REPORT: IMAGE CLASSIFICATION

Atanda Abdullahi Adewale

Introduction: This exercise obtain images from Brodatz images dataset, Extracts and store their Fourier Transform (M1) and Autocorrelation (M2) descriptors or parameter values to represent each family. New images are labelled and classified according to 1NN distance measured in the 2D parameter space.

Fourier Transform Descriptors:

Magnitude spectrum: The magnitude of the FFT coefficients provides information about the frequency content of the image. This can be used to detect features such as edges and textures.

Power spectrum: The power spectrum is the magnitude spectrum squared, and represents the energy distribution in the frequency domain.

Phase spectrum: The phase of the FFT coefficients detect patterns and structures in the image.

Fourier descriptors: These are coefficients obtained by performing Fourier analysis on the contour of the image, and can be used to represent the shape of the object.

Auto-correlation Descriptors:

Mean: The mean value of the auto-correlation function can represent the overall brightness of the image.

Maximum: The maximum value of auto-correlation function can be used to represent the periodicity of the image.

Full-width half-maximum (FWHM): The FWHM of the auto-correlation function can be used to represent the size or scale of the features in the image.

Co-occurrence matrix: The autocorrelation function can be used to construct a co-occurrence matrix, which describes the spatial relationships between pairs of pixels in the image. This can be used to extract features such as texture and shape.

METHODS

1. Obtain images from Brodatz Dataset (1 image represent a family)

2. Compute a function to calculate Distance between the center of the FFT magnitude spectrum and its peak frequency

```
def compute_distancecenter2peak(gray):
    '''
       Distance between the center of the FFT magnitude spectrum and
       its peak frequency
    '''
    N,M=gray.shape
    cx=M//2
    cy=N//2
    f = np.fft.fft2(gray)
    fshift = np.fft.fftshift(f)
    magnitude_spectrum = np.abs(fshift)
    magnitude_spectrum[cy-K:cy+K,cx-int(M/N)*K:cx+int(M/N)*K]=0
    max_value = np.max(magnitude_spectrum)
    max_indices = np.argwhere(magnitude_spectrum == max_value)
    center=np.array([cy,cx])
    manhattan=max_indices - center
    euclidean=[np.linalg.norm(i) for i in manhattan]
    distancecenter2peak = np.mean(euclidean)
    return distancecenter2peak
```

3. Compute a function to calculate the Mean of autocorrelation of an image: using inverse FFT of the product of its Fourier transform and its complex conjugate

```
def compute_autocorr_mean(gray):
    '''
        autocorrelation of an image can be computed as the inverse Fourier transform
        of the product of its Fourier transform and its complex conjugate
    '''
    f = np.fft.fft2(gray)
    f_conj = np.conj(f)
    magnitudeconvolution = np.abs(np.fft.ifft2(f * f_conj))
    mean = np.mean(magnitudeconvolution) / (1e+8) #1e+8) is the scale factor
    return mean
```

4. Define a function to read all image from a (./data) folder consisting each family of Brodatz images

```
def getimagename():
    '''
        Reads all filenames in the directory .\data,
        checks if each file is a JPEG image by its extension, and
        returns a list of filenames for all JPEG images in the directory.
    '''
    name_list=[]
    path_list = os.listdir(r'./data')
    for filename in path_list:
        if os.path.splitext(filename)[1] == '.jpg':
            name_list.append(filename)
    return name_list
```

5. Append feature extracted from FFT and Autocorrelation of each image to M1 and M2 list respectively

```
# Initialize two lists to store feature vectors
m1 = []
m2 = []

# Loop over all the image names
for i in range(len(name_list)):
    # Read the image
    img = cv2.imread(os.path.join(r'./data', name_list[i]))

    # Convert the image to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Compute the distance between the center of the FFT magnitude spectrum and its peak frequency
    distancecenter2peak = compute_distancecenter2peak(gray)

    # Compute the mean of the autocorrelation of the image
    mean = compute_autocorr_mean(gray)

    # Append the features to the lists
    m1.append(distancecenter2peak)
    m2.append(mean)
```

6. Create a 2d array consisting both parameters of [M1, M2] for all image families

```
# Create a list of numpy arrays for all images' features
mdata = [np.array([m1[i], m2[i]]) for i in range(len(m1))]
print('\n m1, m2 list of features in Brodatz image used]: \n',mdata)
```

```
m1, m2 list of features in Brodatz image used]:
[array([ 71.02816343, 347.49021964]), array([89.        , 32.52050438]), array([ 81.0246876 , 118.72820447]), array([ 90
```

...

4. Create a 2d array consisting both parameters of [M1, M2] for the new Test Image

```
# Create a numpy array of the new image's features
mnew = np.array([distancecenter2peaknew, meannew])
print(mnew)
```

Example:

```
[93.         43.0669856]
```

5. Get the Labels from each image family

```
# Define the labels for each class
labels = [os.path.splitext(name)[0] for name in name_list]
print("image family used", labels)
```

```
image family used ['water', 'bubbles', 'sand', 'wall', 'grass']
```

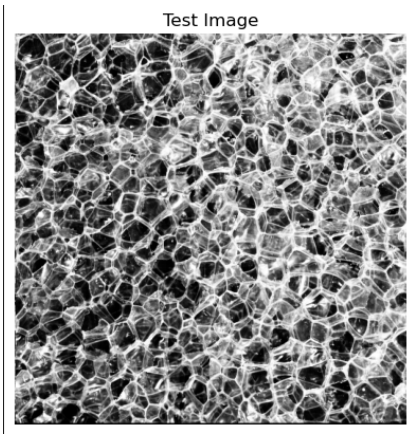
6. Calculate 1NN distance between test image to each families and classify :

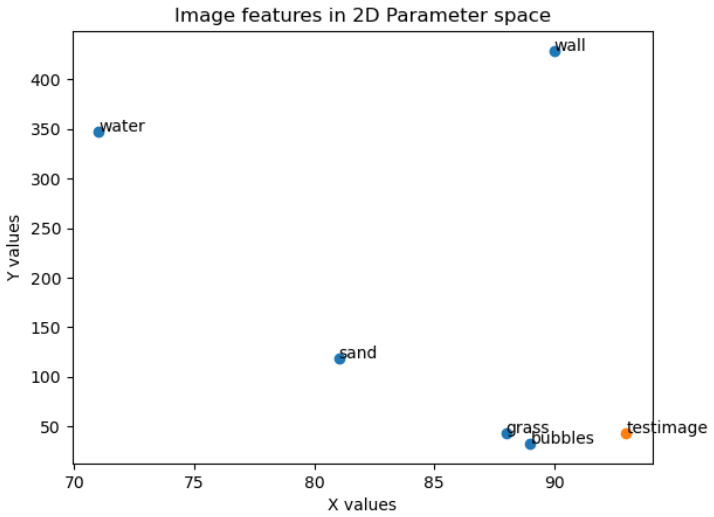
```
# Compute the distances between the new image's features and all other images' features
distancelist = [np.linalg.norm(melement - mnew) for melement in mdata]
print('\n distance of the Test image feature to the other families in 2d parameter space : \n', distancelist)

# Find the index (variable) distancelist: list lest distance to the new image
i = np.argwhere(distancelist == np.min(distancelist))[0, 0]
```

```
distance of the Test image feature to the other families in 2d parameter space :
[305.2151159875356, 11.279550789197856, 76.60305574177848, 385.809248263098, 5.01122733647869]
```

7. Plot 2d Scatter Graph of point distances (M1,M2) of all images and Test Image





8. Display result of classification i.e which family Test Image belongs to

```
# Print the class label of the image that has the smallest distance to the new image
print(f'\n \n The image belongs to  {labels[i]} family')
```

```
The image belongs to  grass family
```

CONCLUSION:

After series of test, this classification algorithm works but not always. To produce better accuracy, FFT and Autocorrelation parameter can be changed and more images should be employed to represent each family of images instead of Just one used