1、

import numpy as np

import cv2

import matplotlib.pyplot as plt

from matplotlib.colors import LogNorm

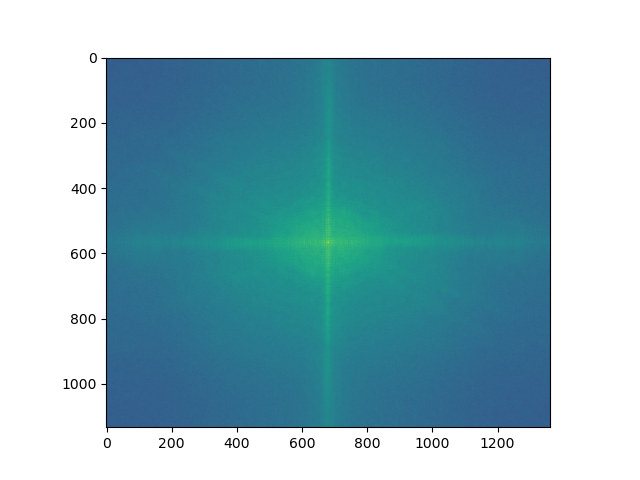
keyboard\_img = cv2.imread("480548\_keyboard.tif", cv2.IMREAD\_GRAYSCALE)

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])

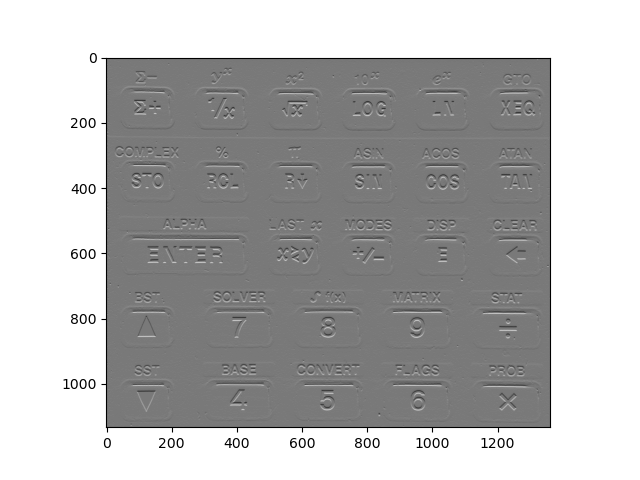
kernel = np.pad(kernel, (((padding\_size[0]+1)//2, padding\_size[0]//2), ((padding\_size[1]+1)//2, padding\_size[1]//2)), 'constant')

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

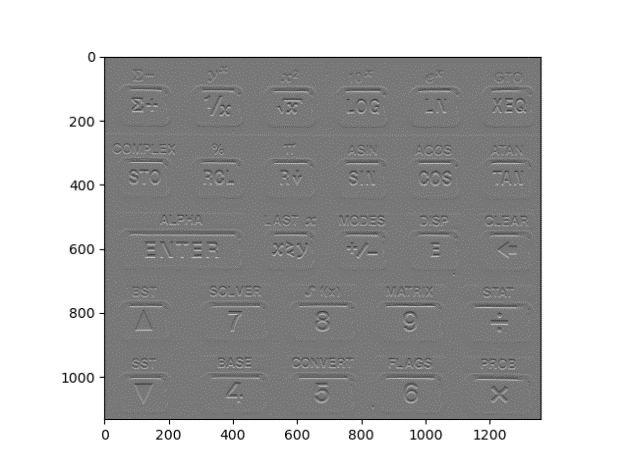
    return result\_mtx.astype('uint8')

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])

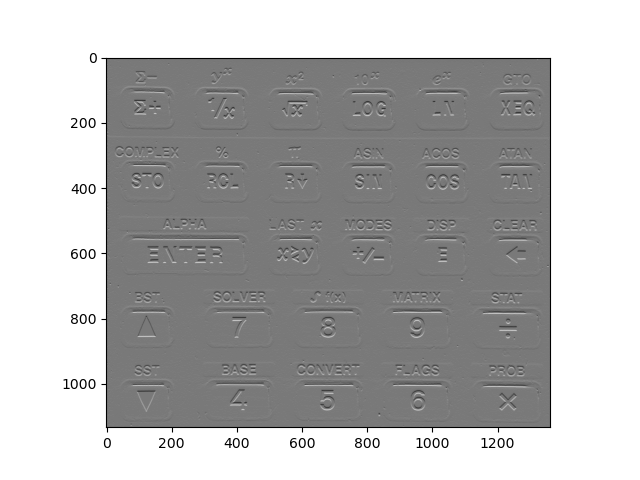
kernel = np.pad(kernel, (((padding\_size[0]+1)//2, padding\_size[0]//2), ((padding\_size[1]+1)//2, padding\_size[1]//2)), 'constant')

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

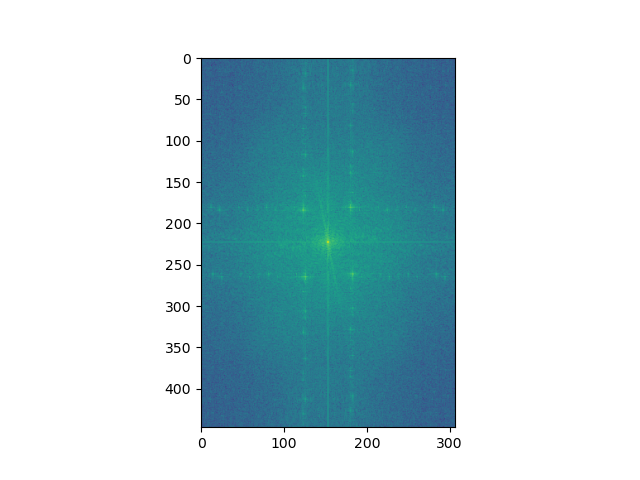
img\_newspaper = cv2.imread("480548\_newspaper.tif", cv2.IMREAD\_GRAYSCALE)

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

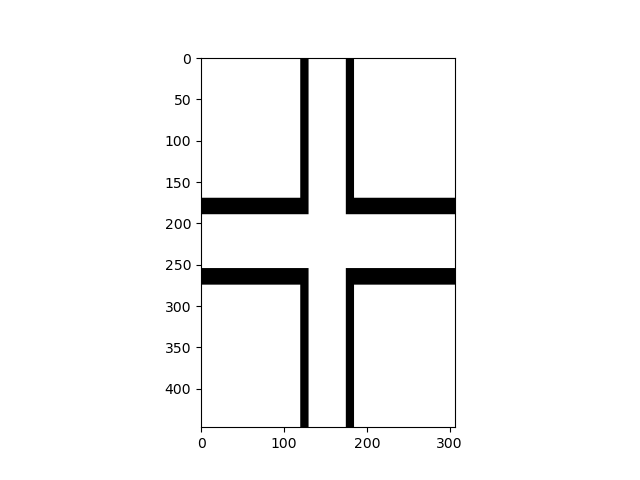
reject\_filter[255:275, :] = 0

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()

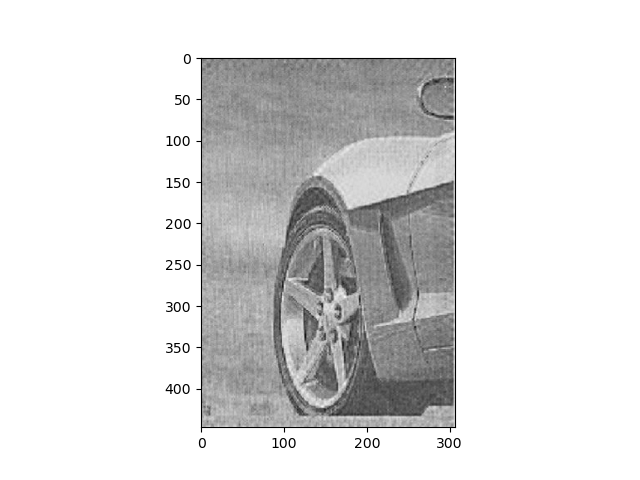


Image processing result

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()

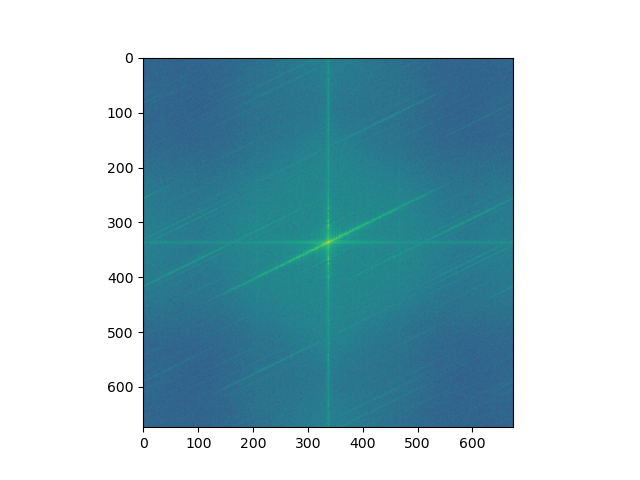


Image spectrum

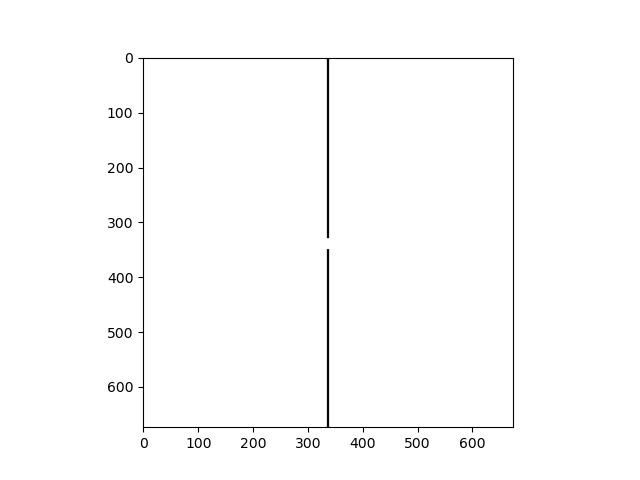
reject\_filter = np.ones(img\_cassini.shape)

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()

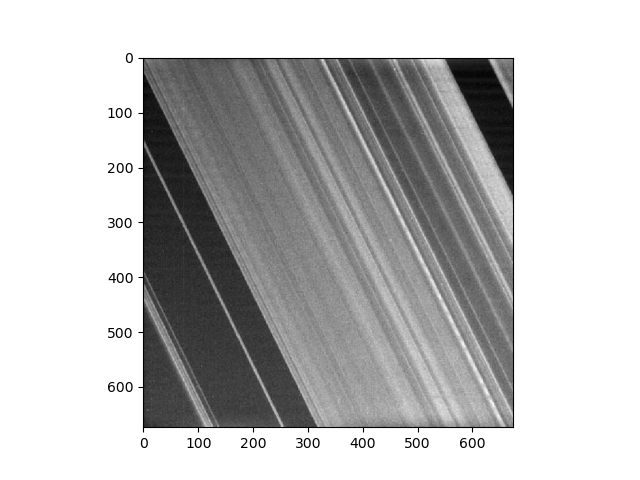


Image Result