Umbralización de Imágenes con OpenCV

---

1. Introducción

La umbralización es una técnica fundamental en el procesamiento de imágenes que permite segmentar objetos en una imagen al separar los píxeles en dos grupos: fondo y primer plano. El siguiente código en Python utiliza la biblioteca OpenCV para aplicar distintos métodos de umbralización y compararlos visualmente.

---

2. Línea por Línea del Código

import numpy

Importa la biblioteca **NumPy**, utilizada para operaciones numéricas, aunque en este caso no se usa directamente. Suele ser requerida implícitamente por OpenCV.

import matplotlib

Importa la biblioteca **Matplotlib**, comúnmente usada para visualización de datos. En este código no se utiliza, pero podría usarse para representar histogramas o comparaciones.

import cv2

Importa la biblioteca **OpenCV**, esencial para el procesamiento de imágenes en Python.

img = cv2.imread('001_Thresholding/fotos/bookpage.jpg')

Lee una imagen a color desde el directorio especificado y la guarda en la variable `img`.

---

3. Umbralización Global Binaria

retval, threshold = cv2.threshold(img, 10, 255, cv2.THRESH_BINARY)

Aplica **umbralización binaria global** directamente sobre la imagen en color. Todos los píxeles con valores menores a 10 se convierten en 0 (negro), y los demás en 255 (blanco). Aunque normalmente esta técnica se aplica a imágenes en escala de grises, también puede aplicarse a cada canal de color.

grayscaled = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

Convierte la imagen original a **escala de grises**, lo cual es necesario para aplicar correctamente los métodos de umbralización que analizan niveles de intensidad.

retval2, threshold2 = cv2.threshold(grayscaled, 12, 255, cv2.THRESH_BINARY)

Aplica **umbralización binaria** sobre la versión en escala de grises. Los valores menores a 12 se convierten en 0; los mayores o iguales, en 255.

---

4. Umbralización Adaptativa

```
gaus = cv2.adaptiveThreshold(grayscaled, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                             cv2.THRESH_BINARY, 115, 1)
```

Aplica un **umbral adaptativo gaussiano**, donde el umbral se calcula para cada región local de la imagen (bloques de 115x115 píxeles). El valor de cada píxel se compara con el promedio ponderado de su vecindad, restando una constante (1). Esto es muy útil cuando la iluminación no es uniforme.

---

5. Umbralización con el Método de Otsu

retval3, otsu = cv2.threshold(grayscaled, 125, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

Aplica el **método de Otsu**, que determina automáticamente el umbral óptimo analizando el histograma de la imagen. Ignora el valor proporcionado manualmente (125) si se activa `cv2.THRESH_OTSU`. Este método es eficaz cuando la imagen presenta una distribución bimodal (dos grupos de intensidades bien diferenciados).

---

6. Visualización de Resultados

```
cv2.imshow('original', img)
cv2.imshow('threshold', threshold)
cv2.imshow('threshold2', threshold2)
cv2.imshow('gaus', gaus)
cv2.imshow('otsu', otsu)
```

Muestra en ventanas independientes:

- La imagen original.
- El resultado de umbralización sobre imagen en color (`threshold`).
- Umbral binario sobre imagen en grises (`threshold2`).
- Umbral adaptativo gaussiano (`gaus`).
- Umbral con el método de Otsu (`otsu`).

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

- `waitKey(0)`: Espera que el usuario presione una tecla para cerrar las ventanas.
- `destroyAllWindows()`: Cierra todas las ventanas abiertas por OpenCV.

---

7. Conclusión

Este código demuestra diferentes técnicas de umbralización con OpenCV. Cada método tiene sus ventajas y aplicaciones específicas:

- **Umbral fijo global**: sencillo, pero sensible a cambios de iluminación.
- **Umbral adaptativo**: ideal para escenas con iluminación desigual.
- **Método de Otsu**: automático, recomendado cuando no se conoce un buen valor de umbral.

Estas herramientas son fundamentales en preprocesamiento de imágenes para tareas como segmentación, reconocimiento de caracteres y detección de objetos.