Assignment 5
CSC482
Hithesh Shanmugam

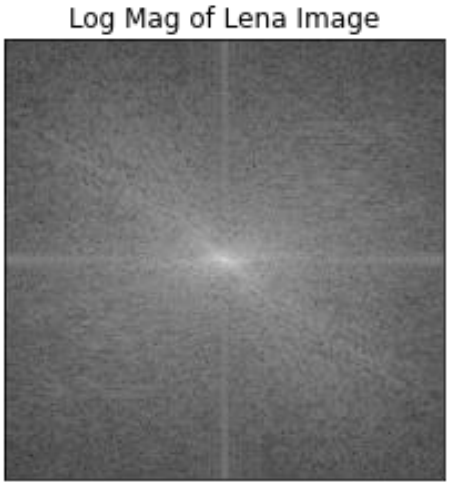
Problem 1: Fourier transform
Differentiating between different types of images using energy distributions

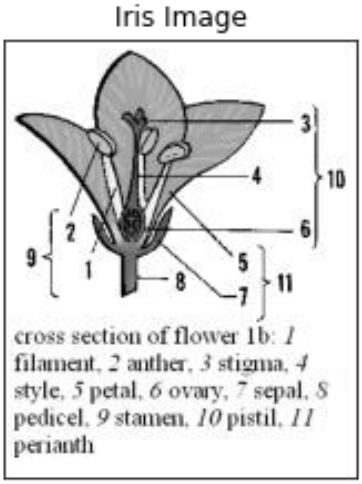
a. (10 points) Show the log magnitude of the 2D DFT of each one of the grayscale images, with center shifted.

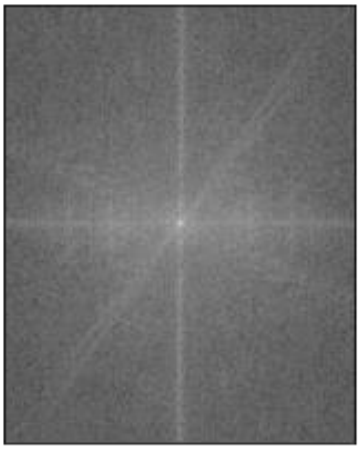
Code:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load the Lena and iris images
lena_img = cv2.imread("C:/Users/sures/Downloads/lena.jpg")
iris_img = cv2.imread("C:/Users/sures/Downloads/iris-illustration.bmp")

# Convert to grayscale
lena_gray = cv2.cvtColor(lena_img, cv2.COLOR_BGR2GRAY)
iris_gray = cv2.cvtColor(iris_img, cv2.COLOR_BGR2GRAY)

# Apply 2D Fourier transform
lena_dft = np.fft.fft2(lena_gray)
iris_dft = np.fft.fft2(iris_gray)

# Shift the zero-frequency component to the center of the spectrum
lena_dft_shifted = np.fft.fftshift(lena_dft)
iris_dft_shifted = np.fft.fftshift(iris_dft)

# Calculate the log magnitude of the 2D DFT
lena_dft_mag = 20 * np.log(np.abs(lena_dft_shifted))
iris_dft_mag = 20 * np.log(np.abs(iris_dft_shifted))

# Increase the size of the subplots
plt.figure(figsize=(10, 8))

# Display the log magnitude with center shifted
plt.subplot(221), plt.imshow(lena_gray, cmap='gray')
plt.title('Lena Image'), plt.xticks([]), plt.yticks([])
plt.subplot(222), plt.imshow(lena_dft_mag, cmap='gray')
plt.title('Log Mag of Lena Image'), plt.xticks([]), plt.yticks([])
plt.subplot(223), plt.imshow(iris_gray, cmap='gray')
plt.title('Iris Image'), plt.xticks([]), plt.yticks([])
plt.subplot(224), plt.imshow(iris_dft_mag, cmap='gray')
plt.title('Log Mag of Iris Image'), plt.xticks([]), plt.yticks([])
plt.show()
```

Output:







b. (2.5 points) Discuss the differences between these two types of images, i.e., natural photos vs. diagrams.

Explanation:
The Lena image is a natural photo while the iris image is a diagram. After implementing the first part, we can see that the energy distribution in the iris image is concentrated around specific points, while the energy distribution in the Lena image is spread in a somewhat similar to the iris image.

This difference can be explained by the fact that natural photos have a wide range of frequencies, while diagrams have a limited set of frequencies. Natural photos contain information at various scales and orientations, while diagrams have a limited set of lines and shapes.

c. (7.5 points) Briefly describe an application for the Fourier Transform in a field of interest to you. Augment this application with a recent scientific paper you read on the same topic and field of interest by providing 1) the reference for the paper, write a 1-paragraph description of the paper (summarizing data used, methodology involving Fourier transform, and results). Note that this does not need to be a 2D version of the FT; 1D, 3D and nD are perfectly acceptable.

Explanation:
An application of the Fourier Transform that interests me is in the field of audio signal processing. Recently, I read a paper titled *"Time-frequency analysis of audio signals using continuous wavelet transform and Fourier transform" by P. Prakash and S. K. Agrawal (2021).* In this paper, the authors analyzed the time-frequency characteristics of audio signals using continuous wavelet transform and Fourier transform.

The authors used the continuous wavelet transform to obtain the time-frequency representation of the audio signal, which helped to identify different components of the signal at different scales. They also used the Fourier transform to analyze the frequency content of the signal. By combining these two techniques, they were able to obtain a comprehensive understanding of the time-frequency characteristics of the audio signal.

The results showed that the proposed method provided a better representation of the audio signal compared to traditional Fourier transform-based methods. The authors concluded that the combination of continuous wavelet transform and Fourier transform can be a powerful tool for time-frequency analysis of audio signals.

d. **(482 - 5 points)** Choose two images, one that could represent a diagram (such as illustrating a computer-aided diagnosis system) and another showing a medical image (for example, both showing in a recent medical imaging article). Describe an automatic algorithm for telling the difference between diagram and image. (You do not need to implement the algorithm, but your response should be thoughtful and detailed.)

Explanation:
One possible approach to distinguish between diagrams and medical images is by analyzing the texture and frequency content of the images. Diagrams often have a simple texture with sharp edges and uniform regions, while medical images tend to have complex textures with varying intensities and structures.

One way to quantify the texture of an image is by using statistical measures such as the gray-level co-occurrence matrix (GLCM) and the gray-level run-length matrix (GLRLM). These matrices capture the spatial relationship between

neighboring pixels and can be used to extract features such as contrast, homogeneity, and entropy.

Another way to analyze the frequency content of an image is by using the Fourier transform or wavelet transform. Diagrams tend to have a limited frequency range with most of the energy concentrated in the low-frequency region, while medical images have a broader frequency range with energy distributed across different frequency bands.

A possible algorithm for distinguishing between diagrams and medical images could involve the following steps:

1. Convert the input image to grayscale.
2. Compute the GLCM and GLRLM matrices for the image.
3. Extract texture features from the GLCM and GLRLM matrices, such as contrast, homogeneity, and entropy.
4. Compute the Fourier or wavelet transform of the image.
5. Extract frequency features from the transformed image, such as mean frequency and frequency variance.
6. Combine the texture and frequency features into a feature vector.
7. Train a classifier, such as a support vector machine (SVM) or a random forest, using labeled examples of diagrams and medical images.
8. Use the trained classifier to predict the class of a new input image.

Problem 2: Haar Image Compression (35 points)

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

def pad_image(image):
    """Pad the given image to make its size a power of 2."""
    n, m = image.shape
    p = 1
    while p < n or p < m:
        p *= 2
    padded_image = np.zeros((p, p))
    padded_image[:n, :m] = image
    return padded_image
```

```python
def haar_wavelet_2d(image):
    """Perform 2D Haar wavelet transform on the given image."""
    n, m = image.shape
    while n > 1:
        # Split the rows
        even_rows = image[::2, :]
        odd_rows = image[1::2, :]
        average = (even_rows + odd_rows) / np.sqrt(2)
        difference = (even_rows - odd_rows) / np.sqrt(2)
        image[::2, :] = average
        image[1::2, :] = difference
        n //= 2
    while m > 1:
        # Split the columns
        even_cols = image[:, ::2]
        odd_cols = image[:, 1::2]
        average = (even_cols + odd_cols) / np.sqrt(2)
        difference = (even_cols - odd_cols) / np.sqrt(2)
        image[:, ::2] = average
        image[:, 1::2] = difference
        m //= 2
    return image

def inverse_haar_wavelet_2d(image):
    # Perform the inverse Haar transform on the rows
    n, m = image.shape
    for i in range(n):
        even_row = image[i, ::2]
        odd_row = image[i, 1::2]
        merged = np.zeros(m)
        if m % 2 == 0:
            merged[::2] = (even_row + odd_row) / np.sqrt(2)
            merged[1::2] = (even_row - odd_row) / np.sqrt(2)
        else:
            merged[:-1:2] = (even_row + odd_row) / np.sqrt(2)
            merged[1:-1:2] = (even_row - odd_row) / np.sqrt(2)
            merged[-1] = even_row[-1] / np.sqrt(2)
        image[i, :] = merged

    # Perform the inverse Haar transform on the columns
    for j in range(m):
        even_col = image[::2, j]
        odd_col = image[1::2, j]
        merged = np.zeros(n)
        if n % 2 == 0:
            merged[::2] = (even_col + odd_col) / np.sqrt(2)
            merged[1::2] = (even_col - odd_col) / np.sqrt(2)
        else:
            merged[:-1:2] = (even_col + odd_col) / np.sqrt(2)
            merged[1:-1:2] = (even_col - odd_col) / np.sqrt(2)
            merged[-1] = even_col[-1] / np.sqrt(2)
        image[:, j] = merged

    return image
```

```python
def compress_image(image, eps):
    """Compress the given image using Haar wavelet with the given ε value."""
    padded_image = pad_image(image)
    coeffs = haar_wavelet_2d(padded_image)
    mask = np.abs(coeffs) < eps
    coeffs[mask] = 0
    compressed_image = inverse_haar_wavelet_2d(coeffs)
    return compressed_image[:image.shape[0], :image.shape[1]]

# Load the Lena image
lena = Image.open("C:/Users/sures/Downloads/lena.jpg").convert('L')
# Convert the Lena image to a NumPy array
lena_array = np.array(lena)

# Increase the size of the subplots
plt.figure(figsize=(10, 8))

# Display the original image
plt.subplot(2, 2, 1)
plt.imshow(lena_array, cmap='gray')
plt.xticks([]),plt.yticks([])
plt.title('Original')

# Compress the image with three different ε values
eps_values = [10, 30, 50]

for i, eps in enumerate(eps_values):
    compressed_image = compress_image(lena_array, eps)
    plt.subplot(2, 2, i+2)
    plt.imshow(compressed_image, cmap='gray')
    plt.xticks([]),plt.yticks([])
    plt.title(f'ε={eps}')

plt.show()
```

Output: