

▼ Project 3

Consider the image Bird 2 degraded, degraded by mild atmospheric turbulence blurring.

- (a) Estimate the parameter k of the model developed by Hufnagel & Stanley.
- (b) Construct and plot the restored image using the $H(u,v)$ obtained.

▼ Figure of the Fourier magnitude spectrum of the degraded image Bird 2 degraded

```
1 import cv2
2 import numpy as np
3 import math
4 from matplotlib import pyplot as plt
5
6 img = cv2.imread('Bird 2 degraded.tif',0)
7 f = np.fft.fft2(img)
8 fshift = np.fft.fftshift(f)
9 magnitude_spectrum = 20*np.log(np.abs(fshift))
10
11 #plot
12 fig, (ax1, ax2) = plt.subplots(figsize=(14,9), nrows=1, ncols=2)
13 ax1.imshow(img, cmap = 'gray')
14 ax1.set_title('input image')
15 ax1.set_xticks([])
16 ax1.set_yticks([])
```

Saving...



```

18 ax2.imshow(magnitude_spectrum, cmap = 'gray')
19 ax2.set_title('magnitude_spectrum')
20 ax2.set_xticks([])
21 ax2.set_yticks([])

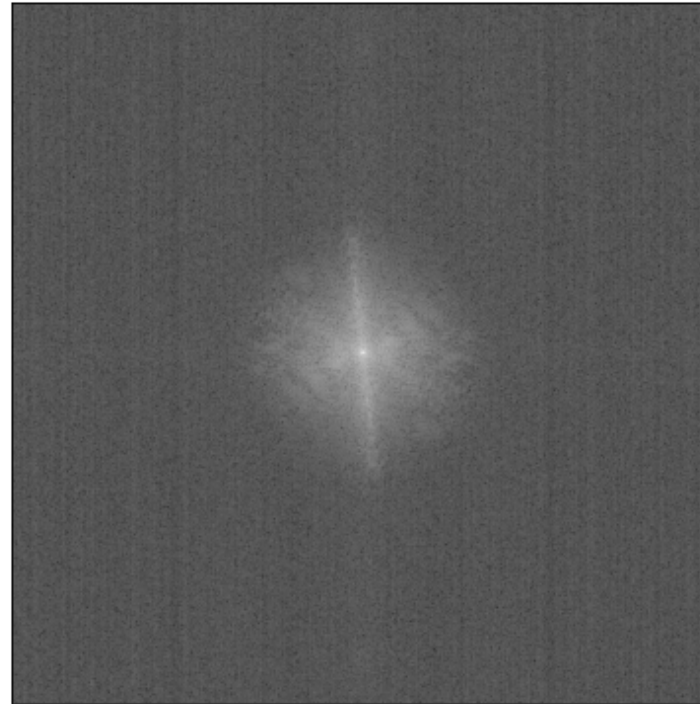
```

[]

input image



magnitude_spectrum



▼ Figure of the Fourier magnitude (frequency response) of degradation model $H(u,v)$

```

1 def H(u,v,k):
2     M=600

```

Saving...

×

$((-k) * (((u-M/2)**2 + (v-N/2)**2)**(5/6)))$

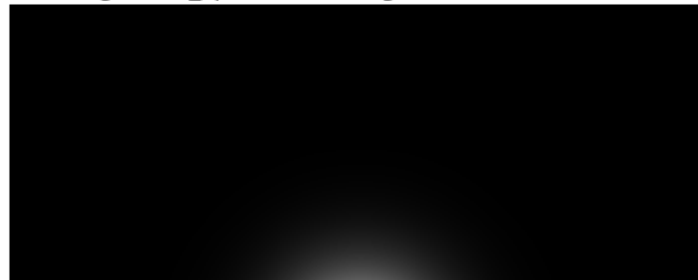
```
5
6
7 degradation=np.zeros( np.shape(magnitude_spectrum) )
8 for u in range(600):
9     for v in range(600):
10         degradation[u][v]=H(u,v,0.001)
11
12 #plot
13 fig, (ax1, ax2) = plt.subplots(figsize=(14,9), nrows=1, ncols=2)
14 ax1.imshow(magnitude_spectrum, cmap = 'gray')
15 ax1.set_title('origin magnitude_spectrum')
16 ax1.set_xticks([])
17 ax1.set_yticks([])
18
19 ax2.imshow(degradation, cmap = 'gray')
20 ax2.set_title('magnitude_spectrum of degradation model H(u,v)')
21 ax2.set_xticks([])
22 ax2.set_yticks([])
```

[]

origin magnitude_spectrum



magnitude_spectrum of degradation model $H(u,v)$



- ▼ Figures of the output images using different radii (50, 85, 120) of inverse filtering

```

1 def filter(image_fshift,r,k):
2     output=image_fshift
3     for u in range(600):
4         for v in range(600):
5             if ((u-300)**2+(v-300)**2)**0.5 < r:
6                 output[u][v]=image_fshift[u][v]/H(u,v,k)
7     return output
8
9 #fft
10 img = cv2.imread('Bird 2 degraded.tif',0)
11 f = np.fft.fft2(img)
12 fshift = np.fft.fftshift(f)
13 # filter & inverse fft to get the image back
14 img_back_50 = np.fft.ifftshift(filter(fshift,50,0.001))
15 img_back_50 = np.fft.ifft2(img_back_50)
16 img_back_50 = np.abs(img_back_50)
17

```

Saving...



```

19 img = cv2.imread('Bird 2 degraded.tif',0)
20 f = np.fft.fft2(img)
21 fshift = np.fft.fftshift(f)
22 # filter & inverse fft to get the image back
23 img_back_85 = np.fft.ifftshift(filter(fshift,85,0.001))
24 img_back_85 = np.fft.ifft2(img_back_85)
25 img_back_85 = np.abs(img_back_85)
26
27 #fft
28 img = cv2.imread('Bird 2 degraded.tif',0)
29 f = np.fft.fft2(img)
30 fshift = np.fft.fftshift(f)
31 # filter & inverse fft to get the image back
32 img_back_120 = np.fft.ifftshift(filter(fshift,120,0.001))
33 img_back_120 = np.fft.ifft2(img_back_120)
34 img_back_120 = np.abs(img_back_120)
35
36
37
38 #plot
39 fig, (ax1, ax2 ,ax3 ,ax4) = plt.subplots(figsize=(25,25), nrows=1, ncols=4)
40 ax1.imshow(img, cmap = 'gray')
41 ax1.set_title('Bird 2 degraded.tif')
42 ax1.set_xticks([])
43 ax1.set_yticks([])
44
45 ax2.imshow(img_back_50, cmap = 'gray')

```

Saving...

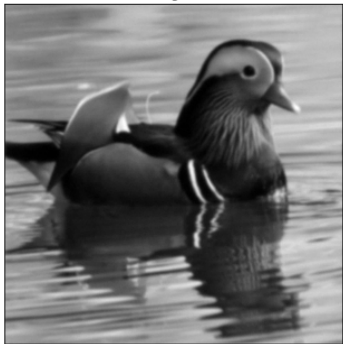
×

=50')

```
48 ax2.set_yticks([])
49
50 ax3.imshow(img_back_85, cmap = 'gray')
51 ax3.set_title('radii=85')
52 ax3.set_xticks([])
53 ax3.set_yticks([])
54
55 ax4.imshow(img_back_120, cmap = 'gray')
56 ax4.set_title('radii=120')
57 ax4.set_xticks([])
58 ax4.set_yticks([])
```

[]

Bird 2 degraded.tif



radii=50



radii=85



radii=120



▼ Model parameter k

choose $k=0.001$ (mild turbulence)

Saving...



```
1 img_without_degraded = cv2.imread('Bird 2.tif',0)
2
3 #fft
4 img = cv2.imread('Bird 2 degraded.tif',0)
5 f = np.fft.fft2(img)
6 fshift = np.fft.fftshift(f)
7 # filter & inverse fft to get the image back
8 img_back_0025 = np.fft.ifftshift(filter(fshift,85,0.0025))
9 img_back_0025 = np.fft.ifft2(img_back_0025)
10 img_back_0025 = np.abs(img_back_0025)
11
12 #fft
13 img = cv2.imread('Bird 2 degraded.tif',0)
14 f = np.fft.fft2(img)
15 fshift = np.fft.fftshift(f)
16 # filter & inverse fft to get the image back
17 img_back_001 = np.fft.ifftshift(filter(fshift,85,0.001))
18 img_back_001 = np.fft.ifft2(img_back_001)
19 img_back_001 = np.abs(img_back_001)
20
21 #fft
22 img = cv2.imread('Bird 2 degraded.tif',0)
23 f = np.fft.fft2(img)
24 fshift = np.fft.fftshift(f)
25 # filter & inverse fft to get the image back
26 img_back_00025 = np.fft.ifftshift(filter(fshift,85,0.00025))
27 img_back_00025 = np.fft.ifft2(img_back_00025)
abs(img_back_00025)
```

Saving...



```
30
31
32 #plot
33 fig, (ax1, ax2 ,ax3 ,ax4) = plt.subplots(figsize=(25,25), nrows=1, ncols=4)
34 ax1.imshow(img_without_degraded, cmap = 'gray')
35 ax1.set_title('img_without_degraded')
36 ax1.set_xticks([])
37 ax1.set_yticks([])
38
39 ax2.imshow(img_back_0025, cmap = 'gray')
40 ax2.set_title('k=0.0025 severe turbulence')
41 ax2.set_xticks([])
42 ax2.set_yticks([])
43
44 ax3.imshow(img_back_001, cmap = 'gray')
45 ax3.set_title('k=0.001 mild turbulence')
46 ax3.set_xticks([])
47 ax3.set_yticks([])
48
49 ax4.imshow(img_back_00025, cmap = 'gray')
50 ax4.set_title('k=0.00025 low turbulence')
51 ax4.set_xticks([])
52 ax4.set_yticks([])
```


[]

img_without_degraded



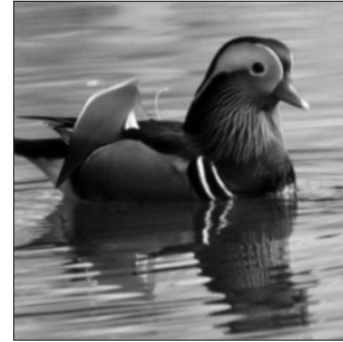
$k=0.0025$ severe turbulence



$k=0.001$ mild turbulence



$k=0.00025$ low turbulence



Saving...

