,200],[180,150]])
M = cv2.getAffineTransform(pts1,pts2)
trasad = cv2.warpAffine(rimg2,M,(cols,rows))

#TRANSPUESTA#
transpuestai=np.transpose(np_arrayi)
imgtrani = transpuestai.astype(np.uint8)
transpuestad=np.transpose(np_arrayd)
imgtrand = transpuestad.astype(np.uint8)
```

```
#La siguiente parte del codigo muestra las imagenes, primero mostrara las dos
#imagenes principales y posteriormente mostrara en el centro las operaciones
#realizadas.

#Mostramos Izquierda en Pantalla
winname = "Izquierda"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 40,300)
cv2.imshow(winname, rimg1)

#Mostramos Derecha en Pantalla
winname = "Derecha"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 1040,300)
cv2.imshow(winname, rimg2)

#Mostramos Suma en Pantalla
winname = "Suma"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, suma)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Resta en Pantalla
winname = "Resta"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, resta)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Division en Pantalla
winname = "Division"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, division)
cv2.waitKey(0)
cv2.destroyWindow(winname)
```

```
#Mostramos Multiplicacion en Pantalla
winname = "Multiplicacion"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, multiplicacion)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Logaritmo Natural de Izquierda en Pantalla
winname = "Logaritmo Natural de Izquierda"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, imglogi)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Logaritmo Natural de Derecha en Pantalla
winname = "Logaritmo Natural de Derecha"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, imglogd)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Raiz Izquierda en Pantalla
winname = "Raiz Izquierda"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, imgraizi)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Raiz Derecha en Pantalla
winname = "Raiz Derecha"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, imgraizd)
cv2.waitKey(0)
cv2.destroyWindow(winname)
```

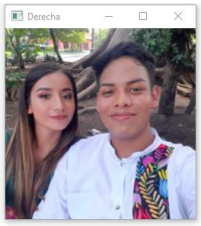
```
#Mostramos Derivada Izquierda en Pantalla
winname = "Derivada Izquierda"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, imgderi)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Derivada Derecha en Pantalla
winname = "Derivada Derecha"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, imgderd)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Potencia Izquierda en Pantalla
winname = "Potencia Izquierda"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, potenciai)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Potencia Derecha en Pantalla
winname = "Potencia Derecha"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, potenciad)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Conjunción en Pantalla
winname = "Conjuncion"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, conjuncion)
cv2.waitKey(0)
cv2.destroyWindow(winname)
```

```
#Mostramos Disyunción en Pantalla
winname = "Disyuncion"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, disyuncion)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Negación Izquierda en Pantalla
winname = "Negacion Izquierda"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, negacioni)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Negación Derecha en Pantalla
winname = "Negacion Derecha"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, negaciond)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Traslación Izquierda en Pantalla
winname = "Traslacion Izquierda"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, traslacioni)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Traslación Derecha en Pantalla
winname = "Traslacion Derecha"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, traslaciond)
cv2.waitKey(0)
cv2.destroyWindow(winname)
```

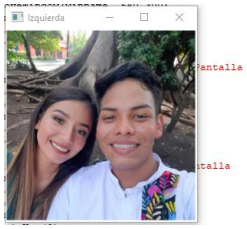
```
    #Mostramos Escalado Izquierda en Pantalla
    winname = "Escalado Izquierda"
    cv2.namedWindow(winname)
    cv2.moveWindow(winname, 540,300)
    cv2.imshow(winname, escaladoi)
    cv2.waitKey(0)
    cv2.destroyWindow(winname)

    #Mostramos Escalado Derecha en Pantalla
    winname = "Escalado Derecha"
    cv2.namedWindow(winname)
    cv2.moveWindow(winname, 540,300)
    cv2.imshow(winname, escaladod)
    cv2.waitKey(0)
    cv2.destroyWindow(winname)

    #Mostramos Rotacion Izquierda en Pantalla
    winname = "Rotacion Izquierda"
    cv2.namedWindow(winname)
    cv2.moveWindow(winname, 540,300)
    cv2.imshow(winname, rotacioni)
    cv2.waitKey(0)
    cv2.destroyWindow(winname)

    #Mostramos Rotacion Derecha en Pantalla
    winname = "Rotacion"
    cv2.namedWindow(winname)
    cv2.moveWindow(winname, 540,300)
    cv2.imshow(winname, rotaciond)
    cv2.waitKey(0)
    cv2.destroyWindow(winname)

    #Mostramos Transladar a Fin Izquierda en Pantalla
    winname = "Transladar a Fin Izquierda"
    cv2.namedWindow(winname)
    cv2.moveWindow(winname, 540,300)
    cv2.imshow(winname, trasai)
    cv2.waitKey(0)
    cv2.destroyWindow(winname)
#Mostramos Transladar a Fin Derecha en Pantalla
winname = "Transladar a Fin Derecha"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, trasad)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Transpuesta Izquierda en Pantalla
winname = "Transpuesta Izquierda"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, imgtrani)
cv2.waitKey(0)
cv2.destroyWindow(winname)

#Mostramos Transpuesta Derecha en Pantalla
winname = "Transpuesta Derecha"
cv2.namedWindow(winname)
cv2.moveWindow(winname, 540,300)
cv2.imshow(winname, imgtrand)
cv2.waitKey(0)
cv2.destroyWindow(winname)

##Final##
cv2.waitKey(0)
cv2.destroyAllWindows()
```

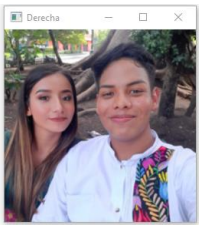
Una vez que tenemos terminado nuestro código procederemos a ver los resultados de las imágenes al realizar las diversas operaciones, para eso pondré algunos ejemplos a continuación.

SUMA

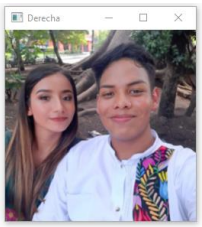


DIVISION





DERIVADA



Con esto se termina la practica.