Ejercicio Práctico: Cálculo de características por SDH

Calcular los histogramas de sumas y diferencias para 12 ventanas iniciando en tamaño 3 y avanzando de 2 en 2 hasta 25. Para cada tamaño de ventana calcule 7 características usando estos histogramas: media, varianza, correlación, contraste, homogeneidad, cluster de sombra, cluster de prominencia.

Calcula estas características del dataset compartido de los cerámicos y entrega tu código y el archivo csv con las características ya calculadas.

```
import numpy as np
import imageio as io
import os
import cv2
import pandas as pd
from tqdm import tqdm
from skimage.feature import local_binary_pattern
from skimage.feature import graycomatrix, graycoprops


#parametros lbp
radio = 1
n_puntos = 8*radio
METHOD  = 'uniform'
tam_ventana = [3,5,7,9,11,13,15,17,19,21,23,25] #3x3
#len(tam_ventana)
```

```
def histogramas_suma_diferencia(im, dx=1, dy=0):
    """Calcula los histogramas de suma y diferencia
    para la imagen im con desplazamiento (dx, dy)"""
    im = im.astype(np.int32)
    h, w = im.shape

    # Píxeles desplazados (validos)
    A = im[:h-dy, :w-dx]
    B = im[dy:, dx:]

    suma = A + B
    diferencia = A - B
```

```python
    # Histogramas normalizados
    max_sum = 2 * 255  # para imágenes 8-bit
    max_diff = 255     # diferencia va de -255 a 255

    hist_suma, _ =
        np.histogram(suma, bins=2*max_sum+1, range=(0, 2*max_sum))
    hist_diff, _ =
        np.histogram(diferencia, bins=2*max_diff+1, range=(-max_diff, max_diff))

    hist_suma = hist_suma / hist_suma.sum()
    hist_diff = hist_diff / hist_diff.sum()

    return hist_suma, hist_diff
```

```python
def caracteristicas_histogramas(hist_suma, hist_diff):
    bins_sum = np.arange(len(hist_suma))
    bins_diff = np.arange(-((len(hist_diff)-1)//2), ((len(hist_diff)-1)//2)+1)
    media_suma = np.sum(bins_sum * hist_suma)
    var_suma = np.sum((bins_sum - media_suma)**2 * hist_suma)
    media_diff = np.sum(bins_diff * hist_diff)
    var_diff = np.sum((bins_diff - media_diff)**2 * hist_diff)
    correlacion = (var_suma - var_diff) / (var_suma + var_diff + 1e-10)
    contraste = var_diff
    homogeneidad = np.sum(hist_diff / (1 + np.abs(bins_diff)))
    sombra = np.sum((bins_diff - media_diff)**3 * hist_diff)
    prominencia = np.sum((bins_diff - media_diff)**4 * hist_diff)
    return {
        "media": media_suma,
        "varianza": var_suma,
        "correlacion": correlacion,
        "contraste": contraste,
        "homogeneidad": homogeneidad,
        "cluster_sombra": sombra,
        "cluster_prominencia": prominencia
    }
```

```python
def extraer_caracteristicas(lbp_img, ventana_size):
    h, w = lbp_img.shape
    step = ventana_size
    caracteristicas_acum =
        {k: [] for k in [
              "media",
              "varianza",
              "correlacion",
              "contraste",
              "homogeneidad",
              "cluster_sombra",
              "cluster_prominencia"
        ]}
    for y in range(0, h - ventana_size + 1, step):
        for x in range(0, w - ventana_size + 1, step):
            ventana = lbp_img[y:y+ventana_size, x:x+ventana_size]
            hist_suma, hist_diff = histogramas_suma_diferencia(ventana)
            car = caracteristicas_histogramas(hist_suma, hist_diff)
            for key in car:
                caracteristicas_acum[key].append(car[key])

    # Promediar características por ventana
    caracteristicas_mean =
    {k: np.mean(v)
        if len(v)>0 else 0
            for k,v in caracteristicas_acum.items()
    }
    return caracteristicas_mean
```

```
def calc_lbp(img_gray):
    return local_binary_pattern(img_gray, n_puntos, radio, METHOD).astype(np.uint8)
```

```
# Parámetros
carpeta = "../Tarea1/dataset_ceramicos_recortes"
tam_ventana = list(range(3, 26, 2))  # 3,5,...,25

# Guardaremos filas como diccionarios (una por imagen)
filas = []

for root, dirs, files in os.walk(carpeta):
    for file in tqdm(files):
        if file.endswith(".tif") and "Falla" in file:
            ruta = os.path.join(root, file)
            img = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE)
            #img = cv2.resize(img, (30,30))
            lbp = calc_lbp(img)
            clase = os.path.basename(os.path.dirname(root))  # Extraer clase del nombre de la car
            fila = {"imagen": file, "clase": clase}
            for size in tam_ventana:
                caracteristicas = extraer_caracteristicas(lbp, size)
                for key, val in caracteristicas.items():
                    colname = f"{key}_w{size}"
                    fila[colname] = val

            filas.append(fila)

# Convertimos a DataFrame y guardamos
df = pd.DataFrame(filas)
df.to_csv("caracteristicas_ceramicos.csv", index=False)
print("Archivo CSV generado con características.")
```

```
0it [00:00, ?it/s]
```

```
0it [00:00, ?it/s]
100%|      | 10/10 [00:00<00:00, 17.92it/s]
100%|      | 10/10 [00:00<00:00, 19.29it/s]
100%|      | 10/10 [00:00<00:00, 19.68it/s]
100%|      | 10/10 [00:00<00:00, 19.07it/s]
100%|      | 10/10 [00:00<00:00, 18.69it/s]
100%|      | 10/10 [00:00<00:00, 18.01it/s]
100%|      | 10/10 [00:00<00:00, 18.19it/s]
100%|      | 10/10 [00:00<00:00, 18.57it/s]
100%|      | 10/10 [00:00<00:00, 19.05it/s]
100%|      | 10/10 [00:00<00:00, 18.75it/s]
0it [00:00, ?it/s]
100%|      | 10/10 [00:00<00:00, 18.29it/s]
100%|      | 10/10 [00:00<00:00, 19.29it/s]
100%|      | 10/10 [00:00<00:00, 18.83it/s]
100%|      | 10/10 [00:00<00:00, 19.38it/s]
100%|      | 10/10 [00:00<00:00, 18.91it/s]
100%|      | 10/10 [00:00<00:00, 19.36it/s]
100%|      | 10/10 [00:00<00:00, 18.73it/s]
100%|      | 10/10 [00:00<00:00, 18.90it/s]
100%|      | 10/10 [00:00<00:00, 18.66it/s]
100%|      | 10/10 [00:00<00:00, 18.63it/s]
0it [00:00, ?it/s]
100%|      | 10/10 [00:00<00:00, 17.92it/s]
100%|      | 10/10 [00:00<00:00, 18.66it/s]
100%|      | 10/10 [00:00<00:00, 18.50it/s]
100%|      | 10/10 [00:00<00:00, 18.68it/s]
100%|      | 10/10 [00:00<00:00, 19.33it/s]
100%|      | 10/10 [00:00<00:00, 18.72it/s]
100%|      | 10/10 [00:00<00:00, 18.85it/s]
100%|      | 10/10 [00:00<00:00, 18.99it/s]
100%|      | 10/10 [00:00<00:00, 18.86it/s]
100%|      | 10/10 [00:00<00:00, 19.10it/s]
0it [00:00, ?it/s]
100%|      | 10/10 [00:00<00:00, 18.44it/s]
100%|      | 10/10 [00:00<00:00, 19.22it/s]
100%|      | 10/10 [00:00<00:00, 18.76it/s]
100%|      | 10/10 [00:00<00:00, 18.51it/s]
100%|      | 10/10 [00:00<00:00, 19.12it/s]
100%|      | 10/10 [00:00<00:00, 18.46it/s]
100%|      | 10/10 [00:00<00:00, 19.01it/s]
100%|      | 10/10 [00:00<00:00, 18.64it/s]
100%|      | 10/10 [00:00<00:00, 18.79it/s]
100%|      | 10/10 [00:00<00:00, 18.41it/s]

Archivo CSV generado con características.
```