完成了所有的必做和选做的作业。

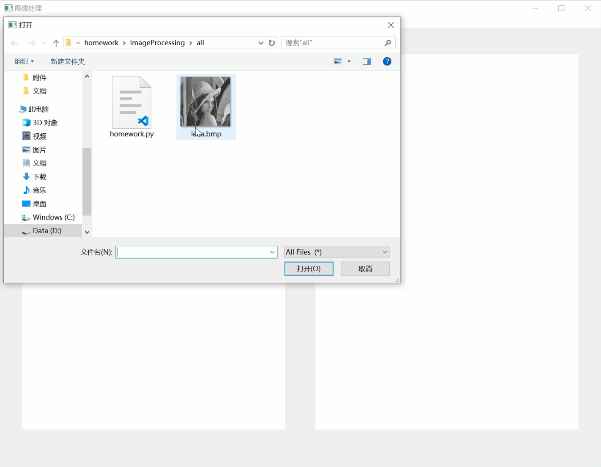
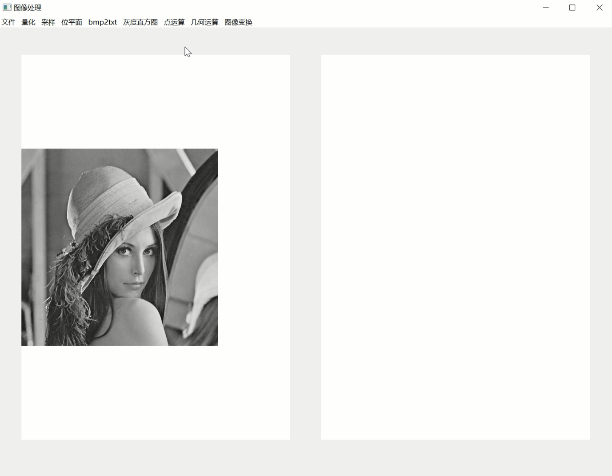
### 第一次作业：

内容在第一次作业展示ppt和homework1.cpp里面

**下面的作业为第二次到十三次，作业均在homework2to13.py里，需要安装的库包含PyQt5,opencv,pywt,numpy,matplotlib,six，运行该python文件。**

### 第二次作业:

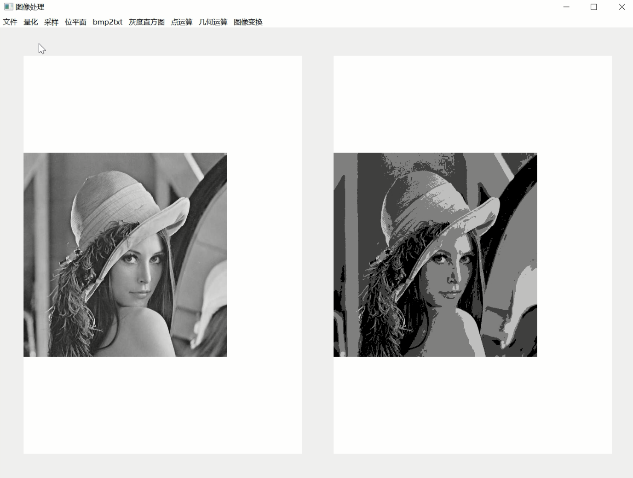
点击文件->打开，载入lena.bmp，点击

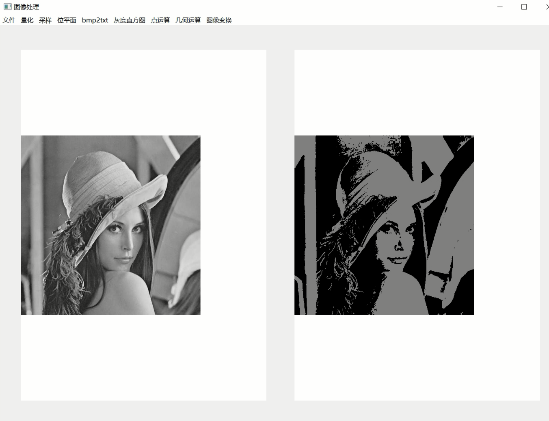
点击量化->量化8



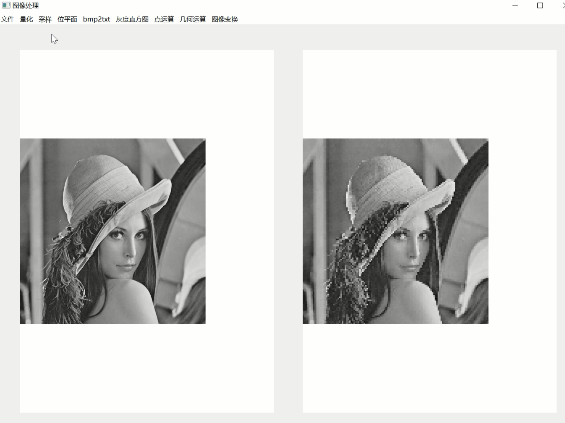
点击量化->量化4



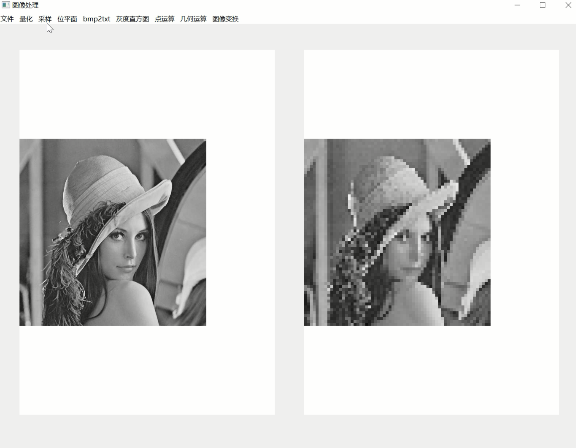
点击量化->量化2



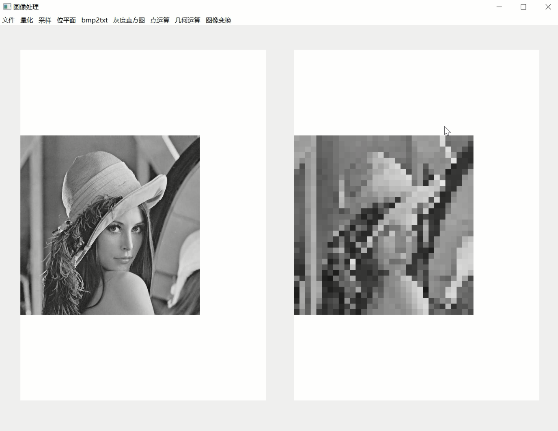
点击采样->采样128x128



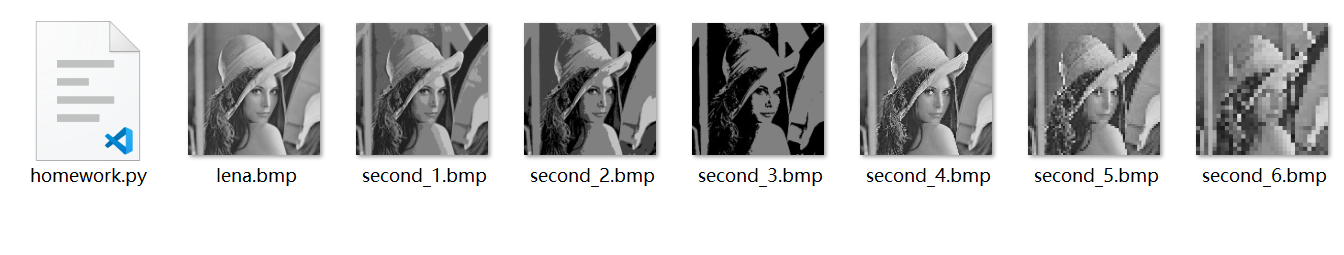
点击采样->采样64x64



点击采样->采样32x32



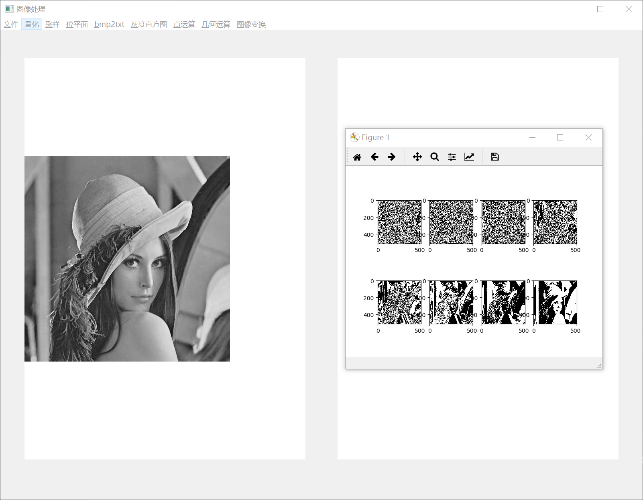
同时所有结果都自动保存在了文件夹里：



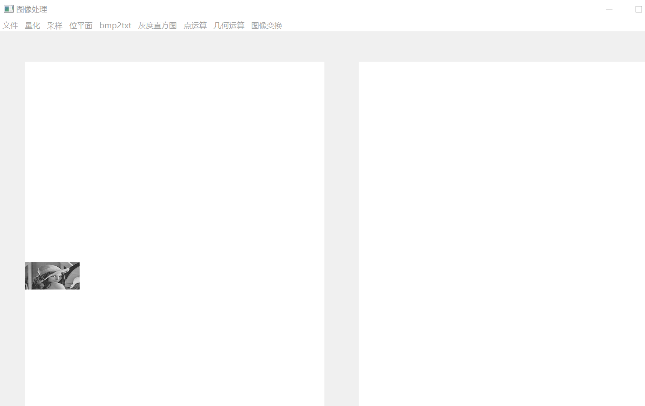
### 第三次作业：

打开lena.bmp

点击位平面->位平面：



打开lena128x64.bmp(由于’$’符号竖向较长，并且编辑器里行高比列宽长，所以我们采用较宽的图形效果更明显):



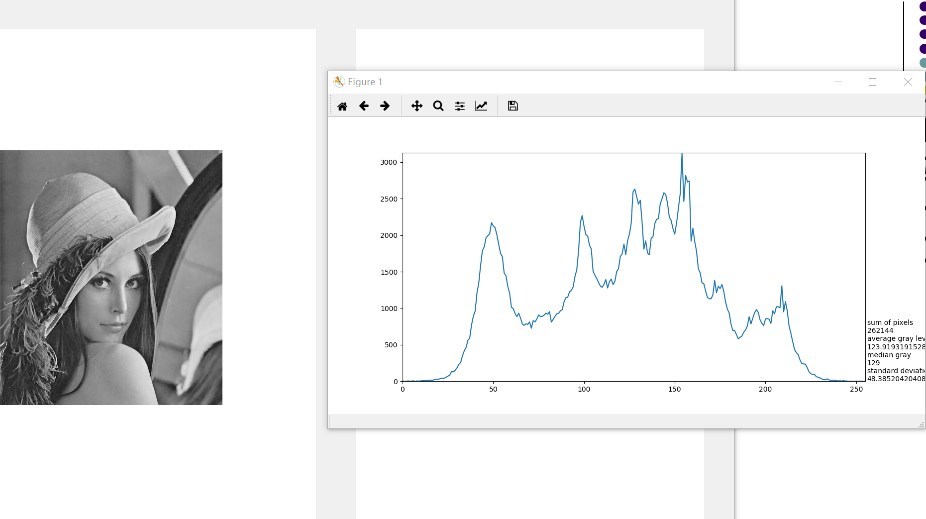
点击bmp2txt->bmp2txt,然后打开生成的bmp2txt.txt:



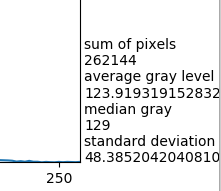
### 第四次作业：

打开lena.bmp

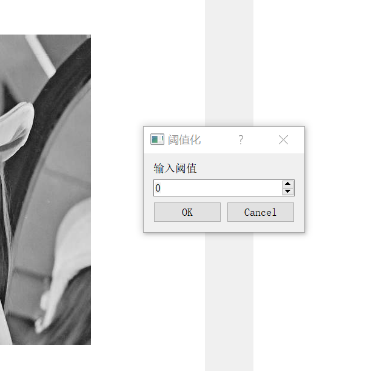
点击灰度直方图->灰度直方图(左):



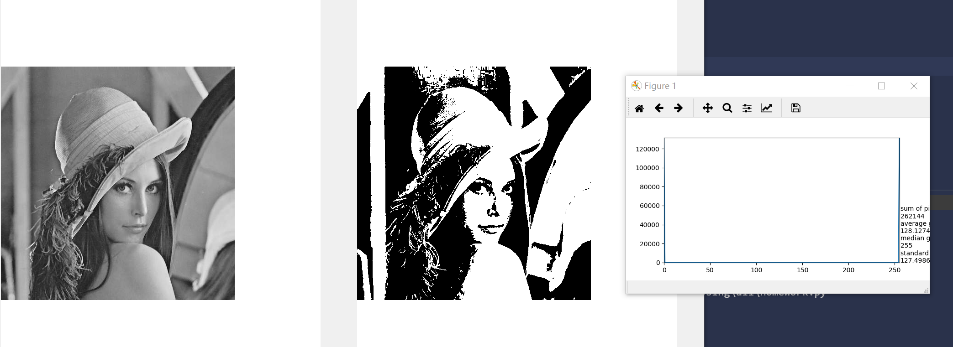
同时右下显示了直方图的数据，像素总数为262144,平均灰度为123.9，中值灰度为129,标准差为48.38



点击灰度直方图->阈值化:



输入阈值128:



再次点击灰度直方图->阈值化:

输入阈值140，可以发现右侧实时变化:



根据章作业要求，对之后的图像都附带灰度直方图显示，在八章之前的作业会自动给出弹出右侧图像的灰度直方图，由于一直弹窗略烦，会在八章后在灰度直方图功能下提供了“灰度直方图(右)”的功能，取消了自动显示，可以有选择的决定是否展示右侧图像的灰度直方图。

### 第五次作业：

打开lena.bmp

点击点运算->线性点运算1:



点击点运算->线性点运算2:



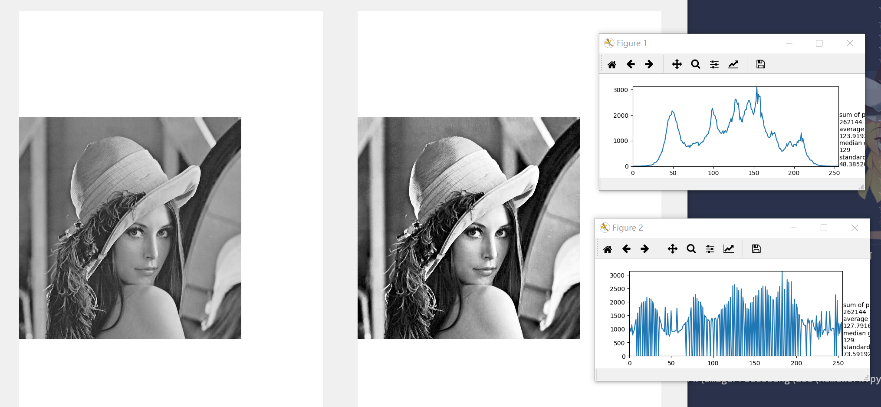
点击点运算->非线性点运算1:



点击点运算->非线性点运算2:



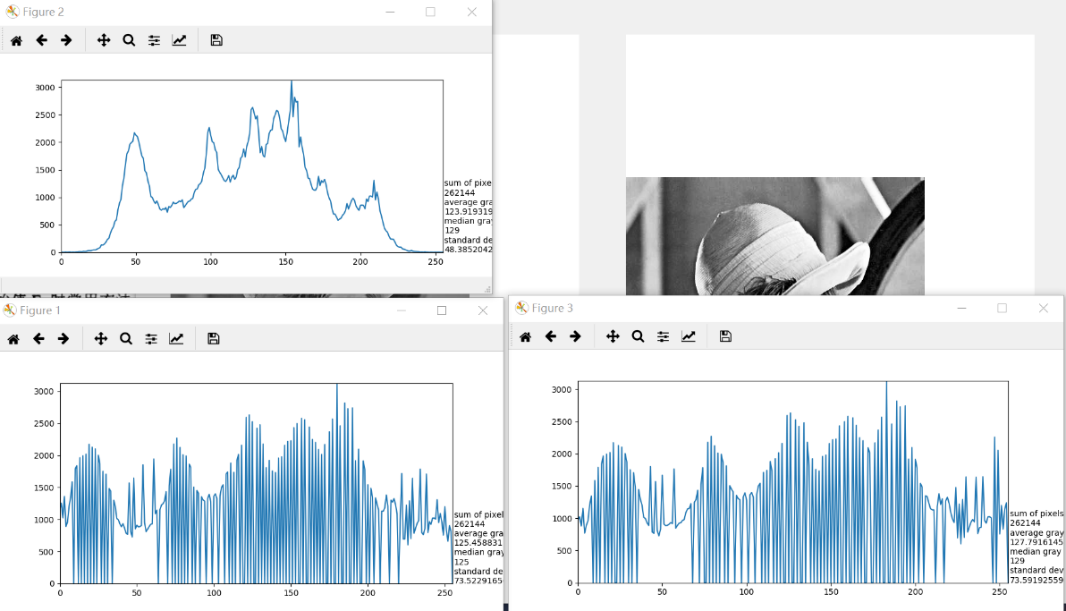
点击点运算->灰度均衡:



点击点运算->灰度均衡优化：

采用的优化算法是对直方图进行平滑处理，指数平滑递推公式为：

其中，为平滑前j点灰度值的像素出现的频数，为平滑后j点灰度值像素出现的频数。在迭代计算的时候经常令=，下图中，左上为原图，左下为优化后的灰度均衡算法，右下为原始灰度均衡算法。



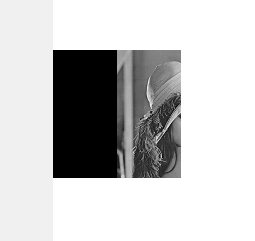
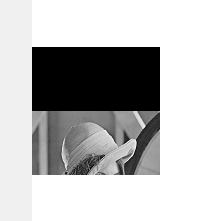
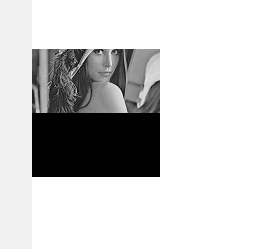
### 第六七章作业：

打开lena128x128.bmp

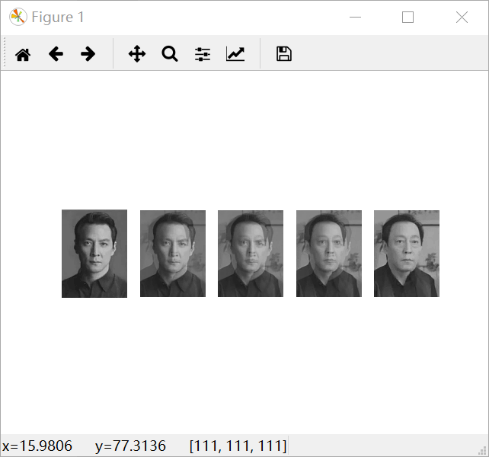
依次点击几何运算的各个功能，实现最近邻算法和双线性插值的缩放，旋转，平移，图A到图B功能。（放大例如放大至2，缩小例如缩小为0.5）

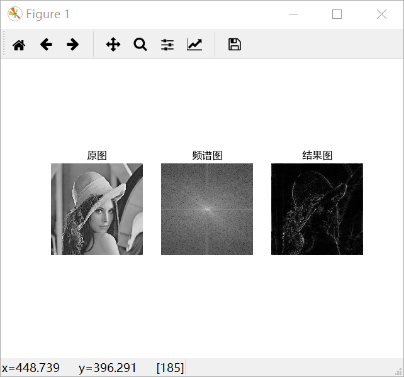
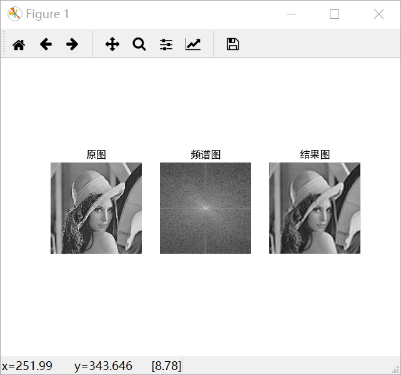


其中，点击图A到图B，会弹出两次打开文件的窗口，分别打开A图和B图,1.bmp和2.bmp，然后显示结果。

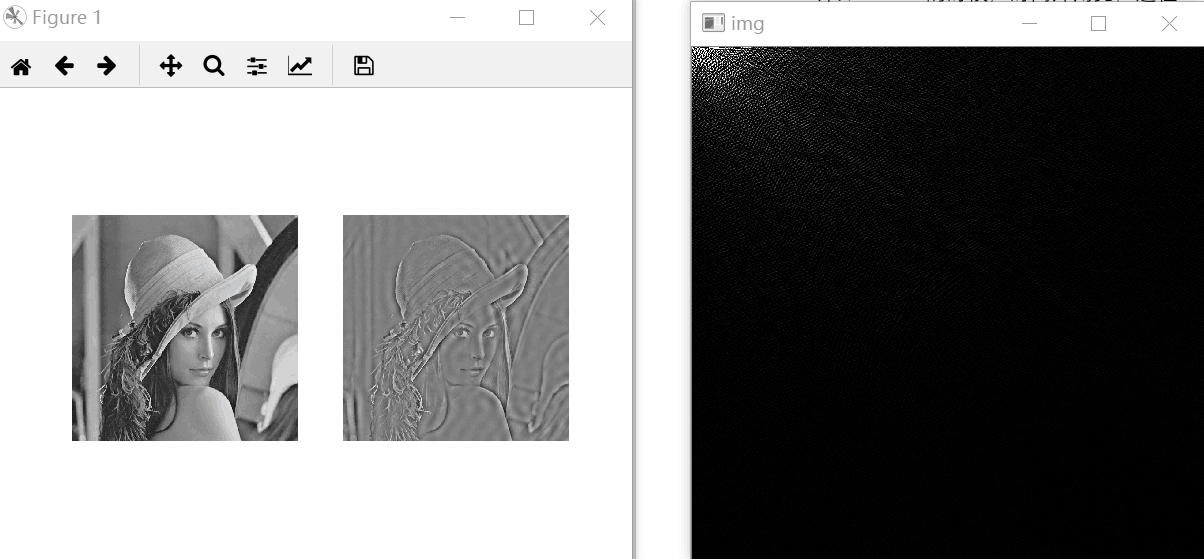


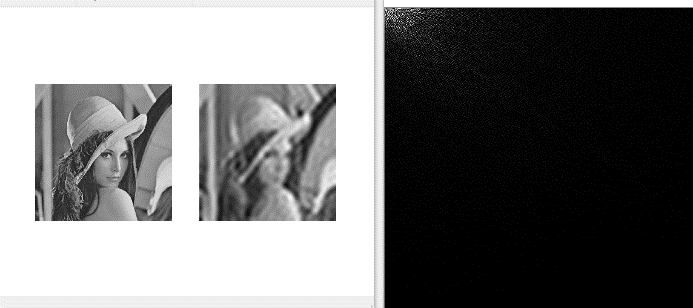
### 第八章作业：

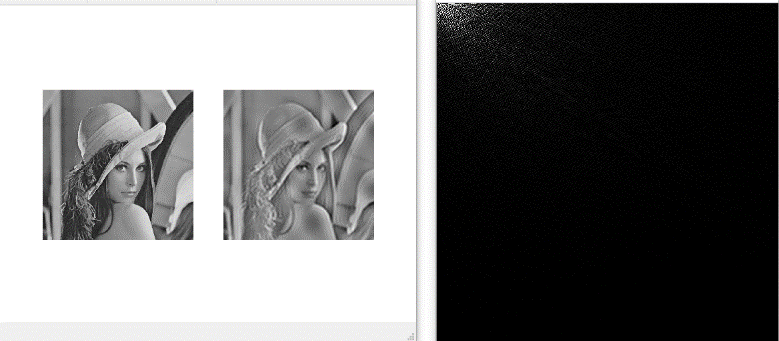
打开lena.bmp，依次点击列表里的功能，左侧是原图，中间是频谱图，右边是结果图，效果如下,：

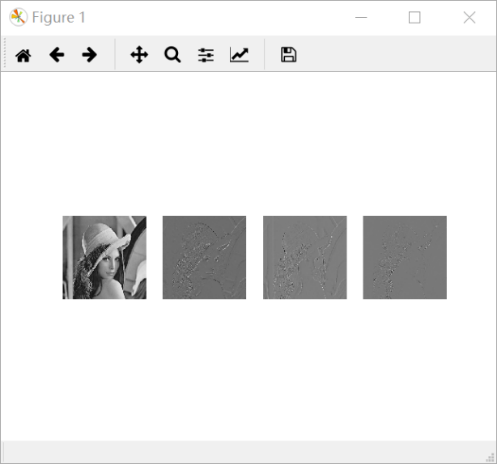
对于离散余弦变换，弹出的是频谱图：







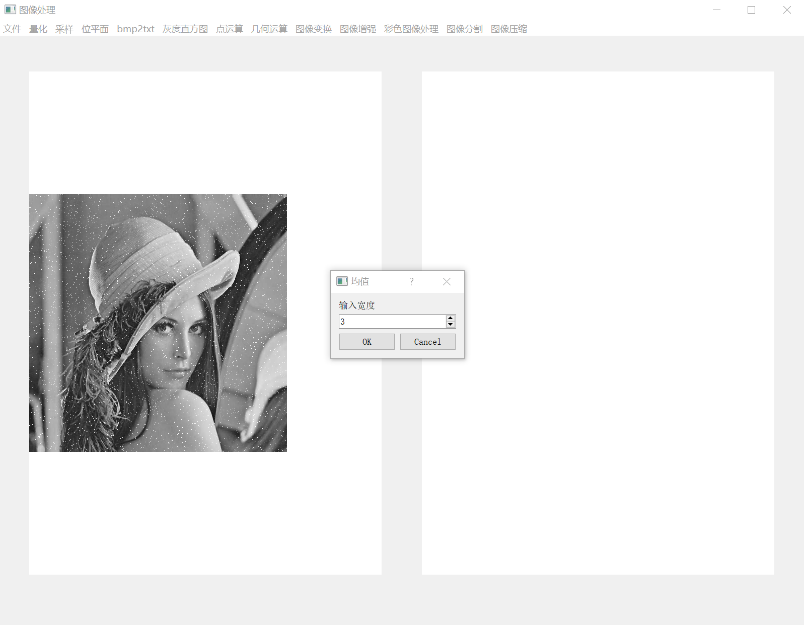
其中小波变换显示的四个图分别为低频分量，水平高频，垂直高频，对角线高频



### 第九、十章：

打开图片lenanoise.bmp,依次点击图像变换下的功能，

图像平滑（均值），输入3，





图像平滑(中值)，输入3：



图像平滑(K邻域)，输入宽度5，输入K为5，(这一步需要稍微等待一会，期间鼠标指针会转圈)

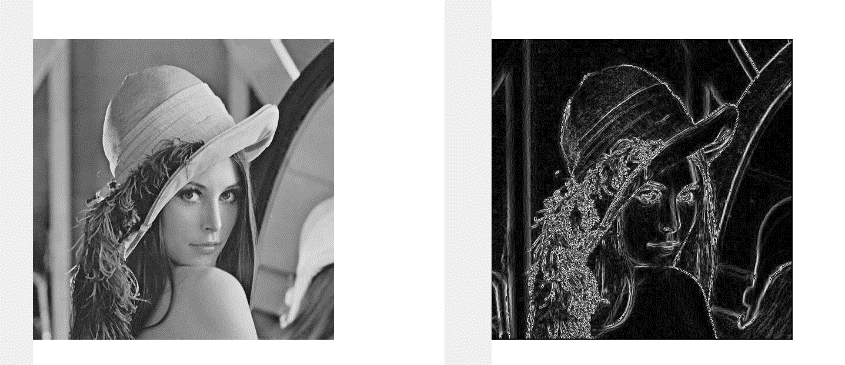


打开lena.bmp

图像锐化(Roberts)，分别为原图，处理完结果，与原图叠加后的效果:



图像锐化(Sobel)，左图为原图，右图为处理完的结果:



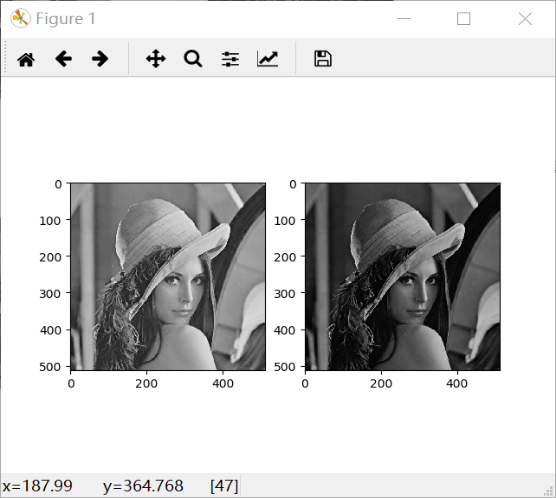
图像锐化(Prewitt),左图为原图，右图为处理完的结果：



任意模板，首先输入3，然后依次输入1 0 1 0 0 0 0 1 0，以空格分隔，



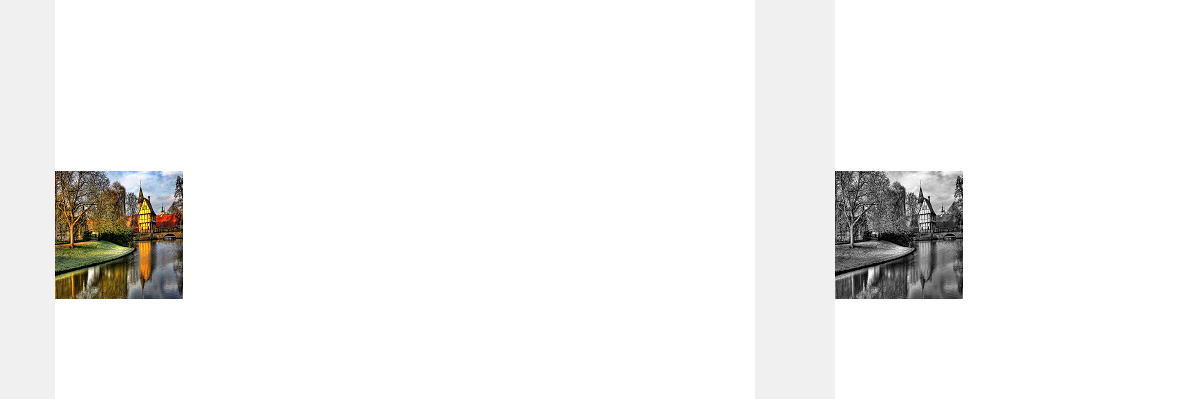
同态滤波：



# 第十一章

打开house.bmp,

彩色图像灰度化:



24位真彩色转256色(这一步的计算量很大，需要耐心等待一会儿，期间鼠标指针会转圈)：

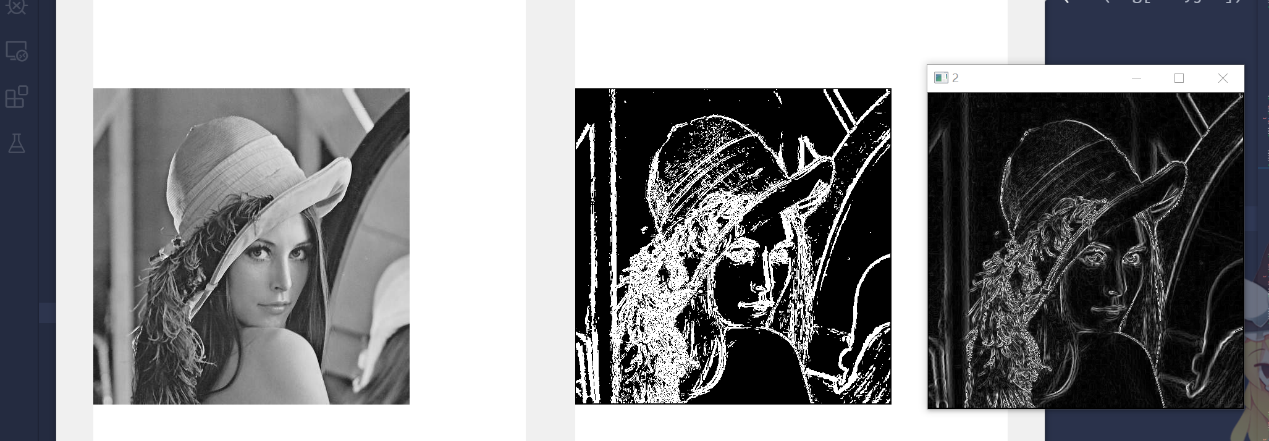


### 第十二章:

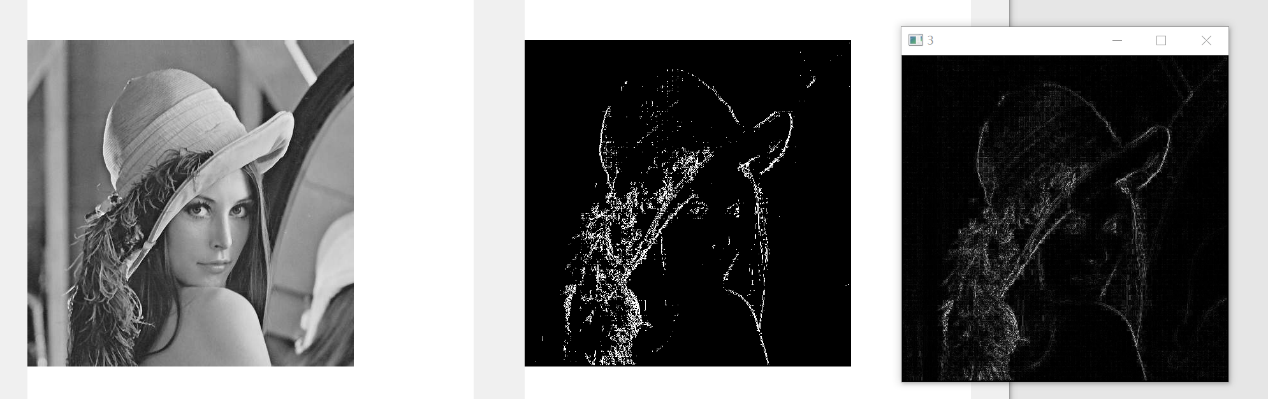
打开lena.bmp

图像分割(Sobel)，左侧为原图，右侧为边缘跟踪结果，弹窗为边缘检测结果:





图像分割(拉普拉斯):



打开3.png

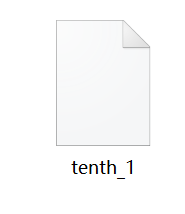
点击霍夫变换(也需要耐心等待一会儿):



### 第十三章

打开lena.bmp

点击Huffman编码，Huffman编码生成的文件在自动保存在文件夹中，：

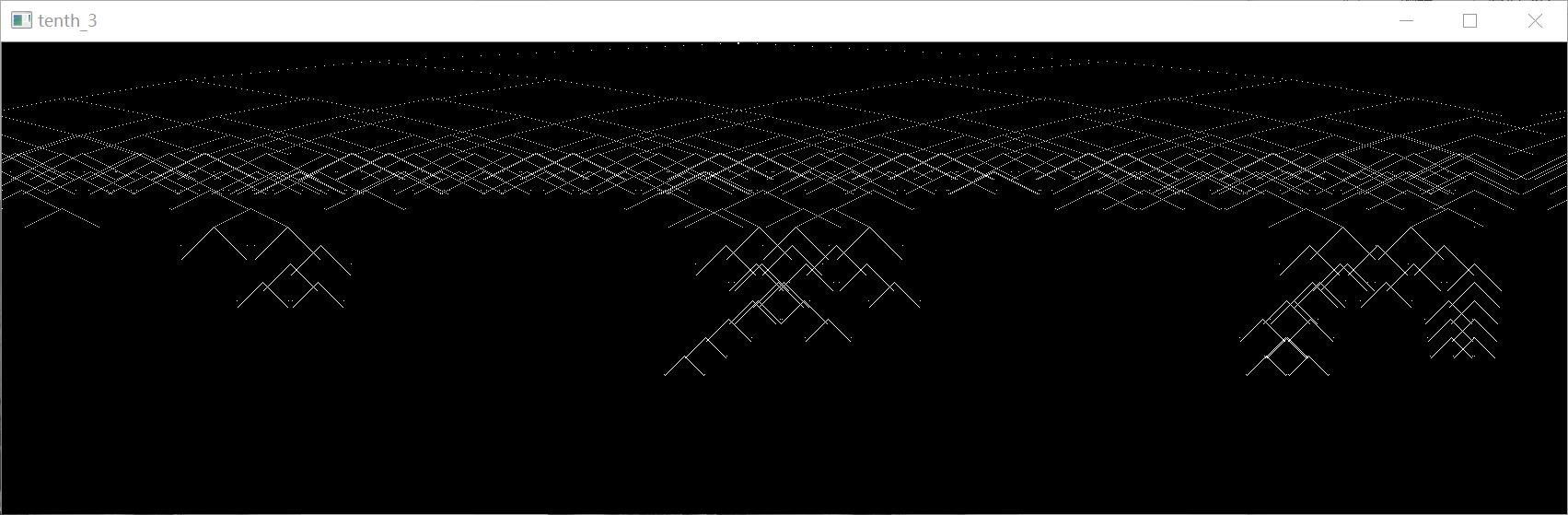


点击huffman解码，解码产生的图片显示在右侧:



Huffman树显示：

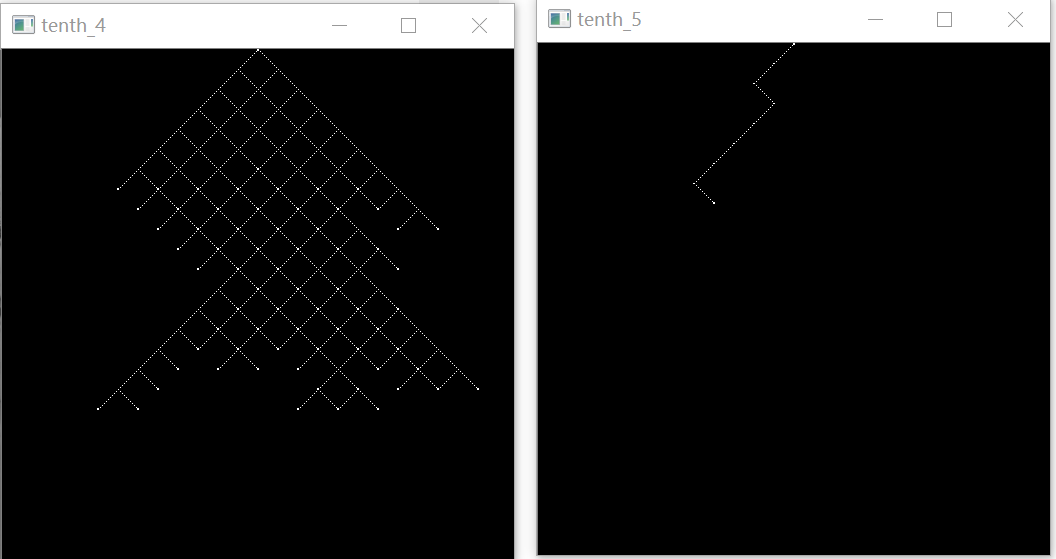
首先尝试了正常显示，该图huffman编码的长度大概有12位，也就是大概有一个12层高的树，按照二分的思想，最后一层的宽度大概有4096，并且需要逐层缩小间距，所以很难正常显示出来，这里举例正常显示的时候，只显示一部分就已经很困难了：



所以之后采用了两种方法显示。

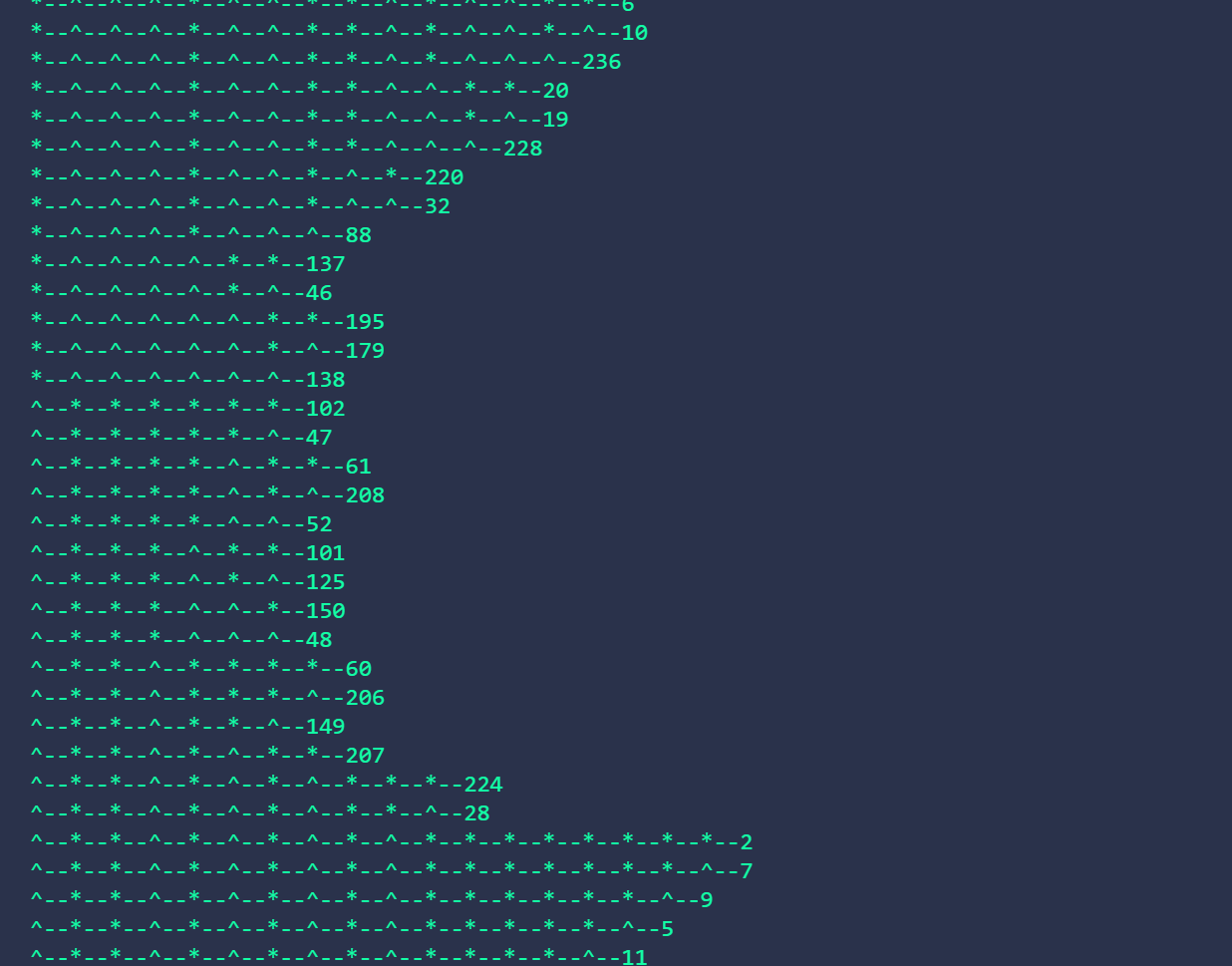
第一种就是不二分，这样会造成大量的点重合，但是却能很好的显示，我们提供了输入，可以输入某个像素来观察它在整个树的位置，比如这里我们点击huffman图形1，输入66。

左图是一棵有大量节点重合的树，右图是66这个像素在树中的位置。通过这种方法可以查看任意一个节点在树中的位置。



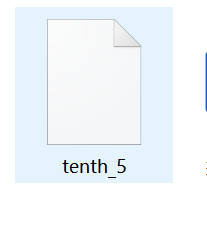
第二种方式是在控制台显示，点击huffman图形2。

根节点不显示，“--\*”表示左子树，“--^”表示右子树。



游程编码

点击游程编码，编码结果存在文件夹中



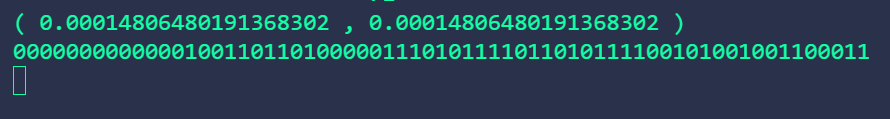
点击游程编码解码，结果显示在右侧：



算术编码

点击算术编码，结果显示在终端，

第一行是最终区间，第二行是在区间中取一个数，我取得是区间中值，以二进制的方式输出。



至此全部完成，所有结果均自动保存，合照如图：

