Descriptores de Textura en una Imagen

(Teoría + Implementación en Python con OpenCV, scikit-image y numpy)

---

1. ¿Qué es la textura?

La **textura** describe la **variación espacial de intensidad** en una región de la imagen. Se mide a nivel **local** (vecindades pequeñas) y se resume en **descriptores globales** por región o imagen completa.

---

2. Principales Categorías de Descriptores de Textura

Categoría	Descriptores	Invarianzas	Uso típico
Estadísticos de primer orden	Media, varianza, skewness, kurtosis	No espacial	Distribución de intensidades
Matriz de Co-ocurrencia (GLCM)	Energía, contraste, correlación, homogeneidad	Rotación (con múltiples ángulos)	Textura direccional
Estadísticos de segundo orden (LBP, Haralick)	LBP, 13 características de Haralick	Rotación (LBP uniforme)	Clasificación robusta
Frecuenciales	Transformada de Gabor, Wavelets, FFT	Escala, orientación	Texturas periódicas
Estructurales	Patrones binarios locales (LBP), HOG	Rotación, escala (con variantes)	Reconocimiento de objetos

---

3. Implementación Completa en Python

python

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.feature import graycomatrix, graycoprops,
local_binary_pattern
from skimage import io, color
from scipy import ndimage as ndi
import warnings
warnings.filterwarnings("ignore")

# ------------------------------------------------
# 1. Cargar imagen (en escala de grises)
# ------------------------------------------------
img_color = io.imread('textura_ejemplo.jpg')  # Cambia por tu
imagen
img = color.rgb2gray(img_color)
```

```
img = (img * 255).astype(np.uint8)

# ------------------------------------------------
# 2. Descriptores Estadisticos (1er orden)
# ------------------------------------------------
def stats_descriptors(roi):
    mean = np.mean(roi)
    var  = np.var(roi)
    skew = np.mean((roi - mean)**3) / (var**1.5 + 1e-8)
    kurt = np.mean((roi - mean)**4) / (var**2 + 1e-8) - 3
    return {'mean': mean, 'var': var, 'skew': skew, 'kurt': kurt}

# ------------------------------------------------
# 3. GLCM (Gray-Level Co-occurrence Matrix)
# ------------------------------------------------
def glcm_descriptors(roi, distances=[1], angles=[0, 45, 90, 135]):
    # Asegurar rango 0-255 y convertir a uint8
    roi = (roi // (roi.max() / 15)).astype(np.uint8) if roi.max()
> 15 else roi.astype(np.uint8)

    glcm = graycomatrix(roi, distances=distances, angles=angles,
                        levels=16, symmetric=True, normed=True)

    props = ['contrast', 'dissimilarity', 'homogeneity', 'energy',
'correlation', 'ASM']
    desc = {}
    for prop in props:
        val = graycoprops(glcm, prop)
        desc[prop] = np.mean(val)  # promedio sobre ángulos
    return desc

# ------------------------------------------------
# 4. LBP (Local Binary Pattern)
# ------------------------------------------------
def lbp_descriptors(roi, P=8, R=1, method='uniform'):
    lbp = local_binary_pattern(roi, P=P, R=R, method=method)
    hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, P+3),
density=True)
    return {'lbp_hist': hist}

# ------------------------------------------------
# 5. Gabor Filters (Frecuencia + Orientación)
# ------------------------------------------------
def gabor_descriptors(roi, frequencies=[0.1, 0.3], thetas=[0, 45,
90, 135]):
    desc = {}
    for freq in frequencies:
        for theta in thetas:
            theta_rad = np.deg2rad(theta)
            kernel = cv2.getGaborKernel((15,15), sigma=3,
theta=theta_rad,
```

```
                                            lambd=1.0/freq, gamma=0.5,
psi=0)
            filtered = cv2.filter2D(roi, cv2.CV_32F, kernel)
            desc[f'gabor_f{freq}_t{theta}_mean'] =
np.mean(filtered)
            desc[f'gabor_f{freq}_t{theta}_std']  =
np.std(filtered)
    return desc

# ------------------------------------------------
# 6. Ejemplo: ROI central (100x100)
# ------------------------------------------------
h, w = img.shape
roi = img[h//2-50:h//2+50, w//2-50:w//2+50]

# Calcular todos los descriptores
stats = stats_descriptors(roi)
glcm  = glcm_descriptors(roi)
lbp   = lbp_descriptors(roi)
gabor = gabor_descriptors(roi)

# ------------------------------------------------
# 7. Mostrar resultados
# ------------------------------------------------
print("=== DESCRIPTORES DE TEXTURA (ROI central) ===")
print("\nEstadísticos (1er orden):")
for k, v in stats.items():
    print(f"  {k:6}: {v:8.3f}")

print("\nGLCM (promedio sobre 4 ángulos):")
for k, v in glcm.items():
    print(f"  {k:12}: {v:8.4f}")

print(f"\nLBP (uniform, P=8,R=1): histograma de
{len(lbp['lbp_hist'])} bins")
print("  Primeros valores:", lbp['lbp_hist'][:5].round(4))

print(f"\nGabor (2 frec, 4 orient): {len(gabor)} valores")
for k in list(gabor.keys())[:6]:
    print(f"  {k}: {gabor[k]:.3f}")

# ------------------------------------------------
# 8. Visualización
# ------------------------------------------------
fig, ax = plt.subplots(2, 3, figsize=(15, 8))

ax[0,0].imshow(img, cmap='gray')
ax[0,0].set_title('Imagen Original')
ax[0,0].axis('off')

ax[0,1].imshow(roi, cmap='gray')
```

```
rect = plt.Rectangle((45,45), 10, 10, linewidth=2, edgecolor='r',
facecolor='none')
ax[0,1].add_patch(rect)
ax[0,1].set_title('ROI (100x100)')

# GLCM (primer ángulo)
glcm_vis = graycomatrix(roi, [1], [0], levels=16, symmetric=True,
normed=True)[:, :, 0, 0]
ax[0,2].imshow(glcm_vis, cmap='jet')
ax[0,2].set_title('GLCM (d=1, 0°)')

# LBP
lbp_map = local_binary_pattern(roi, 8, 1, 'uniform')
ax[1,0].imshow(lbp_map, cmap='gray')
ax[1,0].set_title('Mapa LBP')

# Gabor (ejemplo)
kernel = cv2.getGaborKernel((15,15), 3, np.deg2rad(0), 10, 0.5, 0)
gabor_resp = cv2.filter2D(roi, cv2.CV_32F, kernel)
ax[1,1].imshow(gabor_resp, cmap='gray')
ax[1,1].set_title('Respuesta Gabor (0°, f=0.1)')

ax[1,2].axis('off')

plt.tight_layout()
plt.show()
```

4. Tabla de Descriptores (Resumen)

Descriptor	# Valores	Invariante a	Aplicación
Estadísticos (1er orden)	4	Luz	Detección de brillo
GLCM (Haralick)	6–24	Ángulo (si se promedia)	Clasificación de tejidos
LBP	10 (uniforme)	Rotación, luz	Reconocimiento facial
Gabor	8–32	Escala/orientación	Detección de patrones periódicos
HOG	9–36 por celda	Iluminación	Detección de peatones

5. Ejemplo de Salida (valores típicos)

text

```
=== DESCRIPTORES DE TEXTURA (ROI central) ===

Estadísticos (1er orden):
  mean  :   128.450
  var   :  1456.200
  skew  :     0.120
```

```
  kurt  :     1.850

GLCM (promedio sobre 4 ángulos):
  contrast    :   0.8500
  dissimilarity:   0.3200
  homogeneity :   0.7800
  energy      :   0.4100
  correlation :   0.9200
  ASM         :   0.1680

LBP (uniform, P=8,R=1): histograma de 10 bins
  Primeros valores: [0.12  0.08  0.15  0.09  0.11 ...]

Gabor (2 frec, 4 orient): 16 valores
  gabor_f0.1_t0_mean: 12.340
  gabor_f0.1_t0_std :  8.210
  ...
```

6. Recomendaciones Prácticas

Situación	Descriptor recomendado
Textura direccional (madera, tela)	**GLCM** (4 ángulos)
Textura fina, isótropa (arena, piel)	**LBP uniforme**
Patrones periódicos (rejillas, ondas)	**Gabor**
Clasificación robusta (SVM, RF)	**Combinar GLCM + LBP**
Tiempo real	**LBP** (rápido)