In [ ]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import math
# 本次作业的任务有现成的库可以直接调用解决，但作为一次练习，作业期间不允许使用其他的库
```

Lab 4

第一部分：傅里叶转换和频率域

首先，创建一个边长为1000的信号图形。

In [ ]:
```
# 定义边长
x = np.arange(-500, 501, 1)
# 创建meshgrid
X, Y = np.meshgrid(x, x)
# 设置波长（即1/频率）
wavelength = 200
# 设置信号在空间域的角度
angle = 0
# 创建信号图pattern1，并绘图预览
pattern1 = np.sin(2*np.pi*(X*np.cos(angle) + Y*np.sin(angle)) / wavelength)
plt.imshow(pattern1,cmap=plt.cm.gray);
```

为创建的信号图做快速傅里叶变换FFT，绘制频率域图像。

In [ ]:
```
# 为pattern1计算快速傅里叶变换FFT
ft = np.fft.ifftshift(pattern1)
ft = np.fft.fft2(ft)
ft = np.fft.fftshift(ft)
plt.imshow(abs(ft),cmap=plt.cm.gray)
# 像素点相对原图太小，需将图片剪切至靠近图幅中间的相应位置查看
plt.xlim([480, 520])
plt.ylim([520, 480]);
```

然后，创建另一个不同的信号图，并做FFT。

In [ ]:
```
# 修改波长和角度
x = np.arange(-500, 501, 1)
X, Y = np.meshgrid(x, x)
wavelength = 100
angle = 45
pattern2 = np.sin(2*np.pi*(X*np.cos(angle) + Y*np.sin(angle)) / wavelength)

# 为pattern2计算快速傅里叶变换FFT
ft = np.fft.ifftshift(pattern2)
ft = np.fft.fft2(ft)
ft = np.fft.fftshift(ft)

fig, axes = plt.subplots(1, 2, figsize = (8,8))
axes[0].imshow(pattern2,cmap=plt.cm.gray)
axes[1].imshow(abs(ft),cmap=plt.cm.gray)
axes[1].set_xlim([480, 520])
axes[1].set_ylim([520, 480]);
```

将以上两个信号图叠加后，形成第三个信号图，并对其做FFT。

In [ ]:
```
pattern3 = pattern1 + pattern2

# 为pattern3计算快速傅里叶变换FFT
ft = np.fft.ifftshift(pattern3)
ft = np.fft.fft2(ft)
ft = np.fft.fftshift(ft)

fig, axes = plt.subplots(1, 2, figsize = (8,8))
axes[0].imshow(pattern3,cmap=plt.cm.gray)
axes[1].imshow(abs(ft),cmap=plt.cm.gray)
axes[1].set_xlim([480, 520])
axes[1].set_ylim([520, 480]);
```

如果把叠加改为相乘：

In [ ]:
```
pattern4 = pattern1 * pattern2

# 为pattern3计算快速傅里叶变换FFT
ft = np.fft.ifftshift(pattern4)
ft = np.fft.fft2(ft)
ft = np.fft.fftshift(ft)

fig, axes = plt.subplots(1, 2, figsize = (8,8))
axes[0].imshow(pattern4,cmap=plt.cm.gray)
axes[1].imshow(abs(ft),cmap=plt.cm.gray)
axes[1].set_xlim([480, 520])
axes[1].set_ylim([520, 480]);
```

任务1

创建一个边长为50、sigma为10的高斯滤波器（高斯核），并对其进行FFT。 绘制该高斯滤波器及其频率域图像。

In [ ]:
```
# 在这里完成任务1
```

第二部分：低通、高通、方向位滤波

In [ ]:
```
#导入照片并查看
img = cv2.imread('lab4_img/bridge.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img = cv2.resize(img, (500, 500),
                 interpolation = cv2.INTER_LINEAR)
plt.imshow(img,cmap=plt.cm.gray)
```

对桥梁照片做FFT并查看其频率域图像。

In [ ]:
```
img_fft = np.fft.fftshift(np.fft.fft2(img))
plt.imshow(np.log(abs(img_fft)),cmap=plt.cm.gray)
```

为频率域做一个简单的滤波器，滤掉原图中水平方向的线条（即频率域图像上竖直方向的线条区域）但保留原图竖直方向（即频率域图的水平线条）的所有宏观模式及细节。

经观察，在频率域中，当竖直方向的坐标在230-270区间时，为原图竖直方向的信号，需要保留。

当频率域竖直方向的坐标小于230时，原图水平方向的信号集中在频率域水平方向坐标250-300之间，需要剔除。

当频率域竖直方向的坐标大于270时，原图水平方向的信号集中在频率域水平方向坐标200-250之间，需要剔除。

需要剔除的部分，频率域中对应位置的数值设为1。

In [ ]:
```
# 将需要剔除的部分，频率域中对应位置的数值设为1
img_fft[:230, 250:300] = 1
img_fft[270:, 200:250] = 1
fig, ax = plt.subplots(1,3,figsize=(15,15))
# 显示原图
ax[0].imshow(img, cmap=plt.cm.gray)
ax[0].set_title('Original')
# 显示FFT后的频域图
ax[1].imshow(np.log(abs(img_fft)), cmap=plt.cm.gray)
ax[1].set_title('FFT Filter')
# 逆FFT及显示结果
ax[2].imshow(abs(np.fft.ifft2(img_fft)), cmap=plt.cm.gray)
ax[2].set_title('Transformed');
```

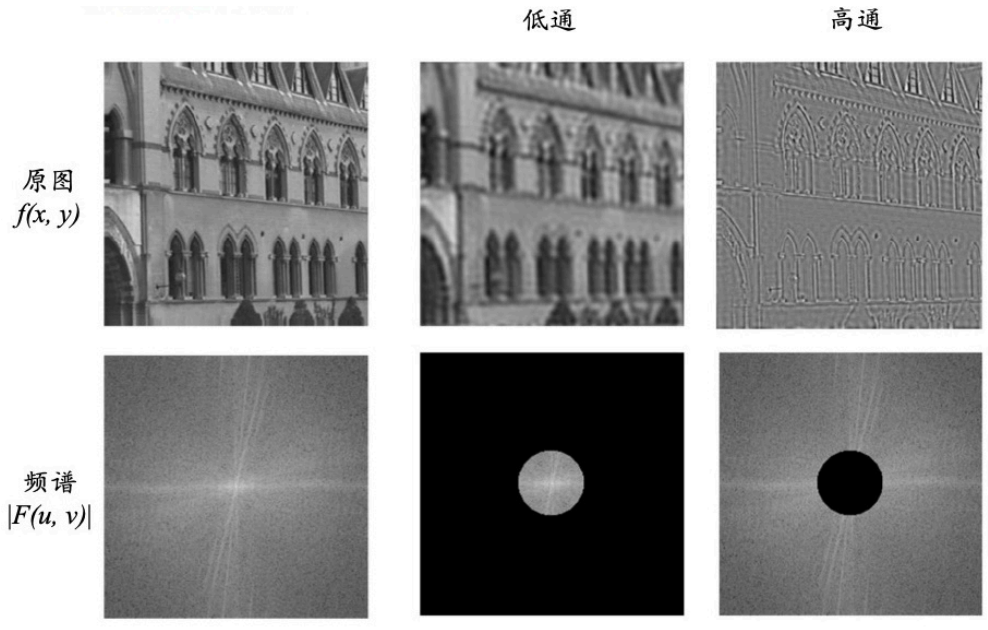
任务2

对原图画面中的竖直线条做滤波处理，并按上方所示，并排展示原图、滤波后的频率域图像、滤波后的原图。

In [ ]:
```
# 在这里完成任务2
```

高斯低通（Gaussian Low Pass）和高斯高通（Gaussian High Pass）

普通的低通滤波器保留原图的大尺度信息（即：保留频率域图像靠近中间的位置）而过滤掉细节（即：过滤掉频率域图像外围的位置）。 普通的高通滤波器保留原图的细节，但过滤掉大尺度信息。



高斯低通和高斯高通在此基础上，对滤波器做了高斯平滑处理，从而获得更好的效果。

高斯低通滤波器的公式为：

$$H(u,v) = e^{-\frac{(D(u,v))^2}{2d^2}}$$

高斯高通滤波器的公式为：

$$H(u,v) = 1 - e^{-\frac{(D(u,v))^2}{2d^2}}$$

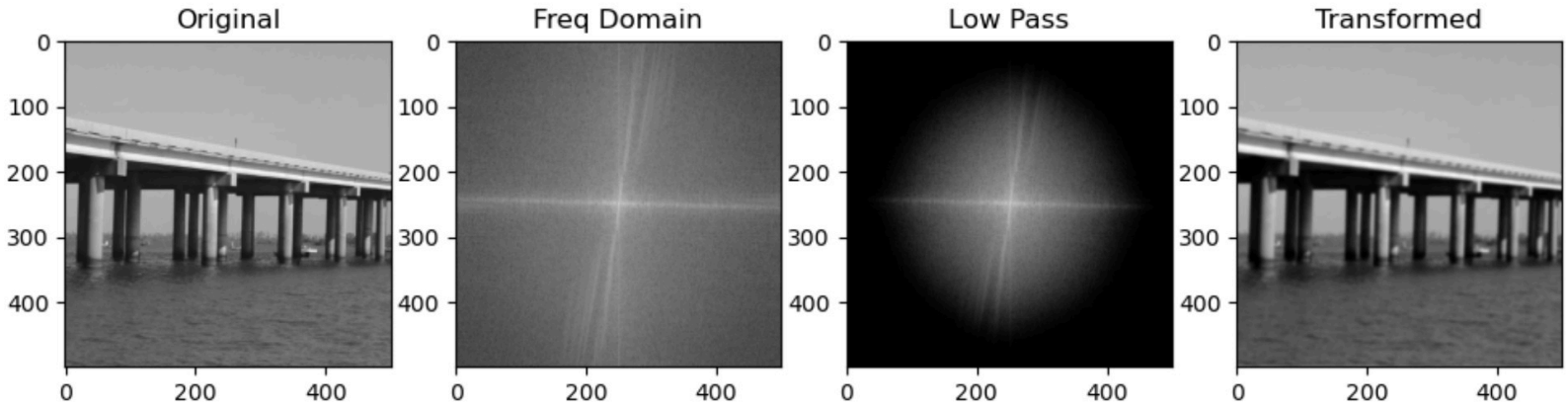
其中，d是截止频率，决定了平滑程度（d越大，高斯滤波器的频带就越宽，平滑程度就越好）。D(u,v）是当前点距矩形频率域图像中心的距离。

任务3

编写gaussianLP()函数，完成高斯低通滤波任务。该函数传入两个参数，分别是截止频率d和图片img（可默认该图长宽相等），它进行上述公式的计算后，返回一个高斯低通滤波器，该滤波器尺度与频率域图像相同，它与频率域图像进行哈达玛积的运算后，能达到高斯低通滤波的效

In [ ]:
```python
# 替换下面的pass，完成高斯低通函数的编写
def gaussianLP(d,img):
    pass
```

完成gaussianLP()的编写后，运行下面的代码。其效果图大致如下但有所区别（示意图为d=50的情况）：



In [ ]:
```python
# 定义阶段频率d
d = 20
# 计算图片的快速傅里叶变换FFT，得到频率域图
img_fft = np.fft.fftshift(np.fft.fft2(img))
# 将gaussianLP()的输出结果和频率域图进行哈达玛积运算，即频率域图和高斯低通滤波器叠加后的效果
img_lp = img_fft * gaussianLP(d,img)

#绘制示意图
fig, ax = plt.subplots(1,4,figsize=(12,12))
# 显示原图
ax[0].imshow(img, cmap=plt.cm.gray)
ax[0].set_title('Original')
# 显示FFT后的频域图
ax[1].imshow(np.log(abs(img_fft)), cmap=plt.cm.gray)
ax[1].set_title('Freq Domain')
# 叠加高斯低通滤波器
ax[2].imshow(np.log(1+np.abs(img_lp)),cmap=plt.cm.gray)
ax[2].set_title('Low Pass')
# 逆FFT及显示结果
ax[3].imshow(abs(np.fft.ifft2(img_lp)), cmap=plt.cm.gray)
ax[3].set_title('Transformed');
```

任务4

编写gaussianHP( )函数，完成高斯高通滤波任务。该函数传入两个参数，分别是截止频率d和图片img（可默认该图长宽相等），它进行上述公式的计算后，返回一个高斯高通滤波器。该滤波器尺度与频率域图像相同，它与频率域图像进行哈达玛积的运算后，能达到高斯高通滤波的效果。

In [ ]:
```
# 替换下面的pass，完成高斯低通函数的编写
def gaussianHP(d,img):
    pass
```

完成gaussianHP( )的编写后，运行下面的代码。

In [ ]:
```
# 定义阶段频率d
d = 40
# 计算图片的快速傅里叶变换FFT，得到频率域图
img_fft = np.fft.fftshift(np.fft.fft2(img))
# 将gaussianHP()的输出结果和频率域图进行哈达玛积运算，即频率域图和高斯高通滤波器叠加后的效果
img_hp = img_fft * gaussianHP(d,img)


fig, ax = plt.subplots(1,4,figsize=(12,12))
# 显示原图
ax[0].imshow(img, cmap=plt.cm.gray)
ax[0].set_title('Original')
# 显示FFT后的频域图
ax[1].imshow(np.log(abs(img_fft)), cmap=plt.cm.gray)
ax[1].set_title('Freq Domain')
# 叠加高斯高通滤波器
ax[2].imshow(np.log(1+np.abs(img_hp)),cmap=plt.cm.gray)
ax[2].set_title('High Pass')
# 逆FFT及显示结果
ax[3].imshow(abs(np.fft.ifft2(img_hp)), cmap=plt.cm.gray)
ax[3].set_title('Transformed');
```

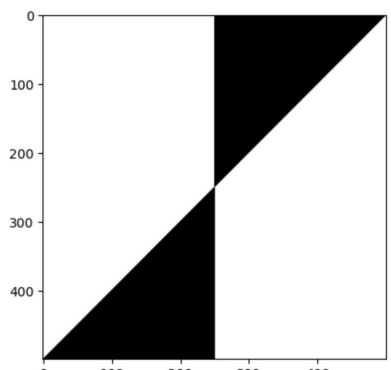
方向位滤波

方向位滤波需要取消某个方向范围内的细节，但保留范围外其他方向的细节。通常它只能在频率域中操作。

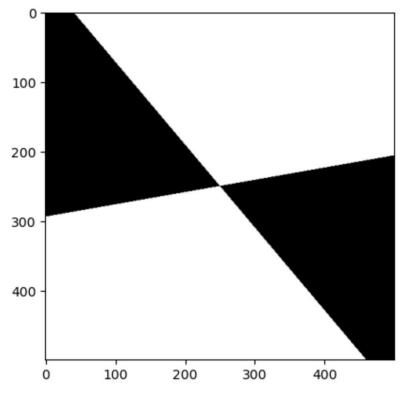
任务5

完成下方的orientation_filter( )函数的编写，它传入三个参数，分别为起始角度angle1、终止角度angle2和图片img（可以默认为长宽相同）。返回一个和img的频率域图像大小相同的模板，该模板中，以中心点为基准，让angle1、angle2角度之间区域的权值为0，其余区域权值为1。angle1、angle2单位为角度(非弧度)，角度以图片中心为基准逆时针转动，取值范围0-180，初始（即0角度）为竖直方向。

下图为orientation_filter(0,45,img)的输出模板(图片边长为500):



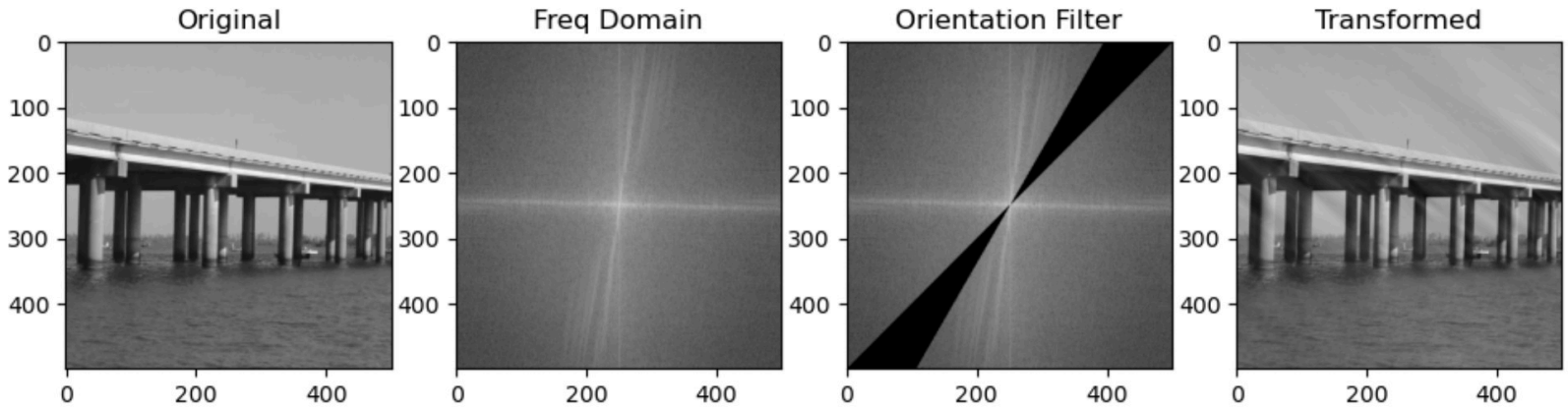
下图为orientation_filter(80,140,img)的输出模板(图片边长为500):



In [ ]:
```
# 替换下面的pass，完成函数的编写
# 提示1：平面直角坐标系中，某一点的横纵坐标和该点到原点连线的角度之间有什么三角函数关系？
# 提示2：numpy中的三角函数传入参数默认为弧度，这里需要注意角度到弧度的转换

def orientation_filter(angle1, angle2, img):
    pass
```

完成orientation_filter()后，运行下方两个cell的代码，其效果图大致如下但有所区别（示意图为angle1=30,angle2=45的情况）：



In [ ]:
```
# 计算图片的快速傅里叶变换FFT
img_fft = np.fft.fftshift(np.fft.fft2(img))
# 叠加方向位滤波器，滤掉5°到20°之间的细节
img_ang = img_fft * orientation_filter(5,20,img)


fig, ax = plt.subplots(1,4,figsize=(12,12))
# 显示原图
ax[0].imshow(img, cmap=plt.cm.gray)
ax[0].set_title('Original')
# 显示FFT后的频域图
ax[1].imshow(np.log(abs(img_fft)), cmap=plt.cm.gray)
ax[1].set_title('Freq Domain')
# 叠加方向位滤波器
ax[2].imshow(np.log(1+np.abs(img_ang)),cmap=plt.cm.gray)
ax[2].set_title('Orientation Filter')
# 逆FFT及显示结果
ax[3].imshow(abs(np.fft.ifft2(img_ang)), cmap=plt.cm.gray)
ax[3].set_title('Transformed');
```

In [ ]:
```
# 计算图片的快速傅里叶变换FFT
img_fft = np.fft.fftshift(np.fft.fft2(img))
# 叠加方向位滤波器，滤掉80°到120°之间的细节
img_ang = img_fft * orientation_filter(80,120,img)


fig, ax = plt.subplots(1,4,figsize=(12,12))
# 显示原图
ax[0].imshow(img, cmap=plt.cm.gray)
ax[0].set_title('Original')
# 显示FFT后的频域图
ax[1].imshow(np.log(abs(img_fft)), cmap=plt.cm.gray)
ax[1].set_title('Freq Domain')
# 叠加高斯高通滤波器
ax[2].imshow(np.log(1+np.abs(img_ang)),cmap=plt.cm.gray)
ax[2].set_title('Orientation Filter')
# 逆FFT及显示结果
ax[3].imshow(abs(np.fft.ifft2(img_ang)), cmap=plt.cm.gray)
ax[3].set_title('Transformed');
```

提交方式

本次作业有5个任务。完成所有cell的运行后，保存为ipynb和PDF格式（保留所有输出）。将导出的ipynb命名为“Lab4+姓名+学号.ipynb”，将导出的PDF命名为“Lab4+姓名+学号.pdf”，并将上述两个文件提交到学习通作业模块的相应位置。 请独立完成练习，参考答案将在截止时间后公布。 截止时间：2024年5月26日23:59。超时1天之内将扣除5%的分数，超时1天以上将扣除10%的分数。