****

数字图像处理

实验报告

姓名： 郑云川

学号： 09015131

东南大学计算机科学与工程学院

School of Computer Science & Engineering

Southeast University

二0 一八 年 一 月

# 实验四 医学图像处理显示简化软件

## 一、实验目标

本实验的任务结合前三次实验的积累，对具有2096级灰度的医学透视图像进行处理并显示，根据自己的理解设计用户界面和功能。通过实验，我们能够：

（1）从自定义格式的文件中读取图像数据（图像文件存放格式定义与实验三相同，但实验使用的图像为真实医用 DR 图像，每像素存放 2 字节，像素灰度范围 0-4095，具体图像参数包含在文件头中）；

（2）实现图像的缩放；

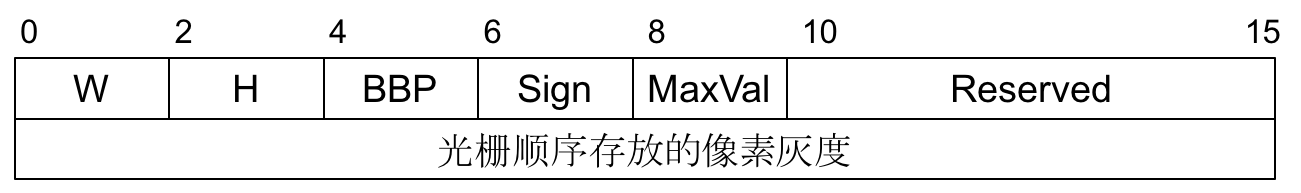
（3）使用“灰度窗方法”显示灰度图像；

（4）进行对比度增强。

## 二、目标图像分析

### 1 自定义图像

与上一次试验类似，根据实验手册所述，自定义图像格式dr的数据分为两部分，图像描述参数和图像数据，其中，图像描述参数描述了图像的总体特性，图像数据为像素的灰度值，像素按光栅扫描顺序存放。具体如下：（字节地址）



W：图像宽（单位像素）；

H：图像高（单位像素）；

BBP：每像素灰度存放的bit数，取值8或16；

Sign：像素灰度是否有符号数，取值0（无符号数）或1（有符号数）

MaxVal：图像所有像素的最大灰度值；

Reserved：保留区。

### 2 图像显示

此次所给的两幅图像与上次实验所给的图像结构相同，不同之处是上一次的两张图的像素灰度都是用8位来表示的，在屏幕上可以直接显示的；而对于本次实验的图像，用文本编辑器打开查看二进制数据，对照图像结构可以知晓，这两张图的每个像素的灰度是用16位来表示，普通计算机的屏幕是不支持4096级灰度的显示，那么必须把这4096级的灰度映射到256级灰度范围中，用到了“灰度窗方法”，涉及窗宽窗位的概念，将会在下面具体介绍。

## 三、软件功能描述

### 1 读取图像并显示

能够读取要处理的两幅具有4096级灰度的DR图像并且在界面中的picture control上进行显示。

### 2 图像平移与缩放

因为本实验室一个简化版的图像处理软件，考虑到用户体验，并结合领域内优秀软件PhotoShop，图像的平移与缩放通过用户鼠标的行为来操控，鼠标在图像上点击并拖动可以平移图像，利用鼠标滚轮的滚动实现图像的缩放。

### 3 灰度窗方法

在软件右侧上方有窗宽和窗位的slider control，拖动它们可以调节窗宽和窗位；

### 4 对比度增强

这部分采用上一次实验实现的ACE算法，细节此处不再赘述，可以看到软件的ACE的那一组slider，拖动它们可以进行对比度的调节。

## 四、实现方法描述

### 1 灰度窗方法

窗技术(Window Technique)是医生用以观察不同密度正常组织或病变的一种显示技术，其包括窗宽(window width)和窗位(window level)。由于各种不同组织结构或病变具有不同的像素值，因些欲显示某一组织结构细节时，应选择适合观察组织结构的窗宽窗位，以获得显示最佳效果。

窗宽是CT/ DR图像上显示的CT/DR值,在此CT/DR值范围内组织和病变均以不同的模拟灰度显示，而CT/DR值高于此范围的组织和病变，无论是高于多少，都均为白影显示，不再有灰度差异，反之，低于此范围的组织，不论是低于多少，均为黑影显示，也无灰度差异。增大窗宽，则图像所示CT/DR值范围加大，显示具有不同密度的组织结构增多，但各结构这间的灰度别减少；减少窗宽，则显示组织结构减少，而各结构这间的灰度别增加。

窗位是窗的中心位置。同样的窗宽，由于窗位不同，其包括CT/DR范围的CT/DR值有差异。例如窗宽(w)同为w =60，当窗位为L =0时，其CT/DR值范围为－30～＋30；如窗位是＋10时，则CT/DR值范围为－20～＋40。通常欲观察某一组织的结构及发生的病变，应以该组织的CT/DR值为窗位。

在本次实验中我的理解就是把指定的一段灰度范围的图像映射到0～255这个区间，原始像素灰度在指定范围之外的，直接映射为0或者255，以突出这个范围内我们感兴趣的组织器官的特征。主要的就是以下的算法：

min = (2\*window\_center - window\_width)/2.0 + 0.5;

max = (2\*window\_center + window\_width)/2.0 + 0.5;

for (i = 0; i != nNumPixels; i++){

disp\_pixel\_val = (pixel\_val - min)\*255.0/(double)(max - min);

}

### 2 图像显示

把图像显示放在灰度窗之后是因为图像显示要用到灰度窗方法，毕竟原始图像是4096级灰度，不能直接显示，在读取图像头、像素数据后对像素数据使用灰度窗方法，映射至0～255。根据实验3的经验，如果直接使用setPixel()，的确能够显示图像，但是图像一行行刷新出来，体验非常不好，按照惯例，把数据塞进BITMAP中进行显示。同时，这里涉及到双缓存的机制。

双缓冲即在内存中创建一个与屏幕绘图区域一致的对象，先将图形绘制到内存中的这个对象上，再一次性将这个对象上的图形拷贝到屏幕上，这样能大大加快绘图的速度。双缓冲实现过程如下：

1、在内存中创建与画布一致的缓冲区

2、在缓冲区画图

3、将缓冲区位图拷贝到当前画布上

4、释放内存缓冲区

在这之后，我给界面加上滚动条，想着如果图片过大，通过滚动条来实现浏览。实现之后却发现，如果滚动时刷新背景，那么屏幕上会出现严重闪烁，如果禁止刷新背景，那么原先是空白的区域就会变得一团糟，同样很影响体验。在网上查找诸多资料后，我依然没有解决这个问题，本身关于纯粹win32 API编程的资料远远少于MFC的，关于这方面的就更少了。经过交流后，仍然无法解决该问题，为用户体验，转战MFC，还好MFC本是win32的包装，二者本质是一样的，代码做少量修改即可移植。再一次查阅资料，利用MFC提供的gdiplus中的Graphics和Image类，结合双缓冲，可以在图像放缩和移动的过程中规避闪烁。Bitmap类继承于Image类，通过给出BITMAPINFO和像素数组即可以构造。

### 3 图像放缩及平移

这里的实现是重写鼠标点击、释放以及滚轮滚动的响应事件。点击事件会得到鼠标当前的位置，计算出相对于picture control所在区域的位置，再与保存的上一次图片的位移值做比较，计算出本次图片需要移动的距离，放缩的话也结合了平移，因为是以鼠标当前的位置为中心放大的。这里借助Image类调用相关成员函数便可以实现缩放的功能，非常方便，但在上一个版本手写的过程中，也发现了问题。

系统在显示bmp图像的时候需要每一行按照四个字节对齐，也就是说如果是24位表示一个像素，那么图像宽度\*3必须为4的整倍数，在前面实验中这个条件满足，所以并没有发现，而在放缩中根据比例不同，就很有可能出现。这样显示出来的图像是错位的。解决方法1是换setPixel()这个一定可以，但是因为性能被否定；方案2是用32位表示一个像素，这样图像宽度不论是多少都可以；方案3，如果lineByte表示图像一行的字节数，那(lineByte + 3)/4 \* 4便可以得到一个大于等于它的最小的是4的倍数的值，在填充像素的时候每一行多出来的补0就行。方案2简单易行，但会占用更多的内存，方案3增加的可以忽略不计，因此选用方案3.

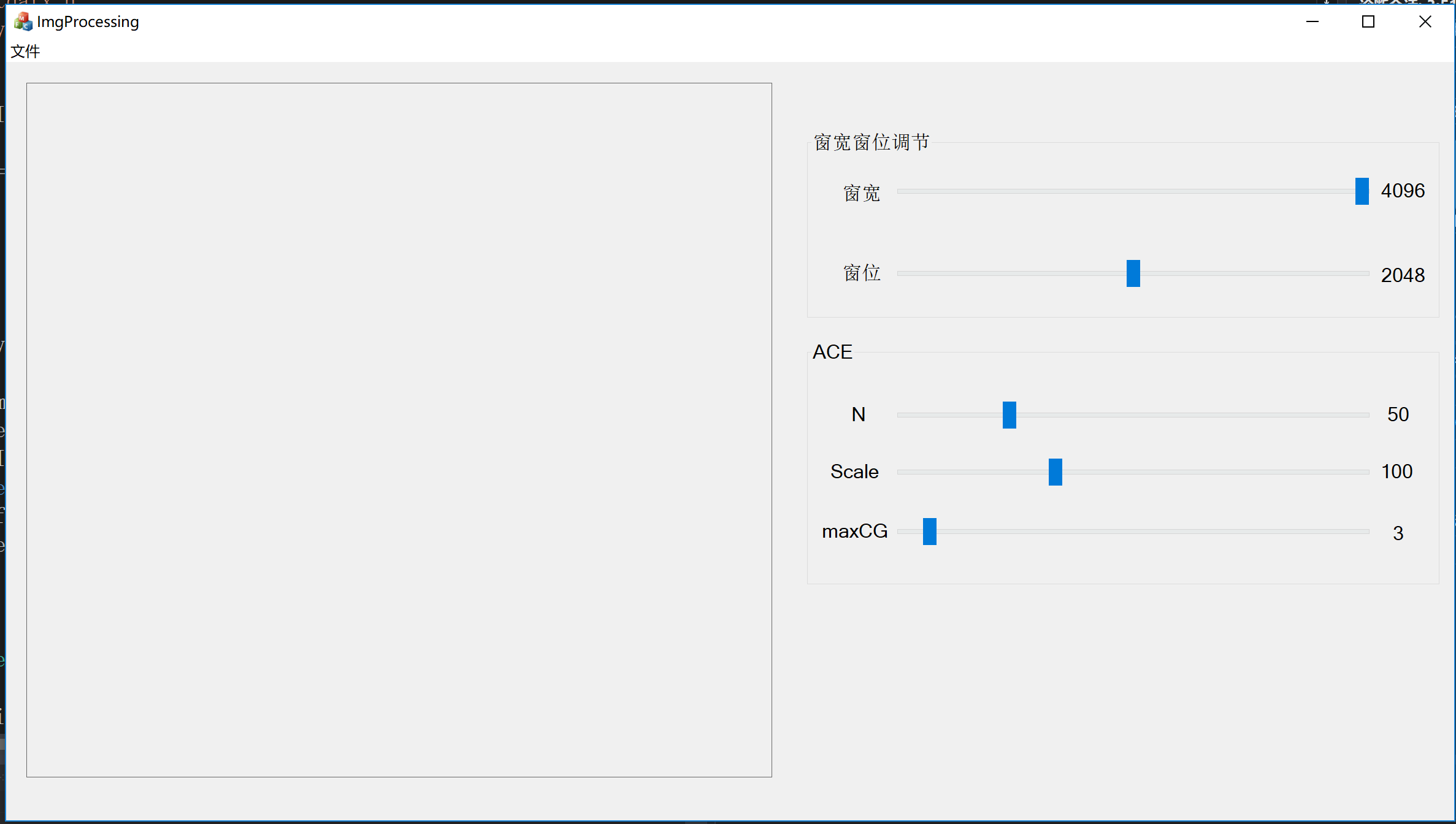
### 4 ACE

同第三次实验的算法一样，根据公式即可：

## 五、软件功能测试

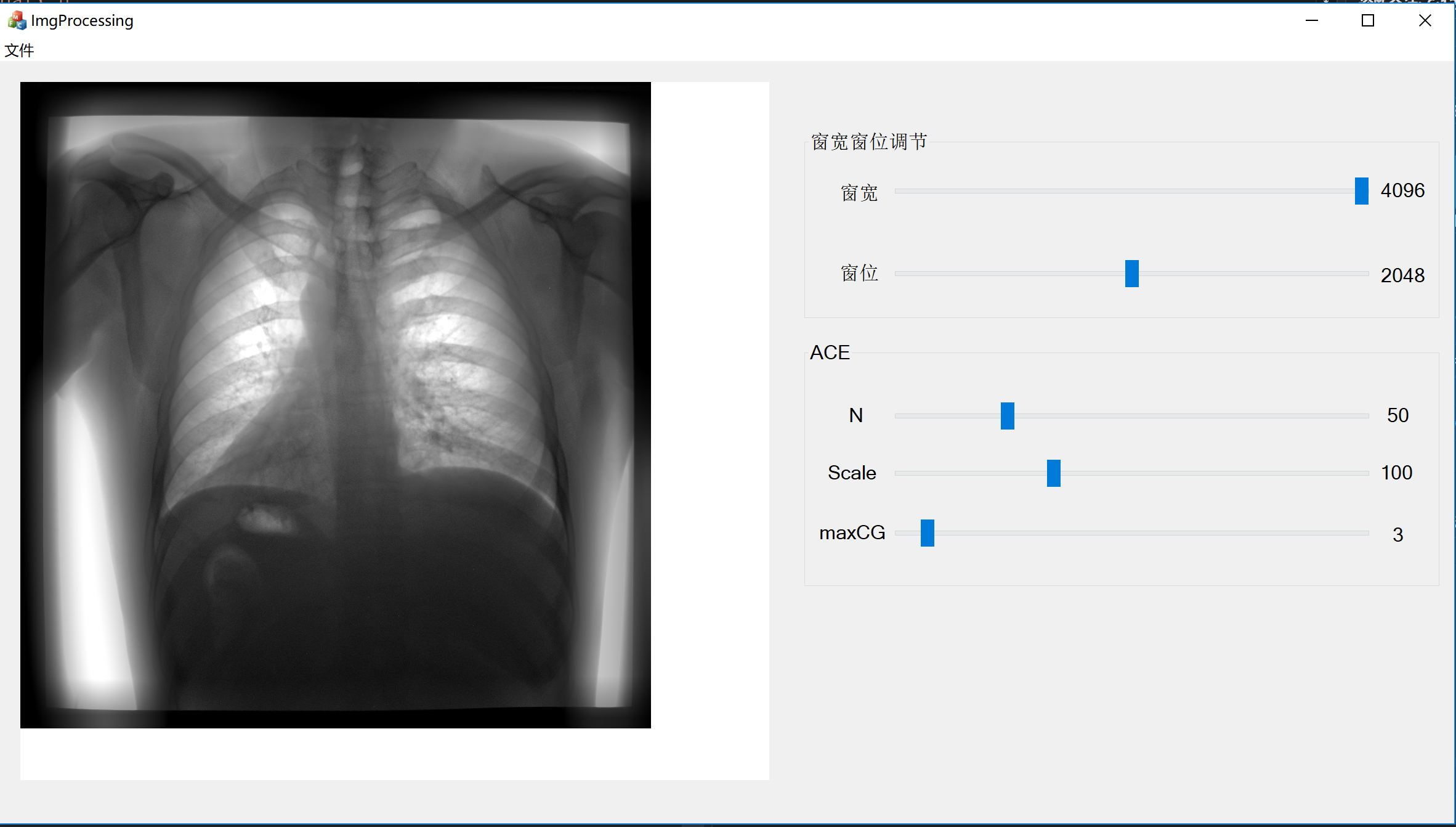
### 1 打开显示图像

点击菜单中文件，打开图像，在对话框中选择实验所给的图片即可。



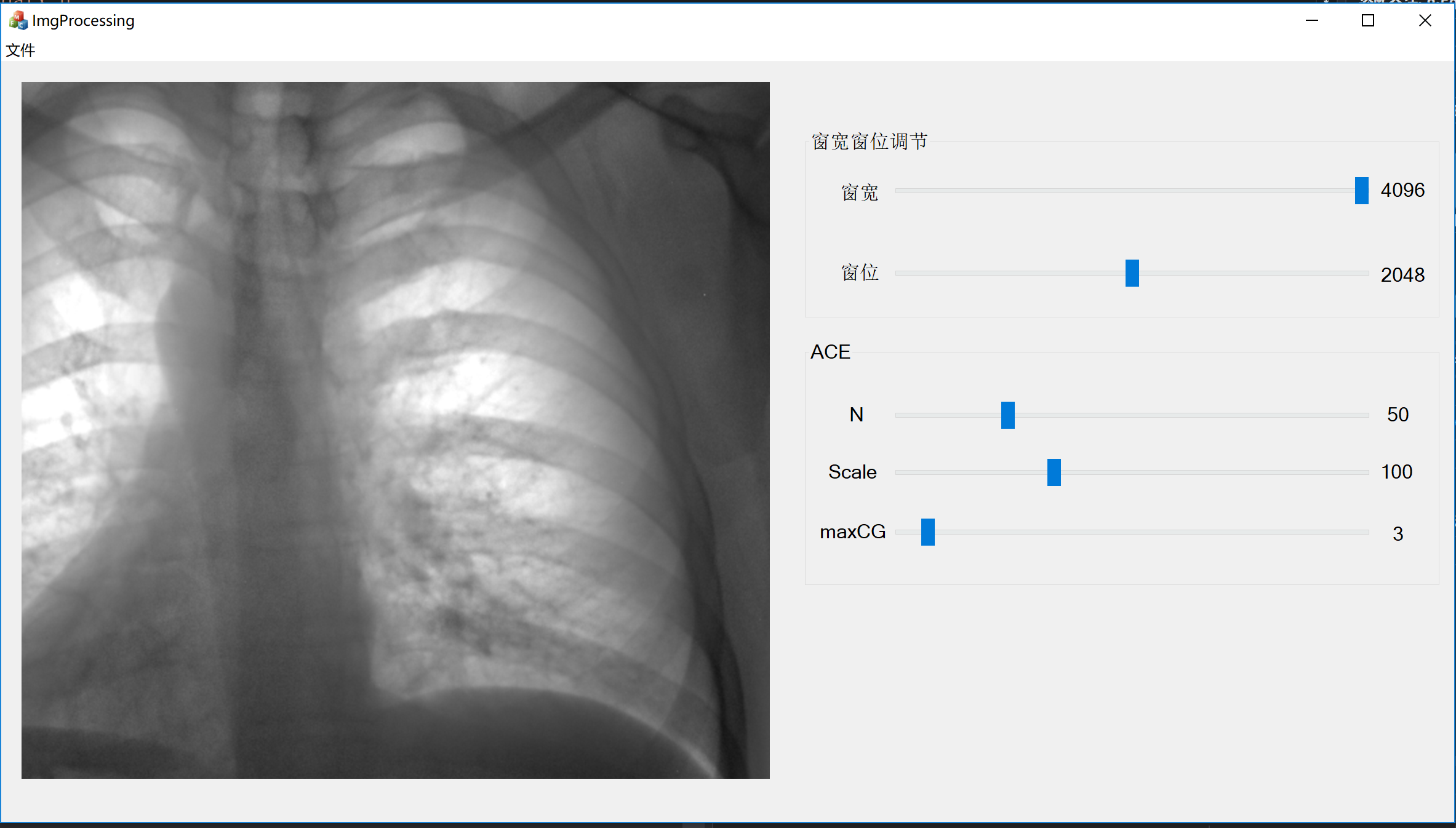
打开图像后能看到图像的显示:





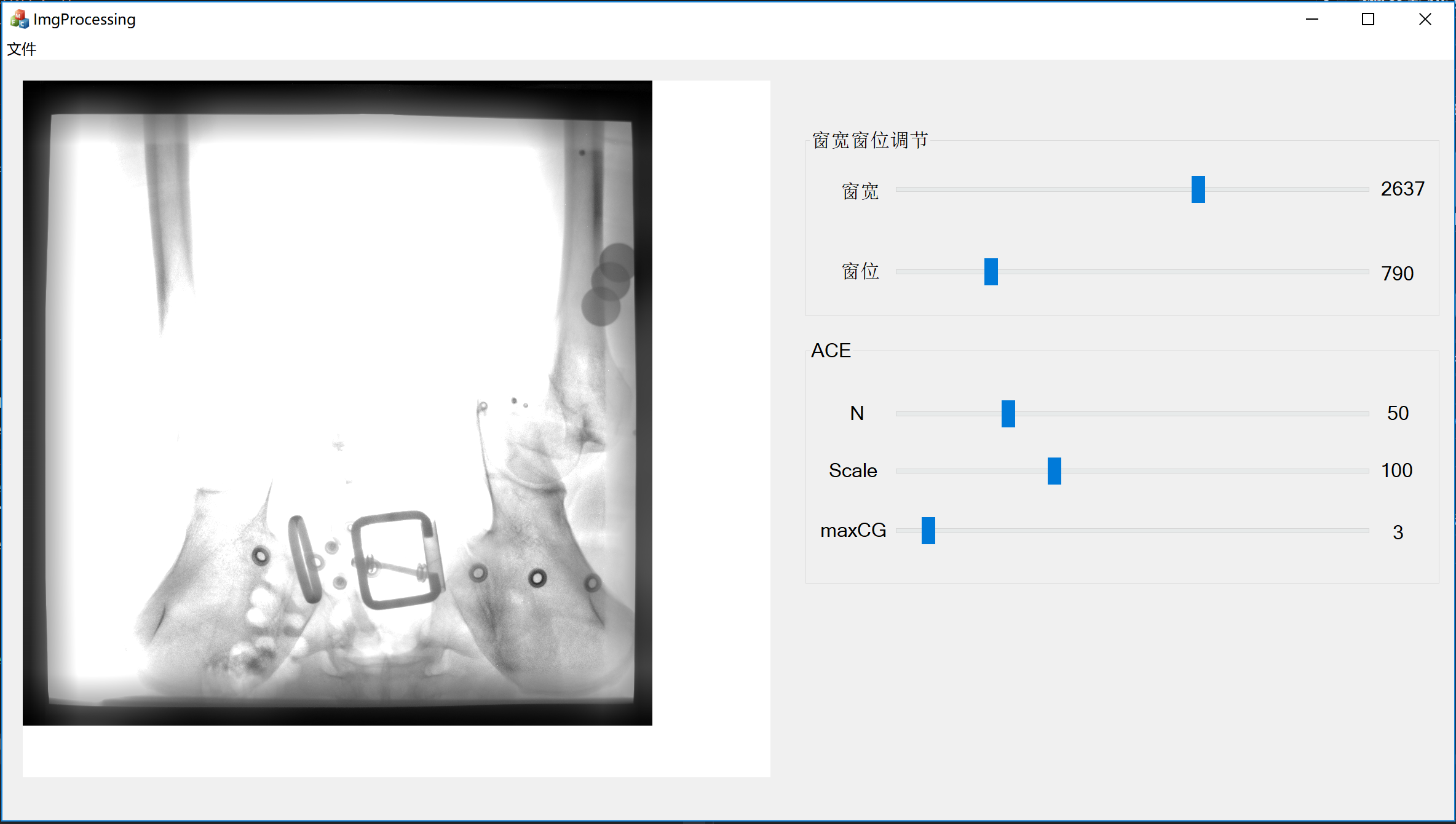
### 2 图像平移、缩放

鼠标拖拽图片即可以实现平移，在想要放大或者缩小的部位滚动滚轮即可实现缩放。



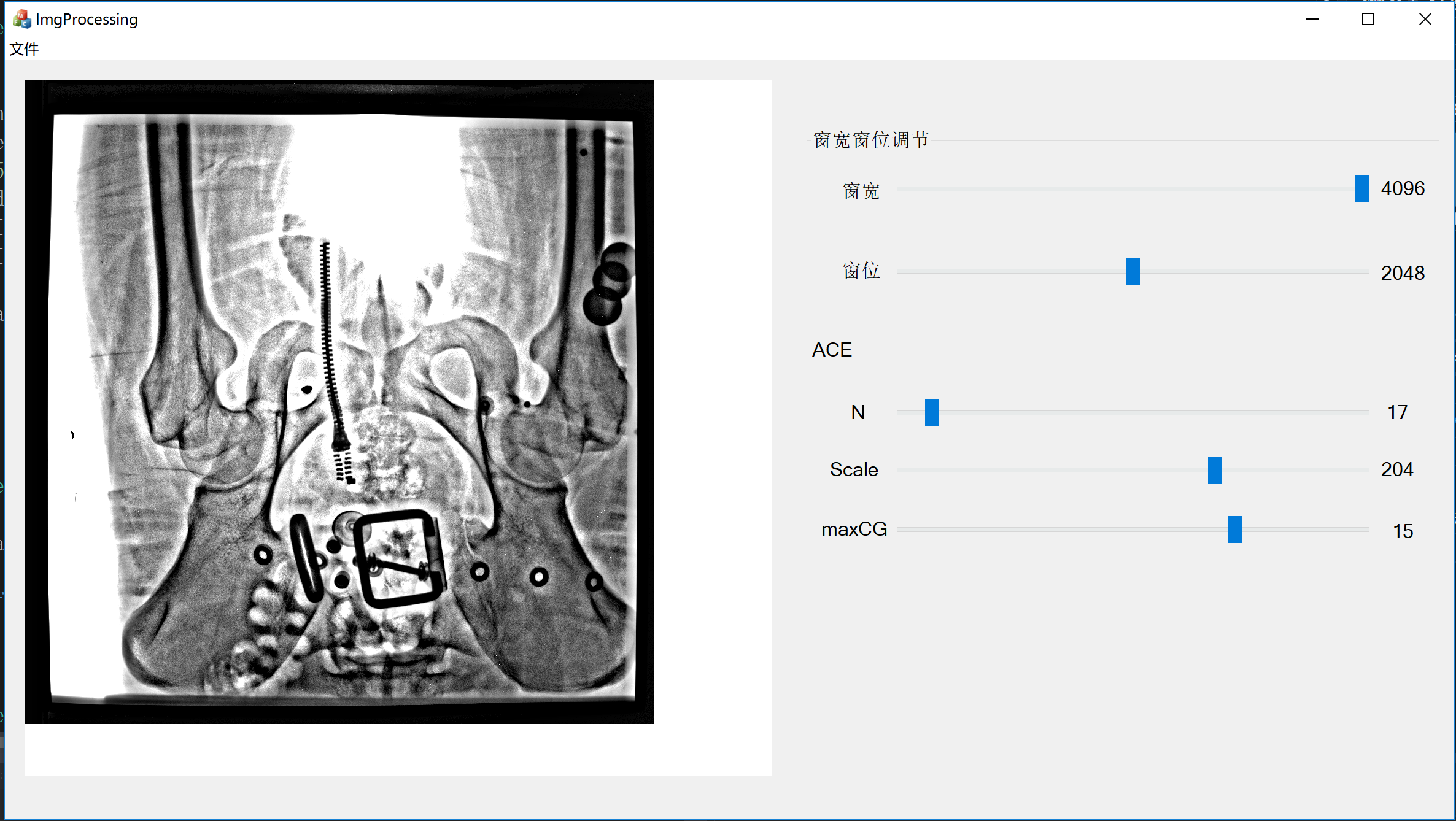
### 3 灰度窗

在窗宽窗位调节中拖动滑动条即可。



### 4 ACE

拖动ACE中那三个滑动条可以调节：



## 六、反思

最大的难度还是在windows编程，对于这方面虽有所训练，但是还远远不够。

很多稳定性的问题没有考虑，例如内存消耗等等，一个小小的实验仔细考虑都还有很多能做的地方，更何况大型的商业稳定软件，开发时要考虑的东西一定数不胜数。

还需要不断学习，更进一步！

## 五、参考资料

1. c++双缓冲技术绘图避免闪烁, ghevinn, http://blog.csdn.net/ghevinn/article/details/36390435.

2. 窗宽窗位与其处理方法，松子茶，http://blog.csdn.net/songzitea/article/details/8505469.

3. 窗位和窗宽，cfqcfqcfqcfqcfq，http://blog.csdn.net/cfqcfqcfqcfqcfq/article/details/51227173.

4. BMP图四字节对齐的问题，http://blog.csdn.net/jkhere/article/details/8155017.

5. MFC GDI+实现以鼠标为中心缩放图片（并且可以拖动），greatchina01，http://blog.csdn.net/greatchina01/article/details/764291056.