**数字音视频大作业实验报告**

姓名：李鑫然

班级：数媒2003班

学号：1191200303

目录

[一、 实验简介 2](#_Toc8742)

[二、 算法原理 3](#_Toc16632)

[3.1统计编码 3](#_Toc20874)

[3.1.1 哈夫曼编码 3](#_Toc21548)

[3.1.2哥伦布编码 3](#_Toc31984)

[3.1.3算术编码 4](#_Toc18330)

[3.2预测编码 5](#_Toc21485)

[3.2.1 DPCM编码 5](#_Toc30832)

[3.2.2余弦变换编码 6](#_Toc17183)

[3.2.3 小波变换编码 7](#_Toc15450)

[3.3 JPEG编解码 8](#_Toc32511)

[三、 实验心得 10](#_Toc15744)

[1. 前端问题： 10](#_Toc31327)

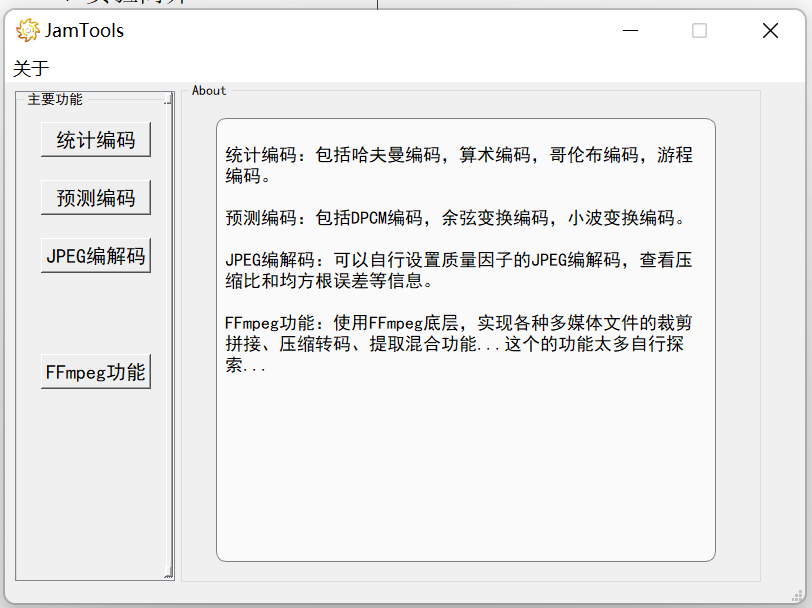
[2. 多线程问题 11](#_Toc30604)

[3. ffmpeg调用 11](#_Toc19930)

1. 实验简介

本软件是使用Python为编写语言，PYQT5作为UI界面工具，结合ffmpeg和各种底层算法完成的一款，以图像，视频，音频编解码功能为主的桌面应用。

下图为软件初始主页面：



1. 算法原理

**3.1统计编码**

统计编码是根据消息出现概率的分布特性而进行的压缩编码。

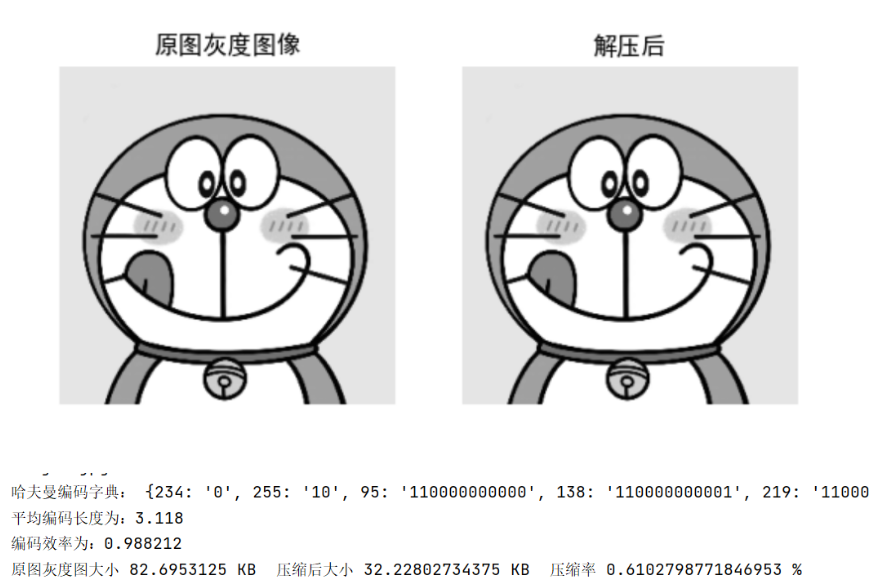
**3.1.1 哈夫曼编码**

哈夫曼编码是一种根据词频变化的变长二进制编码方式，多用于压缩算法。将信源符号按出现概率从大到小排列,然后选2个最小的结合,依次类推,直到剩下2个符号为止。使用[哈夫曼](https://so.csdn.net/so/search?q=%E5%93%88%E5%A4%AB%E6%9B%BC&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_38463737/article/details/_blank)编码，结果不唯一，平均码长相同，接近信源熵，方法容易简单。但是对于接近等概率分布的信源编码效率低。

代码说明：

1.要改变代码中的图片地址（地址不能有中文）

2.注意最后的plt.savefig('1.new.jpg')是保存plt图像，再在软件中进行显示。



该功能使用的压缩率是 1- 压缩后大小/压缩前大小

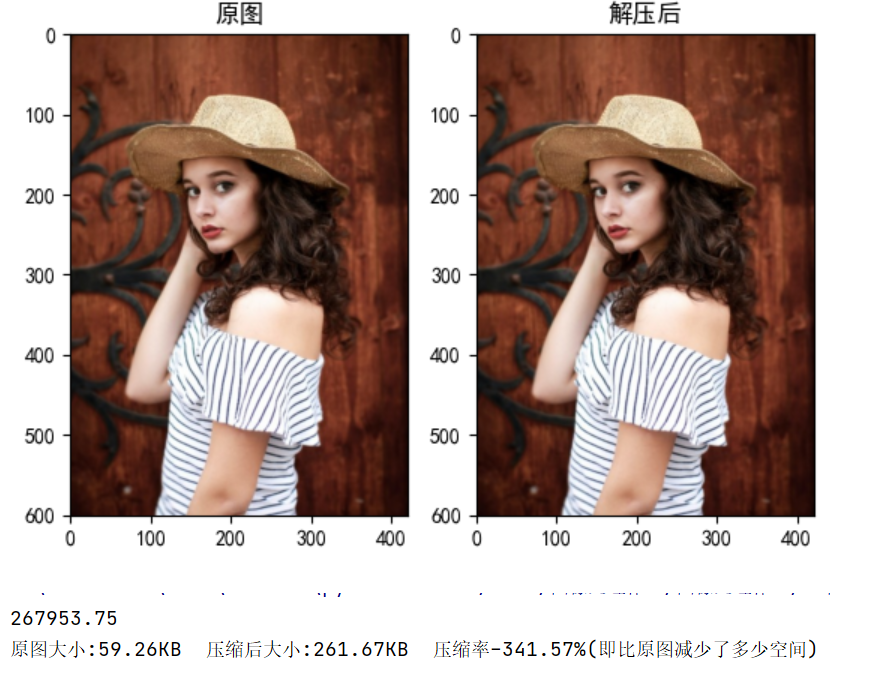
而且不是使用文件格式对比，而是比较图片像素占用的bit

**3.1.2哥伦布编码**

Golomb编码是一种分组编码，需要一个正整数参数m。用m对待编码的数字（n）进行求商（q=int(n/m)）和余（r=n%m），然后对商做一元编码，对余做固定长度的二进制编码（用固定bit数来表示余数）。

代码说明：

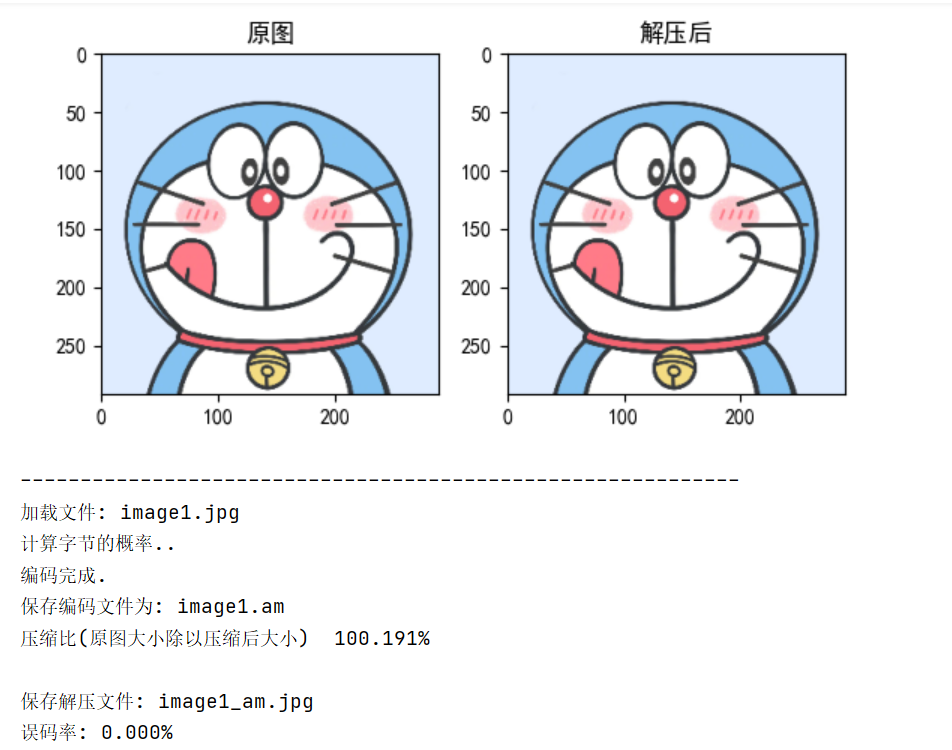
本代码实现是把图像文件按字节的方式读取，按字节编码。通过给定的m值，计算0到255的哥伦布编码，因为一个字节的范围也是0-255，这样就可以使用哥伦布编码替换像素值，从而实现编码。而且本代码实验室把编码好的哥伦布编码保存为ASCII码的文本文件，这样就相当于base64编码，有保密传输的作用。



可以看到空间反而大了很多，所以使用本算法压缩时不适合应用到色彩繁多的图片上。但如果是类似于base64都编码都可以使用

**3.1.3算术编码**

在算术编码中，消息用0到1之间的实数进行编码，算术编码用到两个基本的参数：符号的概率和它的编码间隔。信源符号的概率决定压缩编码的效率，也决定编码过程中信源符号的间隔，而这些间隔包含在0到1之间。编码过程中的间隔决定了符号压缩后的输出。

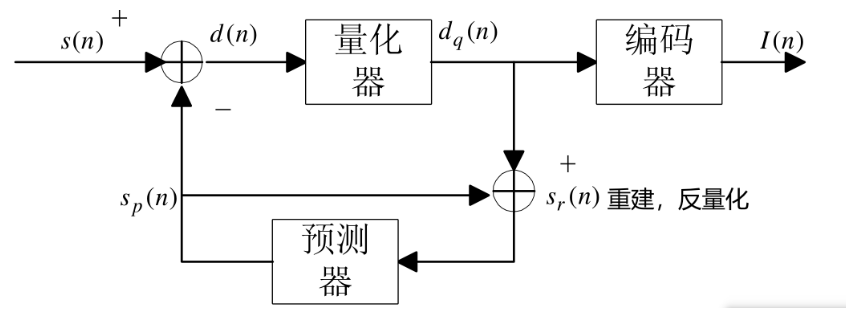


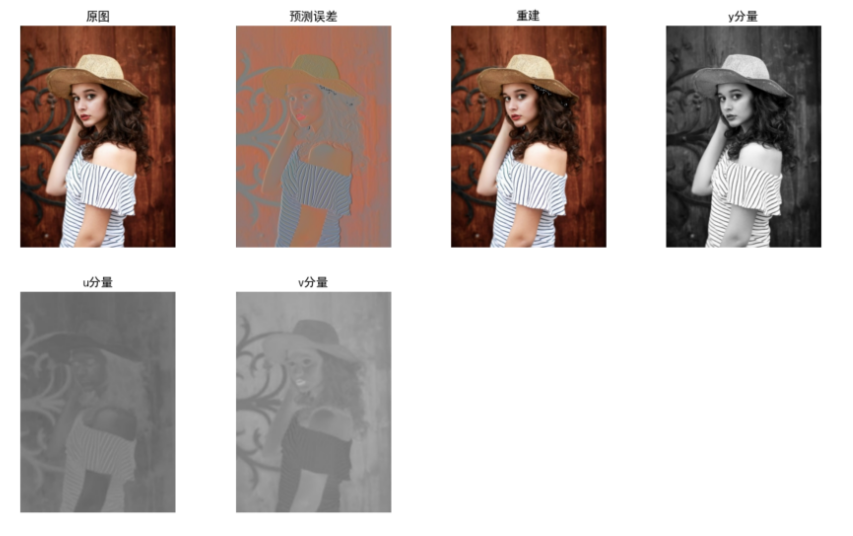
**3.2预测编码**

预测编码是根据离散信号之间存在着一定关联性的特点，利用前面一个或多个信号预测下一个信号进行，然后对实际值和预测值的差（预测误差）进行编码。

**3.2.1 DPCM编码**

DPCM 是差分预测编码调制的缩写，是比较典型的预测编码系统。在 DPCM 系统中，需要注意的是预测器的输入是已经解码以后的样本。之所以不用原始样本来做预测，是因为在解码端无法得到原始样本，只能得到存在误差的样本。因此，在 DPCM 编码器中实际内嵌了一个解码器，如下图编码器中虚线框中所示。





**3.2.2余弦变换编码**

离散余弦变换表示为 DCT( Discrete Cosine Transformation)，常用于图像处理和图像识别等。

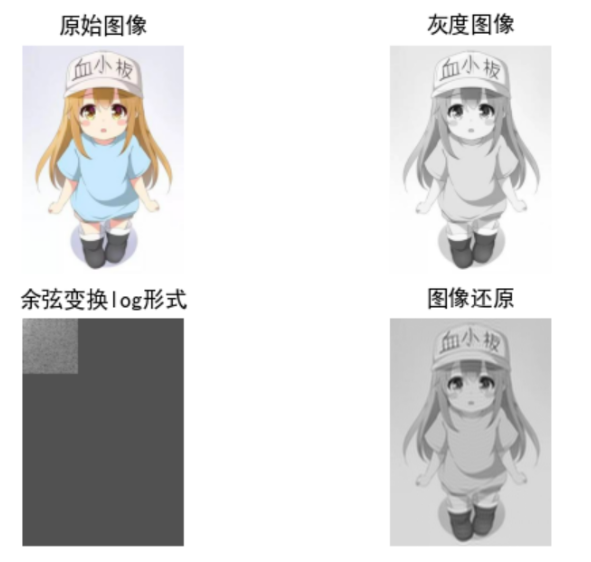
求二维图像的离散余弦变换要进行以下步骤：

1获得图像的二维数据矩阵 f(x,y)；

2求离散余弦变换的系数矩阵[A]；

3求系数矩阵对应的转置矩阵[A]T；

4根据公式公式计算离散余弦变换；

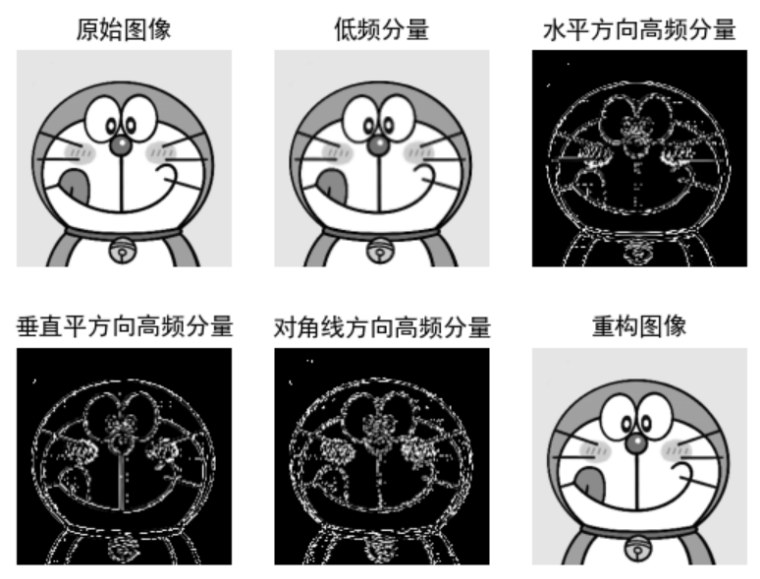


**3.2.3 小波变换编码**

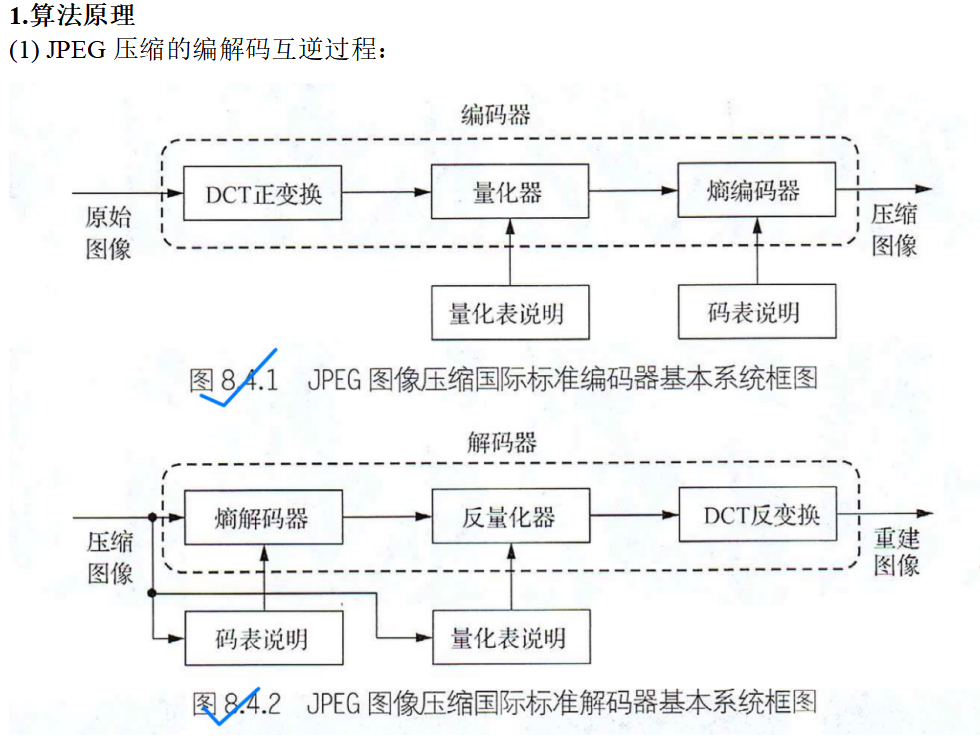
图像的二维离散小波分解和重构过程如下图所示，

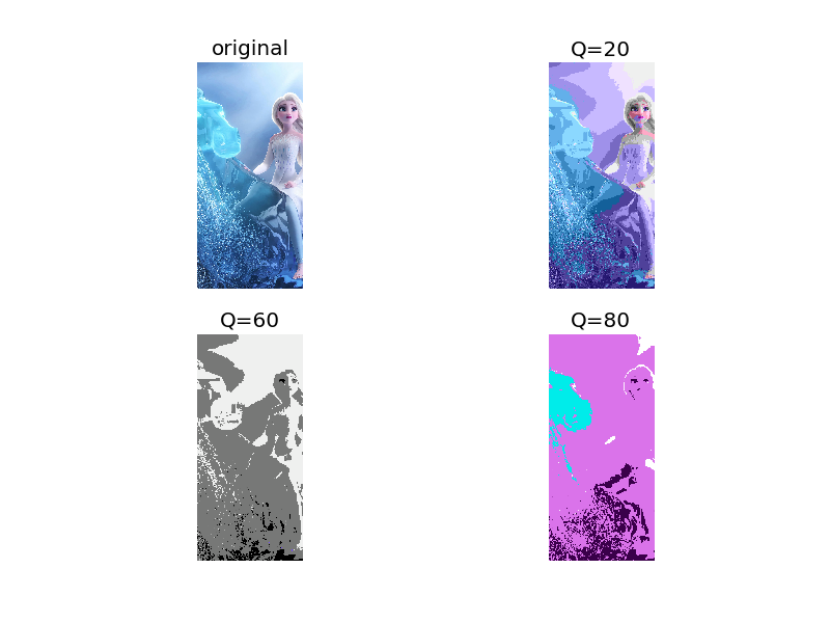
分解过程可描述为：首先对图像的每一行进行 1D-DWT，获得原始图像在水平方向上的低频分量 L 和高频分量 H，然后对变换所得数据的每一列进行 1D-DWT，获得原始图像在水平和垂直方向上的低频分量 LL、水平方向上的低频和垂直方向上的高频 LH、水平方向上的高频和垂直方向上的低频 HL 以及水平和垂直方向上的的高频分量 HH。

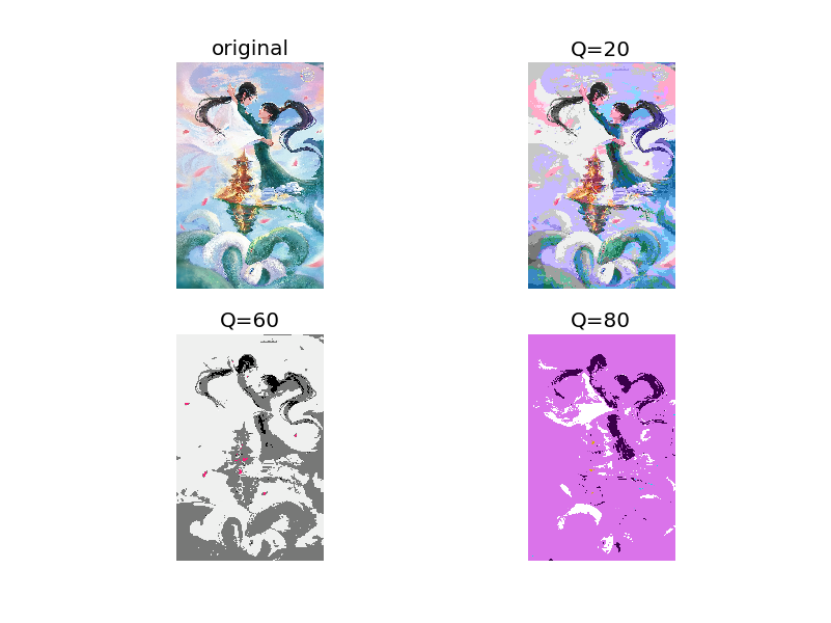
重构过程可描述为：首先对变换结果的每一列进行以为离散小波逆变换，再对变换所得数据的每一行进行一维离散小波逆变换，即可获得重构图像。由上述过程可以看出，图像的小波分解是一个将信号按照低频和有向高频进行分离的过程，分解过程中还可以根据需要对得到的 LL 分量进行进一步的小波分解，直至达到要求。



**3.3 JPEG编解码**



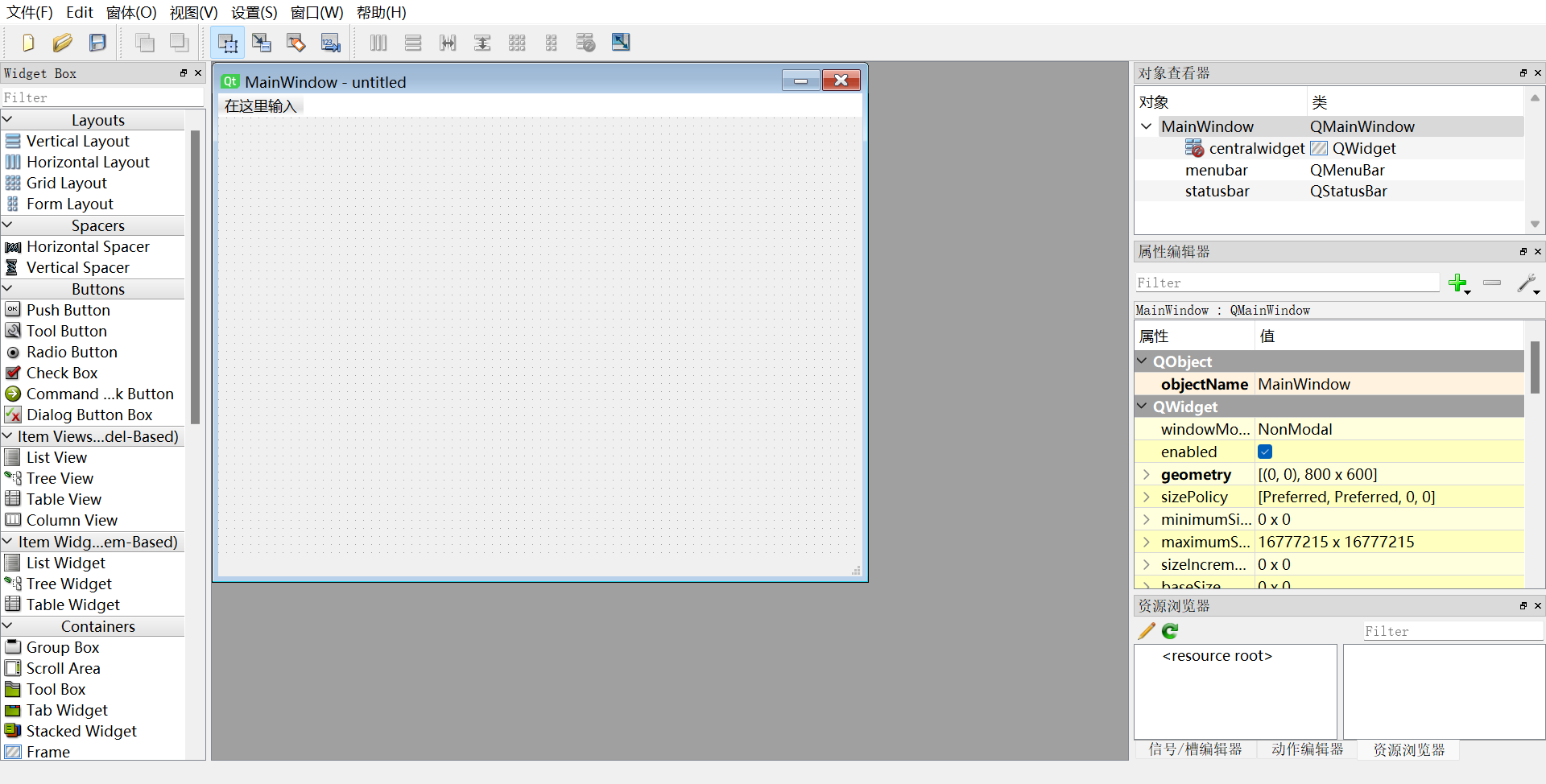




注意：比较压缩率是 比较 原图与gpj文件的 大小 而非 解压后的bmp ，不管Q选择多少 解压后的bmp大小都是一样的。

1. 实验心得
2. 前端问题：

我在编写代码的同时，更深入熟练了pyqt5的使用。感觉比较麻烦的一点是，Python的pyqt不像matlab的前端组件，运行时可以一起运行连接的函数功能。而是必须将用designer设计出的.ui文件转换为.py文件，再进行修改。这样的编写方式不方便做反复修改较多的前端组件，造成了很多的细节变量更改的麻烦。



图：Qtdesigner页面

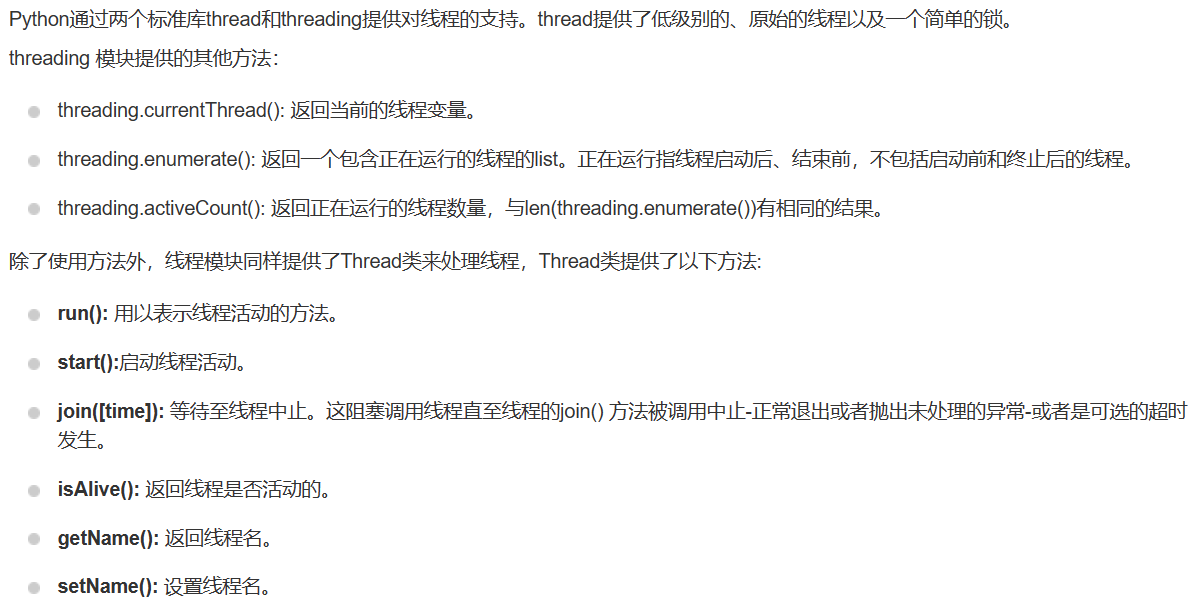
其次在前端图像信息的呈现上做了很多改进：

原本编码的图像是用matplot完成的，考虑到如何将这种结果信息呈现在软件上。

plt有一个功能是plt.save可以保存作图图片，在保存在根文件夹后，可以使用pyqt5的label组件显示规定路径名称的图片，这样就可以将结果实时的呈现，也可以在后台查看更清晰的结果。有一个小细节是，需要在label组件里预先调整好图像的显示大小，否则可能出现显示图像不全的问题。

1. 多线程问题

在不断添加新功能时，考虑实际场景中功能运行时间较长，且功能较多的情况下的解决办法。查阅过后可以在软件程序中添加多线程部分，使用thread库可以实现。完成编写后，发现可以避免等待时间很长的程序处理，软件使用感更好，加快处理速度，也可以了解处理的进度。



1. ffmpeg调用

要实现在python中调用ffmpeg，需要了解一下[subprocess](https://docs.python.org/3/library/subprocess.html)模块。简单来说，subprocess模块就相当于一个包壳的命令行，原则上可以在命令行中实现的事情都可以使用subprocess在python中实现。

最简单的使用：

import subprocess

subprocess.Popen(

'"D:\python\_work\Jamtools/bin/ffmpeg" -thread\_queue\_size 16 -f gdigrab -i desktop -vcodec libx264 "D:/图片/视频/Jam\_screenrecord/test.mp4" -y',

shell=True)

上面的代码实现了python内使用ffmpeg抓屏的功能，主要使用了subprocess的Popen类。

该类第一个参数是要输入的命令，和你在cmd命令行或powershell中输入的完全一样，要用引号括起来(是一个字符串)"D:\python\_work\Jamtools/bin/ffmpeg"是我的ffmpeg的路径，如果你已经把ffmpeg添加进了系统环境变量，可以直接用ffmpeg代替；

然后第二个参数shell=True，表示可以在python的命令行中看到输出；

有一个小问题是应该如果停止ffmpeg的运行；、

ffmpeg在普通命令行中是怎么中止的，是直接按键盘上的‘q’键。要在subprocess接受用户输入，就需要用到subprocess的PIPE(管道)。如果说subprocess是一个包壳的命令行，PIPE就相当于这个命令行的输入(键盘)、输出(显示)。此外，subprocess中可以通过管道的stderr接口获得上图中命令行的输出，分析即可得到设备名称。其中我们通过stderr.read()把输出(格式为字符串)存在relog中，就可以通过普通的字符串分析获取输出内容了。