Guía procesamiento digital de imágenes

Podemos definir de manera resumida El **procesamiento de imágenes digitales** como el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información. El procesamiento de imágenes tiene múltiples aplicaciones en distintos procesos tanto científicos como industriales o de operaciones, se usa en la medicina, inteligencia artificial, sector industrial, entre otros.

¿Qué es una imagen?

Una imagen se define como una función bidimensional, **F (x, y)** , donde ***x,y*** son coordenadas espaciales, y la amplitud de **F** en cualquier par de coordenadas (x, y) se llama **intensidad** de esa imagen en ese punto. Cuando los valores de x, y y amplitud de **F** son finitos, lo llamamos **imagen digital** .
En otras palabras, una imagen se puede definir mediante una matriz bidimensional dispuesta específicamente en filas y columnas.

Representación de imágenes

El espacio de color es la descripción del color bajo ciertos estándares. Los más utilizados son RGB (definido por el ojo humano, el modelo de color orientado al hardware más común), CMY (impresión industrial), HSV (procesamiento digital), HSI (procesamiento digital) Espere.

Tipos de una imagen

1. **IMAGEN BINARIA** : la imagen binaria, como su nombre indica, contiene solo dos elementos de píxeles, es decir, 0 y 1, donde 0 se refiere al negro y 1 al blanco. Esta imagen también se conoce como monocromática.
2. **IMAGEN EN BLANCO Y NEGRO** : la imagen que consta solo de color blanco y negro se llama IMAGEN EN BLANCO Y NEGRO.
3. **FORMATO DE COLOR DE 8 bits** : es el formato de imagen más famoso. Tiene 256 tonos de colores diferentes y se conoce comúnmente como imagen en escala de grises. En este formato, 0 representa el negro, 255 representa el blanco y 127 representa el gris.
4. **FORMATO DE COLOR DE 16 bits** : es un formato de imagen en color. Tiene 65.536 colores diferentes y también se conoce como formato de color de alta densidad. En este formato, la distribución del color no es la misma que la de la imagen en escala de grises.

5. **Espacio de color RGB:** En el modo de color natural, el rojo, el verde y el azul representan los tres colores primarios del espectro visible, cada uno dividido en 256 niveles (8 bits) según el brillo. **Canal de color e**s el canal para guardar la información de color de la imagen. RGB adopta el método de mezcla de colores aditivos, una imagen se forma superponiendo tres canales de color como RGB. Cada canal se puede guardar en una matriz con valores representados con 8 bits.

¿Qué es el procesamiento de imágenes?

El procesamiento de imágenes (también conocido como optimización de imágenes) es el proceso mediante el cual las imágenes originales son codificadas en una manera especial, lo que resulta en un tamaño de archivo comprimido para transferir y almacenar mientras se mantiene la mejor calidad de imagen posible.

Las imágenes de mapa de bits (bitmap, también conocidas como ráster) están formadas por píxeles de colores organizados en cuadrículas donde cada color se almacena como un número binario. Antes de la optimización, la calidad de las imágenes de mapa de bits en la web es afectada principalmente por dos factores: el número de píxeles (o dimensión en píxeles) y la profundidad de color de la imagen (también conocida como profundidad de bits), que es medida en bits por píxel.

¿Qué es Pillow?

Pillow es una biblioteca adicional gratuita y de código abierto para Python que agrega soporte para abrir, manipular y guardar muchos formatos de archivo de imagen diferentes, está construido sobre PIL (Biblioteca de imágenes de Python), Esta librería es esencial para el procesamiento de imágenes con Python, admite la variabilidad de imágenes como jpeg, png, bmp, gif, ppm y tiff y Está disponible para Windows, Mac OS X y Linux.

https://pillow.readthedocs.io/en/stable/

Instalar usando pip:

```
pip install pillow
```

¿Qué es OpenCV?
OpenCV es una librería de computación visual creada por Intel, esta librería esta disponible para múltiples plataformas como: Windows, Linux, Mac, Android, además cuenta con soporte para diferentes lenguajes como: Python, Java, C/C++, .
https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_tutorials.html

como instalar OpenCV?
usando conda

```
conda install -c conda-forge opencv
```

usando pip
```
pip install opencv-python
```

Usando pillow

Para nuestro primer ejemplo, veremos como **abrir, rotar y mostrar** una imagen, este proceso será útil para próximos ejemplos

```
from PIL import Image                #Importamos el modulo
img = Image.open('img1.png')         #Abrimos la imagen
nuevaimg = img.rotate(25)            #Rotamos 25 grados
nuevaimg.show()                      #Mostramos
```

Es posible imprimir caracteristicas basicas de la imagen como formato, tamaño y orden de los canales
```
print(img.format, img.size, img.mode)       #caracteristicas de la imagen
```

Cambiando secuencia de colores (RGB)

```
r,g,b = img.split()                          #Obtenemos los canales RGB de la imagen
nuevaimg = Image.merge("RGB", (b, r, g))  #Cambiamos el orden de los canales
nuevaimg.show()
```

Trabajando con Filtros

El método Image.filter() utilizará las técnicas de filtrado aplicado a las imágenes; veamos algunos ejemplos.

```
from PIL import ImageFilter #Ahora importamos 'ImageFilter" para los filtros
img = Image.open('img1.png')

nuevaimg = img.filter(ImageFilter.BLUR) #Filtro de Blur
nuevaimg.show()

nuevaimg = img.filter(ImageFilter.BoxBlur(50))  #Filtro de BoxBlur(50)
nuevaimg.show()

nuevaimg = img.filter(ImageFilter.CONTOUR)  #Filtro de CONTOUR
nuevaimg.show()

nuevaimg = img.filter(ImageFilter.EDGE_ENHANCE)  #Filtro de EDGE_ENHANCE
nuevaimg.show()

nuevaimg = img.filter(ImageFilter.EMBOSS)  #Filtro de EMBOSS
nuevaimg.show()

nuevaimg = img.filter(ImageFilter.FIND_EDGES)  #Filtro de FIND_EDGES
nuevaimg.show()
```

Guardar los resultados en una nueva imagen:

Se pueden guardar los resultados en un directorio y con un formato especifico con el método **.save()**, en este caso guardaremos el ultimo ejemplo donde utilizamos el filtro **FIND_EDGES**, por lo que nuestro código final quedaría así:

```
#######
from PIL import Image,ImageFilter
img = Image.open('img1.png')
nuevaimg = img.filter(ImageFilter.FIND_EDGES)  #Filtro de FIND_EDGES
nuevaimg.save('img_result.png') #Guardado en un directorio especifico
```

Operaciones adicionales: cortar, pegar y combinar imágenes

La clase Image tiene métodos que permiten manipular regiones de la imagen. Se puede usar el método crop() para extraer una región limitada por un rectángulo:

Copiar un rectángulo de una imagen

```
box = (100, 100, 400, 400)
region = img.crop(box)
```

EJEMPLO con dos imágenes:

```
from PIL import Image                      #Importamos el modulo
img1 = Image.open('img1.png')              #Abrimos la imagen 1
print(img1.format, img1.size, img1.mode)  #propiedades img1

img2 = Image.open('img2.jpeg')              #Abrimos la imagen 2
print(img2.format, img2.size, img2.mode)   #propiedadesimg2


box1 = (100, 100, 350, 350)  #definimos una region de la img1
region1 = img1.crop(box1)     #se copia la region de la imagen en una nueva
variable
region1.show()

box2 = (200, 200, 500, 500) #definimos una region de la img2
region2 = img2.crop(box2)     #se copia la region de la imagen en una nueva
variable
region2.show()
```

Procesar el rectángulo de forma separada y luego pegarlo de nuevo en la región correspondiente. En este caso se usa el método paste()

```
region = region.transpose(Image.Transpose.ROTATE_180) #aplicar cualquier operacion
im.paste(region, box)
```

Ejemplo

```
region1 = region1.filter(ImageFilter.EDGE_ENHANCE)   #aplicar un filtro
img1.paste(region1, box1)                              #pegar en la imagen original
img1.show()                                           #mostrar resultado
```

Mezclar las dos imágenes, poner la región 1 en la imagen 2

```
img2.paste(region1, box1)                              #pegar en la imagen 2
img2.show()
```

Modificar el tamaño o las dimensiones de una imagen

para conocer las dimensiones de una imagen se usa la propiedad size, por ejemplo para la imagen 1 se puede usar `img1.size.` Existen múltiples situaciones donde es necesario cambiar la resolución de una imagen. Bien para estandarizar una base de datos o para reducción del coste computacional al procesarlas en un modelo. Para lograr esto se usa el método resize()

```
from PIL import Image

img1 = Image.open('img1.png')

new_ dim = (500,500) #dar las nuevas dimensiones (alto, ancho)
im_resized = img1.resize(new_dim)

im_resized.save('img1_new_size.png')
```

También es posible cambiar de dimensión de forma proporcional al tamaño original

```
#reducir a la mitad,  el resultado para ancho y alto debe ser un numero entero
new_dim = (img1.width // 2, img1.height // 2)
im_resized = img1.resize(new_dim)
im_resized.save('img1_half.png')
```