Visión Artificial

Título: Practica 2

Nombre: Mario Gerardo Casas Miramontes

Registro: 22310165

Objetivo

El objetivo de esta práctica es comprender como es que función las operaciones dentro del uso de las imágenes en la visión artificial, ósea, los posibles usos e las imágenes a través de operaciones matemáticas.

Funcionamiento

En esta práctica teníamos que realizar varias operaciones a las imágenes:

Suma: Se suman los pixeles que se encuentren en la misma posición.

Resta: Se restan los pixeles que se encuentren en la misma posición.

División: Se dividirán los pixeles que se encuentren en la misma posición, para ello es necesario comprender los distintos casos que se pueden presentar 3 casos:

1. Si la división es menor a uno se pondrá el pixel más oscuro.
2. Si los pixeles son iguales nos dará el color blanco.
3. Si la división nos da 0, se pondrá de color blanco.

Multiplicación: Se realiza una multiplicación pixel por pixel en donde se nos puede presentar tres casos:

1. Si se multiplica un color oscuro por otro color el resultado será oscuro.
2. Si se multiplica el color blanco con otro color, no se altera el color.
3. Si una imagen esta en escala de grises, puede oscurecer o aclarar la otra imagen.

Reducción o aumento de tamaño: Esto se realiza mediante la función de cv2.resizeWindow la cual nos permite ajustar el tamaño de la ventana en la que vemos la imagen.

Rotación: Para esta operación utilizamos la siguiente función para generar una matriz de rotación cv2.getRotationMatrix2D con la cual al aplicar la instrucción cv2.warpAffine se puede generar el cambio en rotación de una imagen.

Traslación: Con esta operación podemos mover la posición de la imagen dentro de la ventana, esto se consigue mediante la función de np.float32([[1, 0, MovX], [0, 1, MovY]]) con la cual podremos generar nuestra función de matriz y con esta matriz podemos aplicar la instrucción de cv2.warpAffine con la cual generaremos una nueva imagen con los cambios de traslación.

Transpuesta: Al igual que una matriz, al hacer una transpuesta es sustituir filas por columnas, ósea, generaremos un cambio en la posición de los pixeles.

Negación: Al negar una imagen, se invierten todos los colores de la imagen, ósea, si teníamos una imagen con negros, estos se volverán blancos, esto se puede apreciar dentro de la operación de mascara con la cual buscamos modificar el logo de Python para poderlo

insertar en los que seria otra imagen, dentro de este3 proceso se hace una negación a la máscara, con la cual podemos ver el cambio contrario de colores.

Finalmente, el código desarrolla un proceso para insertar un logo en otra imagen a través del uso de una máscara. Después tenemos toda una sección con la cual podemos ordenar la posición y tamaño de las ventanas de nuestro programa, debido a que al realizar los procesas de mascara se dejan de considerar imágenes al 100% por lo que las volvemos a modo de ventana.

Código Fuente

```
#Mario Gerardo Casas Miramontes   6°G    22310165

#Practica 2


import cv2

import numpy as np


# Cargar las imágenes

img1 = cv2.imread('3D-Matplotlib.png')  # Imagen de fondo

img2 = cv2.imread('mainlogo.png')  # Logotipo a superponer

img3 = cv2.imread('mainsvmimage.png')

(h, w) = img1.shape[:2] #Sacar valores del tamaño de la imagen


#Operaciones basicas

suma = img1+img3

resta = img1-img3

Mul = cv2.multiply(img1, img3)

Division = cv2.divide(img1, img3)


#Centro de la imagen 1

centro = (w // 2, h // 2)
```

```
# Crear la matriz de rotación
angulo = 90  # Cambia el ángulo según necesites
escala = 1.0  # Factor de escala (1.0 = mismo tamaño)
MatrizR = cv2.getRotationMatrix2D(centro, angulo, escala) #Matriz para la rotacion
Rotacion = cv2.warpAffine(img1, MatrizR, (w, h))

#Operacion: Traslacion
MovX = 100 #Desplazamiento X
MovY = 100 #Despalzamiento Y
MatrizT = np.float32([[1, 0, MovX], [0, 1, MovY]]) #matriz de traslacion
Traslacion = cv2.warpAffine(img1, MatrizT, (w, h))

#Operacion transpuesta
Transpuesta = cv2.transpose(img1)
rows, cols, channels = img2.shape
roi = img1[0:rows, 0:cols]
img2gray = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

# Aplicar umbral para obtener la máscara del logo
ret, mask = cv2.threshold(img2gray, 220, 255, cv2.THRESH_BINARY_INV)

# Invertir la mascara
mask_inv = cv2.bitwise_not(mask)

img1_bg = cv2.bitwise_and(roi, roi, mask=mask_inv)

# Aplicar la máscara al logo para extraer solo la parte del logo sin fondo
img2_fg = cv2.bitwise_and(img2, img2, mask=mask)

# Combinar ambas imágenes (fondo con "hueco" + logo sin fondo)
```

```
dst = cv2.add(img1_bg, img2_fg)

# Colocar el resultado en la imagen principal
img1[0:rows, 0:cols] = dst

# En esta parte sucede que al modificar la imagen, ya no se considera como una imagen principal, sino una modificada, en las siguientes líneas le indicamos que las vuelva normales

cv2.namedWindow('Logo en Escala de Grises', cv2.WINDOW_NORMAL)
cv2.namedWindow('Mascara del Logo', cv2.WINDOW_NORMAL)
cv2.namedWindow('Mascara Invertida', cv2.WINDOW_NORMAL)
cv2.namedWindow('Fondo sin Logo', cv2.WINDOW_NORMAL)
cv2.namedWindow('Solo el Logo', cv2.WINDOW_NORMAL)
#Establecer tamaño de las pestañas
cv2.resizeWindow('Logo en Escala de Grises', 400, 400)
cv2.resizeWindow('Mascara del Logo', 400, 400)
cv2.resizeWindow('Mascara Invertida', 400, 400)
cv2.resizeWindow('Fondo sin Logo', 400, 400)
cv2.resizeWindow('Solo el Logo', 400, 400)
#Acomodar imagenes en n spacio de pantalla
cv2.moveWindow('Logo en Escala de Grises', 50, 50)
cv2.moveWindow('Mascara del Logo', 450, 50)
cv2.moveWindow('Mascara Invertida', 50, 500)
cv2.moveWindow('Fondo sin Logo', 450, 500)
cv2.moveWindow('Solo el Logo', 50, 1350)

#Ya vueltas normales pueden ser interpretadas como imagenes completas
cv2.imshow('Logo en Escala de Grises', img2gray)
cv2.imshow('Mascara del Logo', mask)
cv2.imshow('Mascara Invertida', mask_inv)
```

```
cv2.imshow('Fondo sin Logo', img1_bg)
cv2.imshow('Solo el Logo', img2_fg)
cv2.imshow('Resultado Final', img1)
cv2.imshow('Suma', suma)
cv2.imshow('Resta', resta)
cv2.imshow('Mul', Mul)
cv2.imshow('Division', Division)
cv2.imshow('Rotacion', Rotacion)
cv2.imshow('Traslacion', Traslacion)
cv2.imshow('Traspuesta', Transpuesta)

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

Resultado