Practica Percepción Visual Python

Dr. Julio Cesar Valdez Ahuatzi

Como ya se definió anteriormente, "La percepción visual es la capacidad de interpretar la información y el entorno de los efectos de la luz visible (efecto óptico) que llega al ojo. Dicha percepción es también conocida como la visión."

En la presente ´practica, se realizará el procesamiento de imágenes usando el lenguaje de programación Python.

Deberá tener instalado el paquete opencv en Python (versión 3.9 o 3.10). Puede instalarlo usando la siguiente manera: `pip install opencv-python`. Para más información puede consultar: https://pypi.org/project/opencv-contrib-python/ y también el paquete `numpy`.

OpenCV es una librería de programación de código abierto dirigía a la visión por computadora en tiempo real, usa una licencia open source BSD.

Se iniciará importando los paquetes cv2 y numpy. Luego asignamos una imagen a una variable (img).

```
import cv2
import numpy as np
y = cv2.imread("C:/Users/estarwar/Desktop/imagen1.jpg")
```

Si se introduce la siguiente instrucción, ¿Cuál es el resultado?

```
print (y)
```

Ahora se puede obtener información de la imagen: Número de pixeles, dimensión y se puede ver la imagen, con `imshow`.

```
print("- Number of Pixels: " + str(y.size))
print("- Shape/Dimensions: " + str(y.shape))
cv2.imshow('Imagen Original', y)
```

Usar para mantener la imagen hasta que se cierre o puede asignar tiempo en milisegundos, en cada una de las imágenes resultantres:

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

Cambiar a escala de grises:

```
gray_y = cv2.cvtColor(y, cv2.COLOR_BGR2GRAY)
cv2.imshow("Imagen Original", y)
cv2.imshow("Imagen en escala de grises", gray_y)
```

La imagen se puede rotar con:

```
rotationMatrix = cv2.getRotationMatrix2D((width/2, height/2), 180, .5)
rotatedImage = cv2.warpAffine(y, rotationMatrix, (width, height))
cv2.imshow('Imagen Girada', rotatedImage)
```

Se puede cambiar de tamaño de dos maneras:

1). En los ejes X y Y

```
newImg = cv2.resize(y, (0,0), fx=0.75, fy=0.75)
cv2.imshow('Image Cambio de dimensiones', newImg)
cv2.waitKey(0)
```

2). Por filas y columnas.

```
newImg = cv2.resize(y, (550, 350))
cv2.imshow('Imagen cambio de dimensiones col', newImg)
```

Se puede modificar el contraste con `addWeighted`:

```
contrast_img = cv2.addWeighted(y, 2.5, np.zeros(y.shape, y.dtype), 0, 0)
cv2.imshow('Imagen Original', y)
cv2.imshow('Imagen Contraste', contrast_img)
```

Detectar los bordes:

```
edge_img = cv2.Canny(y,100,200)
cv2.imshow("Bordes detectados ", edge_img)
```

Hacer una imagen borrosa:

```
blur_image = cv2.GaussianBlur(y, (7,7), 0)
cv2.imshow('Imagen original', y)
cv2.imshow('Imagen Borrosa', blur_image)
```

Ejercicio:
Recortar la imagen y mostrar ambas imágenes.
Guardar la imagen recortada.