作业7：理解图像的傅里叶变换

作业目的：

理解频域处理与空间域处理的等效关系与步骤，掌握频域相乘与空间域循环卷积之间的对等关系，理解DFT运算中补零的效果。

作业内容：

（1）对附件中的两幅图像分别进行DFT，以log(1+abs(f))形式显示信号频谱。

从空域观察：两张图片中并没有明显的水平和垂直边缘，但从图像频谱显示其包含有强烈的水平与垂直分量，请分析其原因，提出相应的解决办法并进行验证。

（2）选一张灰度图像，然后顺序进行下列处理操作：

A.对(x,y)位置上的像素值乘以(−1)(x+y)

B.计算图像二维DFT

C.对二维DFT的值取共轭

D.对共轭后的频谱做IDFT运算

E.对IDFT的结果取实部

F.对实部乘以(−1)(x+y)

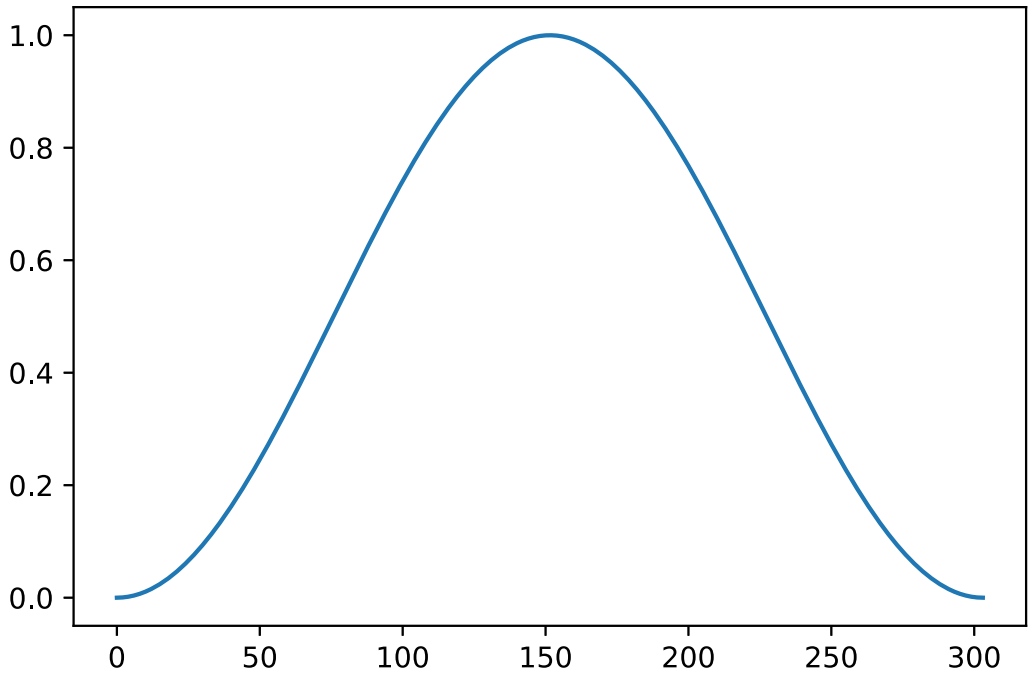
与原始图像进行对比，从理论上分析为什么会出现这种结果。

（1）对附件中的两幅图像分别进行DFT，以log(1+abs(f))形式显示信号频谱。

第一幅图像 DFT，以log(1+abs(f))形式显示信号频谱。

In [43]:
```
#### 汉明窗

plt.plot(signal.hann(img_1.shape[0]))
```

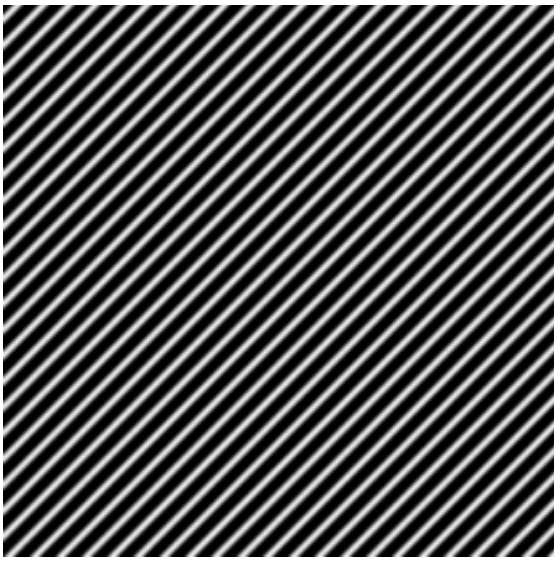
Out[43]: [<matplotlib.lines.Line2D at 0x7fde539d7700>]



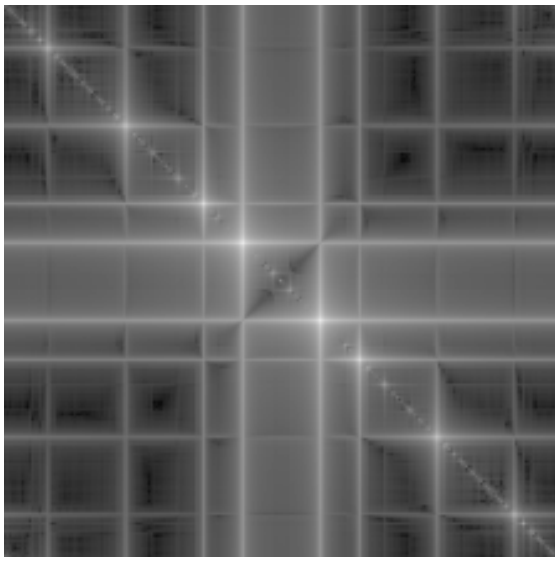
In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io
import scipy.signal as signal
from scipy.fftpack import fft, ifft
import pylab as pl
import scipy.signal as signal
import copy

img_1 = io.imread('img/lines.png')

plt.figure(figsize=(9,9))

plt.subplot(2,2,1)
plt.imshow(img_1,'gray')
plt.title('Original Image')
plt.axis('off')


fft2_1 = np.fft.fft2(img_1)
shift_fft2_1 = np.fft.fftshift(fft2_1)
log_shift_fft2_1 = np.log(1 + np.abs(shift_fft2_1))

plt.subplot(2,2,2)
plt.imshow(log_shift_fft2_1,'gray')
plt.title('logged_shift_fft2_img_1')
plt.axis('off')

img_1=img_1*signal.hann(img_1.shape[0])
img_1=img_1.T*signal.hann(img_1.shape[0])
img_1=img_1.T

plt.subplot(2,2,3)
plt.imshow(img_1,'gray')
plt.title('Original Image After hann')
plt.axis('off')

fft2_1 = np.fft.fft2(img_1)
shift_fft2_1 = np.fft.fftshift(fft2_1)
log_shift_fft2_1 = np.log(1 + np.abs(shift_fft2_1))

plt.subplot(2,2,4)
plt.imshow(log_shift_fft2_1,'gray')
plt.title('After hann Logged_shift_fft2_img_1')
plt.axis('off')
```

Out[2]: (-0.5, 303.5, 303.5, -0.5)

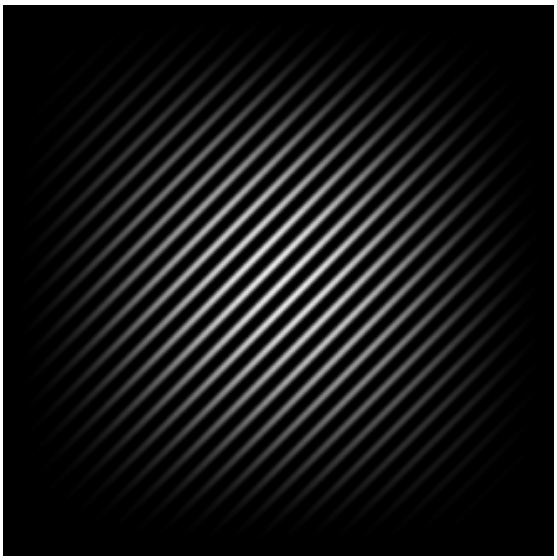
Original Image



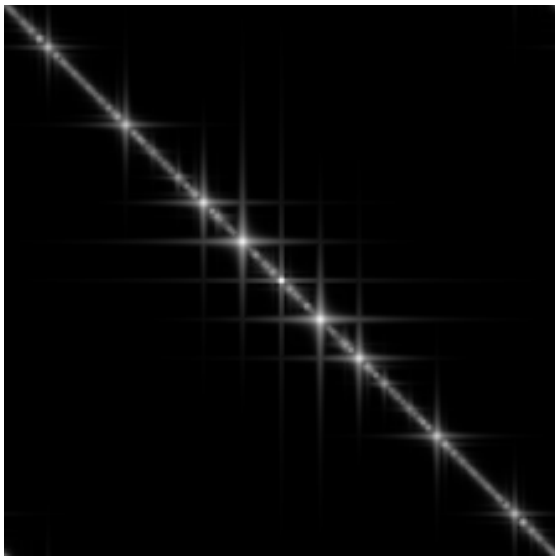
logged_shift_fft2_img_1



Original Image After hann



After hann Logged_shift_fft2_img_1

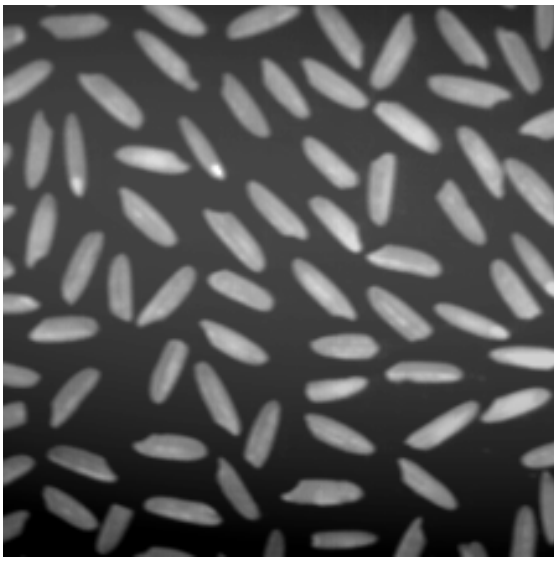


第二幅图像 DFT，以log(1+abs(f))形式显示信号频谱。

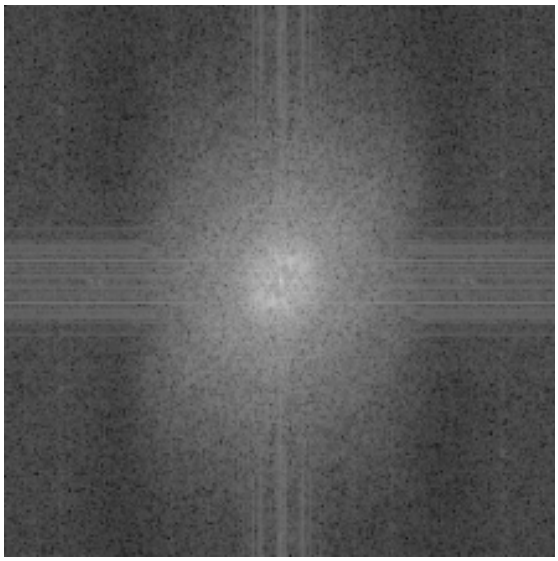
In [63]:
```
img_2 = io.imread('img/rice.tif')

plt.figure(figsize=(9,9))

plt.subplot(2,2,1)
plt.imshow(img_2,'gray')
plt.title('Original Image')
plt.axis('off')


fft2_1 = np.fft.fft2(img_2)
shift_fft2_1 = np.fft.fftshift(fft2_1)
log_shift_fft2_1 = np.log(1 + np.abs(shift_fft2_1))
shift_fft2_1 = np.fft.fftshift(fft2_1)
log_shift_fft2_1 = np.log(1 + np.abs(shift_fft2_1))

plt.subplot(2,2,2)
plt.imshow(log_shift_fft2_1,'gray')
plt.title('logged_shift_fft2_img_2')
plt.axis('off')

img_2=img_2*signal.hann(img_2.shape[0])
img_2=img_2.T*signal.hann(img_2.shape[0])
img_2=img_2.T

plt.subplot(2,2,3)
plt.imshow(img_2,'gray')
plt.title('Original Image After hann')
plt.axis('off')

fft2_1 = np.fft.fft2(img_2)
shift_fft2_1 = np.fft.fftshift(fft2_1)
log_shift_fft2_1 = np.log(1 + np.abs(shift_fft2_1))

plt.subplot(2,2,4)
plt.imshow(log_shift_fft2_1,'gray')
plt.title('After hann Logged_shift_fft2_img_2')
plt.axis('off')
```

Out[63]: (-0.5, 255.5, 255.5, -0.5)

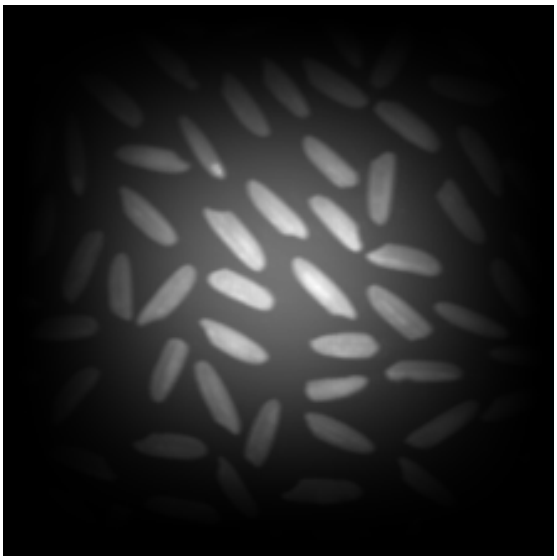
Original Image



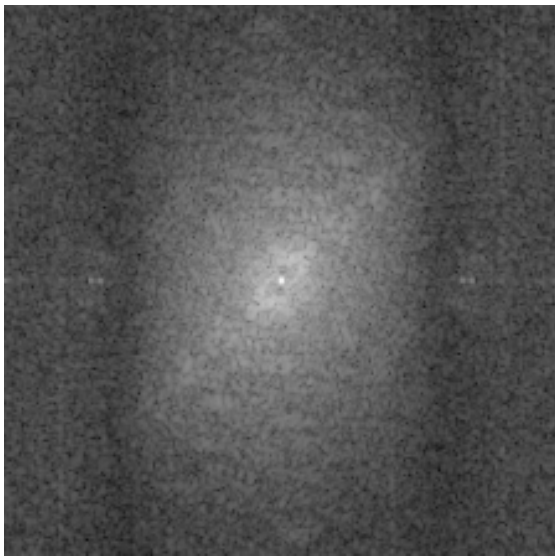
logged_shift_fft2_img_2



Original Image After hann



After hann Logged_shift_fft2_img_2



从空域观察：两张图片中并没有明显的水平和垂直边缘，但从图像频谱显示其包含有强烈的水平与垂直分量，请分析其原因，提出相应的解决办法并进行验证。

（2）对作业1要求分析DFT中水平与垂直分量的比较强的原因，比较对图像加窗后DFT的效果，说明窗函数改变DFT的原因

1.图像频谱显示其包含有强烈的水平与垂直分量，请分析其原因，提出相应的解决办法并进行验证。

原因是因为DFT对应的是离散信号的周期延拓，添加汉明窗，处理图像的边缘，让图像的主要的部分的频率显露出来。

2.频谱混乱的原因有两个：

第一个原因是空域的不连续导致了另一个域的振荡频谱就会看起来非常混乱。

第二个原因就是DFT的周期性。当使用FFT2的时候，所做的计算本质上离散傅里叶级数，DFT的频域计算隐含着周期性，这一周期性不论是在时域还是在频域都是无穷的，都是无限循环的。
所以我们要处理图像的边缘处，因为边缘处的各种不连续，这种不连续直接导致了频率混乱和复杂的频谱。在处理任何图像时，千万不要忽视了图像的边缘。

In [ ]:
```
3.比较对图像加窗后DFT的效果，说明窗函数改变DFT的原因

    由窗函数我们可以看出窗函数的左右两边是衰减的这样，我们就可以将通过在横纵方向上与汉明
```

（2）选一张灰度图像，然后顺序进行下列处理操作：

选取的图片

In [31]:
```
chongqing= io.imread('img/chongqing.png')


print(chongqing.shape)
plt.subplot(1,2,1)
plt.imshow(chongqing)
plt.title('chongqing street pic original')
plt.axis('off')
plt.subplot(1,2,2)

chongqing_gray= (chongqing[:,:,0]*299 + chongqing[:,:,1]*587 + chongqing[:
plt.imshow(chongqing_gray,'gray')
plt.title('chongqing street pic gray')
plt.axis('off')

io.imsave('img/chongqing_gray.jpg',chongqing_gray)
```

```
(4032, 3024, 4)
Lossy conversion from float64 to uint8. Range [0.0, 65.535]. Convert image
to uint8 prior to saving to suppress this warning.
```

In [32]:
```
# img_B=io.imread('img/chongqing_gray.jpg')
# fft2_B = np.fft.fft2(img_B)
# shift_fft2_B = np.fft.fftshift(fft2_B)
# # for x in range(fft2_B.shape[0]):
# #     for y in range(fft2_B.shape[1]):
# #         fft2_B[x,y] = pow(-1,(x+y))*fft2_B[x,y]
# plt.imshow(shift_fft2_B.real,'gray')
```

A.对(x,y)位置上的像素值乘以(-1)(x+y)

In [33]:
```
chongqing_gray=io.imread('img/chongqing_gray.jpg')
chongqing_A=copy.deepcopy(chongqing_gray)

for x in range(chongqing_A.shape[0]):
    for y in range(chongqing_A.shape[1]):
        chongqing_A[x,y] = pow(-1,(x+y))*chongqing_A[x,y]


plt.subplot(1,2,1)
plt.imshow(chongqing_gray,'gray')
plt.title('chongqing street pic gray')
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(chongqing_A,'gray')
plt.title('chongqing street pic gray A')
plt.axis('off')
io.imsave('img/chongqing_A.jpg',chongqing_A)
```

chongqing street pic gray

chongqing street pic gray A

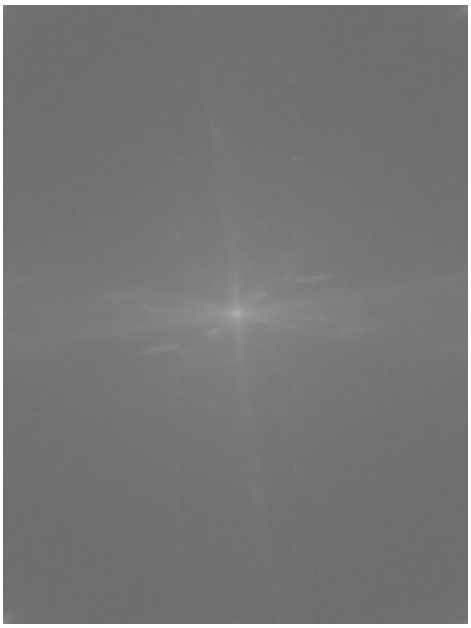
B.计算图像二维DFT

In [11]:
```
chongqing_gray=io.imread('img/chongqing_gray.jpg')
img_B=io.imread('img/chongqing_A.jpg')
fft2_B = np.fft.fft2(img_B)
# shift_fft2_B = np.fft.fftshift(fft2_B)

print(fft2_B)
log_shift_fft2_B = np.log(1 + np.abs(fft2_B))
plt.imshow(log_shift_fft2_B,'gray')
plt.title('chongqing street pic gray DFT')
plt.axis('off')
```

```
[[ 1.55210000e+09      +0.j          -6.53733722e+04 -919710.71122952j
   1.66041894e+06 -486780.44423755j ... -5.25847493e+05 -994633.35345688j
   1.66041894e+06 +486780.44423755j -6.53733722e+04 +919710.71122952j]
 [-1.49678345e+06+5663930.51494139j -1.79733075e+06+1637629.8043742j
   4.54257216e+05-1376542.89856174j ... -1.07091158e+06 -252109.92001386j
   1.20245434e+06 -106220.54346593j -3.92449084e+05-1185723.30773188j]
 [ 8.82271998e+05+1378133.69209418j  1.71488390e+06+1290130.77671626j
   3.32617321e+05 -531488.59489606j ... -1.52914040e+06 +802308.18108083j
   7.75556024e+05 -736280.38835314j -1.26593081e+06 +495592.55053379j]
 ...
 [-1.88532830e+05 -715627.15046238j -3.01644863e+05 -739292.4184919j
   3.84753098e+05 -229146.01429142j ... -8.12242792e+05 +412576.24422273j
  -9.24498525e+04+1017008.78003583j  6.11856586e+05 +605241.77611453j]
 [ 8.82271998e+05-1378133.69209418j -1.26593081e+06 -495592.55053379j
   7.75556024e+05 +736280.38835314j ... -6.30599538e+05 +407985.18358751j
   3.32617321e+05 +531488.59489606j  1.71488390e+06-1290130.77671626j]
 [-1.49678345e+06-5663930.51494139j -3.92449084e+05+1185723.30773188j
   1.20245434e+06 +106220.54346593j ...  6.12984183e+05 +232880.97378454j
   4.54257216e+05+1376542.89856174j -1.79733075e+06-1637629.8043742j ]]
```

Out[11]: (-0.5, 3023.5, 4031.5, -0.5)

chongqing street pic gray DFT



图像的DFT变换结果，目的是体现空间变化与频域的关系：空间变化越快，频谱越宽；空间变化越慢，频谱越窄

C.对二维DFT的值取共轭

In [7]:
```
shift_fft2_B_conjugate=np.conjugate(fft2_B)

print(shift_fft2_B_conjugate)
```

```
[[ 1.55210000e+09      -0.j          -6.53733722e+04 +919710.71122952j
   1.66041894e+06 +486780.44423755j ... -5.25847493e+05 +994633.35345688j
   1.66041894e+06 -486780.44423755j -6.53733722e+04 -919710.71122952j]
 [-1.49678345e+06-5663930.51494139j -1.79733075e+06-1637629.8043742j
   4.54257216e+05+1376542.89856174j ... -1.07091158e+06 +252109.92001386j
   1.20245434e+06 +106220.54346593j -3.92449084e+05+1185723.30773188j]
 [ 8.82271998e+05-1378133.69209418j  1.71488390e+06-1290130.77671626j
   3.32617321e+05 +531488.59489606j ... -1.52914040e+06 -802308.18108083j
   7.75556024e+05 +736280.38835314j -1.26593081e+06 -495592.55053379j]
 ...
 [-1.88532830e+05 +715627.15046238j -3.01644863e+05 +739292.4184919j
   3.84753098e+05 +229146.01429142j ... -8.12242792e+05 -412576.24422273j
  -9.24498525e+04-1017008.78003583j  6.11856586e+05 -605241.77611453j]
 [ 8.82271998e+05+1378133.69209418j -1.26593081e+06 +495592.55053379j
   7.75556024e+05 -736280.38835314j ... -6.30599538e+05 -407985.18358751j
   3.32617321e+05 -531488.59489606j  1.71488390e+06+1290130.77671626j]
 [-1.49678345e+06+5663930.51494139j -3.92449084e+05-1185723.30773188j
   1.20245434e+06 -106220.54346593j ...  6.12984183e+05 -232880.97378454j
   4.54257216e+05-1376542.89856174j -1.79733075e+06+1637629.8043742j ]]
```

D.对共轭后的频谱做IDFT运算

In [50]:
```
ifft_shift_fft2_B = np.fft.ifft2(shift_fft2_B_conjugate)
```

E.对IDFT的结果取实部

In [51]:
```
ifft_shift_fft2_B_real=np.real(ifft_shift_fft2_B)
```

F.对实部乘以(-1)(x+y)

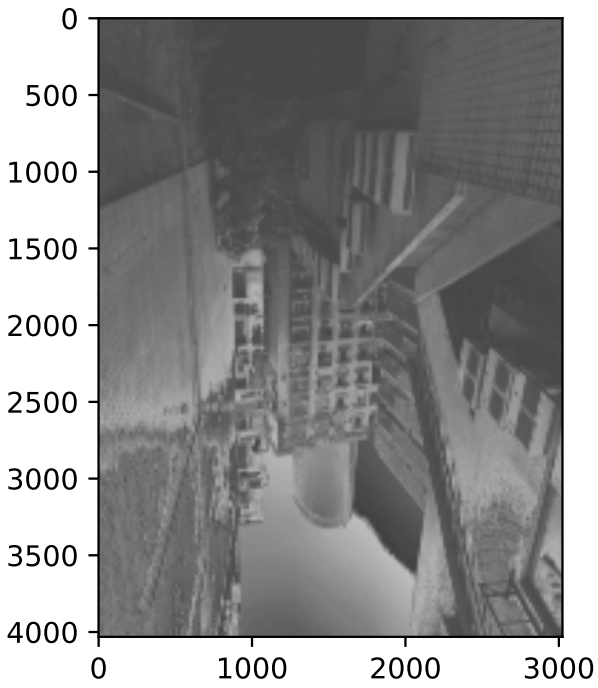
与原始图像进行对比，从理论上分析为什么会出现这种结果。

In [52]:
```
for x in range(ifft_shift_fft2_B_real.shape[0]):
    for y in range(ifft_shift_fft2_B_real.shape[1]):
        ifft_shift_fft2_B_real[x,y] = pow(-1,(x+y))*ifft_shift_fft2_B_real
# plt.plot(ifft_shift_fft2_B_real)
```

In [53]:
```
print(ifft_shift_fft2_B_real.shape)
print(ifft_shift_fft2_B_real)
plt.imshow(ifft_shift_fft2_B_real,'gray')
```

```
(4032, 3024)
[[  61. -146.   99. ... -200.   61. -192.]
 [-186.   26. -220. ...   86. -175.   74.]
 [  80. -221.   60. ... -168.   81. -178.]
 ...
 [-189.  106. -129. ...   64. -181.   75.]
 [  74. -161.  123. ... -180.   86. -163.]
 [-188.  100. -156. ...   68. -179.   82.]]
```

Out[53]: <matplotlib.image.AxesImage at 0x12fff7c40>

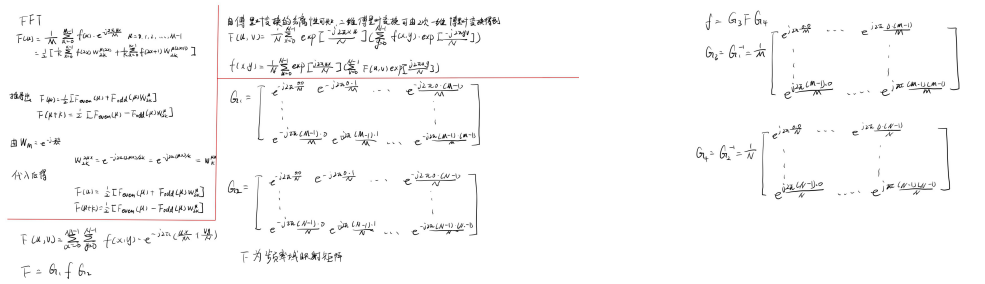


理论推导证明

In [8]:

```
DFT=io.imread('img/DFT.jpg')
IDFT=io.imread('img/IDFT.jpg')
plt.figure(figsize=(128,256))
plt.subplot(1,2,1)
plt.imshow(DFT)
plt.title('DFT')
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(IDFT)
plt.title('IDFT')
plt.axis('off')
```

Out[8]: (-0.5, 2385.5, 1490.5, -0.5)



作业提示：

（1）与DFT对应的是离散信号的周期延拓，请对图像作周期延拓，注意边缘有什么特殊情况

（2）有限长的信号等效于无限长信号与窗函数的乘积。图像也可以看成无限大图像与窗函数的乘积，建议给图像加汉明窗代替矩形窗

作业提交：

（1）提交PDF格式的报告，包括处理前后的图像对比及相应的代码（代码直接放在文档中，不要单独打包）

（2）对作业1要求分析DFT中水平与垂直分量的比较强的原因，比较对图像加窗后DFT的效果，说明窗函数改变DFT的原因

（3）对作业2除给出处理效果外，要给出理论上推导证明。