# 大作业(姓名：————，学号：————)

### 阅读如下给出的程序：

import numpy as np

from PIL import Image

from readimg import ReadImage

def AdaptiveMedianFilter( f, #input image in numpy array

    MinWinSize, #minimum window

    MaxWinSize, #maximum window

    step ):# controls how many possible windows

    # make sure all are odd

    if not MinWinSize&0x1: MinWinSize += 1

    if not MaxWinSize&0x1: MaxWinSize += 1

    #make sure step is even

    if step&0x1: step += 1

    # padding

    HalfMaxWin = MaxWinSize//2

    HalfMinWin = MinWinSize//2

    padded = np.pad( f, HalfMaxWin, 'edge')

    g = np.zeros\_like( f )

    m, n = f.shape

    for i in range( m ):

        for j in range( n ):

            # fetch current pixel value

            zxy = f[i, j]

            processed = 0 # init flag

            for k in range( MinWinSize, MaxWinSize+1, step):

                halfk = k//2

                fromr = i+HalfMaxWin-halfk

                tor = fromr+k

                fromc = j+HalfMaxWin-halfk

                toc = fromc+k

                block = padded[fromr:tor, fromc:toc]

                sorted = np.sort( block, axis=None)

                zmin = sorted[0]

                zmax = sorted[k\*k-1]

                zmed = sorted[k\*k//2]

                if zmed > zmin and zmax > zmed:

                    if zxy > zmin and zmax > zxy: g[i,j] = zxy

                    else: g[i,j] = zmed

                    processed = 1

                    break

            if processed == 0: g[i,j] = zxy

    return g

if \_\_name\_\_=='\_\_main\_\_':

    f = ReadImage( './noisylena.bmp')

    g = AdaptiveMedianFilter( f, 3, 11, 2) #win size should be odd, and step be even

    g = Image.fromarray( g )

    g.save( './denoised.bmp')

其中的readimg.py就是课程中涉及的代码，noisylena.bmp已经和作业一起上传到repo中，针对如上给出的代码：

1. 使用前面中值滤波作业中自己完成的代码，对noisylena.bmp进行滤波，给出滤波结果图像
2. 使用上面给出的程序，对noisylena.bmp进行滤波，给出滤波结果
3. 比较两者的差别，并分析原因
4. 给出上述程序的算法思想，这里算法思想是只具体的对图像中没一点的处理算法
5. （同样，结果和回答都直接写在题目后面）

### 阅读如下给出的程序

import numpy as np

from PIL import Image

from freqfilter import FreqFilter

from readimg import ReadImage

def NotchBandRejectFilter( m, # rows

                          n, #cols

                          u0, # row

                          v0, # col

                          d0 ): # cut off

    a = np.arange( n )

    b = np.arange( m )

    x,y = np.meshgrid( a, b )

    d1 = np.sqrt( np.square(y-m/2-u0) +

                 np.square(x-n/2-v0))

    d2 = np.sqrt( np.square( y-m/2+u0) +

                 np.square(x-n/2+v0))

    sd0 = np.square( d0 )

    #H = np.zeros( (m,n))

    H = 1.- np.exp( -0.5\*( d1\*d2/sd0))

    return H

def PeriodicNoiseRemoval( f, u0, v0, d0 ):

    m, n = f.shape

    H = NotchBandRejectFilter( m, n, u0, v0, d0 )

    g = FreqFilter( f, H )

    return g

if \_\_name\_\_ == '\_\_main\_\_':

    f = ReadImage( "./img\_03\_23.jpg")

    g = PeriodicNoiseRemoval( f, 10, -10, 5)

    g = Image.fromarray(g)

    g.save( "./res.bmp")

其中涉及的readimg.py和FreqFilter.py，就是课程中所涉及的代码。请：

1. 给出该代码的算法思想，以及结果图像，img\_03\_23.jpg在大作业repo中。这里说的写出算法思想要给出滤波器公式
2. 把上述的band reject滤波器改为band pass 滤波器，给出代码和最后的滤波结果图像（代码直接贴在这里，结果也是）

### 给出如图所示的图像，

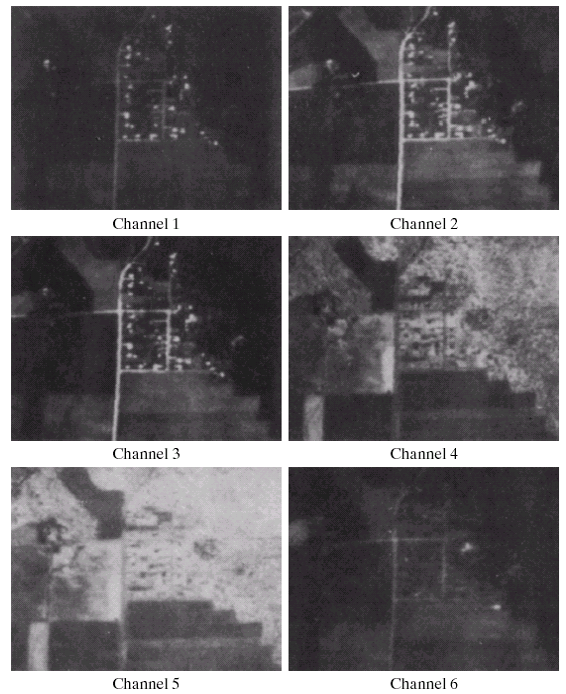


该图像的原始文件coins.jpg随作业一起给出， 在该问题中，需要统计出其中硬币的个数。大致的思路是：a提取边缘；b使用hough变换来检测圆形；c统计圆的个数

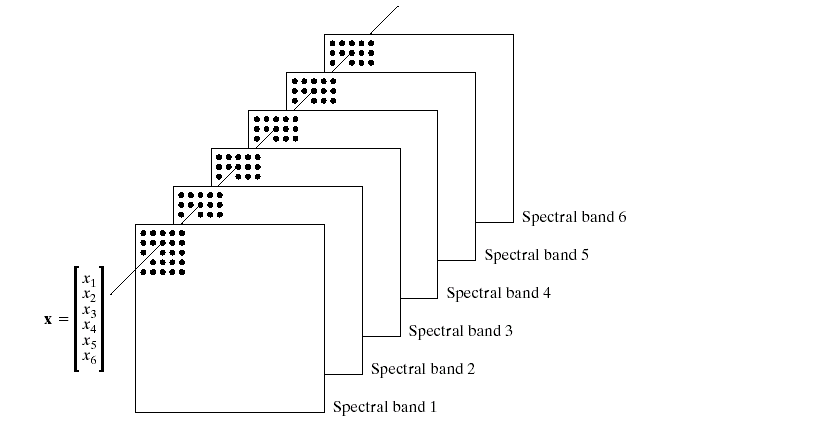
当然，在提取边缘之前可能需要对图像做一定程度的平滑滤波，不限定平滑滤波的方法，也不限定后面边缘检测的方法，不过，所有方法都尽量来自课程学习过的内容，比方不能直接使用Python-Opencv提供的模块，如果想要使用，自己实现类似的功能，在极端认为无法实现的情形下，可以尝试使用课程没有介绍的模块，但这样完成的作业，得分也会根据依赖的外部模块的程度而打折扣。

给出具体的代码（直接贴在问题后面）， 过程描述，以及每一步的结果。

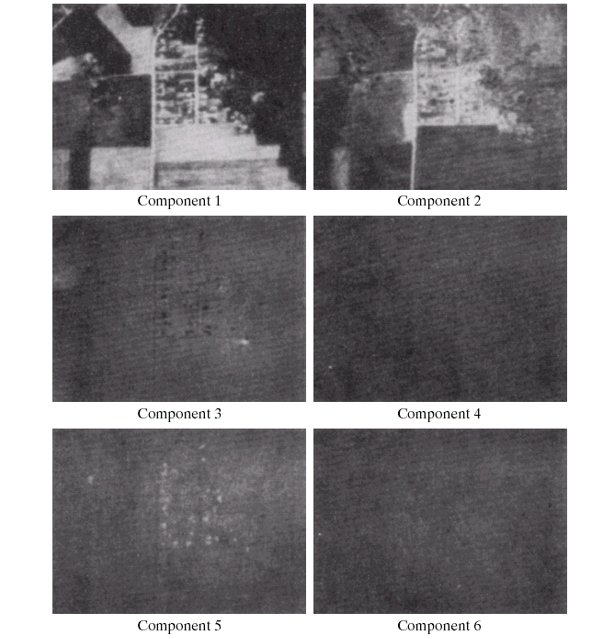
### 参考lecture4-3.ppt中对主分量分析的描述，针对其中给出的六个波段的红外图像，如图（原图分别在WashingtonDC\_Band1\_564.tif到WashingtonDC\_Band6\_564.tif中）



按照如下方式组织矢量：



1. 给出协方差矩阵的计算结果
2. 给出变换矩阵A
3. 给出6个最大分量的本征值
4. 给出使用6个中每一个主分量来近似的结果（留意是每一个而不是用1个，2个之类的），应能够看到如下类似的结果



把代码，结果都贴到本文档后面：

解答：

1.