

2110431 Introduction to Digital Imaging

2147329 Digital Image Processing and Vision Systems

Homework #2

Deadline: October 10, 2023 @23:59

Submissions: only PDF version of this file

Use these commands in colab to download the images.

```
!wget https://drive.google.com/uc?id=1dhh4m9VRLUSmbaHfge2iRSW5Azkpefco -O kitty3.png
!wget https://drive.google.com/uc?id=1o0UMPTyUFzX9CaQp-BwYXgkCholZo6yL -O kitty55.png
!wget https://drive.google.com/uc?id=1Jk0cEtQt4HxkLcKlmTHukpb22gJZ4dmL -O noisy_kitty55.png
!wget https://drive.google.com/uc?id=1xCNA5338nzjlGgGQ5-oBA1dKCWlMvn21 -O hillbefore_noise10%.jpg
```

1. Apply ideal low pass filter in frequency domain on “Kitty3.png” image which has $M \times N$ pixels. Find the minimum cutoff frequency (C) in integer that still maintain the total image power P_T more than 99%. Where the total image power, P_T is calculated by summing the components of spectrum power at each point (u, v) , for $u = 1, 2, \dots, M - 1$ and $v = 1, 2, \dots, N - 1$

$$P_T = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} P(u, v)$$

$P(u, v)$ is the spectrum power provided in the lecture slides

α percent of the image power can be calculated from $100 \times P_{T_f} / P_{T_org}$, where

P_{T_org} is the total image power of the original image and P_{T_f} of the filtered image

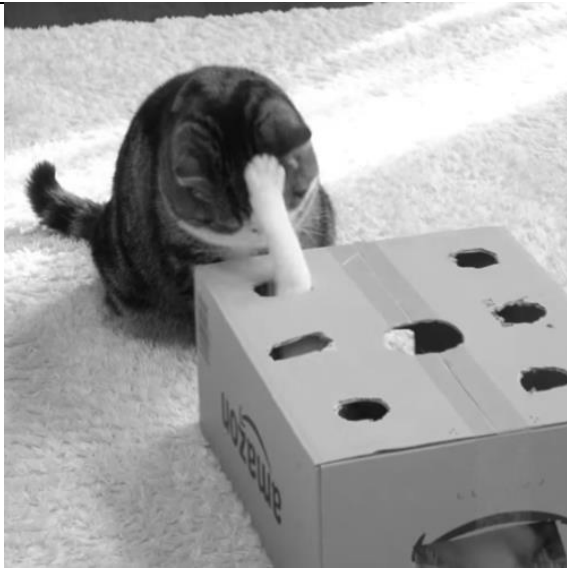
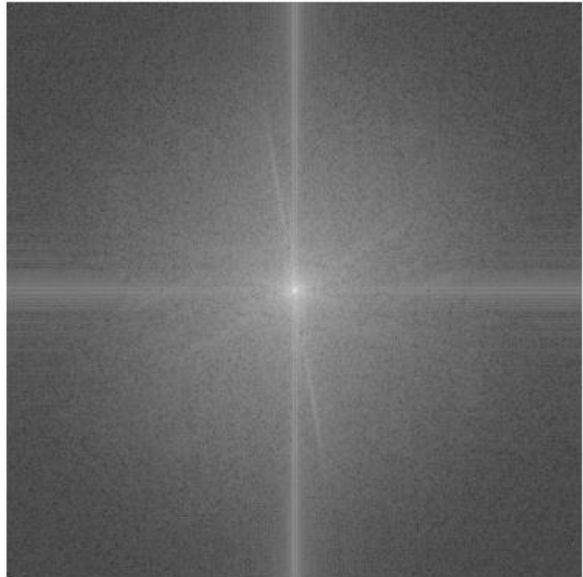
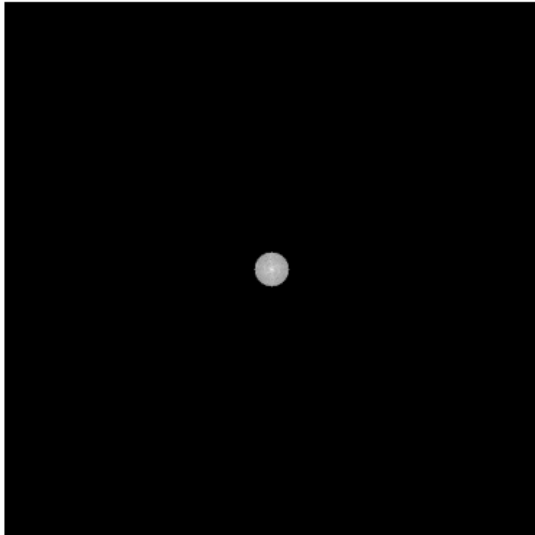

Put your results in the blank box below

Cutoff frequency (D0) =

16

α =

99.03514418234892

Original Image ("kitty55.png")	Fourier Spectrum of the original image
	
Fourier Spectrum of the filtered image	Filtered images ($P_T > 99\%$)
	

Put your code here:

(you can paste all the code or put only the highlighted code for this problem)

```
kitty55 = cv2.imread("kitty55.png")
kitty55 = cv2.cvtColor(kitty55, cv2.COLOR_BGR2GRAY).astype(float)

ff_kitty55 = np.fft.fft2(kitty55)
fftshift_kitty = np.fft.fftshift(ff_kitty55)
mag = (np.log(1 + np.abs(fftshift_kitty))).astype(int)
```

```

plt.imshow( mag, cmap='gray')
plt.axis(False)
plt.show()
plt.imshow(kitty55, cmap = 'gray')
plt.axis(False)
plt.show()

rows, cols = kitty55.shape
center_row, center_col = int(rows / 2), int(cols / 2)
for C in range(1,min(rows, cols) // 2):
    u, v = np.meshgrid(np.arange(rows), np.arange(cols))

    distance = np.sqrt((u - center_row)**2 + (v - center_col)**2)

    ideal_low_pass_filter = np.zeros((rows,cols))
    ideal_low_pass_filter[distance <= C] = 1

    fftshift_fitered_kitty = fftshift_kitty * ideal_low_pass_filter

    PT_org = np.sum(np.abs(fftshift_kitty)**2)
    PT_filtered = np.sum(np.abs(fftshift_fitered_kitty)**2)
    alpha = (PT_filtered * 100) / PT_org

    if alpha > 99:
        print('alpha:' , alpha)
        print('Cut off freq:', C)
        break

image_filtered =
np.fft.ifft2(np.fft.ifftshift(fftshift_fitered_kitty))

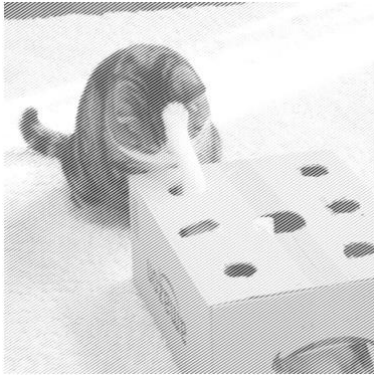
plt.imshow((np.log( 1 + np.abs(fftshift_fitered_kitty))).astype(int),
cmap='gray')
plt.axis(False)
plt.show()
plt.imshow(np.real(image_filtered), cmap='gray')
plt.axis(False)
plt.show()

```

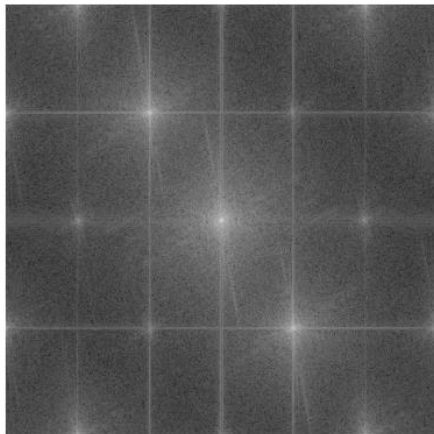
2) There is some periodic noise in “noisy_kitty55.png” image. Remove the periodic noise and calculate PSNR.

Show how to restore the image from periodic noise and display the result,

Noisy_kitty.png



Apply Fourier transform and shifted, so we will get



Apply Notch's filter in rectangle shape

```
fftshift_noisy_kitty[:,80:90] = 0
fftshift_noisy_kitty[:,160:170] = 0
fftshift_noisy_kitty[:,330:340] = 0
fftshift_noisy_kitty[:,410:420] = 0
fftshift_noisy_kitty[120:130,:] = 0
fftshift_noisy_kitty[370:380,:] = 0
```

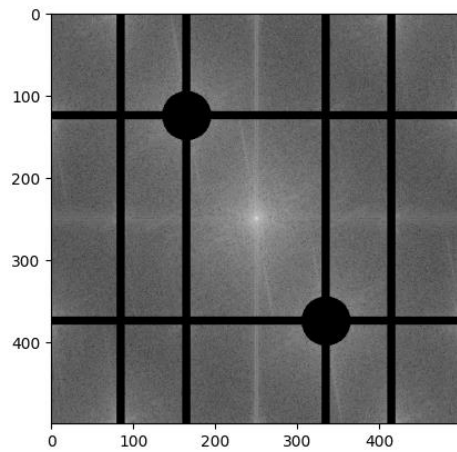
and circle shape, we will get

```
center_x1, center_y1 = 125,165 # coordinates of the notch filter
center_x2, center_y2 = 375,335
radius = 30 # radius of the circular mask
# Create a mask
rows, cols = fftshift_noisy_kitty.shape
center1 = [center_x1, center_y1]
center2 = [center_x2, center_y2]
x, y = np.ogrid[:rows, :cols]
```

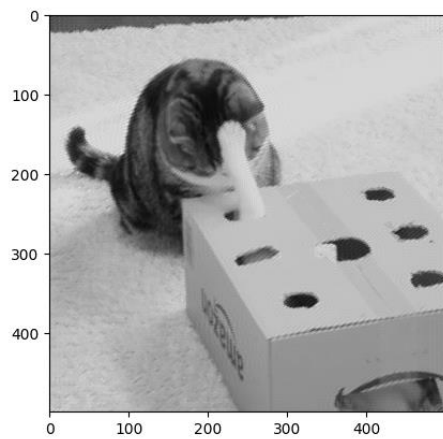
```

mask_area1 = (x - center1[0]) ** 2 + (y - center1[1]) ** 2 <=
radius*radius
mask_area2 = (x - center2[0]) ** 2 + (y - center2[1]) ** 2 <=
radius*radius
fftshift_noisy_kitty[mask_area1] = 0
fftshift_noisy_kitty[mask_area2] = 0

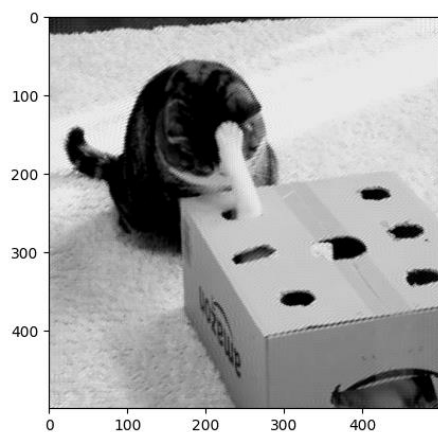
```



Inverse shifted and inverse Fourier transform



Apply implement contrast stretching with $(r1, s1) = (50, 0)$, $(r2, s2) = (225, 255)$



What is the PSNR before adding noise (compared with “kitty55.png”)?

PSNR = 28.78

PSNR of the restored image

PSNR = 29.28

3. Write a paragraph (at least 300 words in Thai or English) to summarize **three** image applications using frequency analysis in your own words and add the references (excluded from the total word count). You will have to present and share the applications you found on Wednesday, October 3, 2023 in a discussion session.

Ans

Image compression using Discrete Cosine Transform (DCT) is a pivotal technique in digital signal processing. DCT functions by converting spatial data into frequency components, emphasizing high-frequency details and low-frequency patterns. In the context of image compression, DCT is widely employed in formats like JPEG. When applied to non-overlapping blocks of an image, DCT transforms pixel values into a set of coefficients. These coefficients, representing the image's frequency content, undergo quantization, allowing for data reduction. By retaining essential visual information while discarding redundant data, DCT-based compression significantly reduces file sizes, making it ideal for applications where efficient storage and transmission are paramount, such as digital photography, multimedia, and internet-based platforms.

Image denoising, specifically targeting salt-and-pepper noise, finds efficacy through Fourier analysis. In this technique, the image, corrupted by sparse, random noise, is transformed into the frequency domain using Fourier Transform. Salt-and-pepper noise, dispersed irregularly across the frequency spectrum, is distinctly separated from the desired signal. By identifying these noise frequencies, denoising algorithms can apply filters, effectively removing the unwanted noise while preserving image details. Fourier analysis enables precise noise isolation, allowing for accurate restoration of the image. This method ensures clarity in images vital for fields like medical imaging or satellite photography, where precision is paramount. The process demonstrates the power of Fourier analysis in enhancing image quality and reliability.

Recognizing fake images through frequency analysis is a critical aspect of digital forensics, ensuring the integrity of visual information. By scrutinizing an image's frequency components using methods like Fourier Transform or Discrete Cosine Transform (DCT), inconsistencies or irregularities indicative of manipulation can be identified. Genuine images possess specific frequency patterns, while manipulations disrupt these patterns, leaving behind detectable signs. Research by Farid (2009) highlights the significance of frequency analysis in digital forensics, emphasizing the identification of discrepancies in noise patterns and lighting within images. This approach proves vital in various fields, from journalism to legal investigations, where ensuring the authenticity of visual content is paramount.

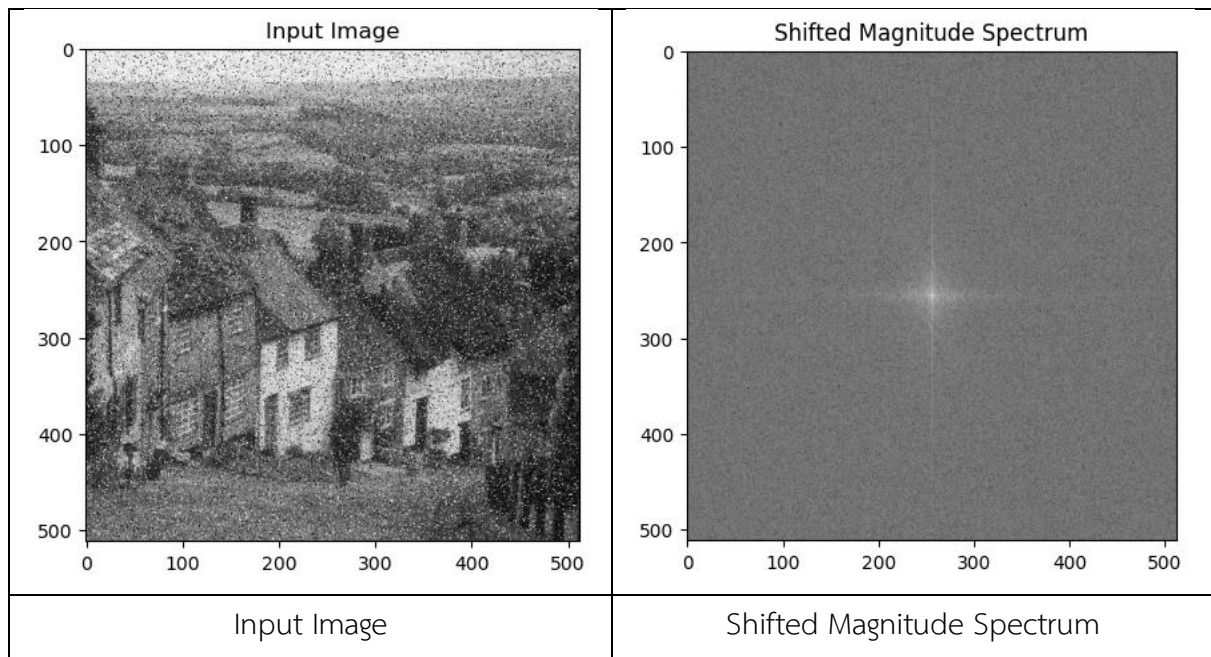
<https://cloudinary.com/glossary/dct-compression>

http://sites.apam.columbia.edu/courses/ap1601y/Watson_MathJour_94.pdf

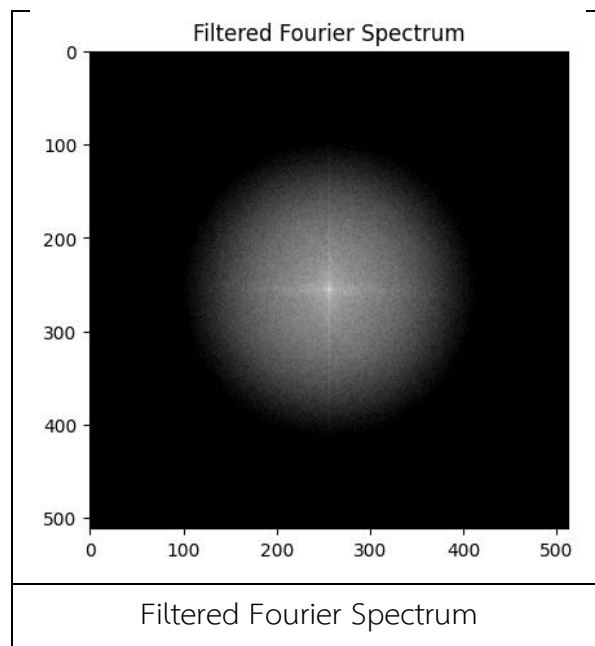
https://www.jstage.jst.go.jp/article/transinf/E100.D/5/E100.D_2016EDP7393/_pdf

4. Apply Gaussian low-pass filter on a noisy image, "hillbefore_noise10%.jpg" below using cutoff frequency, D0 set to 35.

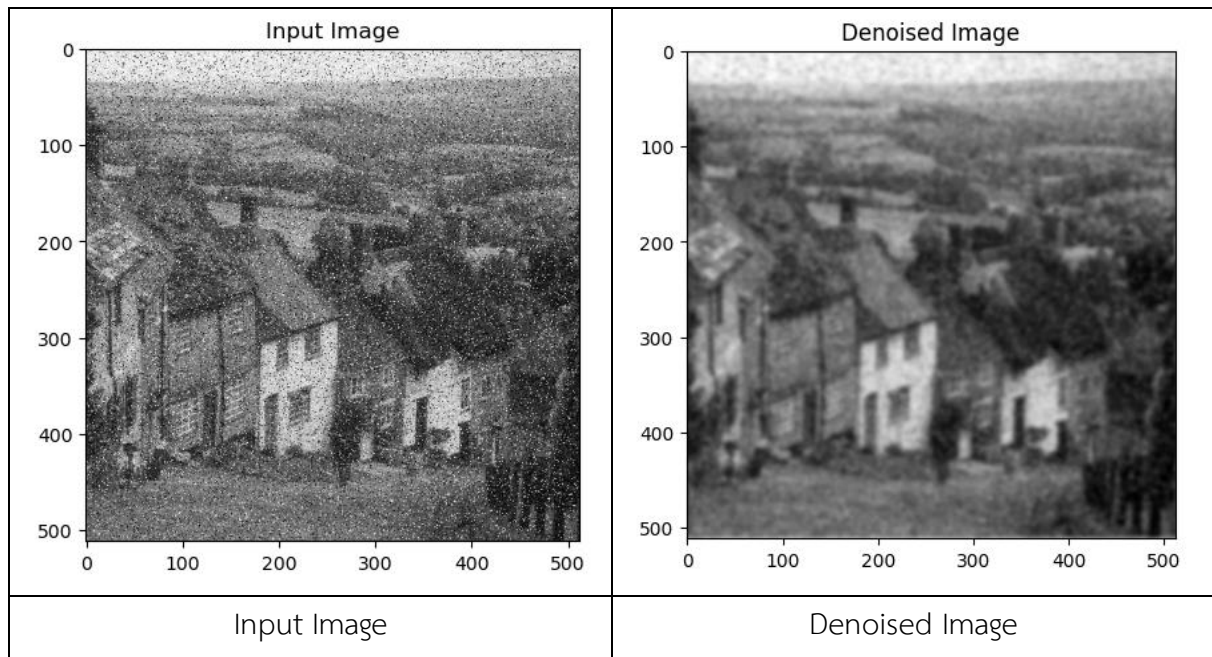
4.1. Generate the Shifted Magnitude Spectrum and put resulted image into the blank box below.



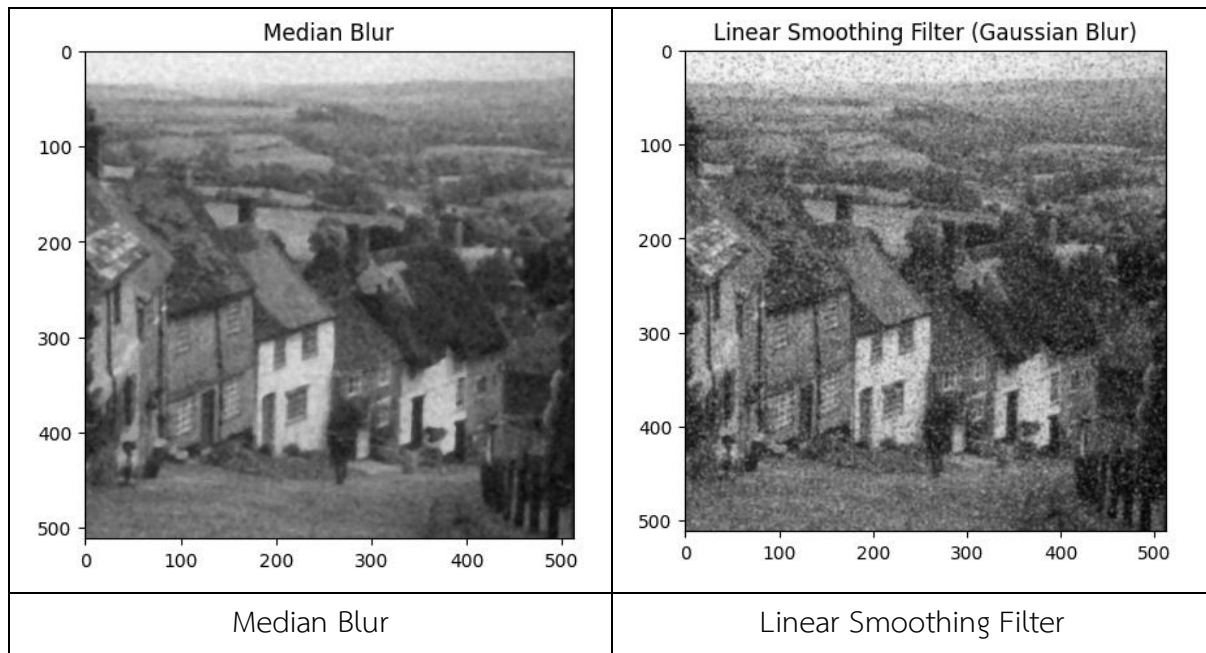
4.2. Display the Filtered Fourier Spectrum and put the resulted image below.



4.3. Show the Image after denoising with the Gaussian low-pass filter method and put the result in the blank box on the right side below.



4.4. Discuss the usage of the Gaussian low-pass filter in comparison to other methods, such as Median Blur and Linear Smoothing Filter. Explain the differences and identify which one might be superior. (Short description)



- Gaussian Low-Pass Filter: Useful for reducing noise while preserving edges, making it suitable for applications where fine details need to be retained. It uses a Gaussian kernel, giving more weight to nearby pixels and less weight to distant ones.
- Median Blur: Replacing each pixel's value with the median of neighboring pixels, preserving edges well and making it a popular choice for noise reduction in images affected by S&P noise.
- Linear Smoothing Filter (Gaussian Blur): Computes the weighted average of neighboring pixels, providing a uniform smoothing effect. It is fast and easy to compute but can blur edges and fine details, especially with larger kernel sizes.
- Comparison between Gaussian Low-Pass Filter, Median Blur, and Linear Smoothing Filter: Upon evaluating the results, it is evident that Median Blur excels in noise reduction while preserving intricate details. Linear Smoothing Filter still exhibits some noise, whereas the Gaussian Low-Pass Filter yields a significantly blurry image. Therefore, Median Blur proves superior in effectively handling S&P noise, providing a balance between noise removal and detail preservation.