Proyecto Final Guia de Laboratorio

Juan Camilo Martinez Ortea CC 1124855139

1. Introducción:

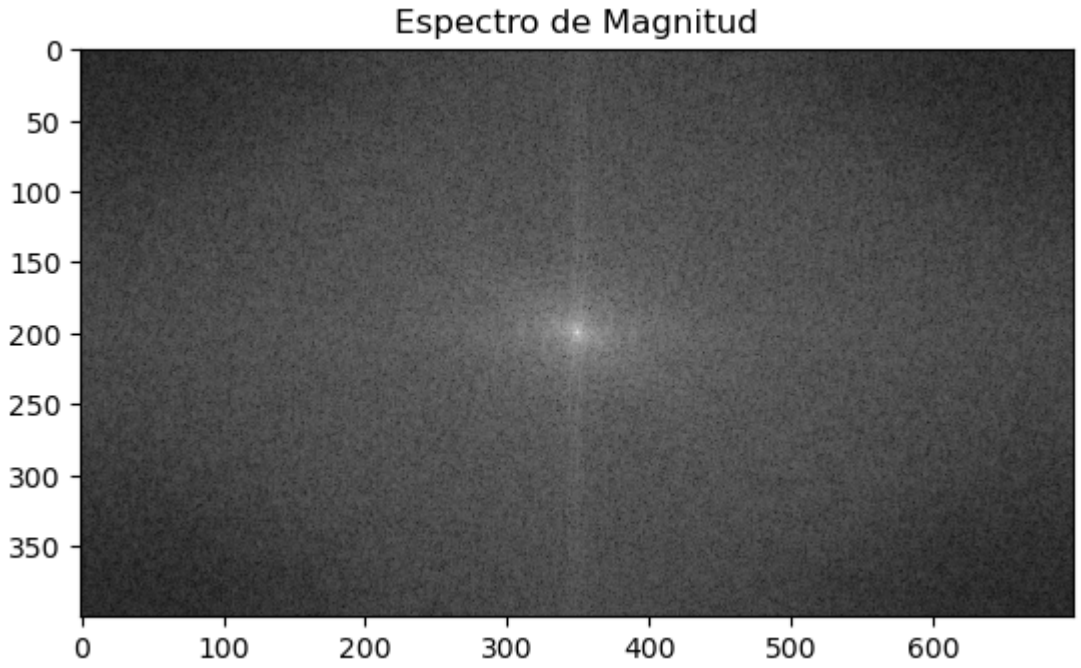
En esta práctica se evaluará el uso de filtros FIR en distintas aplicaciones, puntualmente para este laboratorio se proveerá de una imagen. El objetivo es diseñar filtros que sean capaces de modificar la imagen para resaltar o atenuar ciertos aspectos visuales.

1.1 Cargue la imagen proporcionada y muestre la versión original. Asegúrese de normalizar la imagen y graficarla.

In [57]:
```
import matplotlib.pyplot as plt
from PIL import Image

# Ruta de la imagen descargada
image_path = 'image.jpg'

# Cargar la imagen
image = Image.open(image_path)

# Mostrar la imagen
plt.imshow(image)
plt.axis('off')
plt.show()

image_gray = np.mean(image, axis=2)  # Convertir a escala de grises

# Normalizar la imagen
image_gray = image_gray / 255.0

# Mostrar la imagen original
plt.imshow(image_gray, cmap='gray')
plt.axis('off')
plt.show()
```

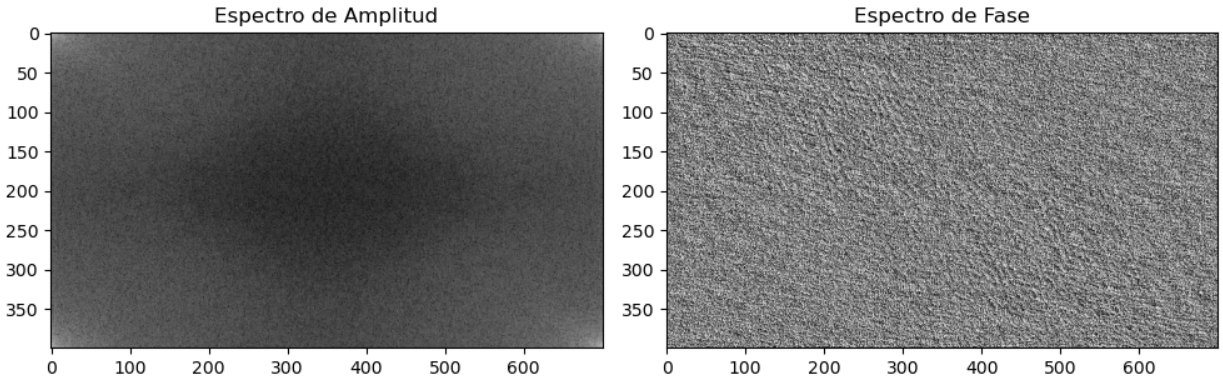
2. Grafique el espectro de amplitud, el espectro de fase y el espectro de magnitud de la imagen y explique características particulares en el espectro

In [60]:
```
# Calcular la transformada de Fourier de la imagen a color
fft_image = np.fft.fft2(image_gray)

# Calcular el espectro de magnitud
magnitude_spectrum = np.abs(np.fft.fftshift(fft_image))

# Graficar el espectro
plt.imshow(np.log(1 + magnitude_spectrum), cmap='gray')
plt.title("Espectro de Magnitud")

plt.show()
```



In [61]:

```python
import numpy as np
import matplotlib.pyplot as plt

# Cargar la imagen
#image = plt.imread("ruta_de_la_imagen.png")

# Aplicar la transformada de Fourier 2D
fourier = np.fft.fft2(image_gray)

# Calcular el espectro de amplitud y espectro de fase
amplitude_spectrum = np.abs(fourier)
phase_spectrum = np.angle(fourier)

# Crear la figura con dos subplots
fig, axes = plt.subplots(1, 2, figsize=(10, 5))

# Graficar el espectro de amplitud
axes[0].imshow(np.log(1 + amplitude_spectrum), cmap='gray')
axes[0].set_title("Espectro de Amplitud")

# Graficar el espectro de fase
axes[1].imshow(phase_spectrum, cmap='gray')
axes[1].set_title("Espectro de Fase")

# Ajustar espaciado entre subplots
plt.tight_layout()

# Mostrar la figura
plt.show()
```

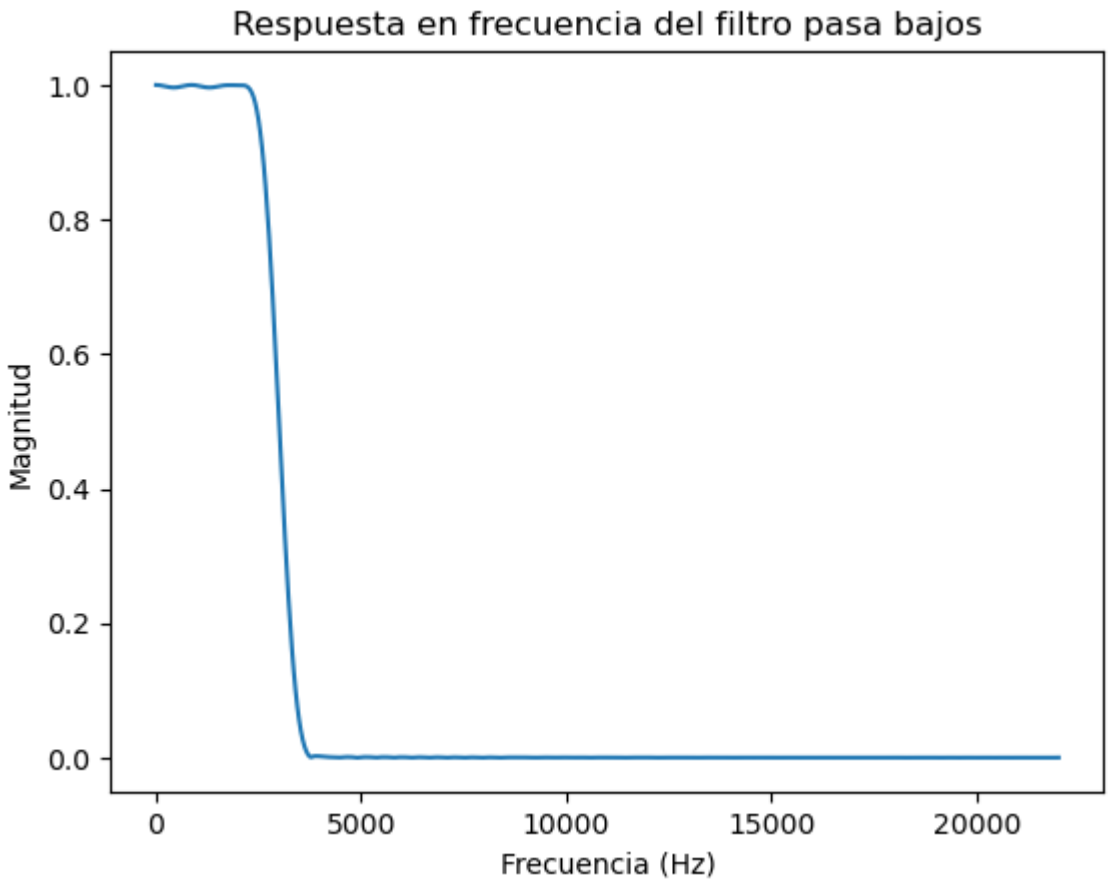


3. Diseño de filtro

Ahora, procederemos a diseñar un filtro FIR para resaltar o atenuar ciertas frecuencias en la imagen.

* 3.1 Diseño del filtro pasa bajos Comenzaremos diseñando un filtro pasa bajos, el cual atenuará las altas frecuencias de la imagen y mantendrá las bajas frecuencias.

In [68]:
```
from scipy.signal import firwin, freqz
# Especificaciones del filtro
cutoff_freq_hz = 3000  # Frecuencia de corte en Hz
filter_order = 101

# Obtener la frecuencia de muestreo (aquí se usa un valor de ejemplo de 44100 Hz)
sampling_freq_hz = 44100

# Convertir la frecuencia de corte a frecuencia normalizada
cutoff_freq_norm = cutoff_freq_hz / (0.5 * sampling_freq_hz)

# Diseño del filtro FIR
fir_filter_s = firwin(filter_order, cutoff_freq_norm)
# Ajustar las dimensiones del filtro
fir_filter_s = np.reshape(fir_filter_s, (filter_order, 1))
# Calcular la respuesta en frecuencia del filtro
w, h = freqz(fir_filter_s)

# Graficar la respuesta en frecuencia
plt.plot(0.5 * sampling_freq_hz * w / np.pi, np.abs(h))
plt.title('Respuesta en frecuencia del filtro pasa bajos')
plt.xlabel('Frecuencia (Hz)')
plt.ylabel('Magnitud')
plt.show()
```



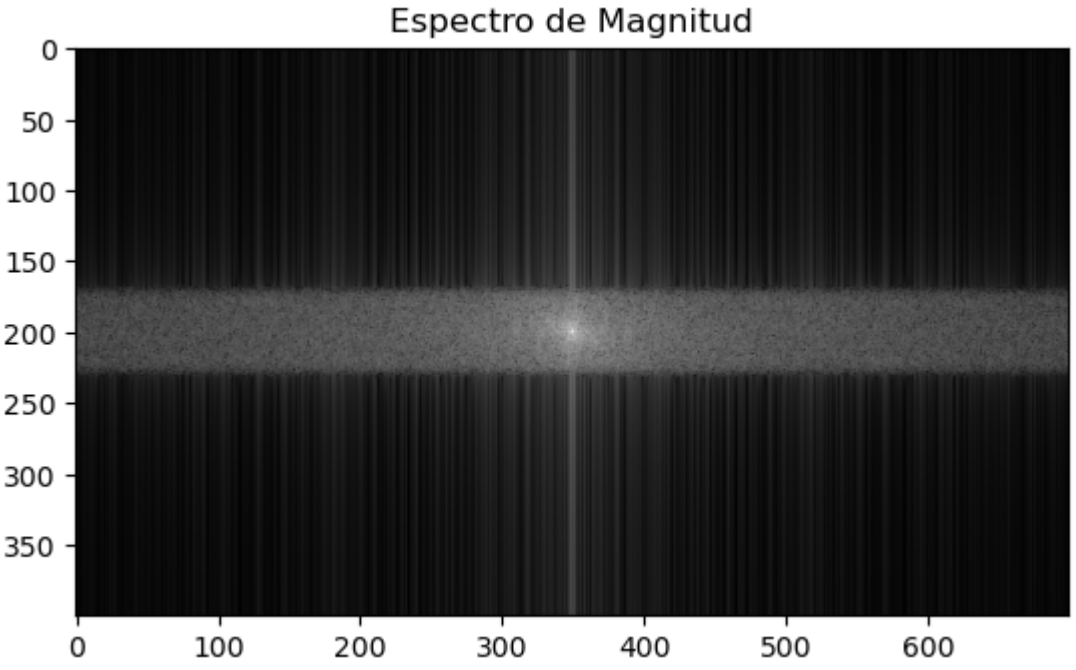
3.2. Aplicación del filtro Aplicaremos el filtro diseñado a la imagen para obtener la versión filtrada.

In [69]:

```
from scipy.signal import convolve2d


# Aplicar el filtro a la imagen
filtered_image = convolve2d(image_gray, fir_filter_s, mode='same', boundary='symm')

# Normalizar la imagen filtrada
filtered_image = filtered_image / np.max(filtered_image)

# Mostrar la imagen filtrada
plt.imshow(filtered_image, cmap='gray')
plt.axis('off')
plt.show()
```

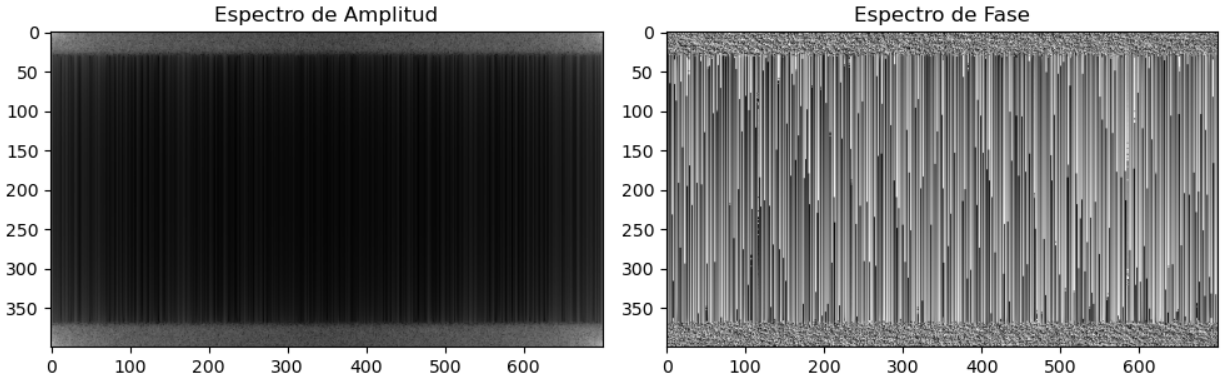
3.3 Grafique el espectro de amplitud, el espectro de fase y el espectro de magnitud de la imagen

In [70]:
```
# Calcular la transformada de Fourier de la imagen a color
fft_image = np.fft.fft2(filtered_image)

# Calcular el espectro de magnitud
magnitude_spectrum = np.abs(np.fft.fftshift(fft_image))

# Graficar el espectro
plt.imshow(np.log(1 + magnitude_spectrum), cmap='gray')
plt.title("Espectro de Magnitud")

plt.show()
```



In [71]:
```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Cargar la imagen
#image = plt.imread("ruta_de_la_imagen.png")

# Aplicar la transformada de Fourier 2D
fourier = np.fft.fft2(filtered_image)

# Calcular el espectro de amplitud y espectro de fase
amplitude_spectrum = np.abs(fourier)
phase_spectrum = np.angle(fourier)

# Crear la figura con dos subplots
fig, axes = plt.subplots(1, 2, figsize=(10, 5))

# Graficar el espectro de amplitud
axes[0].imshow(np.log(1 + amplitude_spectrum), cmap='gray')
axes[0].set_title("Espectro de Amplitud")

# Graficar el espectro de fase
axes[1].imshow(phase_spectrum, cmap='gray')
axes[1].set_title("Espectro de Fase")

# Ajustar espaciado entre subplots
plt.tight_layout()

# Mostrar la figura
plt.show()
```



4. Ahora se trabajara con filtros para imagenes como el filtro Gaussiano y filtro Sobel

4.1 Carque la imagen y muestrela

In [77]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# 4.1 Cargar la imagen
image_path = "image2.png"  # Reemplaza con la ruta de tu imagen
image = cv2.imread(image_path)

# Mostrar la imagen utilizando matplotlib
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
```

```
plt.axis('off')  # Desactivar los ejes
plt.show()
```

4.1.1 Aplicar el filtro Gaussiano con kernel de (5,5) y desviacion de 0, Mostrar la imagen luego cambio los valores de kernel del filtro, que observas en las imagenes

In [98]:
```
# Aplicar el filtro Gaussiano
gaussian_filtered = cv2.GaussianBlur(image, (5,5), 0)  # Puedes ajustar el tamaño del

plt.imshow(cv2.cvtColor(gaussian_filtered, cv2.COLOR_BGR2RGB))
plt.title("Imagen filtrada (Filtro Gaussiano)")
plt.axis("off")
```

Out[98]:
```
(-0.5, 805.5, 474.5, -0.5)
```

In [100…
```
# Aplicar el filtro Gaussiano
```

```
gaussian_filtered = cv2.GaussianBlur(image, (19,19), 0)  # Puedes ajustar el tamaño de

plt.imshow(cv2.cvtColor(gaussian_filtered, cv2.COLOR_BGR2RGB))
plt.title("Imagen filtrada (Filtro Gaussiano)")
plt.axis("off")
```

Out[100]: (-0.5, 805.5, 474.5, -0.5)

4.2 Aplicar el filtro Sobel

In [101…

```
sobel_x = cv2.Sobel(gaussian_filtered, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(gaussian_filtered, cv2.CV_64F, 0, 1, ksize=3)
sobel_filtered = cv2.addWeighted(cv2.convertScaleAbs(sobel_x), 0.5, cv2.convertScaleAb
plt.imshow(cv2.cvtColor(sobel_filtered, cv2.COLOR_BGR2RGB))
plt.title("Imagen filtrada (Filtro Sobel)")
plt.axis("off")

plt.show()
```

In [104…

```python
# Convertir la imagen a escala de grises
imagen_gris = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Aplicar el filtro Sobel
imagen_sobel_x = cv2.Sobel(imagen_gris, cv2.CV_64F, 1, 0, ksize=3)
imagen_sobel_y = cv2.Sobel(imagen_gris, cv2.CV_64F, 0, 1, ksize=3)
# Mostrar las imágenes
cv2.imshow('Imagen original', image)
cv2.imshow('Imagen en escala de grises', imagen_gris)
cv2.imshow('Derivadas horizontales (Sobel X)', imagen_sobel_x)
cv2.imshow('Derivadas verticales (Sobel Y)', imagen_sobel_y)

# Esperar hasta que se presione una tecla
cv2.waitKey(0)

# Cerrar todas las ventanas
cv2.destroyAllWindows()
```

Conclusiones

Luego de realizar las diferentes etapas del laboratorio de Procesamiento Digital de Señales, podemos llegar a las siguientes conclusiones:

1. El análisis espectral nos permite visualizar el espectro de amplitud, el espectro de fase y el espectro de magnitud de la imagen. Al analizar estos espectros, podemos identificar características particulares de la imagen, como la presencia de altas o bajas frecuencias.

2. El diseño del filtro pasa bajos nos permite atenuar las altas frecuencias de la imagen y mantener las bajas frecuencias. Al aplicar el filtro diseñado a la imagen, obtenemos una versión filtrada de la misma.

3. Al analizar nuevamente el espectro de amplitud, el espectro de fase y el espectro de magnitud de la imagen filtrada, podemos observar cómo el filtro ha afectado las

características espectrales de la imagen.

4. Por otro lado, al aplicar el filtro Gaussiano a la imagen original, se logra suavizar la imagen y reducir el ruido. Podemos ajustar los valores del kernel del filtro para obtener diferentes resultados en la imagen filtrada.

5. Finalmente, al aplicar el filtro Sobel a la imagen filtrada con el filtro Gaussiano, se resaltan los bordes y se detectan los cambios de intensidad en la imagen. Esto permite realzar los detalles estructurales de la imagen.

En conclusión, el uso de filtros FIR y el procesamiento digital de señales nos brinda herramientas poderosas para mejorar la calidad visual de las imágenes, resaltar características específicas y obtener información importante a partir de su análisis espectral.