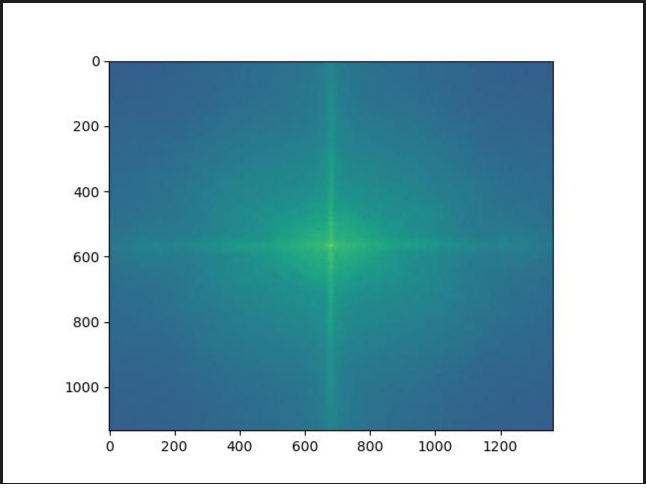
1、

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm

keyboard_img = cv2.imread("480548_keyboard.tif", cv2.IMREAD_GRAYSCALE)
fft_img = np.fft.fft2(keyboard_img)
spec_img = np.fft.fftshift(fft_img)
plt.imshow(np.abs(spec_img), norm=LogNorm(vmin=5))
plt.show()
```

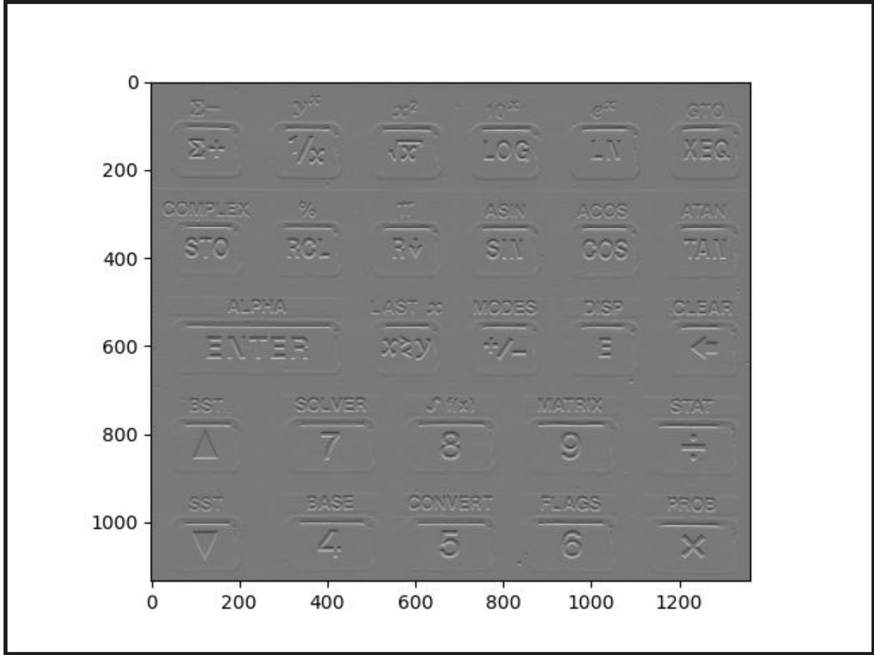


(a) Show the Fourier spectrum of the test image “keyboard” specified for this homework.

```
kernel=np.array([[-1, -2, -1],[0, 0, 0],[1, 2, 1]], dtype="int")
```

(b)Enforce odd symmetry on the kernel.

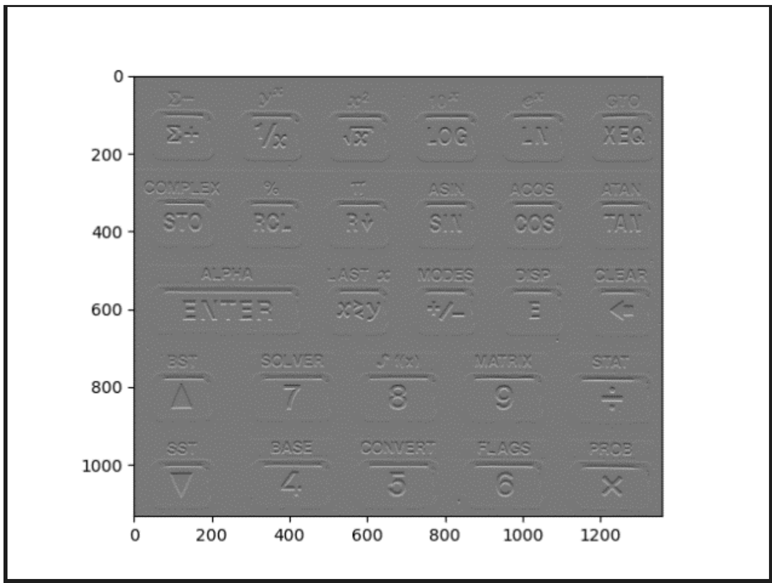
```
# odd symmetry the kernel change to below I don't know the question
exactly “odd symmetry” means
# kernel=np.array([[0, -1, -2],[1, 0, -1],[2, 1, 0]], dtype="int")
# the original also odd symmetry in y axis
padding_size = (keyboard_img.shape[0] - kernel.shape[0],
keyboard_img.shape[1] - kernel.shape[1])
kernel = np.pad(kernel, (((padding_size[0]+1)//2, padding_size[0]//2),
((padding_size[1]+1)//2, padding_size[1]//2)), 'constant')
kernel = np.fft.ifftshift(kernel)
freq_sobel_img = np.real(np.fft.ifft2(np.fft.fft2(keyboard_img) *
np.fft.fft2(kernel)))
plt.imshow(freq_sobel_img, 'gray')
plt.show()
```



(c) Show the result of frequency-domain filtering of the test image using the horizontal Sobel kernel.

```
def imgfilter2D(img, filter, ratio):
    filter = np.flip(filter)
    img_x, img_y = img.shape
    filter_x, filter_y = filter.shape
    result_mtx = np.zeros(((img_x - filter_x + 1), (img_y - filter_y +
1)))
    for i in range(result_mtx.shape[0]):
        for j in range(result_mtx.shape[1]):
            result_mtx[i][j] = np.sum(img[i:i + filter_x, j:j +
filter_y] * filter) / ratio
    result_mtx = ((result_mtx/np.max(np.abs(result_mtx))) + 1)*127.5
    return result_mtx.astype('uint8')

kernel=np.array([[-1, -2, -1],[0, 0, 0],[1, 2, 1]], dtype="int")
result_sobel_y = imgfilter2D(keyboard_img, kernel, 4)
plt.imshow(result_sobel_y, 'gray')
plt.show()
```



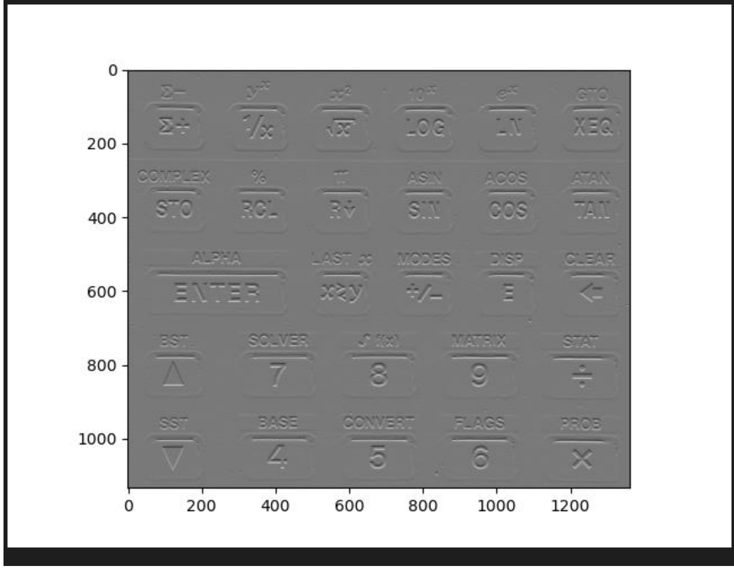
Left is frequency domain right is space domain

(d)Compare your result in (c) with the result of space-domain filtering.

Not obviously different in this case

(e) Show the result of frequency-domain filtering of the test image using the horizontal Sobel kernel without enforcing odd symmetry on the kernel.

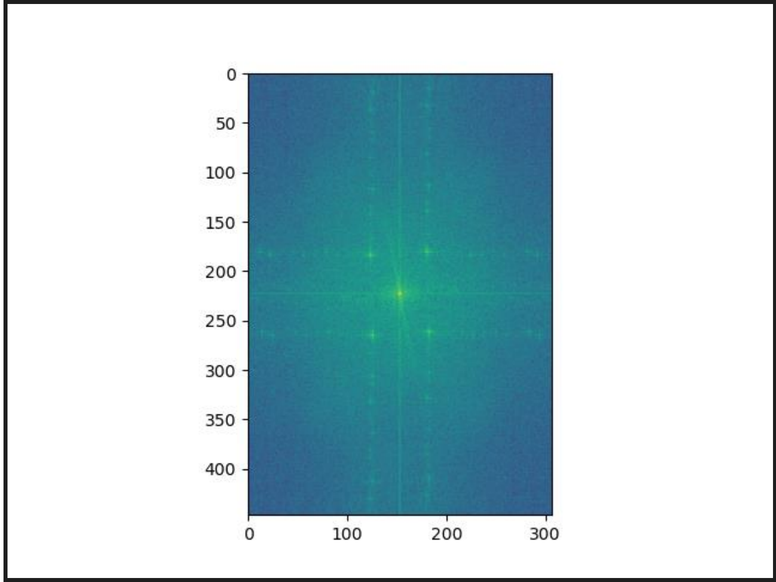
```
kernel=np.array([[-1, -2, -1],[0, 0, 0],[1, 2, 1]], dtype="int")
padding_size = (keyboard_img.shape[0] - kernel.shape[0],
keyboard_img.shape[1] - kernel.shape[1])
kernel = np.pad(kernel, (((padding_size[0]+1)//2, padding_size[0]//2),
((padding_size[1]+1)//2, padding_size[1]//2)), 'constant')
kernel = np.fft.ifftshift(kernel)
freq_sobel_img = np.real(np.fft.ifft2(np.fft.fft2(keyboard_img) *
np.fft.fft2(kernel)))
plt.imshow(freq_sobel_img, 'gray')
plt.show()
```



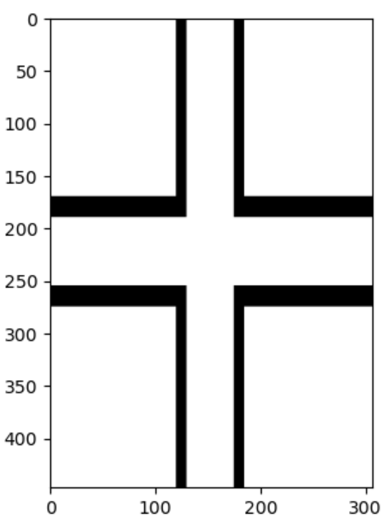
2、

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm

img_newspaper = cv2.imread("480548_newspaper.tif",
cv2.IMREAD_GRAYSCALE)
fft_img = np.fft.fft2(img_newspaper)
spec_img = np.fft.fftshift(fft_img)
plt.imshow(np.abs(spec_img), norm=LogNorm(vmin=5))
plt.show()
```



```
Get spectrum
reject_filter = np.ones(img_newspaper.shape)
reject_filter[:, 120:130] = 0
reject_filter[:, 175:185] = 0
reject_filter[190:255, 120:130] = 1
reject_filter[190:255, 175:185] = 1
reject_filter[170:190, :] = 0
reject_filter[255:275, :] = 0
reject_filter[170:190, 130:175] = 1
reject_filter[255:275, 130:175] = 1
plt.imshow(reject_filter*255, "gray")
plt.show()
```



Our design reject filter

```
filter_spectrum = spec_img * reject_filter
filter_spectrum = np.fft.ifftshift(filter_spectrum)
result_img = np.fft.ifft2(filter_spectrum)
plt.imshow(np.abs(result_img), "gray")
plt.show()
```

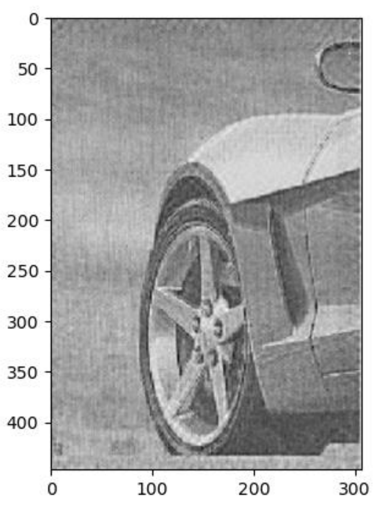


Image processing result

```
img_cassini = cv2.imread("480548_cassini.tif", cv2.IMREAD_GRAYSCALE)
fft_img = np.fft.fft2(img_cassini)
spec_img = np.fft.fftshift(fft_img)
plt.imshow(np.abs(spec_img), norm=LogNorm(vmin=5))
plt.show()
```

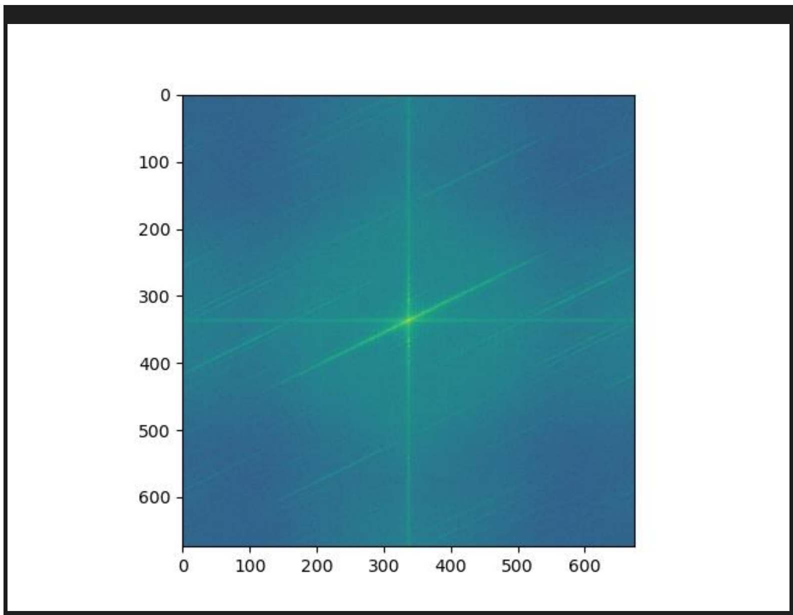
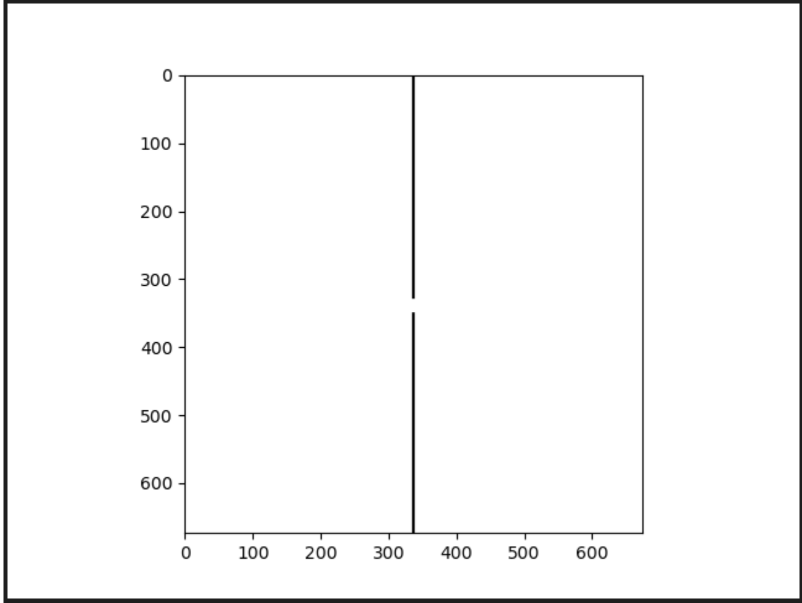


Image spectrum

```
reject_filter = np.ones(img_cassini.shape)
reject_filter[:, 335:339] = 0
reject_filter[330:350, 335:339] = 1
plt.imshow(reject_filter*255, "gray")
plt.show()
```



Reject filter

```
filter_spectrum = spec_img * reject_filter
filter_spectrum = np.fft.ifftshift(filter_spectrum)
result_img = np.fft.ifft2(filter_spectrum)
plt.imshow(np.abs(result_img), "gray")
plt.show()
```

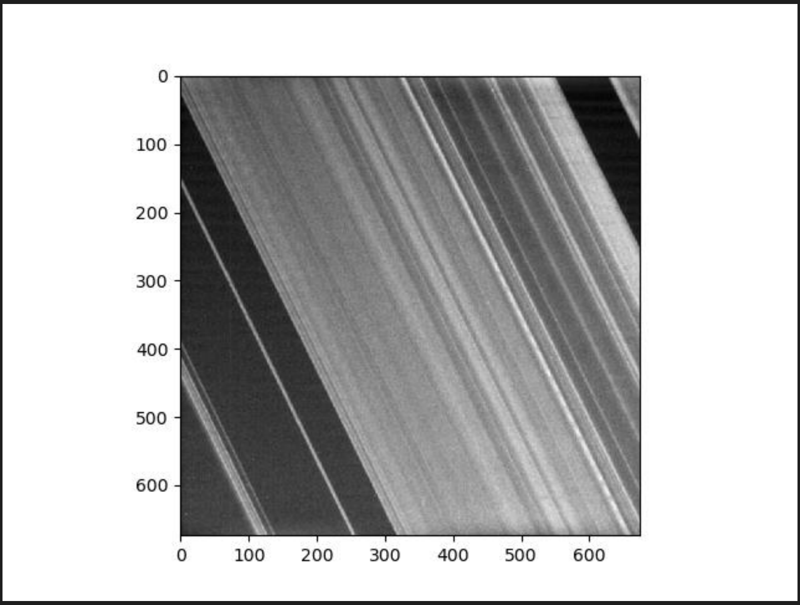


Image Result