```
img_sinc = cv2.imread('./imgs/sinc.png')
fourier_img = []
for img in [img_car, img_lena_periodic_noise, img_newspaper_shot_woman,␣
 ↪img_periodic_noise, img_sinc]:
  fourier_img.append(apply_fourier_transform(img))
```

Comparando com o ImageJ, os resultados são iguais. Abaixo, as mesmas imagens do espectro de magnitude obtidas com o ImageJ.

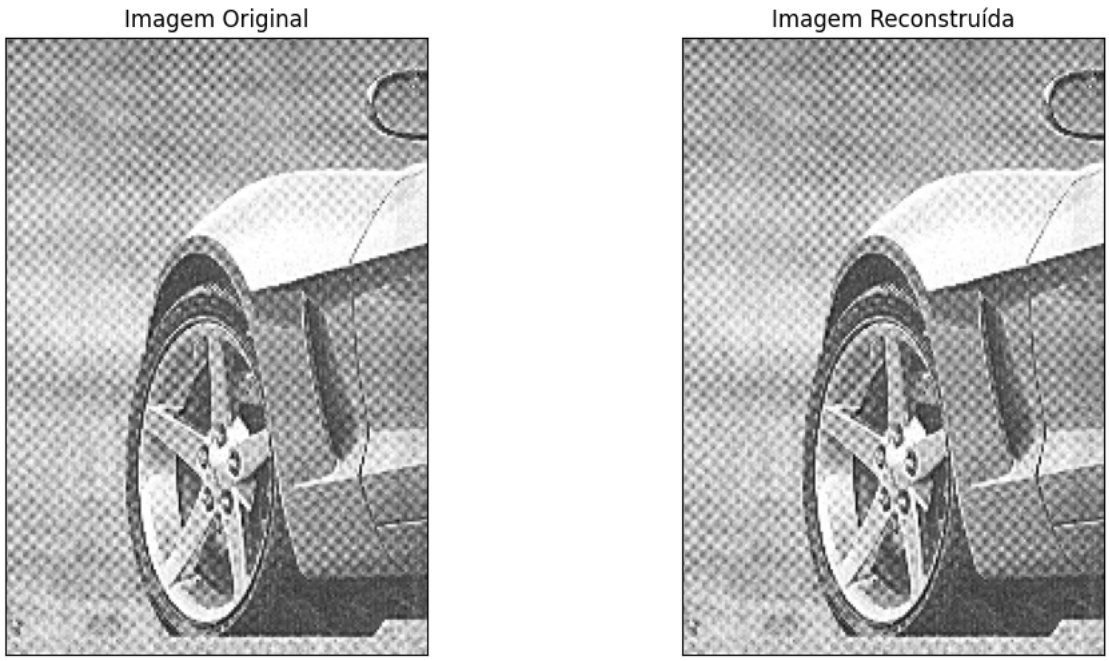
1.5.3 apply_inverse_fourier_transform :: inverse_transformed_image

A função apply_inverse_fourier_transform recebe uma imagem original e seu espectro de frequências obtido pela Transformada de Fourier Discreta (DFT), desfaz o deslocamento das frequências, aplica a Transformada Inversa de Fourier para reconstruir a imagem, calcula o espectro de magnitude da imagem reconstruída e normaliza seus valores para melhor visualização. Em seguida, ela converte o espectro de magnitude em uma imagem em tons de cinza, exibe a imagem original e a imagem reconstruída em subplots e retorna a imagem reconstruída.

[61]:
```
def apply_inverse_fourier_transform(original_img, dft_shift):

    # Desfazer o deslocamento (shift) do espectro de Fourier
    f_transform_unshifted = np.fft.ifftshift(dft_shift)

    # Aplicar a Transformada Inversa de Fourier 2D usando o OpenCV
    # inverse_transform = cv2.idft(f_transform_unshifted)
    inverse_transform = np.fft.ifft2(f_transform_unshifted)

    # Calcular o espectro de magnitude da transformada inversa
    # inverse_magnitude_spectrum = cv2.magnitude(inverse_transform[:, :, 0],␣
 ↪inverse_transform[:, :, 1])
    inverse_magnitude_spectrum = np.abs(inverse_transform)

    # Normalizar os valores para o intervalo de 0 a 255
    inverse_magnitude_spectrum = cv2.normalize(inverse_magnitude_spectrum,␣
 ↪None, 0, 255, cv2.NORM_MINMAX)

    # Converter para tipo de dados uint8 (imagem em tons de cinza)
    inverse_transformed_image = np.uint8(inverse_magnitude_spectrum)


    # Cria uma figura para exibir a imagem original e a imagem reconstruída
    plt.subplots_adjust(wspace=0.2, hspace=0.01)
    plt.figure(figsize=(12, 6))

    # Subplot 1: Imagem original em escala de cinza
    plt1 = plt.subplot(1, 2, 1)
    plt1.set_title('Imagem Original')
```

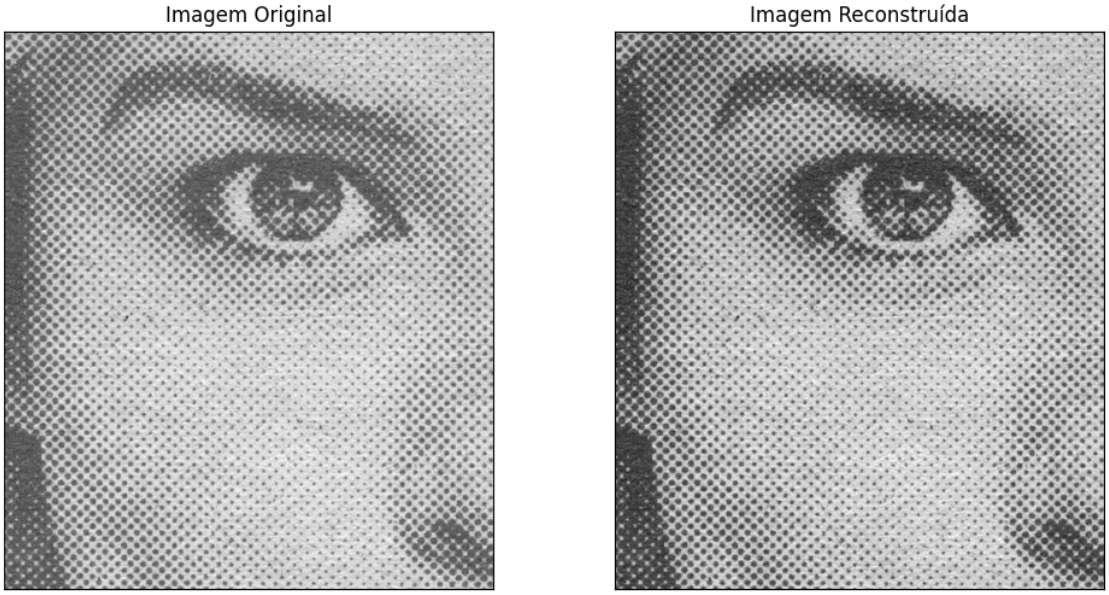
```
    plt1.set_xticks([]), plt1.set_yticks([])
    plt1.imshow(original_img, cmap='gray')

    # Subplot 2: Imagem reconstruída a partir do espectro de magnitude
    plt2 = plt.subplot(1, 2, 2)
    plt2.set_title('Imagem Reconstruída')
    plt2.set_xticks([]), plt2.set_yticks([])
    plt2.imshow(inverse_transformed_image, cmap='gray')

    # Exibe a figura com as duas imagens
    plt.show()


    return inverse_transformed_image

i = 0
for img in [img_car, img_lena_periodic_noise, img_newspaper_shot_woman,␣
 ↪img_periodic_noise, img_sinc]:
    inverse_fourier_img = apply_inverse_fourier_transform(img,fourier_img[i][2])
    i+=1
```

```
<Figure size 640x480 with 0 Axes>
```

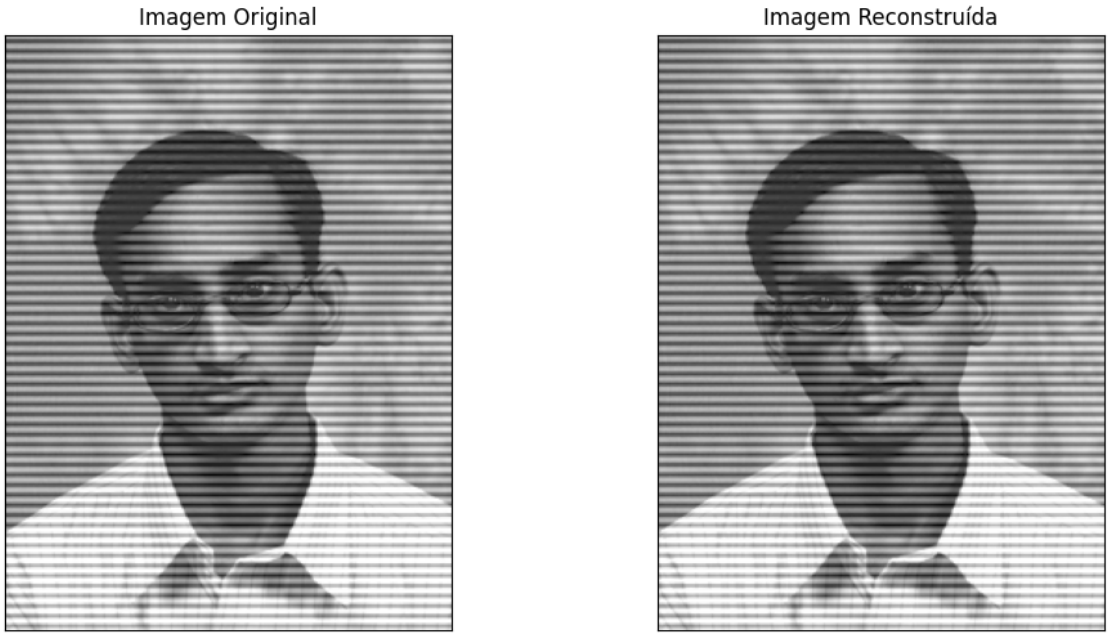



```
<Figure size 640x480 with 0 Axes>
```

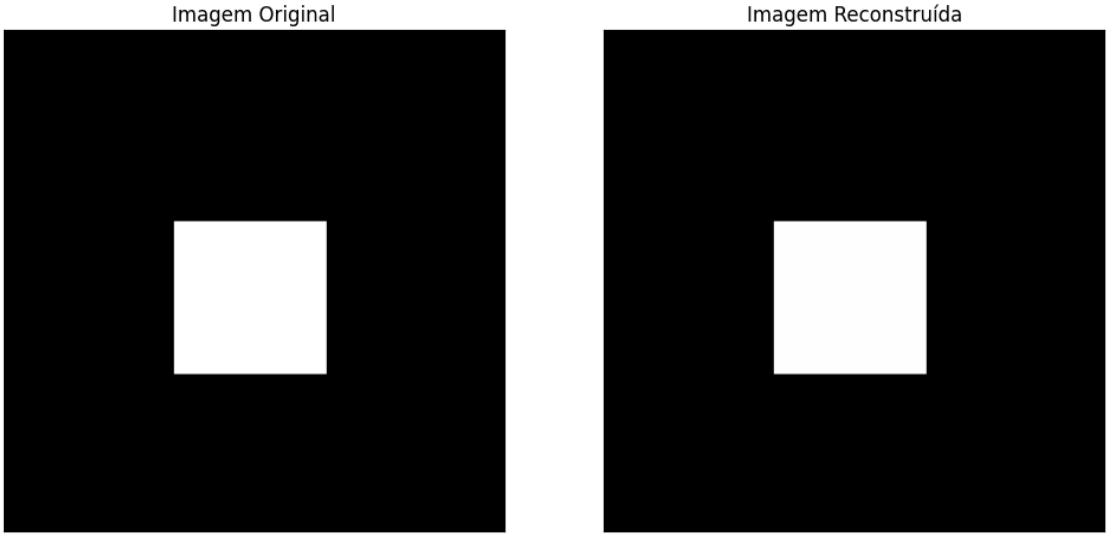

```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```



O código a seguir aplica a Transformada de Fourier Discreta (DFT) e a Transformada Inversa de Fourier em uma imagem com fundo preto e quadrado branco no centro representando a função sinc(x,y) e exibe os resultados plotando as imagens.

[62]:
```
img_white_square = np.zeros((600, 600, 3), dtype=np.uint8)
img_white_square[225:375, 225:375] = (255, 255, 255)
```