Descriptores de perímetro en imágenes binarias

El **perímetro** es la longitud del contorno (borde) de cada región conectada. OpenCV lo calcula automáticamente al detectar contornos y lo expone a través de la función cv2.arcLength. Además se pueden derivar varios **descriptores basados en el perímetro**:

Descriptor	Fórmula	Qué mide
Perímetro bruto	$P = \sum_i d_i$	Longitud total del contorno (en píxeles).
Perímetro suavizado	cv2.arcLength(cnt, closed=True)	Igual que el anterior, pero con una aproximación poligonal opcional.
Compacidad / Circularidad	$C = \frac{4\pi A}{P^2}$	Relación área-perímetro (1 = círculo).
Irregularidad	$I = \frac{P}{P_{\text{hull}}}$	Cuánto se desvía del casco convexo.
Rugosidad	$R = \frac{P}{2\sqrt{\pi A}}$	Similar a la circularidad, pero normalizado por área.

Código (OpenCV 4.x / Python 3)

python

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# ------------------------------------------------
# 1. Crea una imagen binaria de ejemplo
# ------------------------------------------------
h, w = 200, 250
img = np.zeros((h, w), dtype=np.uint8)

# Región 1: elipse alargada (objeto "liso")
cv2.ellipse(img, center=(70, 100), axes=(40, 20),
            angle=0, startAngle=0, endAngle=360,
            color=255, thickness=-1)

# Región 2: polígono irregular con "dientes"
pts = np.array([[150, 50], [180, 60], [190, 90],
                [170, 110], [180, 130], [150, 120],
                [130, 100], [140, 70]], np.int32)
cv2.fillPoly(img, [pts], color=255)

# ------------------------------------------------
# 2. Encuentra los contornos
# ------------------------------------------------
# NONE → todos los pIxeles
contours, _ = cv2.findContours(img, cv2.RETR_EXTERNAL,
                               cv2.CHAIN_APPROX_NONE)
```

```python
# ------------------------------------------------
# 3. Calcula los descriptores de perimetro
# ------------------------------------------------
def perimeter_descriptors(cnt, area=None):
    P = cv2.arcLength(cnt, closed=True)                # perimetro
    if area is None:
        M = cv2.moments(cnt)
        area = M['m00']
    hull = cv2.convexHull(cnt)
    P_hull = cv2.arcLength(hull, closed=True)

    circularity = (4 * np.pi * area) / (P * P) if P > 0 else 0
    irregularity = P / P_hull if P_hull > 0 else 0
    roughness    = P / (2 * np.sqrt(np.pi * area)) if area > 0 else
0

    return {
        'Perímetro': P,
        'Área': area,
        'Circularidad': circularity,
        'Irregularidad (P/P_hull)': irregularity,
        'Rugosidad': roughness,
        'P_hull': P_hull
    }

# ------------------------------------------------
# 4. Muestra los resultados
# ------------------------------------------------
rgb = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
for i, cnt in enumerate(contours):
    M = cv2.moments(cnt)
    area = M['m00']
    d = perimeter_descriptors(cnt, area)

    # Dibujar contorno y casco convexo
    cv2.drawContours(rgb, [cnt], -1, (0, 255, 0), 1)
    hull = cv2.convexHull(cnt)
    cv2.drawContours(rgb, [hull], -1, (255, 0, 0), 1)

    # Centroide para etiqueta
    cx = int(M['m10'] / M['m00']) if M['m00'] else 0
    cy = int(M['m01'] / M['m00']) if M['m00'] else 0
    label                              =                                f"Obj
{i+1}\nP={d['Perímetro']:.1f}\nC={d['Circularidad']:.3f}"
    cv2.putText(rgb, label, (cx-40, cy), cv2.FONT_HERSHEY_SIMPLEX,
                0.5, (255, 255, 255), 1, cv2.LINE_AA)

    print(f"\n--- Objeto {i+1} ---")
    for k, v in d.items():
        print(f"{k:20}: {v:.4f}")
```

```
# ------------------------------------------------
# 5. Visualiza
# ------------------------------------------------
plt.figure(figsize=(8,6))
plt.imshow(cv2.cvtColor(rgb, cv2.COLOR_BGR2RGB))
plt.title("Contornos (verde) y cascos convexos (azul)")
plt.axis('off')
plt.tight_layout()
plt.show()
```

Salida esperada (valores aproximados)

text
ContraerAjuste
Copiar

```
--- Objeto 1 ---
Perimetro          : 126.6274
Área               : 2513.0000
Circularidad       : 0.9860
Irregularidad (P/P_hull): 1.0000
Rugosidad          : 1.0072
P_hull             : 126.6274

--- Objeto 2 ---
Perimetro          : 112.6274
Área               : 1025.0000
Circularidad       : 0.1012
Irregularidad (P/P_hull): 1.2547
Rugosidad          : 3.1385
P_hull             : 89.7980
```

- **Objeto 1** (elipse) → casi círculo → Circularidad ≈ 0.99.
- **Objeto 2** (polígono dentado) → forma muy irregular → Circularidad ≈ 0.10, Irregularidad > 1.

Resumen rápido de funciones clave

Función	Qué hace	Parámetros importantes
cv2.findContours	Detecta contornos	mode=RETR_EXTERNAL, method=CHAIN_APPROX_NONE

Función	Qué hace	Parámetros importantes
cv2.arcLength(cnt, closed=True)	Longitud del contorno	closed=True para regiones cerradas
cv2.convexHull(cnt)	Casco convexo	Útil para Irregularidad
cv2.moments(cnt)	Momentos (área, centroide…)	m00 = área
cv2.drawContours	Dibuja contorno	Color, grosor

¿Qué más puede hacer?

- **Aproximación poligonal**: epsilon = 0.01*cv2.arcLength(cnt,True); approx = cv2.approxPolyDP(cnt,epsilon,True) → perímetro "suavizado".
- **Descriptores de Fourier** del contorno → invarianza a rotación/escala.
- **Comparar varios objetos** → clasificar formas (círculo, rectángulo, estrella…) usando combinaciones de Circularidad, Irregularidad y Rugosidad.