Fourier Transform

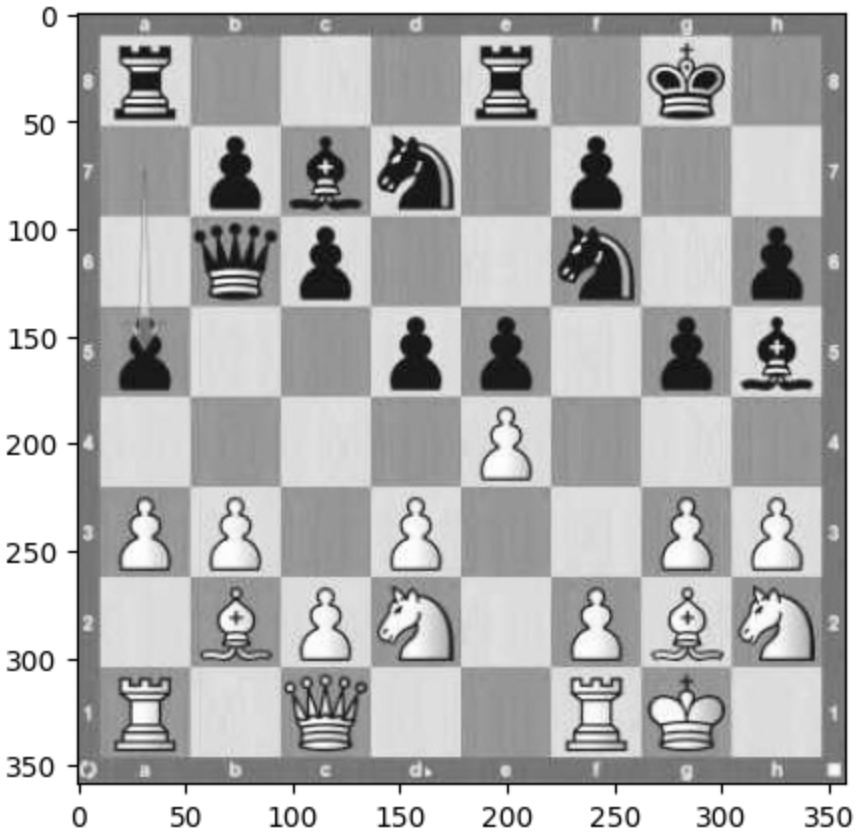
In [254]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

- **Low-pass filter** được sử dụng để loại bỏ nhiễu và giảm độ nét của hình ảnh hoặc tín hiệu. Nó làm mờ hoặc làm giảm độ biến đổi nhanh chói trong ảnh hoặc tín hiệu, giúp tạo ra một phiên bản "mượt" và giảm nhiễu hơn của dữ liệu đầu vào.
- **High-pass filter** được sử dụng để tách ra các thành phần tần số cao (biến đổi nhanh) trong hình ảnh hoặc tín hiệu, trong khi loại bỏ các thành phần tần số thấp (biến đổi chậm). Nó thường được sử dụng để nổi bật các đặc điểm cụ thể hoặc cạnh trong hình ảnh hoặc tín hiệu.

1. Low pass frequency by using Distance smooth function

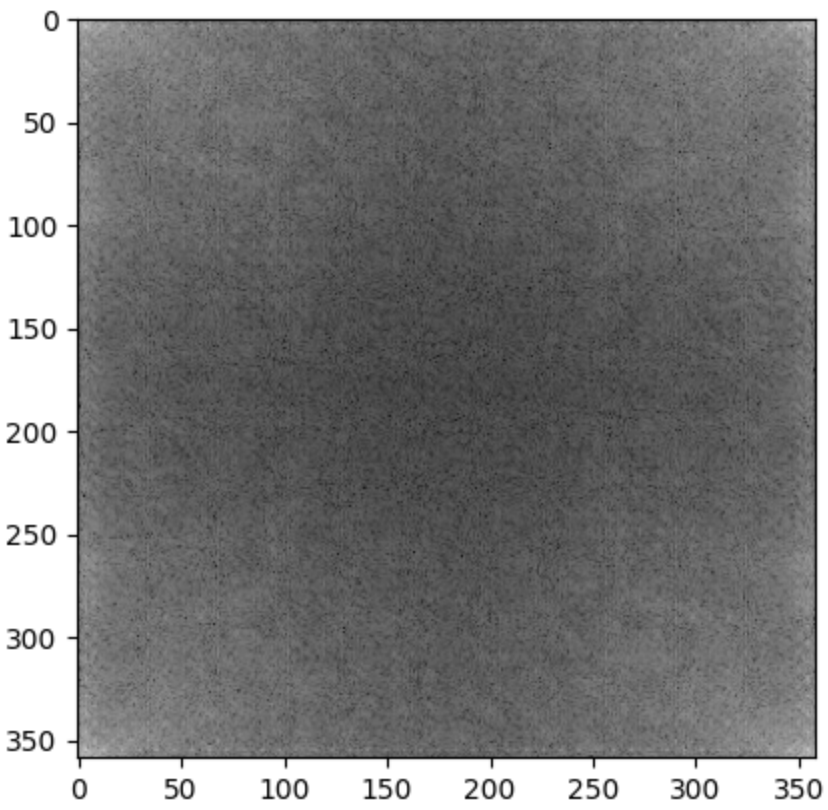
In [255]:
```
img = cv2.imread('chest.png')
#(1) chuyển sang gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.figure(figsize = (5, 5))
plt.imshow(gray, cmap = 'gray')
```

Out[255]: <matplotlib.image.AxesImage at 0x7f47f99211e0>



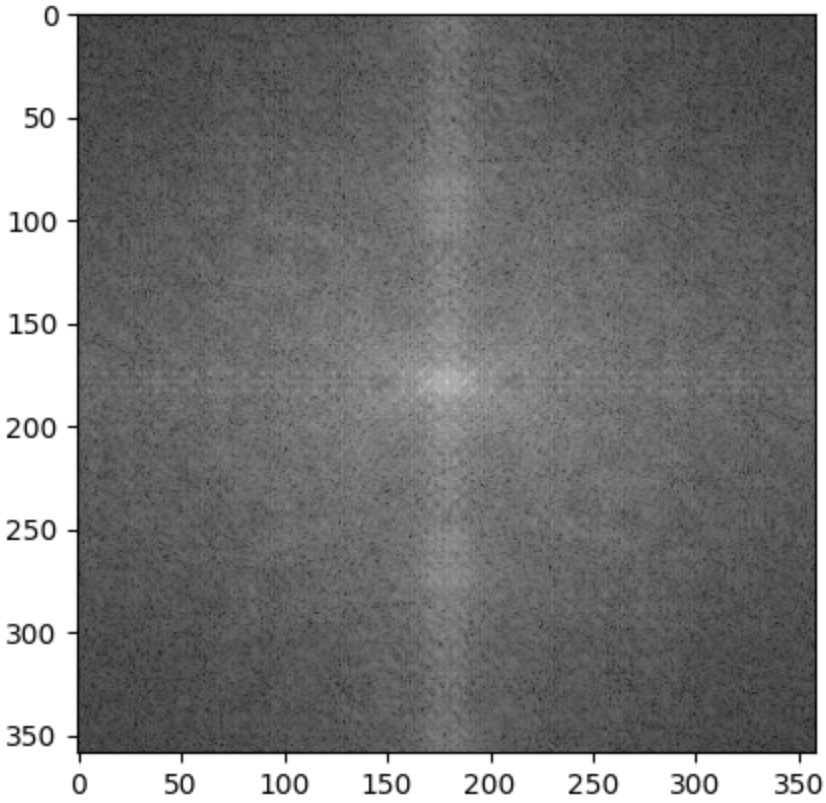
In [256]:
```
#(2) Fourier transformation
f= np.fft.fft2(gray)
plt.imshow(np.log(np.abs(f)), cmap ='gray')
```

Out[256]: <matplotlib.image.AxesImage at 0x7f47f8f91cc0>



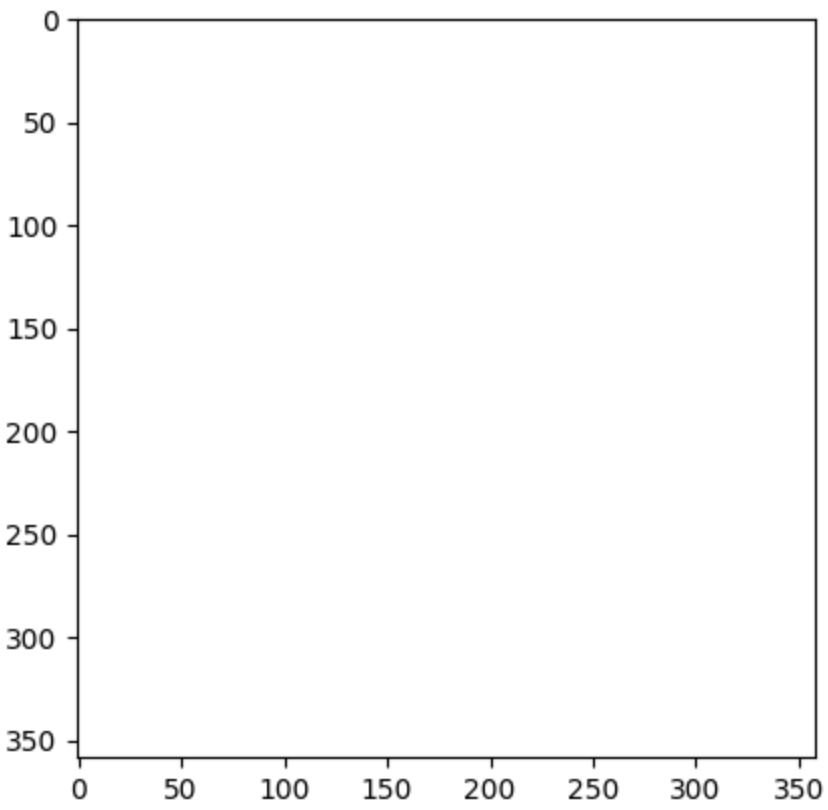
In [257]:
```
#(3) frequency shift
shifted_f = np. fft.fftshift(f)
plt.imshow(np.log(np.abs (shifted_f)),cmap ='gray')
```

Out[257]: <matplotlib.image.AxesImage at 0x7f47f8e0e260>



In [258]:
```
#(4a) low pass frequency
low_pass = np.ones (shape = gray.shape)
plt.imshow(low_pass, cmap = 'gray', vmax=1, vmin= 0)
```

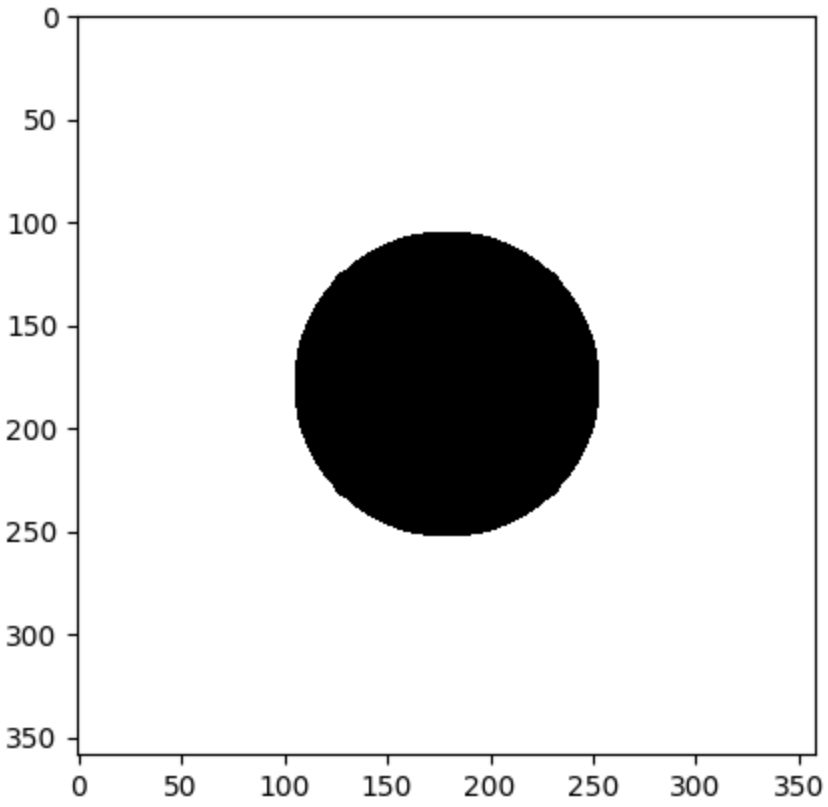
Out[258]: <matplotlib.image.AxesImage at 0x7f47f8e8ead0>



In [259]:
```
# (4b)
# lấy tâm bức ảnh
xc = gray.shape[1] // 2
yc = gray.shape[0] // 2
Do = 75
#bán kính :Do
# tính khoang cách từ 1 điểm đến tâm với công thức
# sqrt( (x-xc)**2 + (y-yc)**2)
def d(x, y):
  return np.sqrt((x-xc)**2+ (y-yc)**2)
for x in range (gray.shape[1]):
  for y in range (gray.shape[0]):
    if d(x,y) > Do:
      low_pass [y,x] = 0
f=shifted_f*low_pass
plt.imshow(np.log(np.abs(low_pass)), cmap ='gray')
```

```
<ipython-input-259-d0b0df90e730>:16: RuntimeWarning: divide by zero encounter
ed in log
  plt.imshow(np.log(np.abs(low_pass)), cmap ='gray')
```

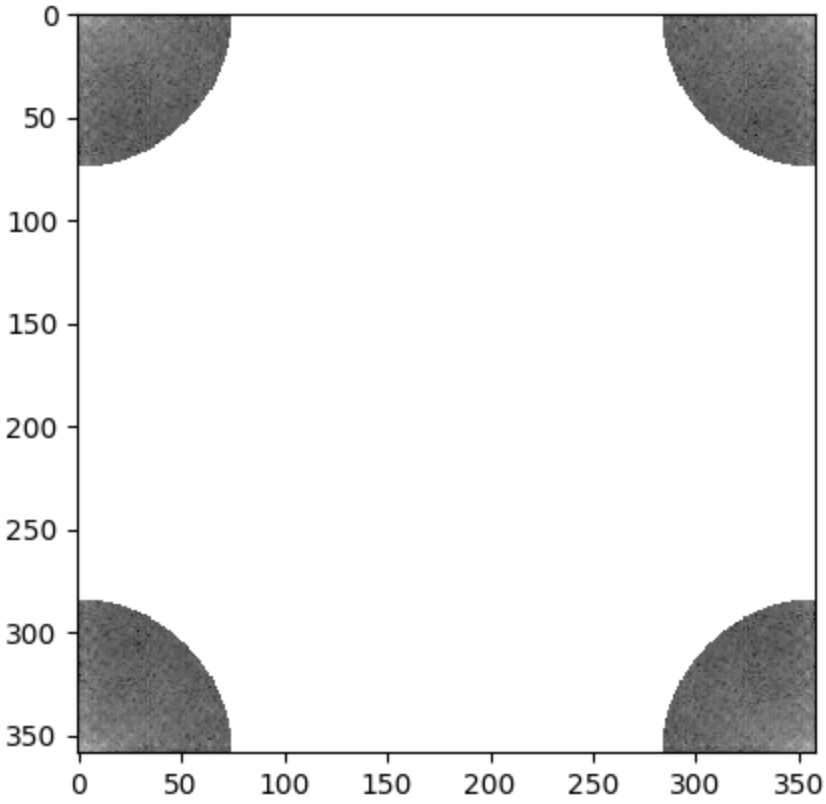
Out[259]: <matplotlib.image.AxesImage at 0x7f47f8cf27d0>



In [260]:
```
#(5) invert shift
f = np.fft.ifftshift(f)
plt.imshow(np.log(np.abs(f)), cmap='gray')
```

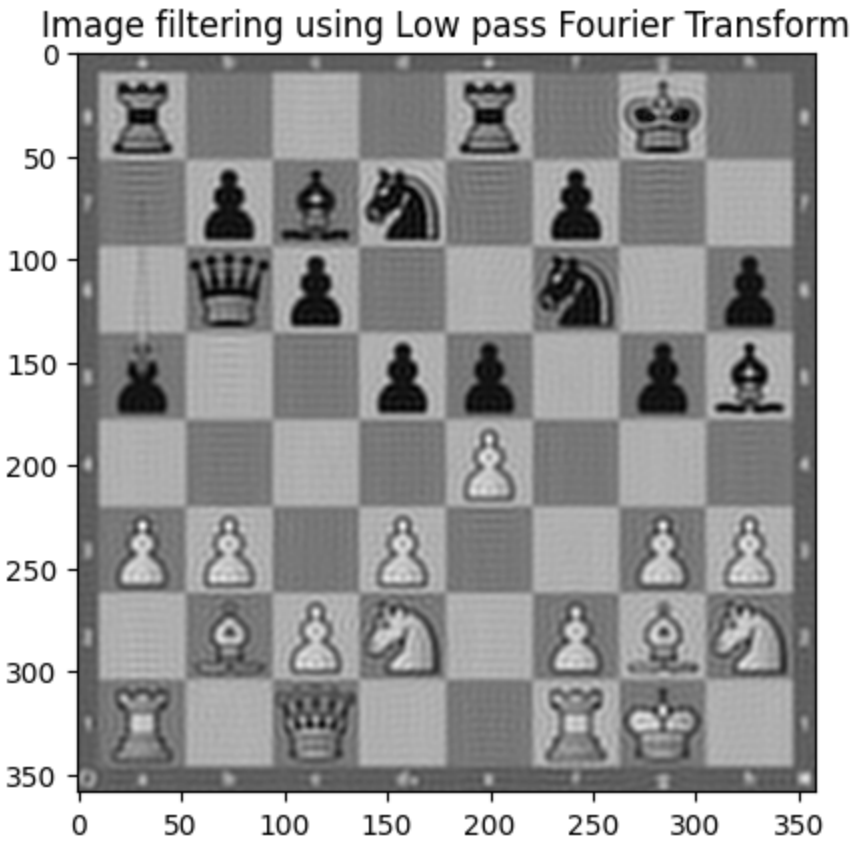
```
<ipython-input-260-6c9184b88ca8>:3: RuntimeWarning: divide by zero encountere
d in log
  plt.imshow(np.log(np.abs(f)), cmap='gray')
```

Out[260]: <matplotlib.image.AxesImage at 0x7f47f8d97a90>



In [261]:
```
#(6) invert fourier transform
new_img_DSF = np.abs(np.fft.ifft2(f))
plt.title("Image filtering using Low pass Fourier Transform")
plt.imshow(new_img_DSF, cmap = 'gray')
```

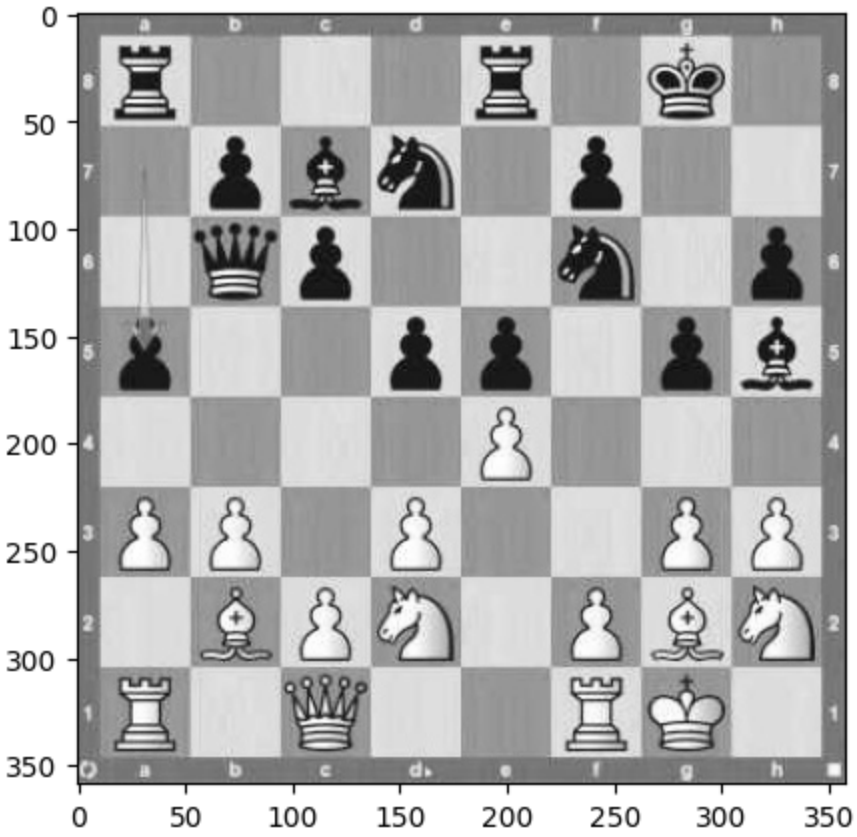
Out[261]: <matplotlib.image.AxesImage at 0x7f47f8c0ab00>



2. High pass frequency by using Distance smooth function

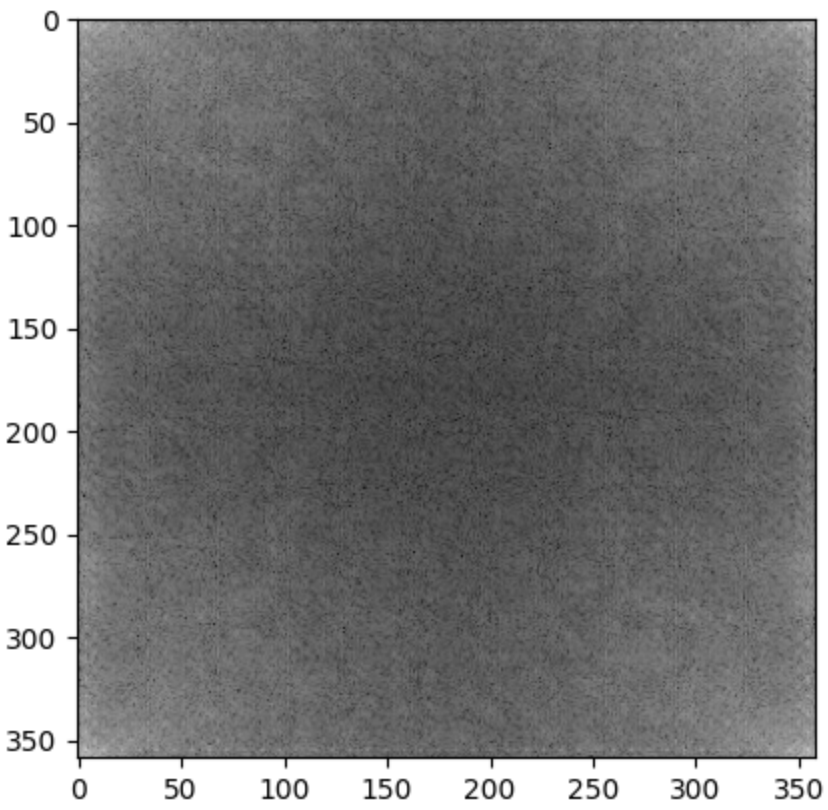
In [262]:
```
img = cv2.imread('chest.png')
#(1) chuyển sang gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.figure(figsize = (5, 5))
plt.imshow(gray, cmap = 'gray')
```

Out[262]: <matplotlib.image.AxesImage at 0x7f47f8ca4850>



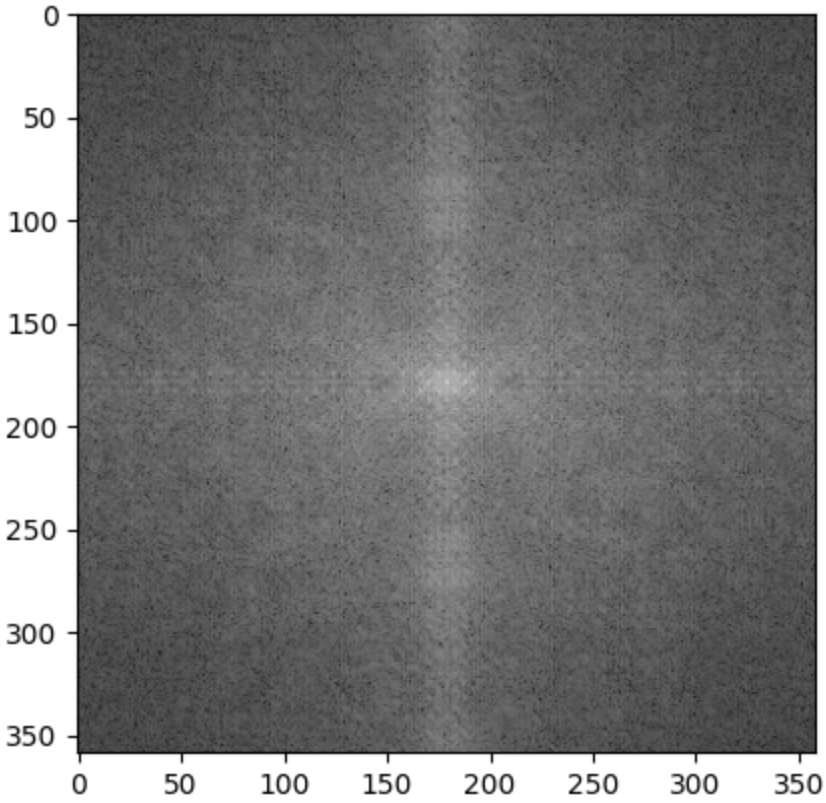
In [263]:
```
#(2) Fourier transformation
f= np.fft.fft2(gray)
plt.imshow(np.log(np.abs(f)), cmap ='gray')
```

Out[263]: <matplotlib.image.AxesImage at 0x7f47f8b287f0>



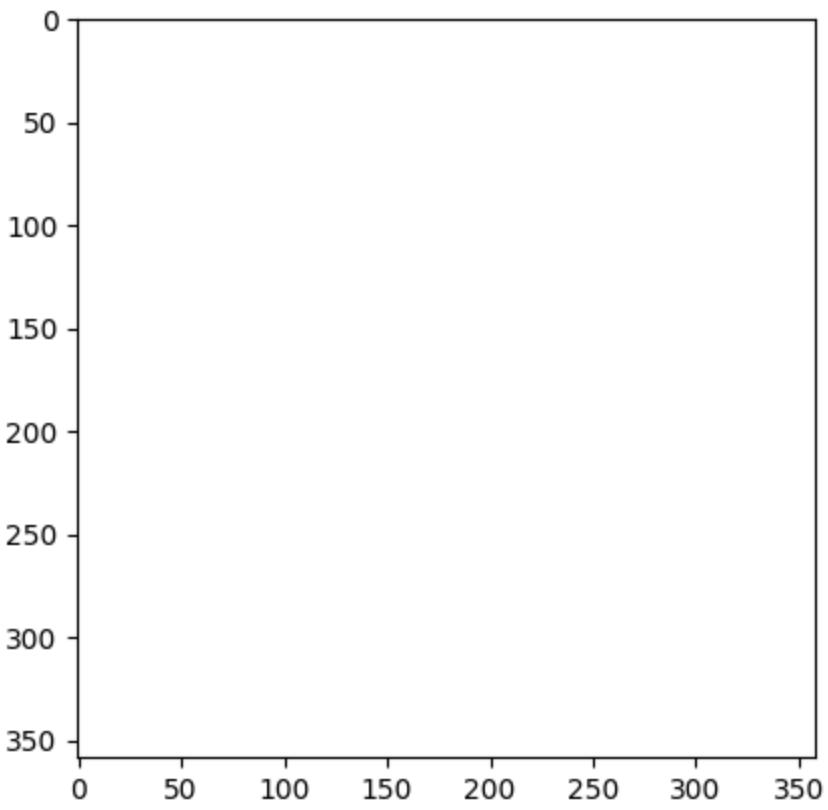
In [264]:
```
#(3) frequency shift
shifted_f = np. fft.fftshift(f)
plt.imshow(np.log(np.abs (shifted_f)),cmap ='gray')
```

Out[264]: <matplotlib.image.AxesImage at 0x7f47f8ba4be0>



In [265]:
```
#(4a) high pass frequency
high_pass = np.ones (shape = gray.shape)
plt.imshow(high_pass, cmap = 'gray', vmax=1, vmin= 0)
```

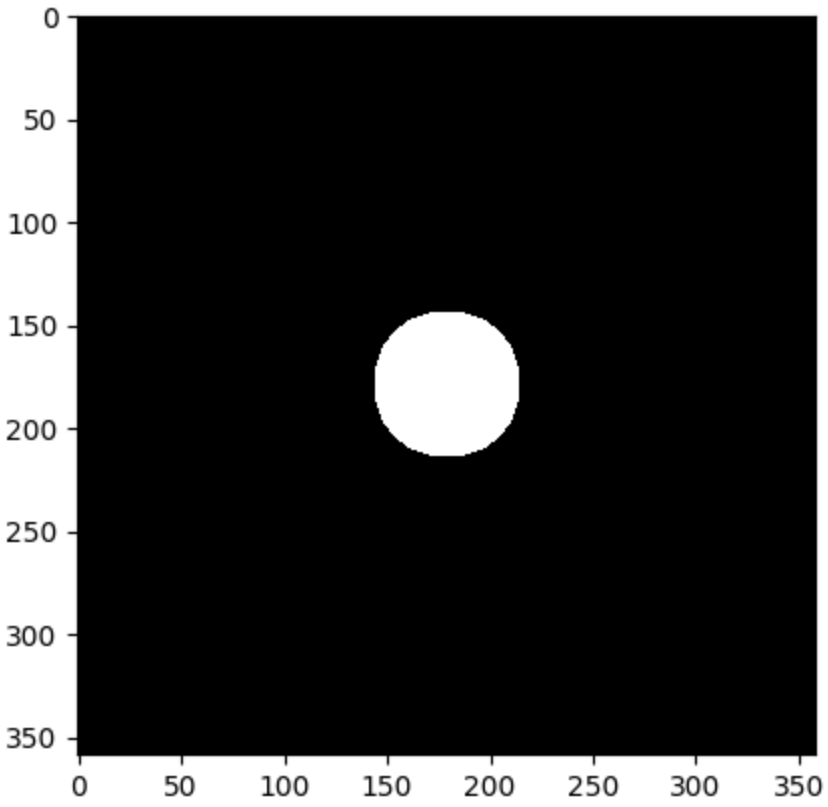
Out[265]: <matplotlib.image.AxesImage at 0x7f47f8a259c0>



In [266]:
```
# (4b)
# lấy tâm bức ảnh
xc = gray.shape[1] // 2
yc = gray.shape[0] // 2
Do = 35
#bán kính :Do
# tính khoang cách từ 1 điểm đến tâm với công thức
# sqrt( (x-xc)**2 + (y-yc)**2)
def d(x, y):
  return np.sqrt((x-xc)**2+ (y-yc)**2)
for x in range (gray.shape[1]):
  for y in range (gray.shape[0]):
    if d(x,y) < Do:
      high_pass [y,x] = 0
f=shifted_f*high_pass
plt.imshow(np.log(np.abs(high_pass)), cmap ='gray')
```

```
<ipython-input-266-421bbb4168ea>:16: RuntimeWarning: divide by zero encounter
ed in log
  plt.imshow(np.log(np.abs(high_pass)), cmap ='gray')
```

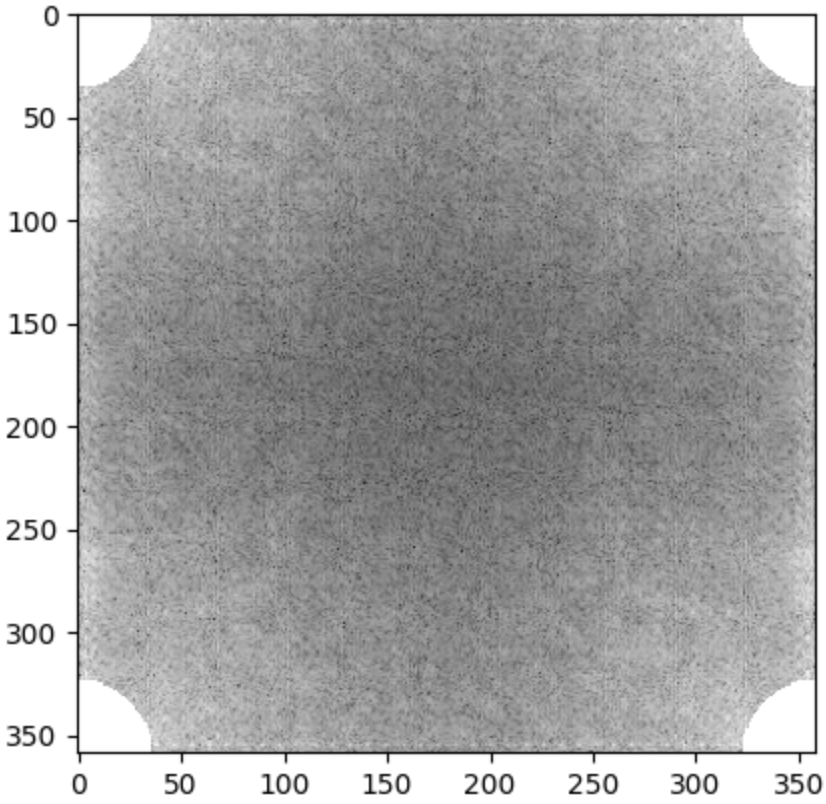
Out[266]: <matplotlib.image.AxesImage at 0x7f47f8a9ded0>



In [267]:
```
#(5) invert shift
f = np.fft.ifftshift(f)
plt.imshow(np.log(np.abs(f)), cmap='gray')
```

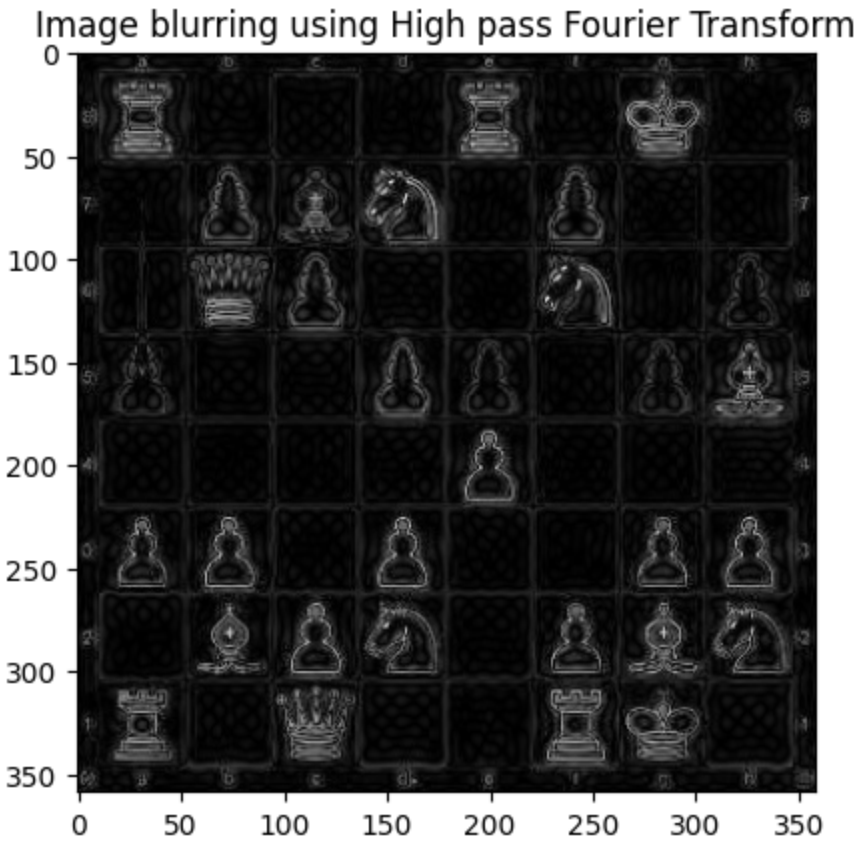
```
<ipython-input-267-6c9184b88ca8>:3: RuntimeWarning: divide by zero encountere
d in log
  plt.imshow(np.log(np.abs(f)), cmap='gray')
```

Out[267]:
```
<matplotlib.image.AxesImage at 0x7f47fbad0160>
```



In [268]:
```
#(6) invert fourier transform
new_img = np.abs (np.fft.ifft2(f))
plt.title("Image blurring using High pass Fourier Transform")
plt.imshow(new_img, cmap = 'gray')
```

Out[268]: <matplotlib.image.AxesImage at 0x7f47fa28e1a0>

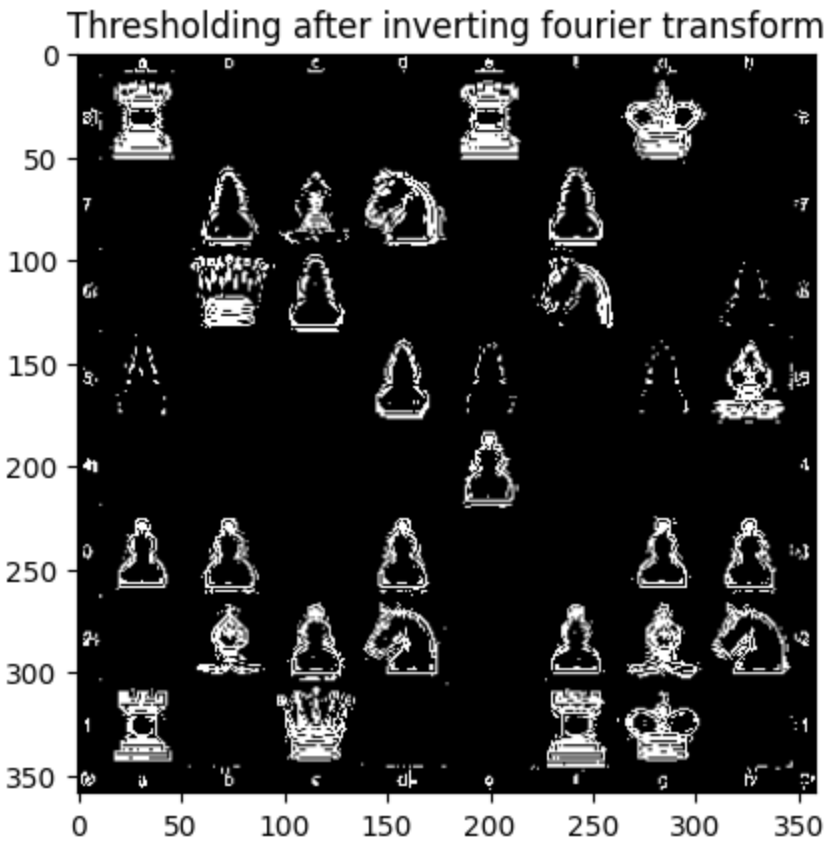


In [269]:
```
# Đặt ngưỡng
thresh = 47

# Sử dụng hàm cv2.threshold() để loại bỏ những điểm có độ sáng nhỏ hơn ngưỡng
ret, thresh_img_DSF = cv2.threshold(new_img, thresh, 255, cv2.THRESH_BINARY)

# Hiển thị ảnh sau khi đã áp dụng ngưỡng
plt.title("Thresholding after inverting fourier transform")
plt.imshow(thresh_img_DSF, cmap = 'gray')
```

Out[269]: <matplotlib.image.AxesImage at 0x7f47f97e0040>

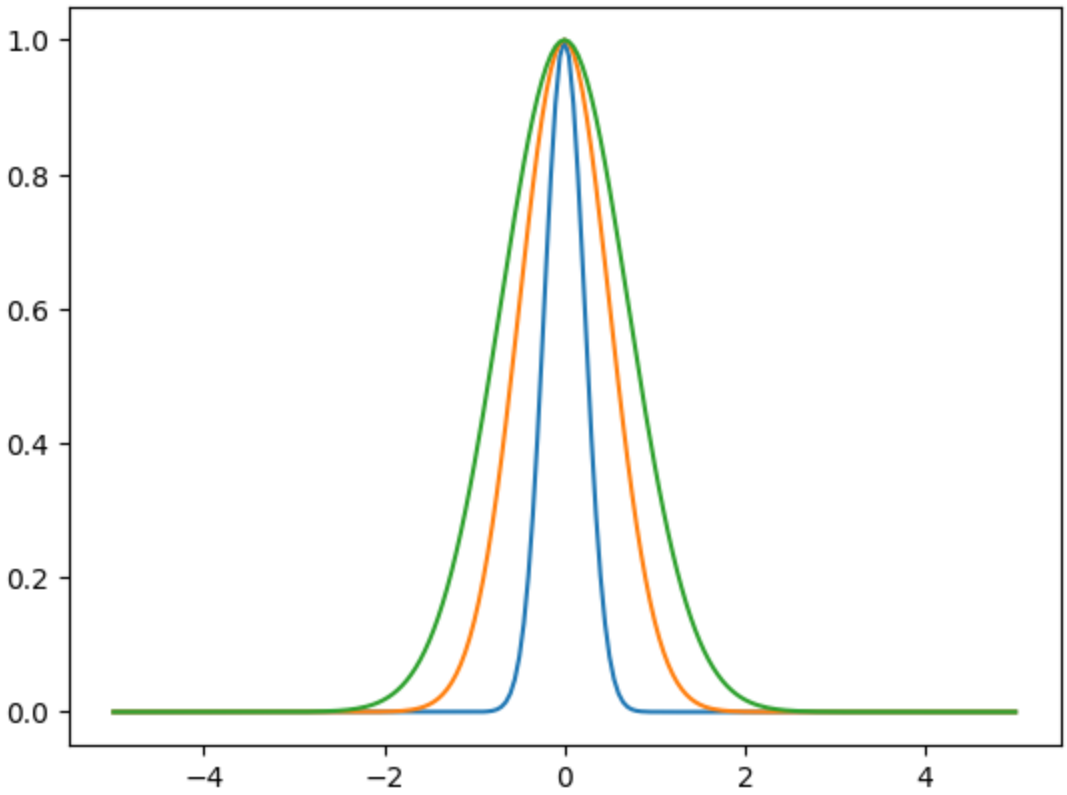


In [270]:
```
# # Erosion followed by dilation để lọc gợn ảnh
# kernel = np.ones((2,1),np.uint8) # Lọc theo chiều ngang
# opening1 = cv2.morphologyEx(thresh_img_DSF, cv2.MORPH_OPEN, kernel)
# kernel = np.ones((1,2),np.uint8) # Lọc theo chiều dọc
# opening2 = cv2.morphologyEx(thresh_img_DSF, cv2.MORPH_OPEN, kernel)
# # Cộng 2 ảnh vừa lọc
# opening_DSF = cv2.addWeighted(opening1, 0.5, opening2, 0.5, 0.0)
# plt.title("Erosion followed by dilation")
# plt.imshow(opening_DSF, cmap = 'gray')
```

Draw frequency curves and plot 3D function

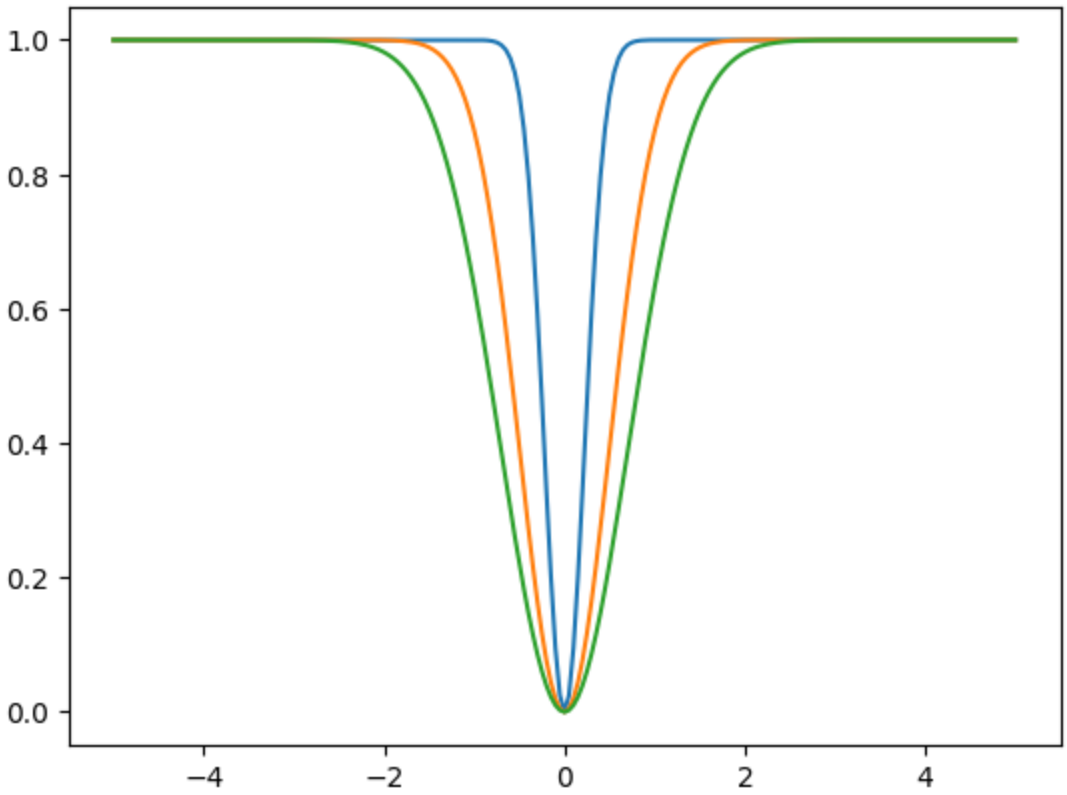
In [271]:
```
for sigma in [0.1, 0.5, 1.0]:
  x = np.linspace (-5, 5, 201)
  y = np.exp(-x**2 / sigma)
  plt.plot(x,y, label="".format(sigma))
plt.legend
```

Out[271]:
```
<function matplotlib.pyplot.legend(*args, **kwargs)>
```

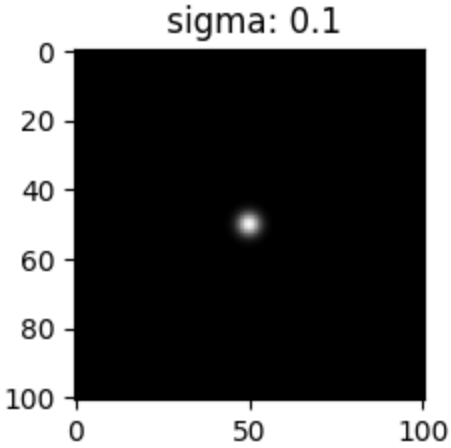


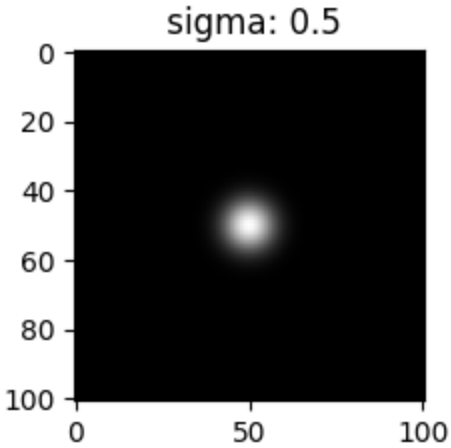
In [272]:
```
for sigma in [0.1, 0.5, 1.0]:
  x = np. linspace (-5, 5, 201)
  y = 1 - np.exp(-x**2 / sigma)
  plt.plot(x,y, label = "".format(sigma))
plt.legend
```

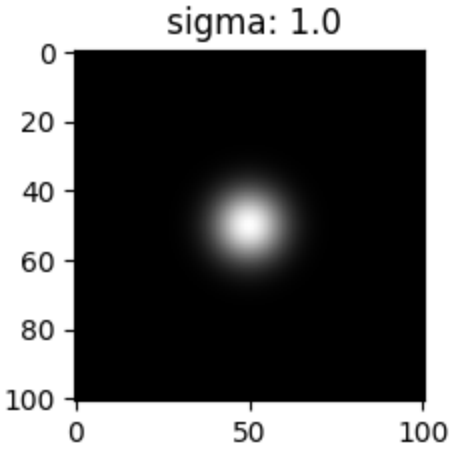
Out[272]:
```
<function matplotlib.pyplot.legend(*args, **kwargs)>
```

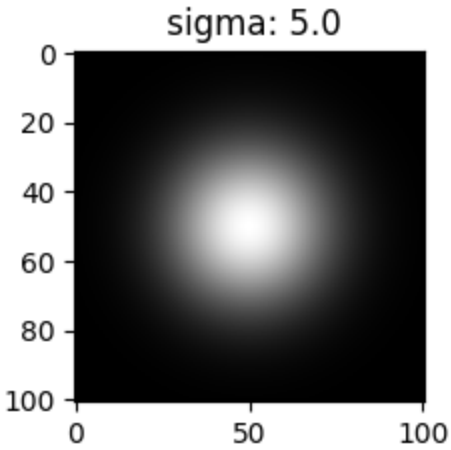


In [273]:
```
#plot 3D funtion y = exp(-(x**2 + y**2) / sigma)
x= np.linspace (-5,5,101)
y = np.linspace (-5,5,101)
X, Y = np.meshgrid(x, y)
for i, sigma in enumerate([0.1, 0.5, 1.0, 5.0]):
  z = np.exp(-(X**2+ Y**2) / sigma)
  plt.figure(figsize = (5,5))
  plt.subplot(2,2,i+1)
  plt.imshow(z, cmap = 'gray')
  plt.title(f" sigma: {sigma}")
```


sigma: 0.1
0
20
40
60
80
100
0
50
100


sigma: 0.5
0
20
40
60
80
100
0
50
100

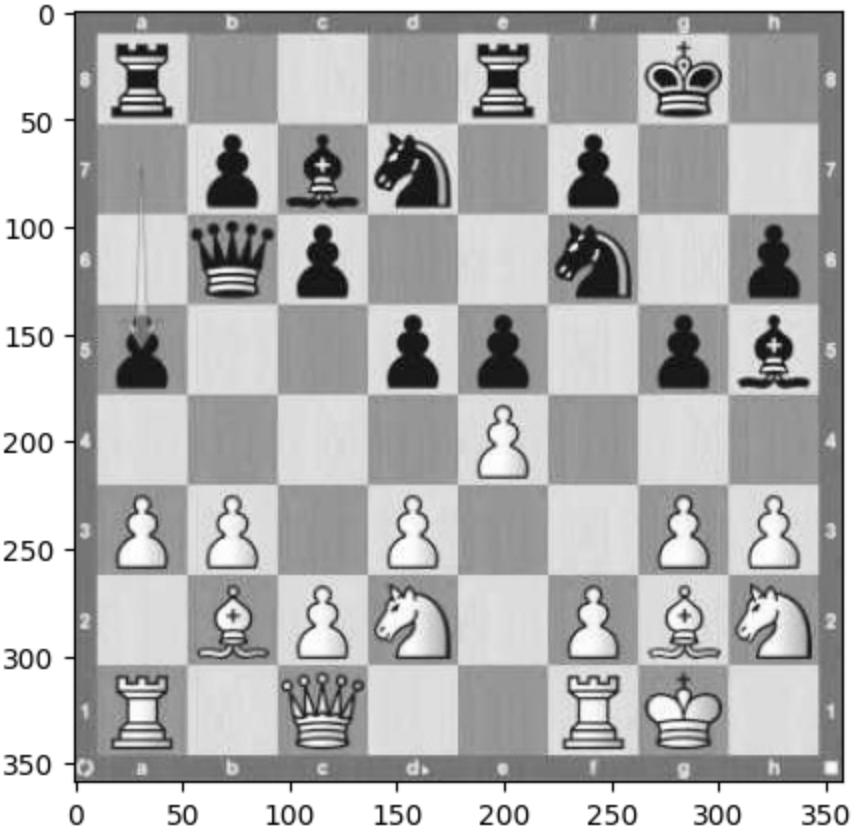

sigma: 1.0
0
20
40
60
80
100
0
50
100



3. Low pass frequency by using Gaussian smooth function

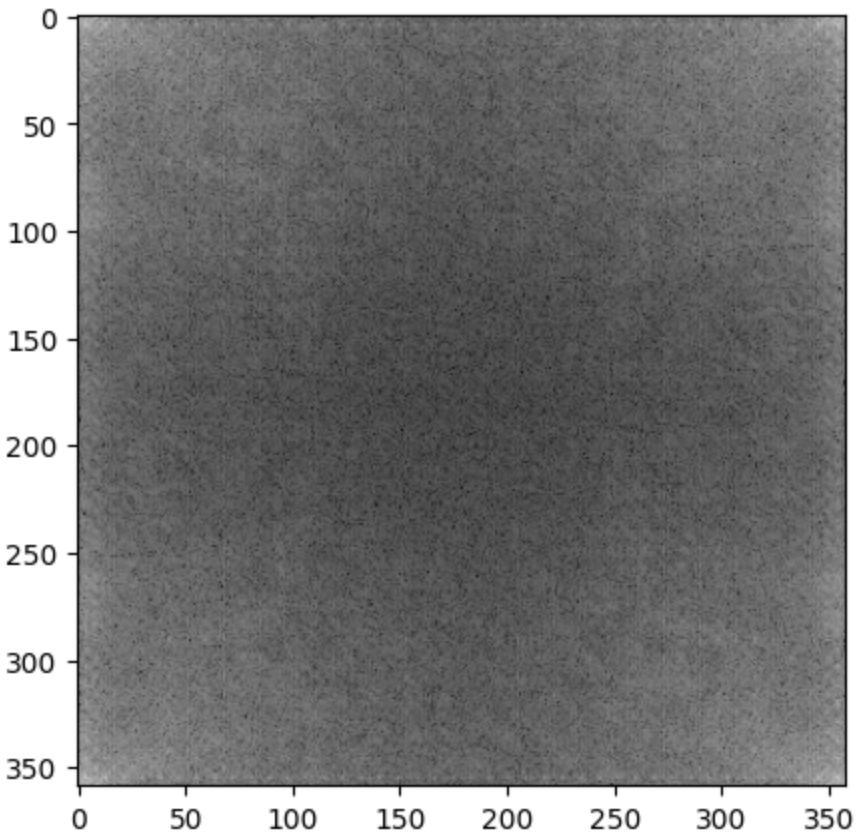
In [274]:
```
img = cv2.imread('chest.png')
#(1) chuyển sang gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.figure(figsize = (5, 5))
plt.imshow(gray, cmap = 'gray')
```

Out[274]: <matplotlib.image.AxesImage at 0x7f47f8700910>



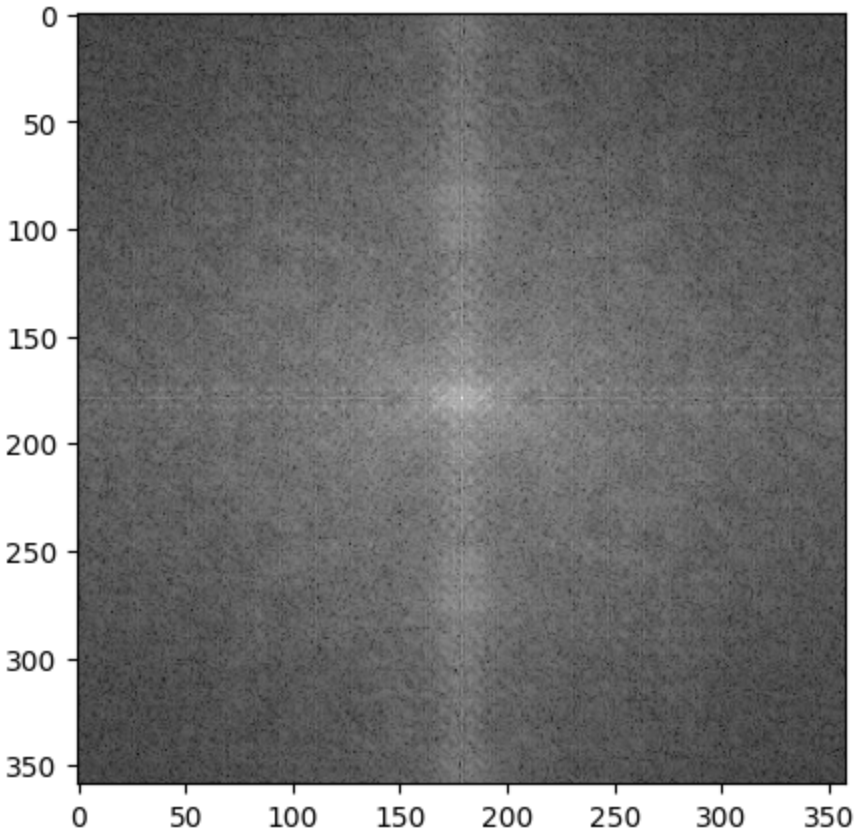
In [275]:
```
#(2) Fourier transformation
f= np.fft.fft2(gray)
plt.figure(figsize = (5, 5))
plt.imshow(np.log(np.abs (f)), cmap ='gray')
```

Out[275]: <matplotlib.image.AxesImage at 0x7f47f875f490>



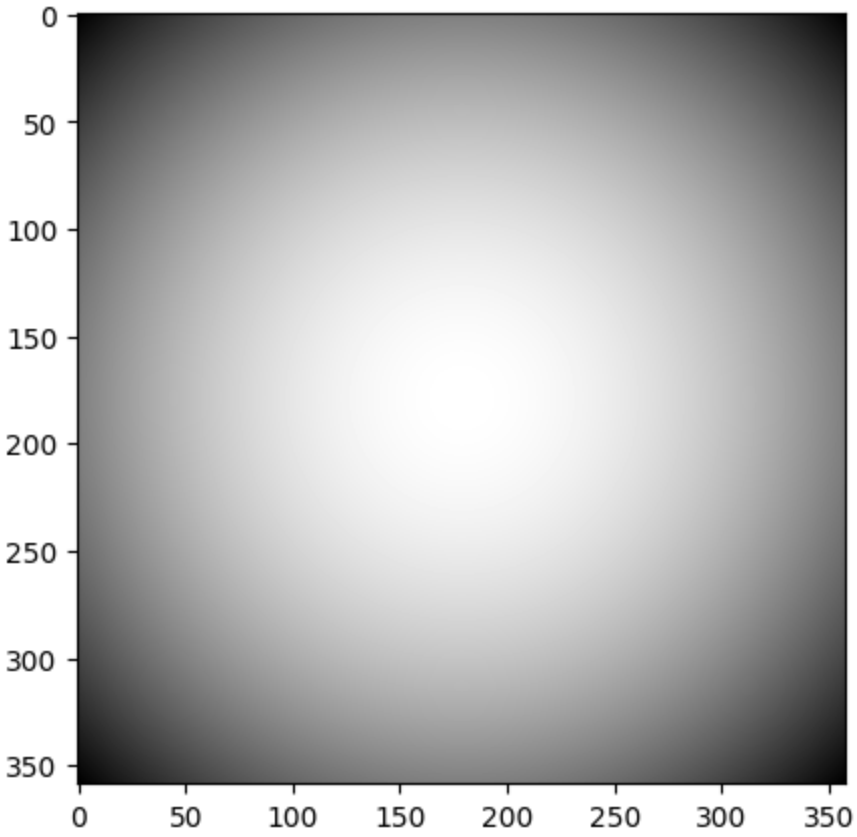
In [276]:
```
#(3) frequency shift
shifted_f = np. fft.fftshift(f)
plt.figure(figsize = (5, 5))
plt.imshow(np.log(np.abs (shifted_f)), cmap='gray')
```

Out[276]: <matplotlib.image.AxesImage at 0x7f47f85f2650>



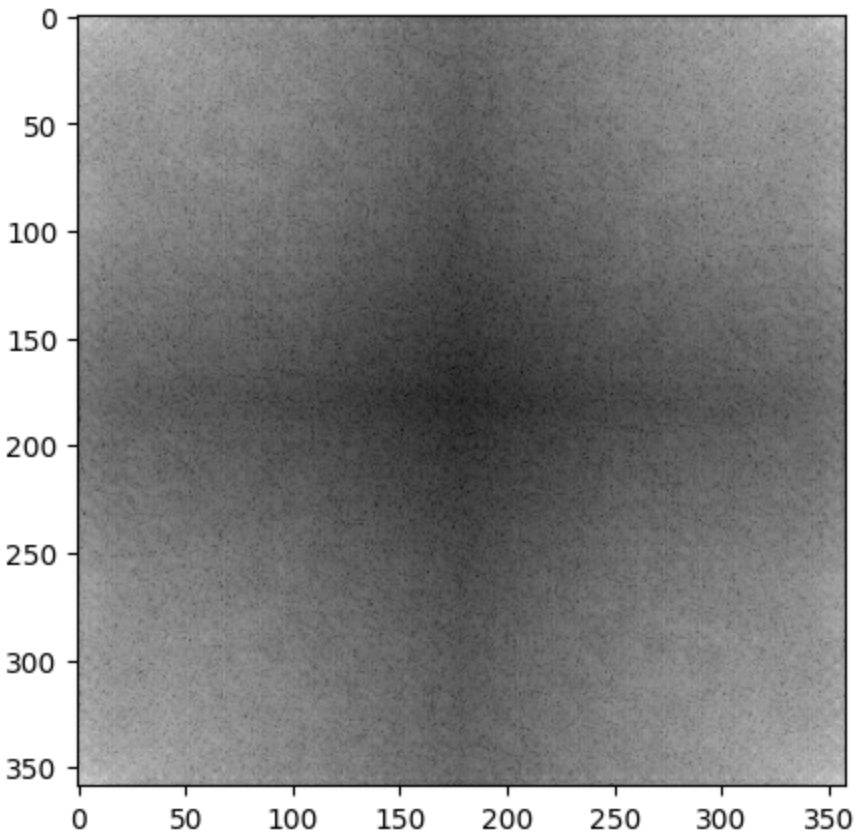
In [277]:
```
#(4) low pass frequency
low_pass = np.ones(shape = gray.shape)
#Láy tâm bức anh
xc = gray.shape[0] // 2
yc = gray.shape[1] // 2
#bán kính :Do
sigma = 75
#tính khoảng cách từ 1 điểm đến tâm với công thức : # sqrt( (x-xc)**2 + (y-yc)
**2)
def d(x, y):
  return np.sqrt((x-xc)**2+ (y-yc)**2)
for x in range (gray.shape[0]):
  for y in range (gray.shape[1]):
    low_pass [x,y] = np.exp(-d(x, y) **2 / (2*sigma**2))
f= shifted_f*low_pass
plt.figure(figsize = (5, 5))
plt.imshow(np.log(np.abs (low_pass)), cmap = 'gray')
```

Out[277]: <matplotlib.image.AxesImage at 0x7f47f85f0a30>



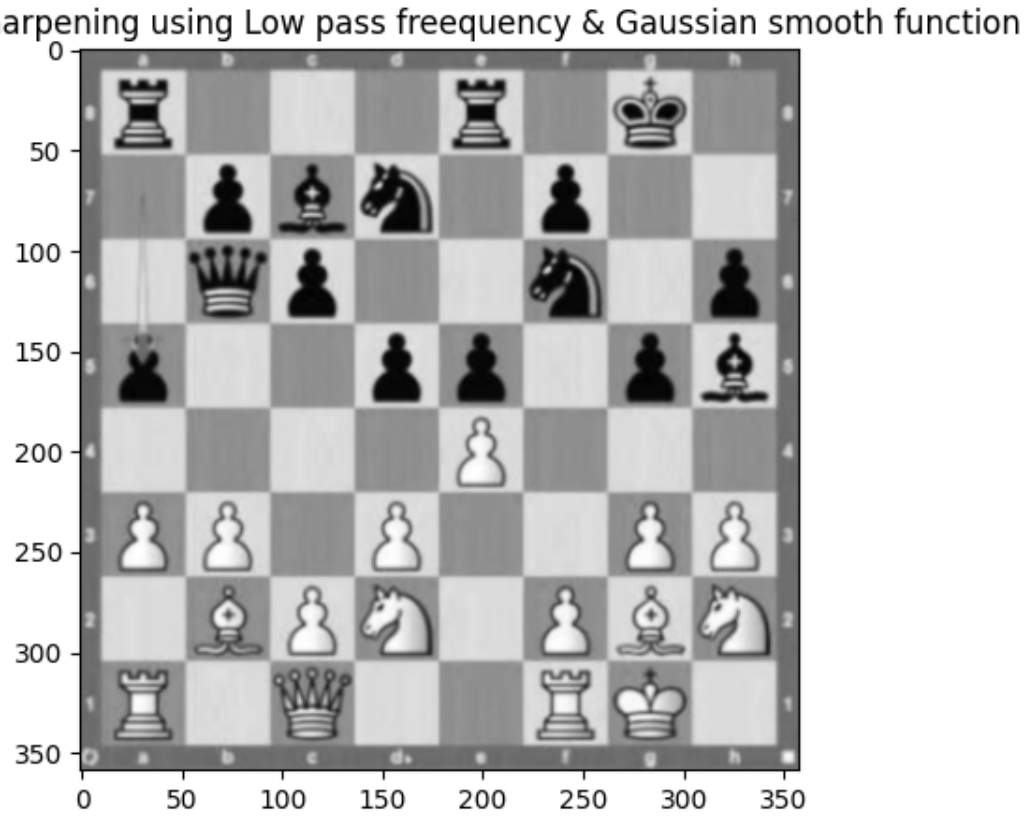
In [278]:
```
#(5) invert shift
plt.figure(figsize = (5, 5))
f = np.fft.ifftshift(f)
plt.imshow(np.log(np.abs (f)), cmap='gray')
```

Out[278]: <matplotlib.image.AxesImage at 0x7f47f84f8850>



In [279]:
```
#(6) invert fourier transform
new_img_GSF = np.abs (np.fft.ifft2(f))
plt.figure(figsize = (5, 5))
plt.title("Image sharpening using Low pass freequency & Gaussian smooth functi
on")
plt.imshow(new_img_GSF, cmap = 'gray')
```

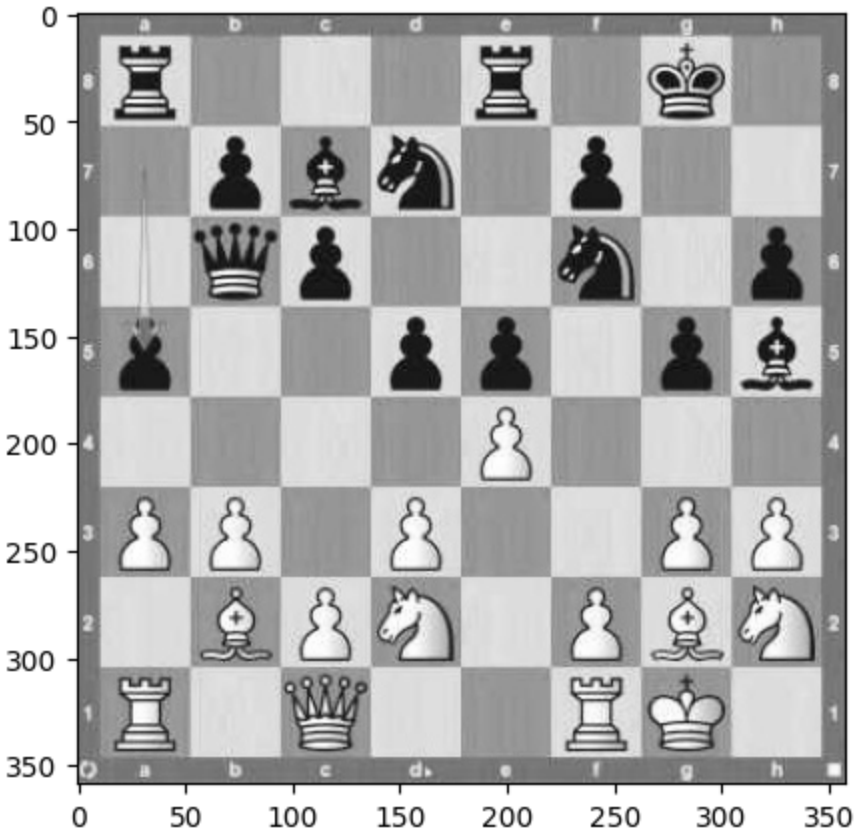
Out[279]: <matplotlib.image.AxesImage at 0x7f47f85721a0>



4. High pass frequency by using Gaussian smooth function

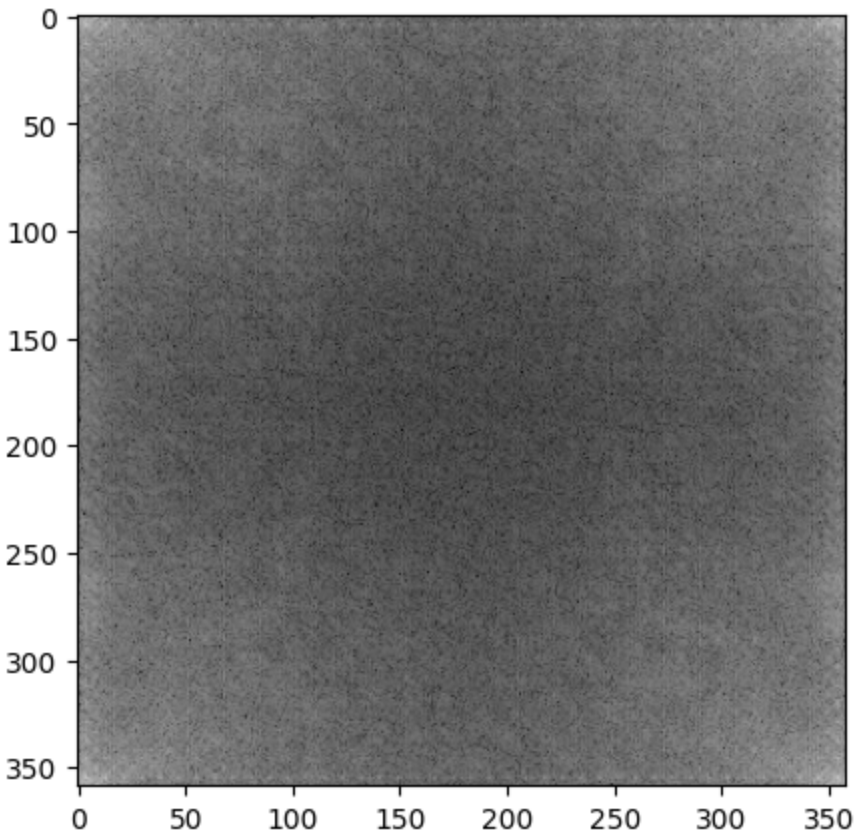
In [280]:
```
img = cv2.imread('chest.png')
#(1) chuyển sang gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.figure(figsize = (5, 5))
plt.imshow(gray, cmap = 'gray')
```

Out[280]: <matplotlib.image.AxesImage at 0x7f47f83ea9b0>



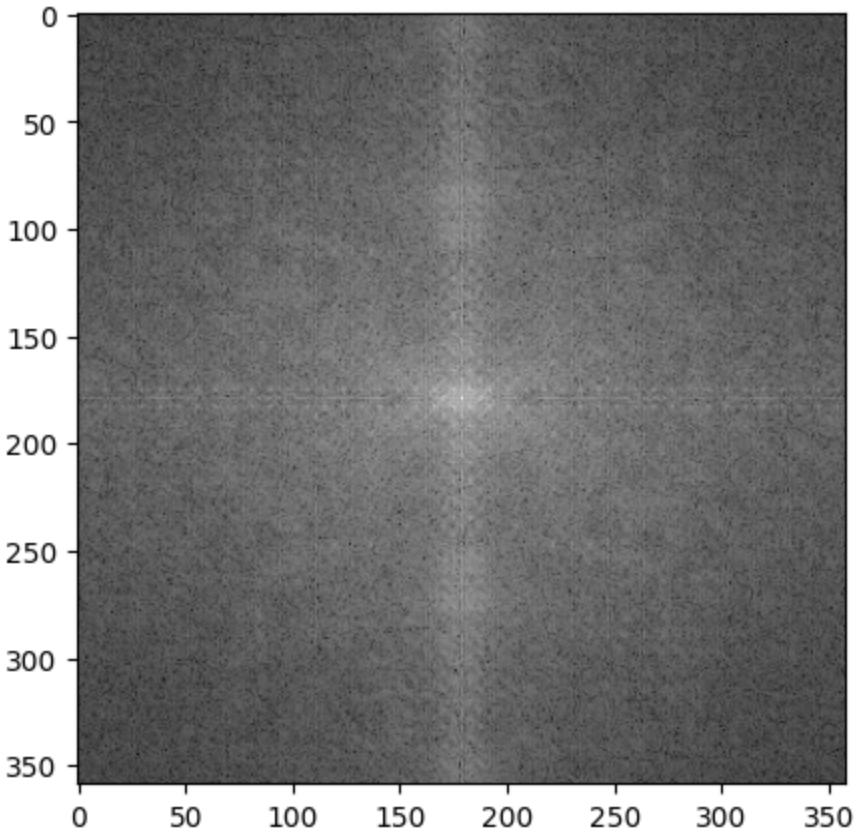
In [281]:
```
#(2) Fourier transformation
f= np.fft.fft2(gray)
plt.figure(figsize = (5, 5))
plt.imshow(np.log(np.abs (f)), cmap ='gray')
```

Out[281]: <matplotlib.image.AxesImage at 0x7f47f8466aa0>



In [282]:
```
#(3) frequency shift
shifted_f = np. fft.fftshift(f)
plt.figure(figsize = (5, 5))
plt.imshow(np.log(np.abs (shifted_f)), cmap='gray')
```

Out[282]: <matplotlib.image.AxesImage at 0x7f47f82e7820>



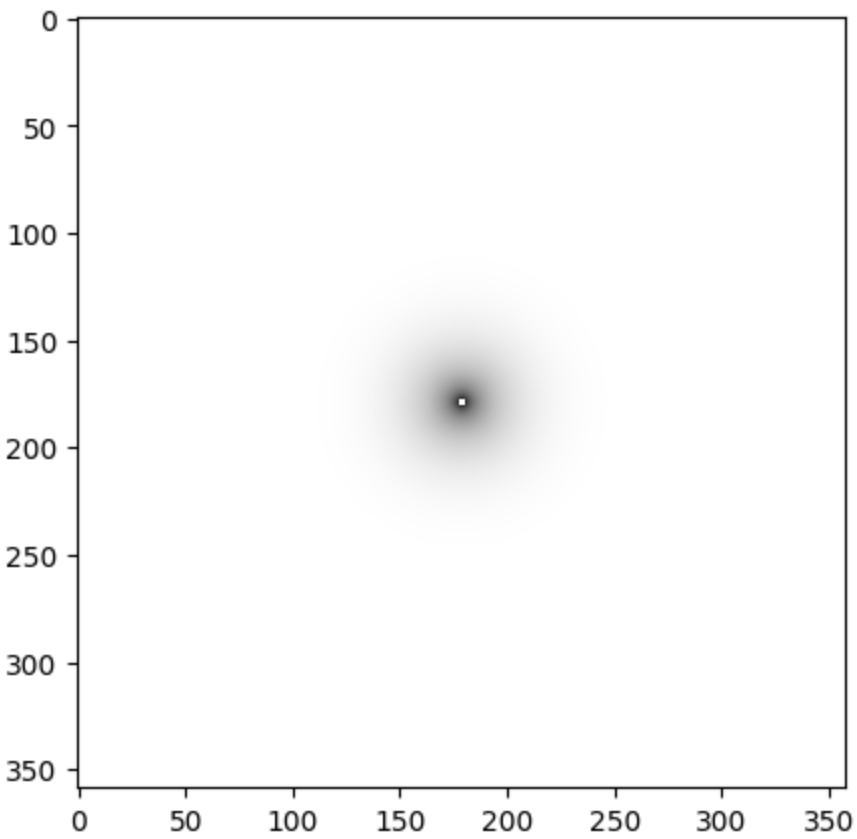
In [283]:
```
# (4) high pass frequency
high_pass = np.ones(shape=gray.shape)
# Get the center of the image
xc = gray.shape[1] // 2
yc = gray.shape[0] // 2
# Radius: Do
sigma = 25
# Calculate the distance from each point to the center using the formula: sqrt
((x-xc)**2 + (y-yc)**2)
def d(x, y):
    return np.sqrt((x - xc)**2 + (y - yc)**2)

for x in range(gray.shape[1]):
    for y in range(gray.shape[0]):
        high_pass[y, x] = 1 - np.exp(-d(x, y)**2 / (2 * sigma**2))  # Modify t
he filter equation

f = shifted_f * high_pass
plt.figure(figsize=(5, 5))
plt.imshow(np.log(np.abs(high_pass)), cmap='gray')
```

```
<ipython-input-283-235c0bbd875e>:18: RuntimeWarning: divide by zero encounter
ed in log
  plt.imshow(np.log(np.abs(high_pass)), cmap='gray')
```

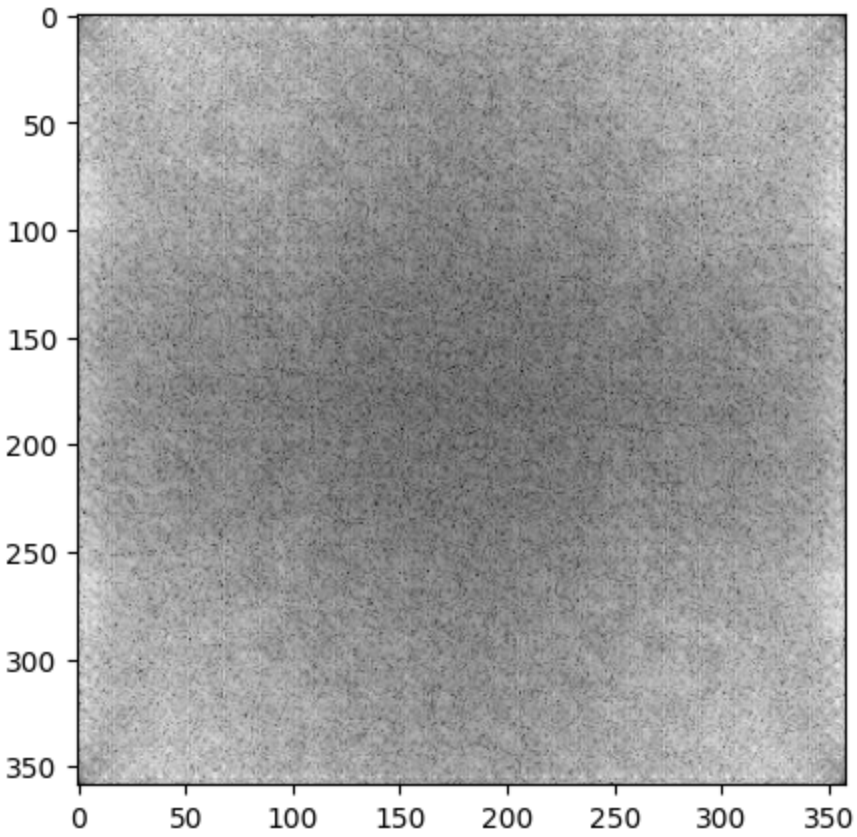
Out[283]:
```
<matplotlib.image.AxesImage at 0x7f47f8389930>
```



In [284]:
```
#(5) invert shift
plt.figure(figsize = (5, 5))
f = np.fft.ifftshift(f)
plt.imshow(np.log(np.abs (f)), cmap='gray')
```

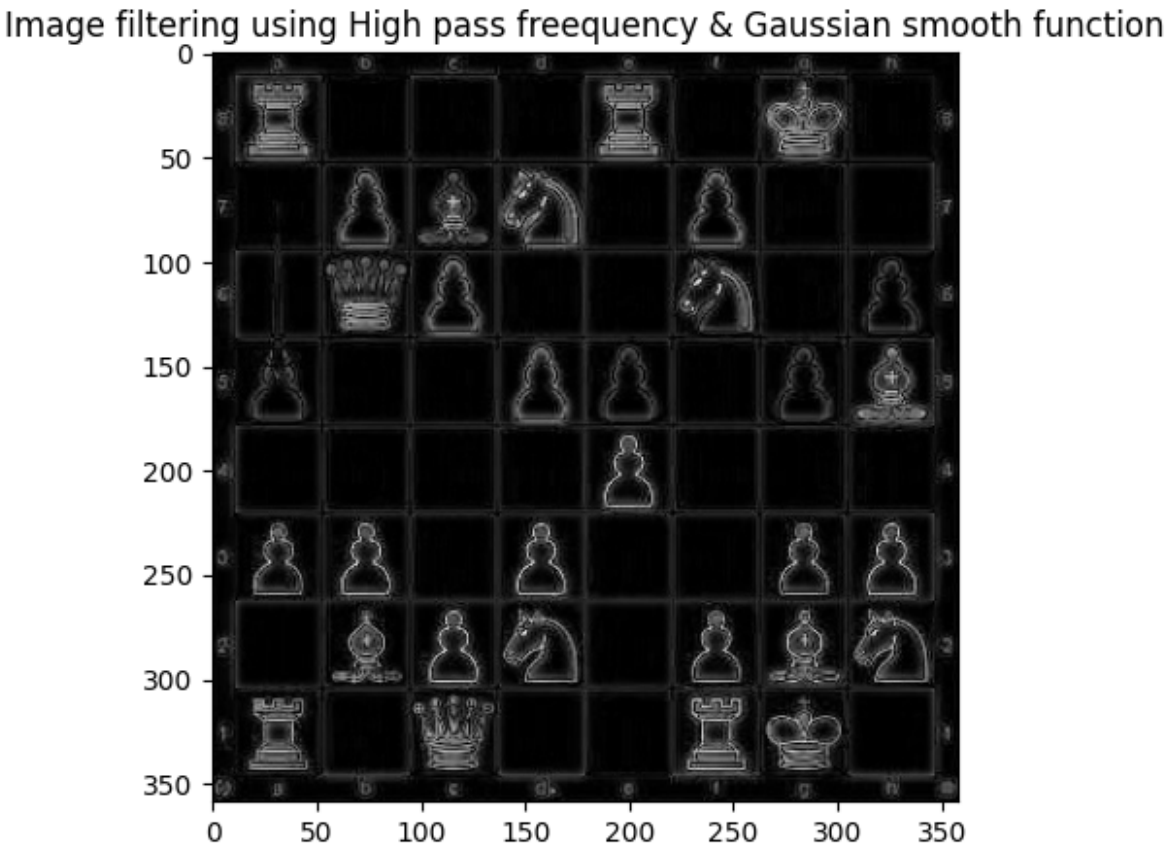
```
<ipython-input-284-3f831ad3f181>:4: RuntimeWarning: divide by zero encountere
d in log
  plt.imshow(np.log(np.abs (f)), cmap='gray')
```

Out[284]:
```
<matplotlib.image.AxesImage at 0x7f47f820cd90>
```



In [285]:
```
#(6) invert fourier transform
new_img = np.abs (np.fft.ifft2(f))
plt.figure(figsize = (5, 5))
plt.title("Image filtering using High pass freequency & Gaussian smooth functi
on")
plt.imshow(new_img, cmap = 'gray')
```

Out[285]: <matplotlib.image.AxesImage at 0x7f47f827cd60>

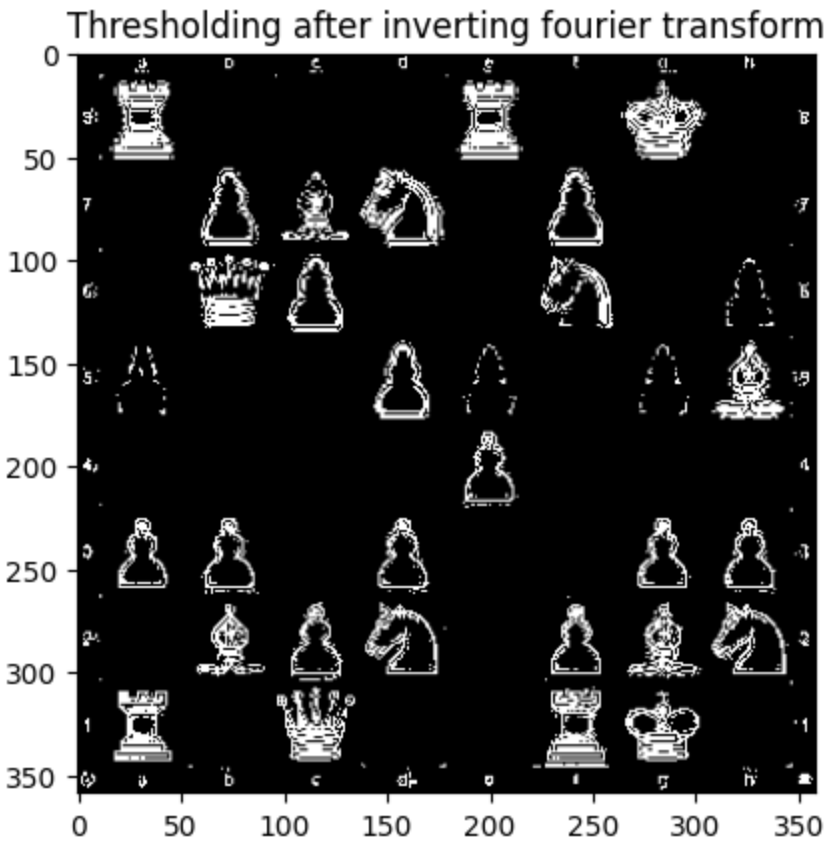


In [286]:
```
# Đặt ngưỡng
thresh = 47

# Sử dụng hàm cv2.threshold() để loại bỏ những điểm có độ sáng nhỏ hơn ngưỡng
ret, thresh_img_GSF = cv2.threshold(new_img, thresh, 255, cv2.THRESH_BINARY)

# Hiển thị ảnh sau khi đã áp dụng ngưỡng
plt.title("Thresholding after inverting fourier transform")
plt.imshow(thresh_img_GSF, cmap = 'gray')
```

Out[286]: <matplotlib.image.AxesImage at 0x7f47fdf60b80>

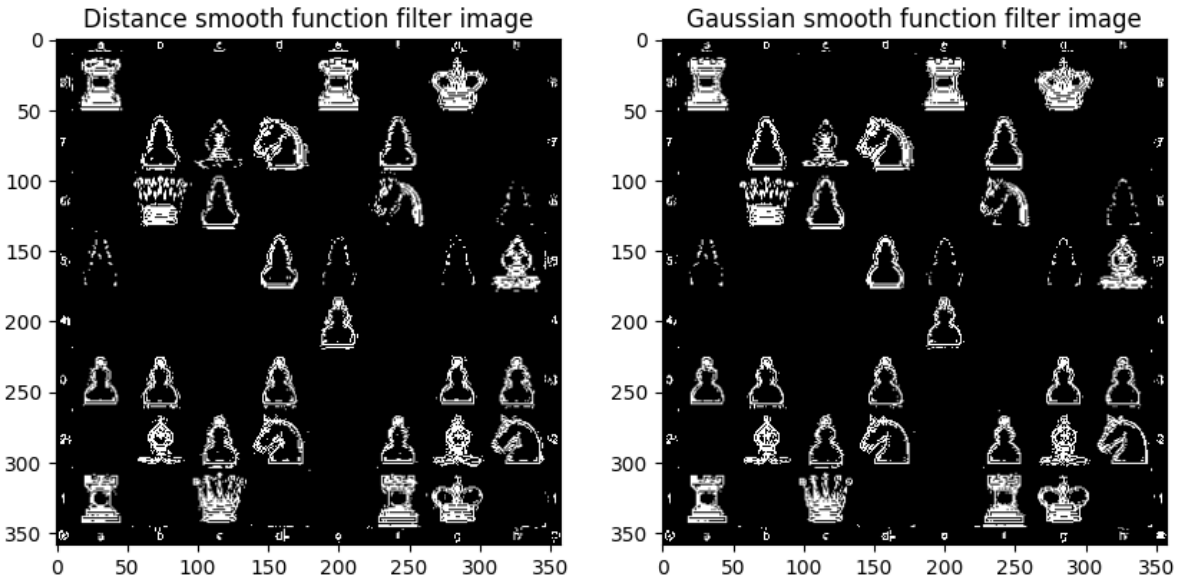


In [287]:
```
# # Erosion followed by dilation để lọc gợn ảnh
# kernel = np.ones((2,1),np.uint8) # Lọc theo chiều ngang
# opening1 = cv2.morphologyEx(thresh_img_GSF, cv2.MORPH_OPEN, kernel)
# kernel = np.ones((1,2),np.uint8) # Lọc theo chiều dọc
# opening2 = cv2.morphologyEx(thresh_img_GSF, cv2.MORPH_OPEN, kernel)
# # Cộng 2 ảnh vừa lọc
# opening_GSF = cv2.addWeighted(opening1, 0.5, opening2, 0.5, 0.0)
# plt.title("Erosion followed by dilation")
# plt.imshow(opening_GSF, cmap = 'gray')
```

In [288]:
```
print("Comparison between image high pass filtering using Distance smooth func
tion and Gaussian smooth function")
plt.figure(figsize=(10,10))
plt.subplot(1,2,1)
plt.title("Distance smooth function filter image")
plt.imshow(thresh_img_DSF, cmap = 'gray')
plt.subplot(1,2,2)
plt.title("Gaussian smooth function filter image")
plt.imshow(thresh_img_GSF, cmap = 'gray')
```

```
Comparison between image high pass filtering using Distance smooth function a
nd Gaussian smooth function
```

Out[288]:
```
<matplotlib.image.AxesImage at 0x7f47f811f370>
```



In [289]:
```
print("Comparison between image low pass filtering using Distance smooth funct
ion and Gaussian smooth function")
print("With same radius")
plt.figure(figsize=(10,10))
plt.subplot(1,2,1)
plt.title("Distance smooth function filter image")
plt.imshow(new_img_DSF, cmap = 'gray')
plt.subplot(1,2,2)
plt.title("Gaussian smooth function filter image")
plt.imshow(new_img_GSF, cmap = 'gray')
```

```
Comparison between image low pass filtering using Distance smooth function an
d Gaussian smooth function
With same radius
```

Out[289]:
```
<matplotlib.image.AxesImage at 0x7f47f8097040>
```