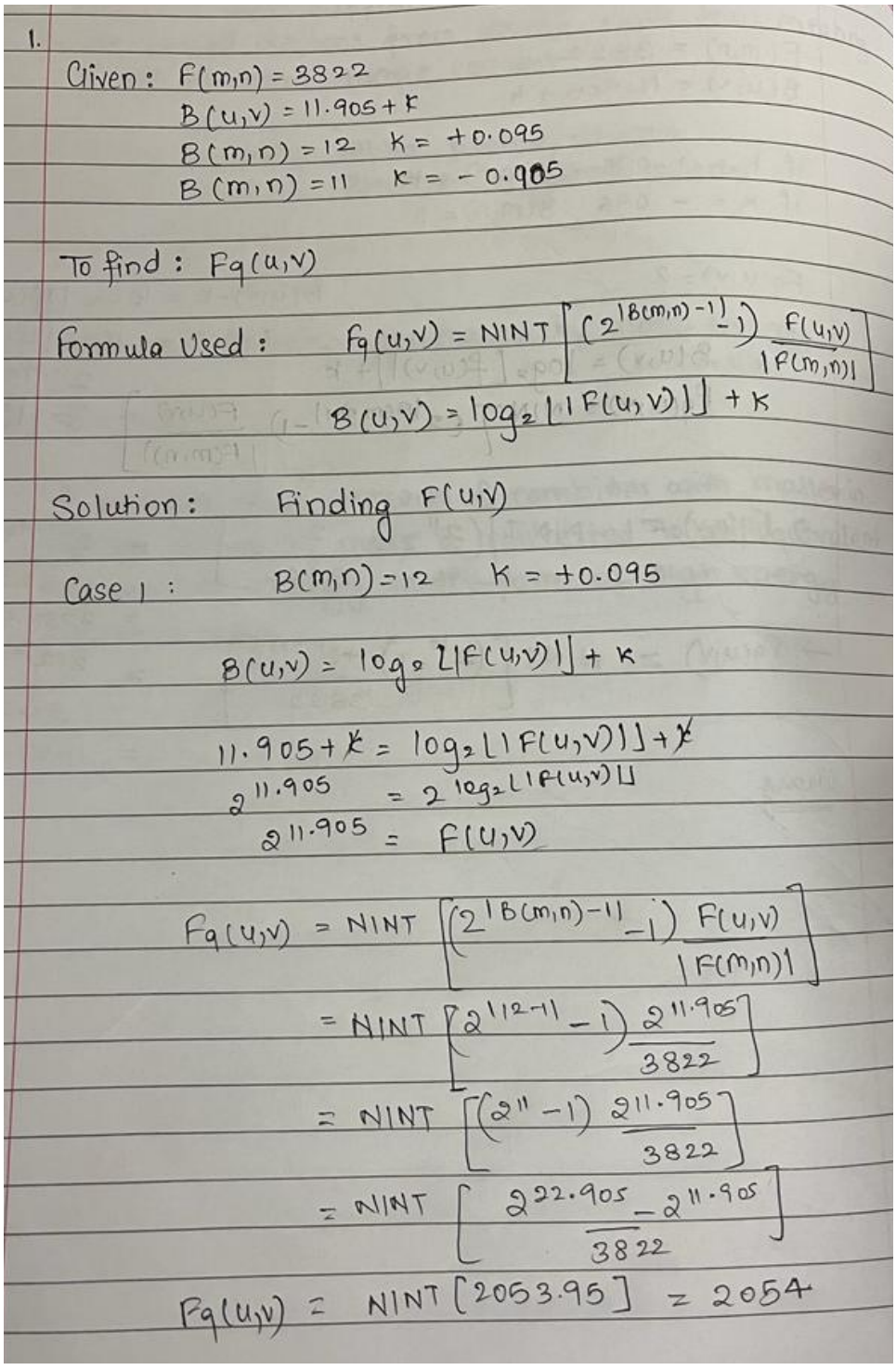
Assignment 2

1.



1.

Given: $F(m,n) = 3822$

$B(u,v) = 11.905 + K$

$B(m,n) = 12 \quad K = +0.095$

$B(m,n) = 11 \quad K = -0.905$

To find: $Fq(u,v)$

Formula Used: $Fq(u,v) = NINT\left[\left(2^{|B(m,n)-11|} - 1\right)\frac{F(u,v)}{|F(m,n)|}\right]$

$B(u,v) = \log_2[|F(u,v)|] + K$

Solution: Finding $F(u,v)$

Case 1: $B(m,n) = 12 \quad K = +0.095$

$B(u,v) = \log_2[|F(u,v)|] + K$

$11.905 + K = \log_2[|F(u,v)|] + K$

$2^{11.905} = 2^{\log_2[|F(u,v)|]}$

$2^{11.905} = F(u,v)$

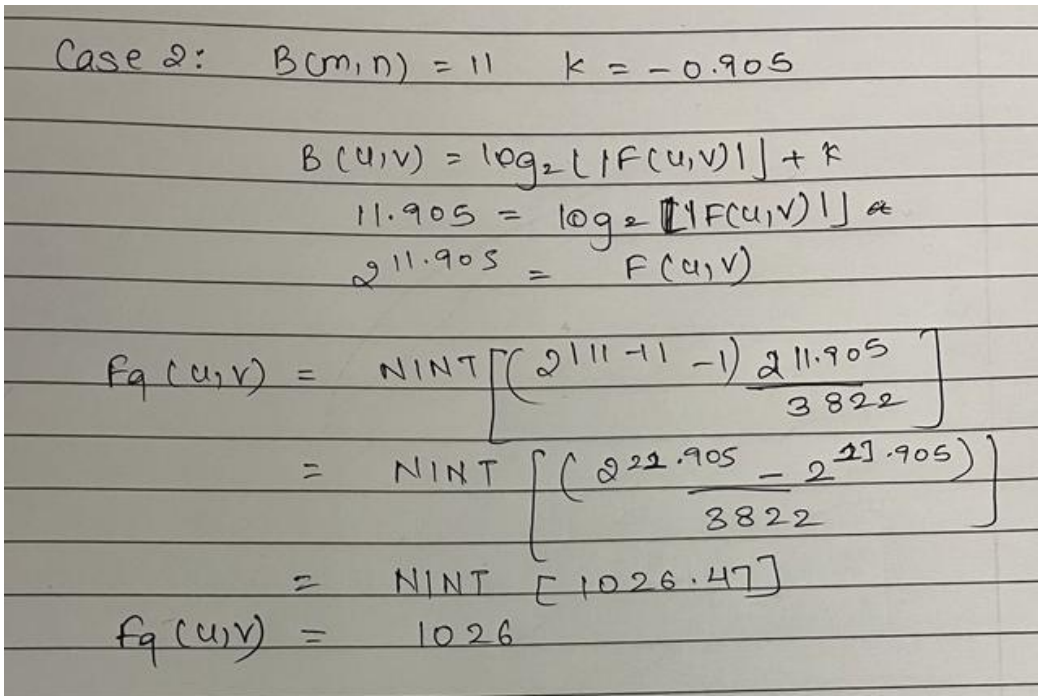
$Fq(u,v) = NINT\left[\left(2^{|B(m,n)-11|} - 1\right)\frac{F(u,v)}{|F(m,n)|}\right]$

$= NINT\left[\left(2^{|12-11|} - 1\right)\frac{2^{11.905}}{3822}\right]$

$= NINT\left[\left(2^{1} - 1\right)\frac{2^{11.905}}{3822}\right]$

$= NINT\left[\frac{2^{22.905}}{3822} - \frac{2^{11.905}}{3822}\right]$

$Fq(u,v) = NINT[2053.95] = 2054$



F_q(u,v) = 2054 for B(m,n)=12
F_q(u,v) = 1026 for B(m,n)=11

2. Cosine Transform Compression:

Full Frame	Block
Advantage - Simple to store and transmit due to the whole image being compressed as one unit - Preserves the image better as stores it as a whole.	**Advantage** - It would allow threading and parallel processing, speeding up the process. - Different compression ratios can be applied to different parts.
Disadvantage - Loss of data in one part of the frame can affect the entire image. - Can introduce edge artifacts.	**Disadvantage** - May lose some fine details, especially at block boundaries - Requires additional information to specify block arrangement and order during compression and decompression.

3. Summarize the concept of image compression using the wavelet transformation.

→ A basis wavelet function is derived from a Mother Wavelet function by Dilation and Translation. It has both the Spatial and Frequency domain.

In wavelet transformation, we decompose a Signal into a Series of Smooth Signals and Detailed Signals at Different Resolution Levels.

Basis Function:- $\Psi_{a,b}(a,b) = \frac{1}{\sqrt{a}} \Psi(\frac{x-b}{a})$

….[x-b is the translation, and /a is the scaling component]

Wavelet Transformation:- $F_w(a,b) = \frac{1}{\sqrt{a}} \int_{\infty}^{-\infty} f(x) \Psi^*(\frac{x-b}{a})\, dx$

So, it decomposes a signal into low and high-frequency level components, which means low for smooth signal and high for detailed signal. This is done by -

a) Sampling at every other data point (based on the Nyquist rule of sampling).
b) Running it through a high pass filter and a low pass filter based on the wavelet transform function.
c) Then, these 2 signals are split into 2 again.

Equations for the low pass filter and high pass filter are as follows:-

Smooth Signal:- $f_{m+1}(n) = \sum_k h(2n-k) f_m(k)$

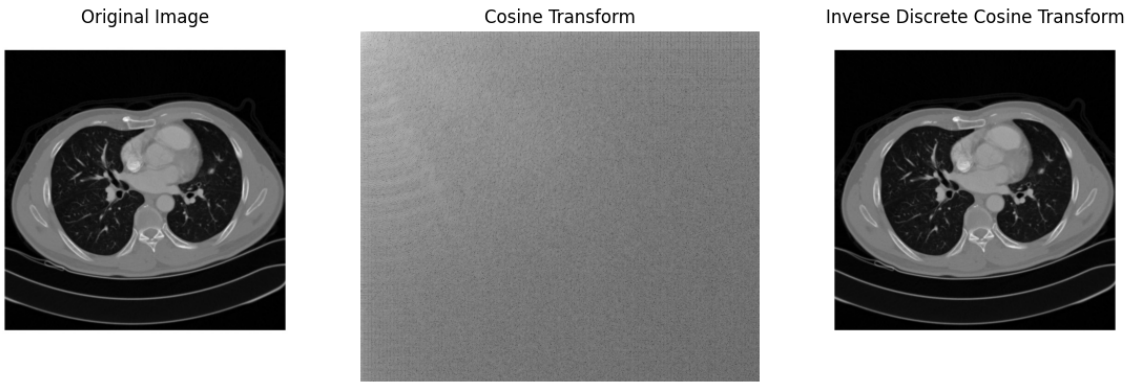
Detailed Signal:- $f'_{m+1}(n) = \sum_k g(2n-k) f_m(k)$

For the image compression process using wavelet transformation, we perform the forward 3D wavelet transform and then perform scalar quantization (Quantization of the coefficients). Finally, we do entropy coding (run-length and Huffman coding), resulting in a compressed image. For image decompression, we perform entropy decoding and then do scalar dequantization. Finally, we take the inverse 3D wavelet transform to get back the image.

4. Code:

```
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
from scipy.fftpack import dct, idct
from skimage.color import rgb2gray
from skimage.io import imread
from scipy.fft import fft2, fftshift, ifft2, ifftshift

# Load the image
image = Image.open('/content/lungs.jpg')

# Convert the image to grayscale and then to numpy array
gray_image = rgb2gray(imread('/content/lungs.jpg'))

# Perform 2D Discrete Cosine Transform (DCT)
cosine_transform = dct(dct(gray_image.T, norm='ortho').T, norm='ortho')
reconstructed = idct(idct(cosine_transform.T, norm='ortho').T, norm='ortho')


# Display the images
plt.figure(figsize=(12, 12))

plt.subplot(1, 3, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(np.log(np.abs(cosine_transform)))
plt.title('Cosine Transform')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(reconstructed)
plt.title('Inverse Discrete Cosine Transform')
plt.axis('off')


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

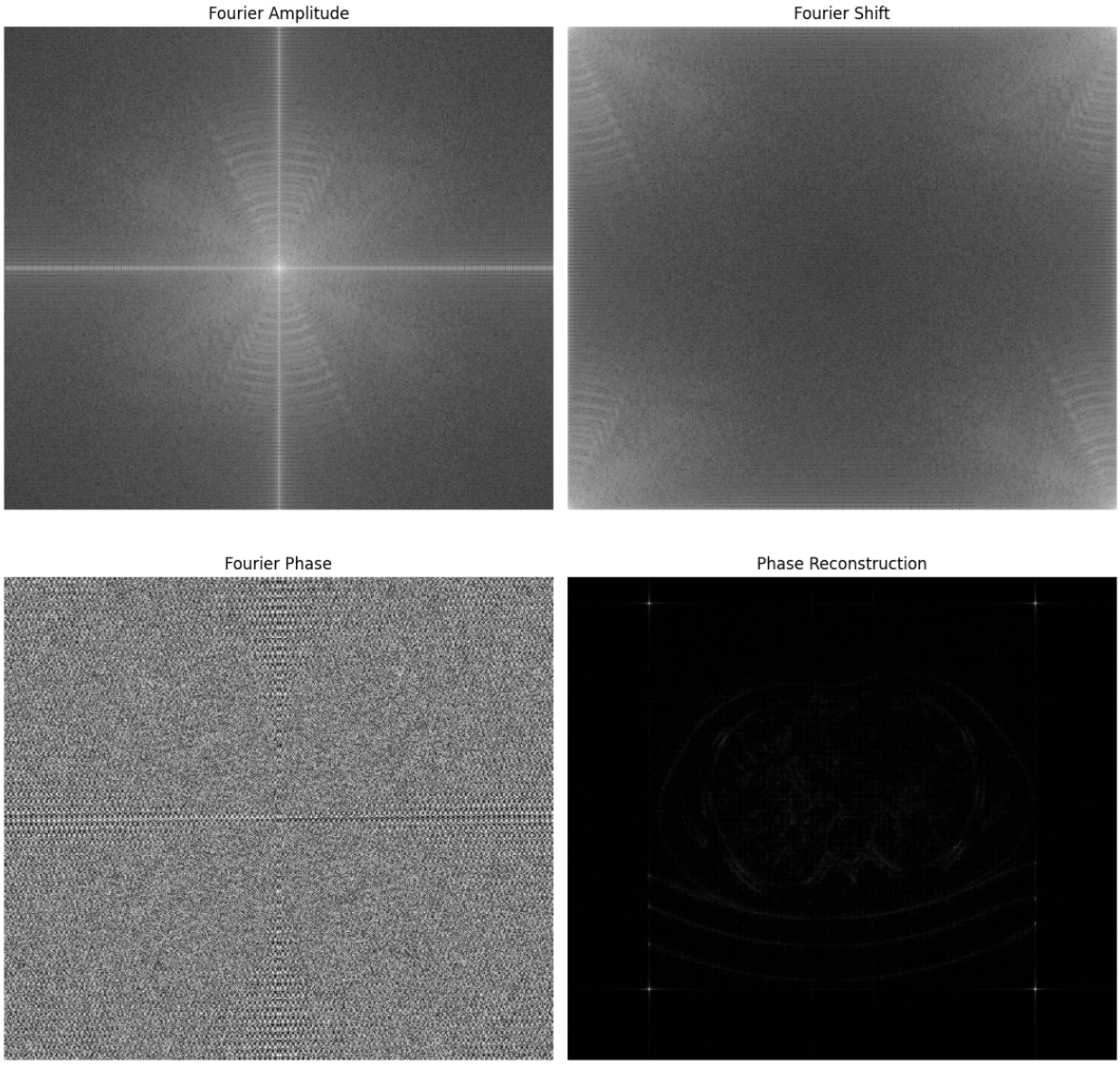
```
# Load the original image
original_image = Image.open('/content/lungs.jpg').convert('L')
original_array = np.array(original_image)

# Perform 2D Fourier Transform
fourier_transform = fftshift(fft2(original_array))

# Fourier Amplitude
fourier_amplitude = np.abs(fourier_transform)

# Fourier Phase
fourier_phase = np.angle(fourier_transform)

# Inverse Fourier Transform for Phase Reconstruction
reconstructed_phase = ifft2(ifftshift(np.exp(1j * fourier_phase)))

# Display the images
plt.figure(figsize=(12, 12))

plt.subplot(2, 2, 1)
plt.imshow(np.log(fourier_amplitude), cmap='gray')
plt.title('Fourier Amplitude')
plt.axis('off')

plt.subplot(2, 2, 2)
plt.imshow(np.log(np.abs(fftshift(fourier_transform))), cmap='gray')
plt.title('Fourier Shift')
plt.axis('off')

plt.subplot(2, 2, 3)
plt.imshow(fourier_phase, cmap='gray')
plt.title('Fourier Phase')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.imshow(np.abs(reconstructed_phase), cmap='gray')
plt.title('Phase Reconstruction')
plt.axis('off')

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



We are transforming the image from its spatial domain to its frequency domain. Each coefficient in the transformed image represents the contribution of a particular frequency component.

When you reconstruct the image using Inverse Discrete Cosine Transform we get an image very similar to the original image.

In the cosine transform image, we can see on the left side corner there are a few lines going through representing high-frequency components.



In the Fourier Amplitude Image, we see the strength of the corresponding frequency component. Higher amplitudes indicate more pronounced features. It represents the intensity of the different frequencies in the image. Therefore, it holds the geometrical structure of features in the image (i.e., changes in the spatial domain). The vertical, horizontal white lines along with the concentric circular lines starting from the bright center projected on the vertical white line, represent the high-frequency components. The rest of the image represents low-frequency components.

When we apply a Fourier shift, we notice a shift of brighter areas toward the corners of the image, indicating a transformation in the spatial domain due to the phase shift in the frequency domain. The corners with white lines indicate the high-frequency components. The rest of them are low-frequency components.

The Fourier Phase, on the other hand, represents the phase shift of each frequency component. It stores the information on the locations of the features of the image.

In the Phase Reconstruction Image, upon zooming in, faint traces of the original image's high-frequency components may be visible, showcasing how phase information is crucial for image reconstruction.

The phase information in the Fourier transform is a critical component, but the cosine transform doesn't provide phase information. In the Cosine transform, we observe horizontal and vertical patterns. In the Fourier transform magnitude, we see a more complex pattern of frequencies, including diagonal and circular patterns.

The Fourier transform provides a more comprehensive representation with amplitude and phase information. The Cosine transform simplifies the representation but may be less suitable for precise image reconstruction. The choice between the two depends on the specific requirements of the analysis or application.