Lab 04 Home tasks

In [50]:
```
#libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

HOME TASK 1

Task 2 Linear Filtering:

1. Implement a Gaussian blur filter using convolution for image smoothing.
2. Apply a Sobel filter to perform edge detection on a grayscale image.
3. Perform image sharpening using the Laplacian filter.
4. Implement a mean filter for noise reduction in an image.

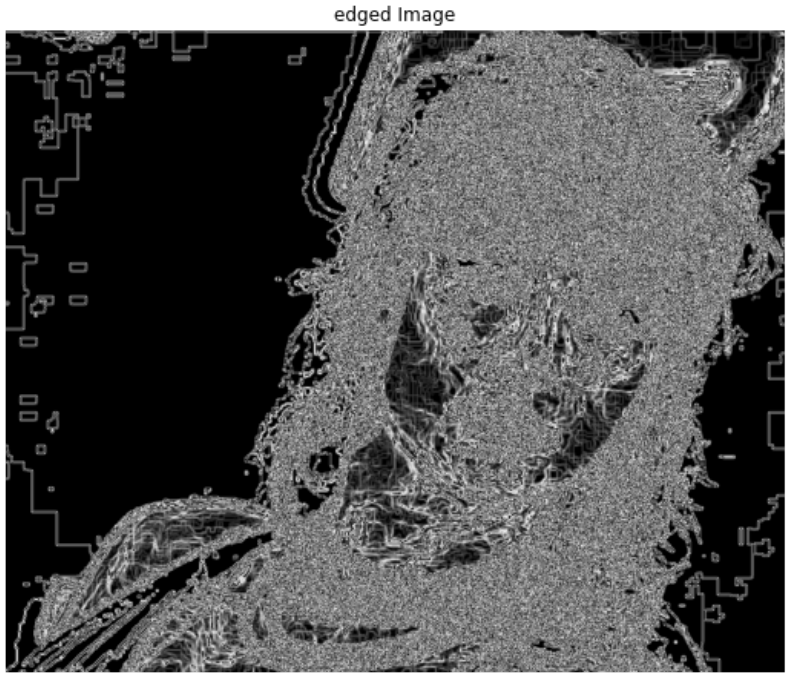
In [24]:
```
# Implement a Gaussian blur filter using convolution for image smoothing.
rgb_image = cv2.imread(r'D:\FastSemesters\semester7\Computer Vision\lab\4\mom.PNG')
sigma = 1.5
gaussian_filtered_image = cv2.GaussianBlur(rgb_image, (0, 0), sigma)
```

In [25]:

```
plt.figure(figsize=(20, 12))
plt.subplot(1, 2, 1)
plt.axis('off')
plt.imshow(cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(gaussian_filtered_image, cv2.COLOR_BGR2RGB))
plt.title('Smoothened Image')
plt.axis('off')
plt.show()
```

In [26]:
```
# Apply a Sobel filter to perform edge detection on a grayscale image.
image = cv2.imread(r'D:\FastSemesters\semester7\Computer Vision\lab\4\mom.PNG', cv2.IMREAD_GRAYSCALE)
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=5)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=5)
edge_image = cv2.magnitude(sobel_x, sobel_y)
```

In [27]:
```
plt.figure(figsize=(20, 12))
plt.subplot(1, 2, 1)
plt.axis('off')
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(edge_image.astype(np.uint8), cv2.COLOR_BGR2RGB))
plt.title('edged Image')
plt.axis('off')
plt.show()
```



In [31]:
```
# Perform image sharpening using the Laplacian filter.
laplacian = cv2.Laplacian(rgb_image, cv2.CV_64F)
laplacian_abs = cv2.convertScaleAbs(laplacian)
sharpened_image_laplacian = cv2.add(rgb_image, laplacian_abs)
sharpened_image_laplacian = np.clip(sharpened_image_laplacian, 0, 255).astype(np.uint8)
```

In [35]:
```
plt.figure(figsize=(20, 12))
plt.subplot(1, 2, 1)
plt.axis('off')
plt.imshow(cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')

plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(sharpened_image_laplacian, cv2.COLOR_BGR2RGB))
plt.title('Sharpened Image')
plt.axis('off')
plt.show()
```

In [36]:

```
# Implement a mean filter for noise reduction in an image.
mean_filtered_image = cv2.blur(rgb_image, (5, 5))

plt.figure(figsize=(20, 12))
plt.subplot(1, 2, 1)
plt.axis('off')
plt.imshow(cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')

plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(mean_filtered_image, cv2.COLOR_BGR2RGB))
plt.title('smoothened Image')
plt.axis('off')
plt.show()
```

HOME TASK 2

Task 4 Fourier Transformations:

1. Calculate the 1D Fourier Transform of a signal and visualize its magnitude and phase spectra.
2. Apply a 2D Fourier Transform to an image and display its magnitude spectrum.
3. Implement a high-pass filter in the frequency domain to emphasize edges in an image.
4. Perform image compression using the Fourier Transformation

Calculate the 1D Fourier Transform of a signal and visualize its magnitude and phase spectra.

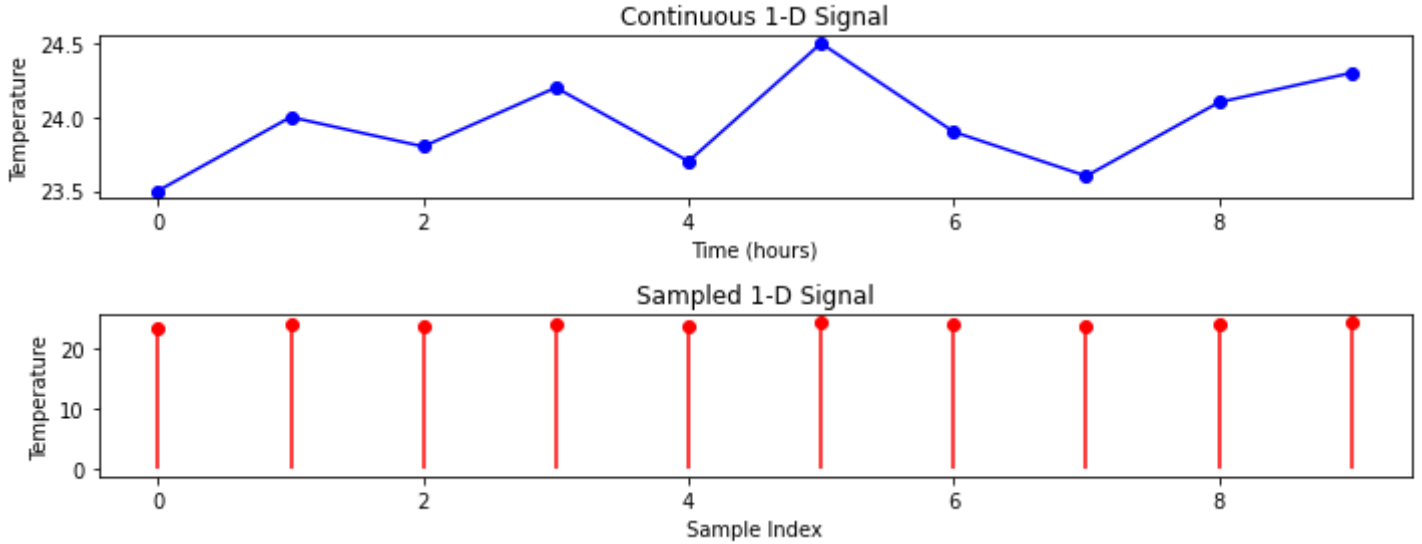
In [83]:

```
import numpy as np
import matplotlib.pyplot as plt
# Continuous signal (e.g., temperature readings)
continuous_signal = np.array([23.5, 24.0, 23.8, 24.2, 23.7, 24.5, 23.9, 23.6, 24.1, 24.3])
# Sampling at hourly intervals
sampled_indices = np.arange(0, len(continuous_signal))
sampled_signal = continuous_signal[sampled_indices]
# Create a time axis for plotting
time_axis = np.arange(0, len(continuous_signal))
```

In [84]:

```
plt.figure(figsize=(10, 4))
plt.subplot(2, 1, 1)
plt.plot(time_axis, continuous_signal, marker='o', linestyle='-', color='b')
plt.title("Continuous 1-D Signal")
plt.xlabel("Time (hours)")
plt.ylabel("Temperature")
plt.subplot(2, 1, 2)
plt.stem(sampled_indices, sampled_signal, basefmt=" ", markerfmt="ro", linefmt="-r")
plt.title("Sampled 1-D Signal")
plt.xlabel("Sample Index")
plt.ylabel("Temperature")

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



In [85]:

```
fourier_transform = np.fft.fft(sampled_signal)

# corresponding frequency values
N = len(sampled_signal)  # Number of data points
freq = np.fft.fftfreq(N)

# magnitude spectrum
magnitude_spectrum = np.abs(fourier_transform)

# phase spectrum
phase_spectrum = np.angle(fourier_transform)
```

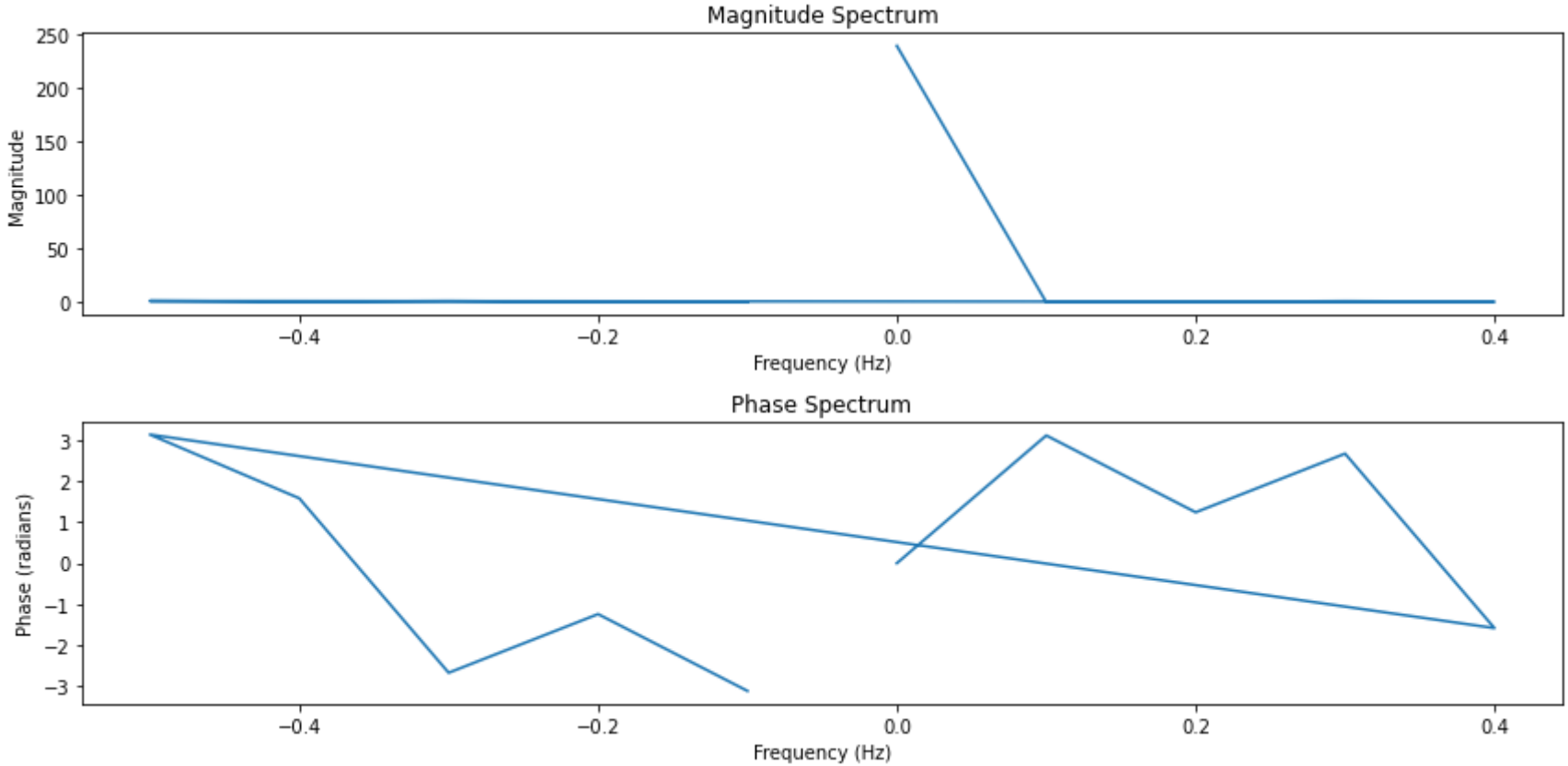
In [86]:

```
# Plot the magnitude spectrum
plt.figure(figsize=(12, 6))

plt.subplot(2, 1, 1)
plt.plot(freq, magnitude_spectrum)
plt.title('Magnitude Spectrum')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')

# Plot the phase spectrum
plt.subplot(2, 1, 2)
plt.plot(freq, phase_spectrum)
plt.title('Phase Spectrum')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Phase (radians)')

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


Magnitude Spectrum
Magnitude
250
200
150
100
50
0
−0.4
−0.2
0.0
0.2
0.4
Frequency (Hz)
Phase Spectrum
Phase (radians)
3
2
1
0
−1
−2
−3
−0.4
−0.2
0.0
0.2
0.4
Frequency (Hz)

Apply a 2D Fourier Transform to an image and display its magnitude spectrum.

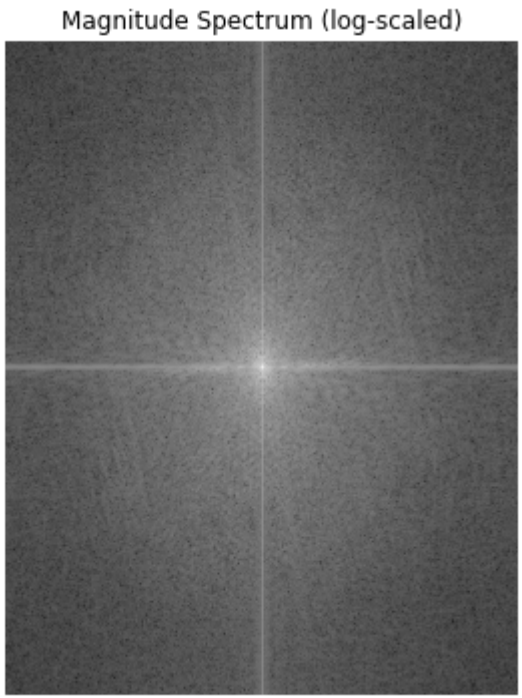
In [76]:

```
image = cv2.imread(r'D:\FastSemesters\semester7\Computer Vision\lab\4\marylin.PNG', cv2.IMREAD_GRAYSCALE)

# Perform the 2D Fourier Transform
fourier_transform = np.fft.fft2(image)
fourier_transform_shifted = np.fft.fftshift(fourier_transform)
magnitude_spectrum = np.log(np.abs(fourier_transform_shifted) + 1)

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1), plt.imshow(image, cmap='gray'), plt.title('Original Image'), plt.axis('off')
plt.subplot(1, 2, 2), plt.imshow(magnitude_spectrum, cmap='gray'), plt.title('Magnitude Spectrum (log-scaled)
plt.show()
```



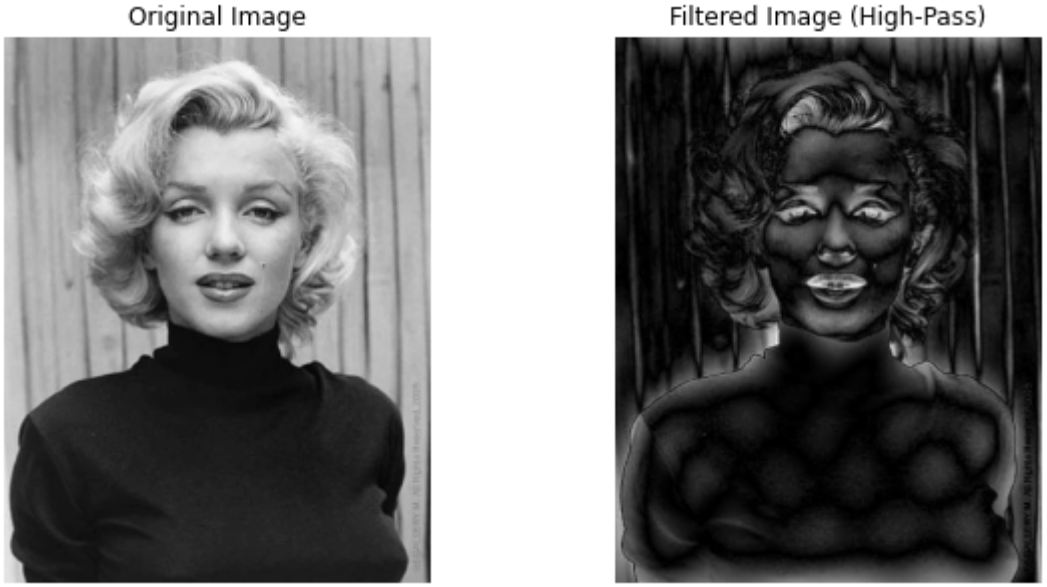
Implement a high-pass filter in the frequency domain to emphasize edges in an image.

In [77]:

```
# Create a high-pass filter in the frequency domain
rows, cols = image.shape
center_row, center_col = rows // 2, cols // 2
D = 5
high_pass_filter_mask = np.ones((rows, cols), dtype=np.uint8)
high_pass_filter_mask[center_row - D:center_row + D + 1, center_col - D:center_col + D + 1] = 0

# Apply the high-pass filter by multiplication in the frequency domain
filtered_image = fourier_transform_shifted * high_pass_filter_mask
filtered_image = np.abs(np.fft.ifft2(np.fft.ifftshift(filtered_image))).astype(np.uint8)

plt.figure(figsize=(10, 5))
plt.subplot(121), plt.imshow(image, cmap='gray'), plt.title('Original Image'), plt.axis('off')
plt.subplot(122), plt.imshow(filtered_image, cmap='gray'), plt.title('Filtered Image (High-Pass)'), plt.axis(
plt.show()
```



Perform image compression using the Fourier Transformation

In [94]:

```
image = cv2.imread(r'D:\FastSemesters\semester7\Computer Vision\lab\4\marylin.PNG', cv2.IMREAD_GRAYSCALE)

filtered_fourier_transform = np.fft.fft2(image)
threshold = 20000
filtered_fourier_transform[abs(filtered_fourier_transform) < threshold] = 0
compressed_image = np.abs(np.fft.ifft2(np.fft.ifftshift(filtered_fourier_transform)))

plt.figure(figsize=(10, 5))
plt.subplot(121), plt.imshow(image, cmap='gray'), plt.title('Original Image'), plt.axis('off')
plt.subplot(122), plt.imshow(compressed_image, cmap='gray'), plt.title('Compressed Image'), plt.axis('off')
plt.show()
```