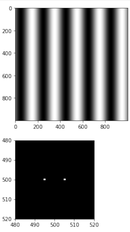
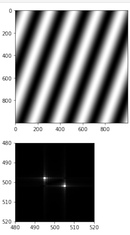
In [2]:
```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
```

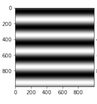
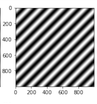
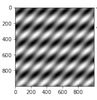
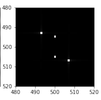
In [49]:
```
x = np.arange(-500, 500, 1)
X, Y = np.meshgrid(x, x)
wavelength = 200
grating = np.sin(2 * np.pi * X / wavelength)
plt.set_cmap('gray')
plt.imshow(grating)
plt.show()

# calculate the fourier transform of grating
ft = np.fft.ifftshift(grating)
ft = np.fft.fft2(ft)
ft = np.fft.fftshift(ft)
plt.subplot(122)
plt.imshow(abs(ft))
plt.xlim([480, 520])
plt.ylim([520, 480]) # note, order is reversed for y
plt.show()
```



In [66]:
```
x = np.arange(-500, 500, 1)
X, Y = np.meshgrid(x, x)
wavelength = 200
angle = np.pi / 9
grating = np.sin(2 * np.pi * (X * np.cos(angle) + Y * np.sin(angle)) / wavelength)
plt.set_cmap('gray')
plt.imshow(grating)
plt.show()

# calculate the fourier transform of grating
ft = np.fft.ifftshift(grating)
ft = np.fft.fft2(ft)
ft = np.fft.fftshift(ft)
plt.subplot(122)
plt.imshow(abs(ft))
plt.xlim([480, 520])
plt.ylim([520, 480]) # note, order is reversed for y
plt.show()
```



In [75]:
```
x = np.arange(-500, 500, 1)
X, Y = np.meshgrid(x, x)

wavelength1 = 200
angle1 = 0
grating1 = np.sin(2 * np.pi * (Y * np.cos(angle1) + X * np.sin(angle1)) / wavelength1)

wavelength2 = 100
angle2 = np.pi / 4
grating2 = np.sin(2 * np.pi * (Y * np.cos(angle2) + X * np.sin(angle2)) / wavelength2)

plt.set_cmap("gray")
plt.subplot(121)
plt.imshow(grating1)
plt.subplot(122)
plt.imshow(grating2)
plt.show()

gratings = grating1 + grating2

# calculate the fourier transform of grating
ft = np.fft.ifftshift(gratings)
ft = np.fft.fft2(ft)
ft = np.fft.fftshift(ft)
plt.figure()
plt.subplot(121)
plt.imshow(gratings)
plt.subplot(122)
plt.imshow(abs(ft))
plt.xlim([480, 520])
plt.ylim([520, 480]) # note, order is reversed for y
plt.show()
```

In [78]:
```
img = cv.imread("boy.bmp")
```

In [80]:
```
kernel = np.ones((5,5),np.float32)/25
dst = cv.filter2D(img,-1,kernel)
blur = cv.blur(img,(5,5))

cv.imshow('original',img)
cv.imshow('filtered',dst)
cv.imshow('blured',blur)
cv.waitKey(0)
cv.destroyAllWindows()
cv.waitKey(1)
```

Out[80]:
```
-1
```

In [82]:
```
blur = cv.GaussianBlur(img,(5,5),0)
median5 = cv.medianBlur(img,5)
median9 = cv.medianBlur(img,9)
```

In [83]:
```
from random import *
mean = 0
var = 40
sigma = var ** 0.5
gaussian = np.random.normal(mean, sigma, (img.shape[0],img.shape[1]))
noisy_image=img
noisy_image[:, :, 0] = img[:, :, 0] + gaussian
noisy_image[:, :, 1] = img[:, :, 1] + gaussian
noisy_image[:, :, 2] = img[:, :, 2] + gaussian
```

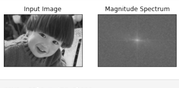
In [84]:
```
#cv2.imshow('gaussian noise',noisy_image)
median5 = cv.medianBlur(noisy_image,5)
median9 = cv.medianBlur(noisy_image,9)

cv.imshow('median5 filtered',median5)
cv.imshow('median9 filtered',median9)
cv.imshow('noisy imaged',noisy_image)
cv.waitKey(0)
cv.destroyAllWindows()
cv.waitKey(1)
```

Out[84]:
```
-1
```

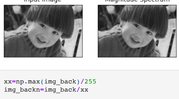
In [85]:
```
img = cv.imread('boy.bmp',0)
img=cv.cvtColor(noisy_image, cv.COLOR_BGR2GRAY)
cv.imshow('gray',img)
cv.waitKey(0)
cv.destroyAllWindows()
cv.waitKey(1)
```

Out[85]:
```
-1
```

In [86]:
```
dft = cv.dft(np.float32(img),flags = cv.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
magnitude_spectrum = 20*np.log(cv.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))
plt.subplot(121),plt.imshow(img, cmap = 'gray')

plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



In [88]:
```
rows, cols = img.shape
crow,ccol = int(rows/2) , int(cols/2)

# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows,cols,2),np.uint8)
mask[crow-60:crow+60, ccol-60:ccol+60] = 1

# apply mask and inverse DFT
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_back = cv.idft(f_ishift)
img_back = cv.magnitude(img_back[:,:,0],img_back[:,:,1])

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



In [89]:
```
xx=np.max(img_back)/255
img_backn=img_back/xx
```

In [90]:
```
ximg=img_backn.astype(int)
```

In [95]:
```
cv.imshow('gray',ximg)
cv.waitKey(0)
cv.destroyAllWindows()
cv.waitKey(1)
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
Input In [95], in <cell line: 1>()
----> 1 cv.imshow('gray',ximg)
      2 cv.waitKey(0)
      3 cv.destroyAllWindows()

error: OpenCV(4.5.4) ../modules/highgui/src/precomp.hpp:155: error: (-215:Assertion failed) src_depth != CV_16F && src_depth != CV_32S in function 'convertToShow'
```

In [93]:
```
cv.imshow('gray',img)
cv.waitKey(0)
cv.destroyAllWindows()
cv.waitKey(1)
```

Out[93]:
```
-1
```

In [94]:
```
np.max(img_back)
```

Out[94]:
```
102302900.0
```

In [ ]: