

Evidencia 1

Barozzi Eugenia


Procesamiento de imágenes con Pillow y CV2



Profesora: Sol Del Valle FIGUEROA

Barozzi Eugenia

ISPC | Primer evidencia de Procesamiento de imágenes

 Copia de EV1Procesamiento.Img.ipynb

Introducción

En este proyecto, vamos a mostrar cómo podemos modificar una imagen usando dos librerías muy útiles: OpenCV (cv2) y Pillow. Veremos cómo estas librerías nos permiten hacer diferentes transformaciones con las imágenes.

Pillow y OpenCV: punto de partida en procesamiento de imágenes

Pillow y OpenCV son dos de las bibliotecas más fundamentales y accesibles para comenzar a trabajar con procesamiento de imágenes en Python. Ambas permiten realizar tareas básicas y avanzadas con imágenes, desde su carga hasta transformaciones complejas, pero cada una tiene su propio enfoque y ventajas.

Pillow (también conocido como PIL - Python Imaging Library) es una biblioteca sencilla y poderosa que facilita el manejo de imágenes en Python. Permite abrir, editar, guardar y aplicar filtros a imágenes en una gran variedad de formatos. Es especialmente útil para tareas básicas como cambiar el tamaño, rotar, modificar el brillo o el contraste, aplicar desenfoques o convertir imágenes a escala de grises. Su sintaxis es amigable, lo que la convierte en una excelente opción para quienes recién comienzan.

OpenCV (Open Source Computer Vision Library), por su parte, es una biblioteca mucho más extensa, diseñada específicamente para la visión por computadora. Ofrece capacidades de procesamiento mucho más potentes y detalladas, incluyendo detección de objetos, seguimiento, transformaciones geométricas avanzadas, filtros personalizados mediante convolución, detección de bordes, análisis de video, y mucho más. Aunque su curva de aprendizaje es un poco mayor, es una herramienta indispensable en proyectos que requieren un mayor control y precisión.

Comenzamos las transformaciones

Escala de grises con Pil

Una de las primeras transformaciones más comunes y necesarias es la conversión de una imagen a escala de grises. Esto simplifica la información visual al eliminar el color, permitiendo centrarse en la intensidad de luz de cada píxel.

```
# 2. Leer la imagen con OpenCV
imagen_opencv = cv2.imread(filename)

# 3. Convertir de OpenCV a PIL (cambiar BGR a RGB)
imagen_pil = Image.fromarray(cv2.cvtColor(imagen_opencv, cv2.COLOR_BGR2RGB))
```

OpenCV por defecto carga las imágenes en formato BGR (Blue-Green-Red) PIL y matplotlib esperan el formato RGB (Red-Green-Blue)

```
# 4. Aplicar filtros artísticos con PIL # Convertir a escala de grises
imagen_gris = imagen_pil.convert('L')

# 5. Mostrar la imagen en escala de grises
plt.imshow(imagen_gris, cmap='gray')
plt.axis('off') # Ocultar ejes
plt.show()
```



Resaltamos bordes con Pil

Una vez que la imagen ha sido convertida a escala de grises, un paso natural en el flujo de procesamiento es el realce de bordes. Con Pillow, resaltar bordes es sencillo gracias a los filtros ya incorporados en la librería, como `ImageFilter.CONTOUR` o `ImageFilter.FIND_EDGES`. Estos filtros aplican técnicas de detección de bordes que permiten identificar las regiones donde se producen diferencias notorias de intensidad entre píxeles vecinos.

```
from PIL import Image, ImageFilter # Import the ImageFilter class

# 6 Resaltar bordes
imagen_bordes = imagen_gris.filter(ImageFilter.FIND_EDGES)

# 7. Mostrar la imagen con bordes resaltados
plt.imshow(imagen_bordes, cmap='gray')
plt.axis('off')
plt.show()
```



Rotación y recorte con PIL

Con Pillow, se puede trabajar de manera intuitiva con imágenes. Por ejemplo, rotar una imagen implica girar una cantidad específica de grados, lo cual es útil cuando las imágenes no están correctamente alineadas o cuando se desea aplicar una transformación creativa. Por otro lado, el recorte permite seleccionar una región específica de la imagen (una subimagen rectangular), lo cual es muy útil para enfocar el análisis en un área de interés particular o eliminar elementos irrelevantes.

```
from PIL import Image

# Abrir imagen
img = Image.open('imagen.png')

# Rotar 45 grados
img_rotada = img.rotate(45, expand=True)

# Recortar (izquierda, superior, derecha, inferior)
img_recortada = img.crop((100, 100, 400, 400))

# Guardar resultados
img_rotada.save("rotada.jpg")
img_recortada.save("recortada.jpg")
```

Imagen Rotada



Imagen Recortada



Enfoque Gaussiano con Pil

El filtro gaussiano actúa suavizando los cambios bruscos de intensidad en la imagen, reduciendo el ruido o variaciones no deseadas. Esto es especialmente útil en tareas donde se necesita una imagen más "limpia" para extraer características, como bordes o formas, con mayor precisión. A diferencia de otros desenfoques simples, el gaussiano utiliza una distribución normal (campana de Gauss) para calcular cómo se combinan los píxeles vecinos, produciendo una transición más suave y realista.

```
from PIL import Image, ImageFilter

img = Image.open('imagen.png')

# Desenfoque gaussiano
img_blur = img.filter(ImageFilter.GaussianBlur(radius=5))

# Detección de bordes
img_edges = img.filter(ImageFilter.FIND_EDGES)

img_blur.save("blur.jpg")
img_edges.save("bordes.jpg")
```

Imagen con Desenfoque Gaussiano



Imagen con Deteccion de Bordes



Corrección de Perspectiva con OpenCV (CV2)

La corrección de perspectiva con OpenCV permite transformar una imagen distorsionada o inclinada (como una foto tomada en ángulo). Esto se hace utilizando cuatro puntos de origen (los vértices de una figura en perspectiva) y mapeándolos a cuatro puntos de destino (los vértices de la figura corregida). Este proceso es útil cuando queremos trabajar con documentos escaneados torcidos o cualquier imagen que no esté bien alineada.

```
# Puntos de origen y destino (ej: corrección de perspectiva)
puntos_origen = np.float32([[65, 25], [468, 52], [28, 387], [389, 390]])
puntos_destino = np.float32([[0, 0], [300, 0], [0, 300], [300, 300]])

matriz_perspectiva = cv2.getPerspectiveTransform(puntos_origen, puntos_destino)
img_transformada = cv2.warpPerspective(img, matriz_perspectiva, (300, 300))

cv2.imshow('img_transformada')
```




Operaciones Avanzadas con OpenCV Convolución y Filtrado

El valor central del kernel es positivo y elevado, lo que intensifica el píxel central, mientras que los valores negativos en los píxeles vecinos restan intensidad, y las esquinas no influyen. El resultado es una imagen con mayor contraste local, donde los bordes y los detalles finos se destacan más. Este tipo de filtrado se usa para compensar desenfoques leves, mejorar la visualización de detalles, especialmente en imágenes médicas, y como paso previo a la detección de bordes.

```
# Kernel para enfocar
kernel = np.array([[0, -1, 0],
                  [-1, 6, -1],
                  [0, -2, 0]])

img_filtrada = cv2.filter2D(img, -1, kernel)
cv2.imshow(img_filtrada)
```



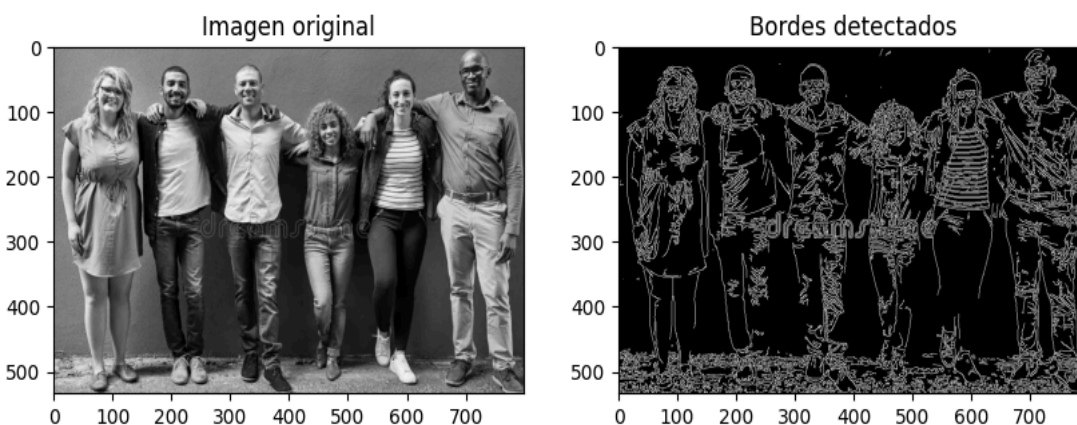
Umbralización (Binarización) con cv2

La umbralización o binarización es una técnica de procesamiento de imágenes que convierte una imagen en escala de grises en una imagen blanco y negro. Se basa en comparar cada píxel con un valor de umbral predefinido: si el valor del píxel es igual o mayor al umbral, se asigna blanco; si es menor, se asigna negro. Esta técnica se utiliza comúnmente en reconocimiento de texto (OCR), detección de objetos, limpieza de documentos escaneados y segmentación en imágenes médicas.



Detección de bordes Canny con CV2

La detección de bordes Canny es un algoritmo muy utilizado en visión por computadora para identificar bordes en imágenes de forma precisa y robusta. Permite resaltar contornos y límites de objetos dentro de una imagen. Se aplica en tareas como reconocimiento de objetos, vehículos o rostros, detección de carriles en sistemas de conducción autónoma, mejora de textos en OCR y análisis de estructuras en imágenes médicas.



La segmentación de imágenes utilizando scikit-learn

Usamos OpenCV para obtener los datos de la imagen (que se almacenan como arrays de NumPy) y NumPy para manipular esos arrays de manera eficiente antes de aplicar K-Means de la librería scikit-learn).

K-Means es un algoritmo de clustering o agrupamiento. Su objetivo es tomar un conjunto de datos (en nuestro caso, los valores RGB de los píxeles) y dividirlos en un número predefinido de grupos (clusters) de tal manera que los puntos dentro de cada grupo sean lo más similares posible entre sí, y los grupos sean lo más diferentes posible entre ellos."

K-Means en imágenes es una técnica poderosa para simplificar la información de color y segmentar la imagen en regiones basadas en la similitud de color, lo cual tiene diversas aplicaciones en el procesamiento y análisis de imágenes.



Con este proyecto tenemos un vistazo general de las transformaciones que podemos realizar con las librerías de pillow (PIL) y OpenCV (cv2). Mientras Pillow se destaca por su enfoque intuitivo en la edición y conversión de imágenes, OpenCV emerge como una potente herramienta para tareas avanzadas de visión por computadora y análisis. La comprensión de las capacidades distintivas de cada librería permite a los desarrolladores seleccionar la herramienta más adecuada para abordar los requerimientos específicos de sus proyectos de procesamiento de imágenes.
