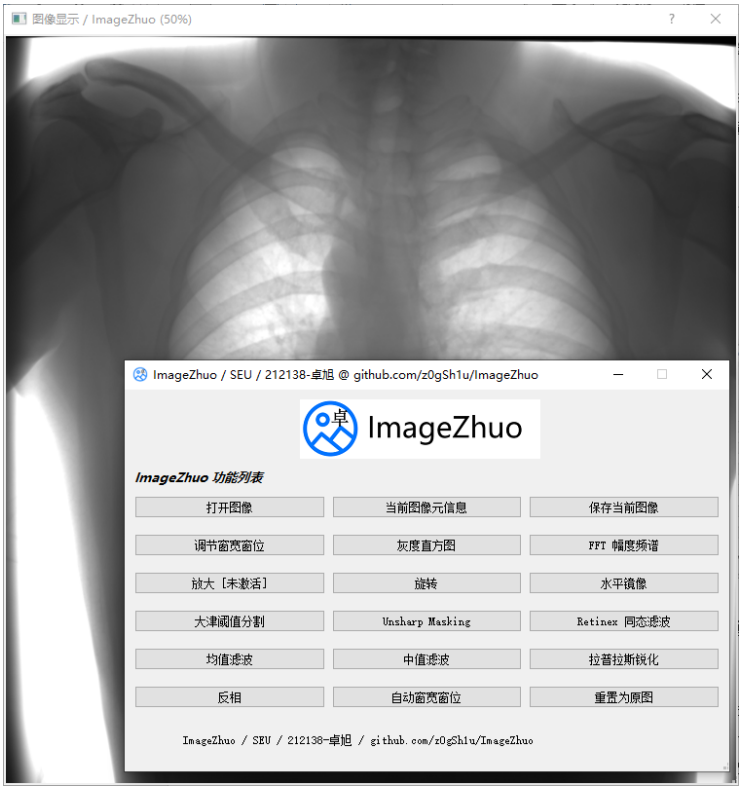
医学 DR 图像基本阅片软件 ImageZhuo 设计报告

——《数字图像处理基础》课程实验 东南大学 212138-卓旭

一、 软件用途

ImageZhuo 是一款医学 DR 图像基本阅片软件。其主要用途是帮助医生与学者完成对 4096 级灰度的 DR 图像的阅览，并提供图像增强、图像参数查看、窗宽窗位调整等基础功能。

ImageZhuo 的界面一瞥如下图所示：

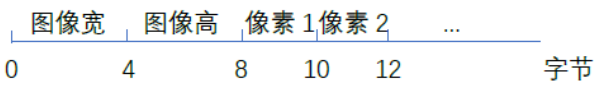


二、 需求分析

ImageZhuo 使用 Python 语言进行开发，以快速粘合各类算法，并使用 PyQt 库完成 GUI 界面的开发，实现图形界面的需求。

ImageZhuo 设计的基本需求有如下五点：

1. 读入按给定的格式保存的图像数据文件。此需求需要程序以二进制方式读入 raw 格式图像文件，并按照数据存放约定进行宽高和像素的解析。给定的数据存放格式约定如下图所示。字节序为小端序；像素数值格式为 16 位无符号整数，高 4 位无效；宽高格式为 32 位无符号整数；像素排列顺序为行扫描。本文后续将该种图像格式称为“鲍老师格式”：



2. 灰度窗映射。此需求需要对 4096 级灰度图像进行分段线性映射，将在窗口范围内的灰度线性缩放到 256 级灰度范围进行显示。具体实现细节将在第四节介绍。

3. 图像局部放大。此需求需要程序对用户框定的感兴趣区域进行等比放大，放大目标尺寸为原图像尺寸。在放大时，需要进行合适的插值。具体实现细节将在第四节介绍。

4. 图像细节增强。此需求要求在对图像的细节进行增强的同时，不明显放大噪声。结合阅读的是医学 DR 图像的先验知识，ImageZhuo 采用同态滤波（Retinex）方法实现本需求。具体实现细节和算法选择理由将在第四节和第六节介绍。

5. 灰度图像显示。在读入图像且调整好灰度窗后，即可显示 4096 级灰度图像。

ImageZhuo 在设计的基本需求之外，还添加了诸如灰度直方图显示、图像元信息查看、FFT 幅度频谱、大津阈值分割等一系列实用模块，以增强软件的功能。具体细节将在第四节介绍。

三、 总体设计

ImageZhuo 的代码架构采用分模块式设计，如下表所示：

模块名	模块功能
reader	读取器模块。用于对各类约定格式的图像的二进制读取。
writer	写入器模块。负责将经过一系列处理后的图像保存为各类约定格式的图像。
ui	用户界面。包含了所有图形界面的设计与绘制，以及对各项功能的调用逻辑。
function	功能模块。包含了一系列图像处理算法，供程序调用组合。算法均与 UI 和图像格式约定解耦，可移植到任意图像矩阵。
misc	杂项模块。主要包括工具方法和自定义图像包装类 MyImage。

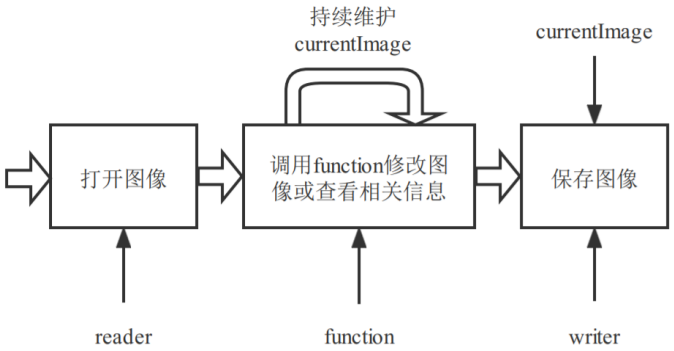
当用户打开图像后，将调用读取器读取图像，构造读取器实例 `reader: T extends _BaseReader`，同时初始化挂在在主窗口 mainWindow 的全局当前图像对象 `currentImage: MyImage`。在用户进一步调用各种功能对图像进行处理时，程序将不断修改 `currentImage`。即 `currentImage` 始终是目前展示的图像，而 `reader` 内保存的图像始终为打开的原始图像。

程序将持续维护 `currentImage` 对象的下表内的这些属性：

属性	说明
h	图像高度（int）
w	图像宽度（int）
data	图像数据（numpy.ndarray）
ww	窗宽（int）
wl	窗位（int）
dtype	图像数据数据类型（np.uint16, np.uint8, float 等）
data8bit	按 ww 和 wl 加窗后的 256 级灰度图像（numpy.ndarray）
PILImg8bit	按 ww 和 wl 加窗后的 256 级灰度图像（PIL.Image）

图像显示窗口 `ImageDisplay` 显示的图像始终是 256 级灰度图像，依据 `PILImg8bit` 进行显示。由于进行图像处理后 data 会发生改变，故可调用 `MyImage.reGen8bit()`函数重新计算在当前窗设定下的 256 级灰度图像，以供显示窗口显示。

综上所述，用户使用 ImageZhuo 的典型流程如下图所示。当用户没有打开图像，或者关闭了图像显示窗口后，使用功能时，程序会检查出 currentImage 为 None，从而报错，阻断后续流程。



在这样的架构设计下，调用不同功能模块对当前图像进行处理的代码都类似于如下结构（以图像水平翻转为例）：

```
def on_btn_flip_clicked(self):
    global currentImage
    ensureCurrentOpen() # 检查当前有图像打开
    flipRes = horizontalFlip(currentImage.data) # 调用相关功能模块
    currentImage.data = flipRes # 替换结果图像
    currentImage.reGen8bit() # 重新生成加窗后图像
    imageDisplay.loadFromMyImage(currentImage) # 刷新图像显示窗口
```

四、 模块设计

读取器模块（reader.*）

为支持多种不同类型约定格式的图像文件的读取，可编写多种继承了基类_BaseReader 的读取器模块。要求它们填充 h（高）、w（宽）、data（图像数据）、path（路径）、filename（文件名）这些属性，并在_MetaInfo.py 中登记注册，即可接入 ImageZhuo 工作。

为“鲍老师格式”编写的读取器在 BaoReader.py。首先读入 4 个字节，按小端方式（低位在低地址）解析为无符号整数，即为图像的宽；同理读入 4 个字节，为图像的高。然后反复每次读入 2 个字节，按小端方式解析为无符号整数，逻辑与上 0x0FFF 抹去高 4 位的无效数据，即为像素值。按照行扫描格式填充 data 属性，即完成了图像的读取，构造好了 reader 对象。核心代码如下：

```
while True:
    pixel = f.read(2)
    if pixel == b'':
        break
    pixel = np.uint16(bytesToInt(pixel))
    # 抹去高 4bit 的杂数据
    pixel = pixel & 0x0fff
    self.data.append(pixel)
self.data = np.array(self.data, dtype=np.uint16).reshape((self.h, self.w))
```

读取器模块是由打开窗口 openDialog 调用的。在完成 reader 对象组装后，窗口会通过信号，将 reader 托管给 mainWindow。mainWindow 将依据 reader 填充 currentImage，并将图像显示到显示窗口 imageDisplay，根据图像的宽高调整显示窗口以适合图像的显示（过大的图像会被适当缩小后展示）。至此，图像的打开和显示完成。openDialog 示意图如下：

图像元信息查看模块（function.metadata）

本模块接收 reader 和 currentImage，读取相关属性，告知用户当前图像的文件名、宽、高、像素格式、窗宽、窗位、像素最大值、像素最小值、像素均值、像素方差。这些元信息能够帮助用户更好地了解图像情况。示意图如下，示例图像为 lung.raw：

窗宽窗位调节模块（function.window）

本模块将根据设定的窗宽（WW）和窗位（WL），对 currentImage.data 进行分段线性映射，将窗口内的灰度映射到 min~max（默认为 0~255，256 级灰度）。映射关系式如下：

$$x' = \begin{cases} \max, & x > r \\ \dfrac{x-l}{r-l}(\max - \min) + \min, & l \le x \le r \\ \min, & x < l \end{cases}$$

其中$l = WL - WW/2, r = WL + WW/2$。

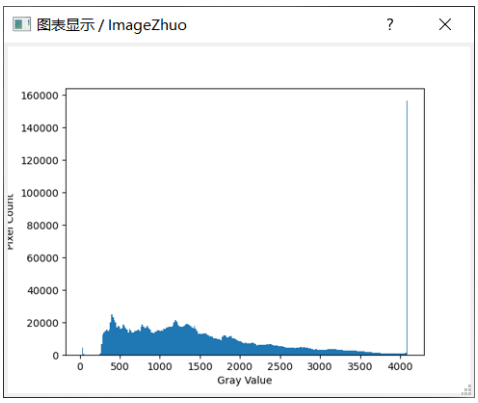
对窗宽窗位进行调节，有助于充分利用图像的动态范围，优化显示效果，便于医生观察感兴趣灰度范围。窗宽窗位调节模块的示意图见软件测试章节。

另外需要说明的是，对于新打开的图像，默认的窗宽设定是图像的最大灰度值与最小灰度值的差，窗位设定是窗宽的二分之一处。也就是说，将在全灰度级上进行到 256 级的灰度映射。

灰度直方图模块（function.hist）

灰度直方图模块能够统计 currentImage.data 的各灰度及其出现次数，从而使用户更好地掌握图像的灰度分布信息，尤其是能够指导窗宽窗位的调节。

本模块使用 matplotlib.pyplot 的 hist 功能绘制灰度直方图，将其转换为 PIL.Image 后，显示到图标显示窗口 figureDisplay。示意图如下，示例图像为 lung.raw：



FFT 幅度频谱模块（function.fft）

声明：本模块代码修改自本人本科的《数字信号处理》（舒华忠）课程实验的代码，学号 09017227。实验报告可见 https://www.cnblogs.com/zxuuu/p/12425321.html 。

当实现了一维 FFT 算法时，借助二维离散傅里叶变换的可分离性：

$$F(u,v)=\sum_{x=0}^{N-1}W_N^{xu}\sum_{y=0}^{N-1}f(x,y)W_N^{yv}$$

可将二维 FFT 分解为沿行方向进行一次一维 FFT，再对结果在列方向进行一次一维 FFT，即为最终的二维 FFT 结果。将复的结果取模$\sqrt{(\mathrm{Re})^2+(\mathrm{Im})^2}$，即构成幅度谱。

另有一些细节问题需要说明：

在进行图像的二维 FFT 前，通常将数据归一化到 [0,1] 区间进行处理。相关公式为 $f'=(f-f_{\min})/(f_{\max}-f_{\min})$。

在绘制图像的幅度频谱时，习惯将低频分量移动到频谱图中心而非四角（fftshift）。此要求可借助 DFT 的频域循环移位性质，来调整频域原点。具体做法等价于使用$(-1)^{x+y}f(x,y)$替代原函数进行傅里叶变换。

在绘制图像的幅度频谱时，通常进行对数变换以优化显示效果。即使用$\ln F$的值作为幅度频谱图的像素值。

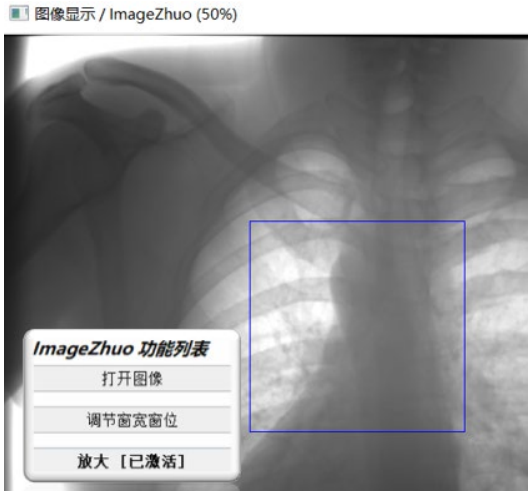
在完成上述细节处理后，将幅度频谱图拉伸回 256 级灰度，交由 figureDisplay 显示即可。FFT 幅度频谱模块的示意图见软件测试章节。

局部放大模块（function.zoom）

声明：本模块代码综合修改自本人研究生的《数字图像处理基础》（鲍旭东）课程实验 1：图像的旋转和缩放；以及本人本科的《数字图像处理》（鲍旭东）课程实验，学号 09017227。

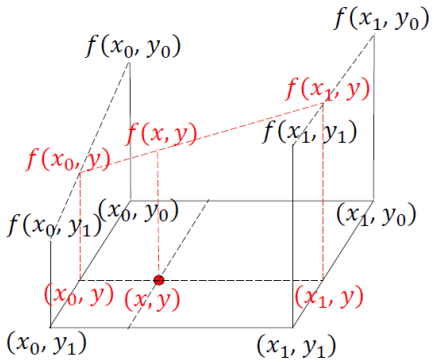
ImageZhuo 的图像局部放大操作采用如下逻辑：用户点击“放大 [未激活]”按钮后，可进入放大激活模式，然后在图像显示窗口中框定需要放大的区域，即可完成放大。放大

区域的长宽比和放大的目标尺寸都与 currentImage 一致，而倍率取决于框定区域的大小。也就是说，进行的是框定的感兴趣区域的放大，而不是整张图像的放大。这样运算的复杂度较低，实现比较简单。操作示意图如下：



对于这类图像到图像型的变换，做法是进行“反向映射”，即遍历目标图像的像素，计算与其相关的原始图像的像素位置，对这些像素进行有关处理以计算目标图像的像素值。这样可以确保目标图像的每个像素位置都有值。

在对图像进行放大的过程中，需要进行插值。ImageZhuo 采用双线性插值方案，有关公式如下：



$$f(P) = uvf(Q_{22}) + (1-u)vf(Q_{12}) + u(1-v)f(Q_{21}) + (1-u)(1-v)f(Q_{11})$$

其中f为图像；P为被插值点(x, y)；$Q_{11} = (x_0, y_0)$、$Q_{12} = (x_0, y_1)$、$Q_{21} = (x_1, y_0)$、$Q_{22} = (x_1, y_1)$为四个参考点；$x_1 = x_0 + 1$，$y_1 = y_0 + 1$；u、v表示被插值点超出参考点Q_{11}的坐标的小数部分。

旋转模块（function.rotate）

声明：本模块代码综合修改自本人研究生的《数字图像处理基础》（鲍旭东）课程实验1：图像的旋转和缩放。

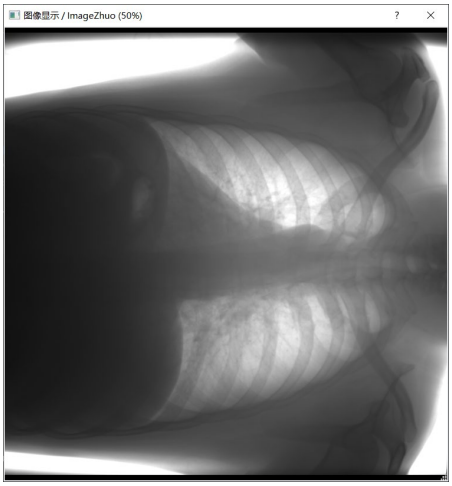
图像的旋转和图像的局部放大类似，也采用“反向映射”。图像旋转的映射关系式如下：

$$r' = (-h/2 + r)\cos\theta + (w/2 - c)\sin\theta + h/2$$

$$c' = (-h/2 + r)\sin\theta + (-w/2 + c)\cos\theta + w/2$$

其中θ为旋转角度（顺时针为正）；h、w为图像的高、宽；r为行坐标，c为列坐标，有撇号上标的表示目标图像上的坐标。

作为医学 DR 图像阅片软件，ImageZhuo 只提供常用的 90°、180°、270°三种旋转度数选项。对 lung.raw 旋转 90°的示意图如下：



需要说明的是，对长宽不等的图像的旋转，目标图像的长宽与原始图像不一定相等。ImageZhuo 的处理方式是，规定目标图像的长宽等于原始图像，对于溢出的部分裁剪，对于不足的部分取 0。

水平镜像模块（function.flip）

图像的水平镜像翻转依然采用“反向映射”。图像水平翻转的映射关系式如下：

$$r' = r$$

$$c' = w - c - 1$$

符号体系与图像的旋转模块一致。对 lung.raw 的水平翻转结果示意图如下（左边为翻转前，右边为翻转后）：

大津阈值分割模块（function.otsuSegmentation）

声明：本模块代码修改自本人本科的《数字图像处理》（鲍旭东）课程实验，学号 09017227。

大津阈值分割法是一种常用的阈值自适应型二分类分割算法。ImageZhuo 在进行大津阈值分割的过程中，是在 currentImage 经过加窗的 256 级灰度的图像（而非原始图像）上

进行阈值确定和分割的。这样能够降低运算的复杂度，并且考虑了针对不同欲分割部位，先加窗进行强调后再做分割的做法，更有道理。

大津阈值分割算法流程如下：

对于 256 灰度级图，从 0 到 255 遍历各灰度级作为当前阈值。称灰度值大于阈值的像素为前景，反之为背景。对于每一个阈值，计算：

前景比例w_0 = 前景像素数 ÷ 总像素数

背景比例w_1 = 背景像素数 ÷ 总像素数

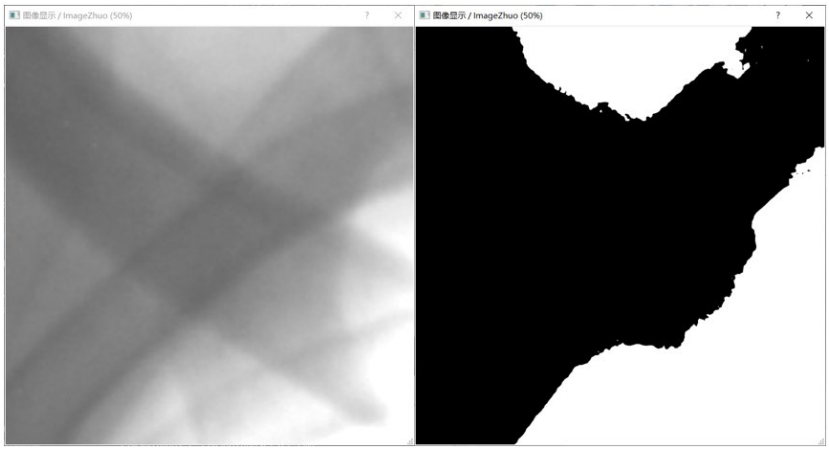
前景平均灰度μ_0 = 前景像素总灰度值 ÷ 前景像素数

背景平均灰度μ_1 = 背景像素总灰度值 ÷ 背景像素数

类间方差$Var = w_0 w_1 (\mu_1 - \mu_0)^2$（该公式经化简）

最后，能够获得最大类间方差的阈值，就是最佳分割阈值。

对于大于阈值的像素，认为属于前景，赋白色（255）；否则认为是背景，赋黑色（0）。对于 lung.raw 的一个部分（约位于左上角），分割示意图如下（左侧为分割前，窗设定：WW/WL=1350/1500；右侧为分割后，大津阈值=169）：



Unsharp Masking 模块（function.unsharpMasking）

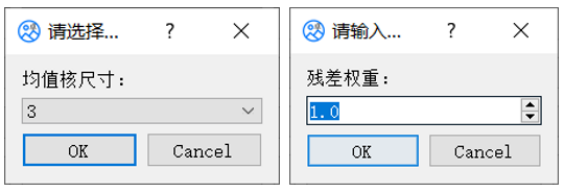
Unsharp Masking 的算法流程如下：

使用模糊核 *k* 对图像 *I* 进行模糊，得到 *blur*

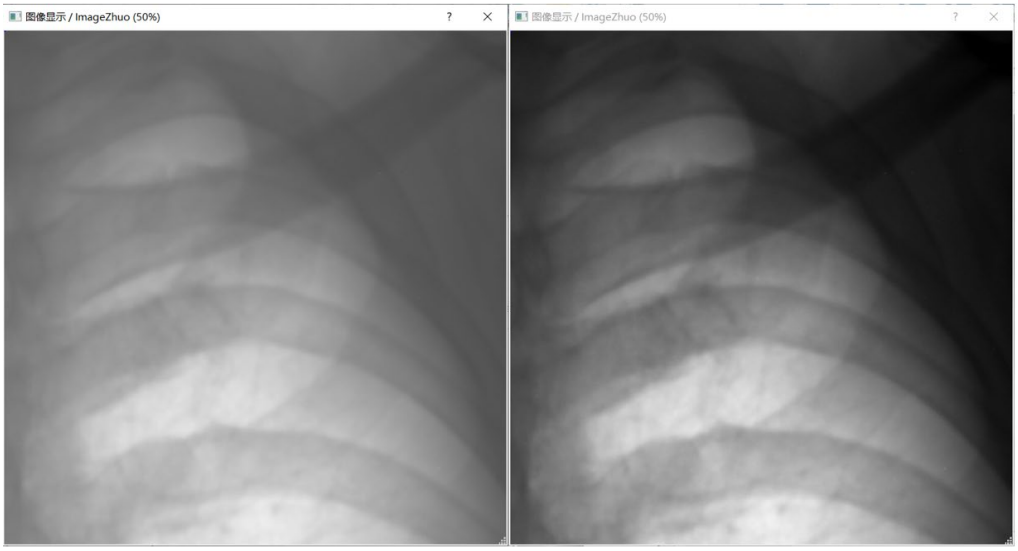
获得锐化掩膜 *mask* = *I* − *blur*

Unsharp Masking 结果 *res* = *I* + *k*×*mask*，*k* >= 1 为权重参数，用于控制锐化程度

ImageZhuo 进行该算法使用的模糊核是均值滤波器。在用户调用该功能进行处理时，会询问用户使用的均值滤波器的尺寸，以及权重参数 *k*。如下图所示：



一组对 lung.raw 的处理样例图如下所示，左侧为处理前的图像（WW/WL 为默认值），右侧为处理后的图像（WW/WL=2800/1500），使用的参数是核尺寸=5，*k*=1.2。



Retinex 同态滤波模块（function.retinex）

声明：本模块代码修改自本人本科的《数字图像处理》（鲍旭东）课程实验，学号 09017227。

Retinex（同态滤波）相当于对数域的 Unsharp Masking，是一种著名的对比度增强算法。其算法流程描述如下：

Retinex 理论认为，观测到的图像 I 是外界照度 L 与物体反射 R 综合作用的结果。数学表达式为$I(x,y)=L(x,y)R(x,y)$。我们的目标是尽量消除外界照度 L 的影响，尽量强调 R，即来自物体本身的信息。同态滤波使用下面的方法来处理：

积性得出的 $I(x,y)$不利于我们对 L 和 R 分别处理，故左右取对数转入对数域：

$$\log I(x,y)=\log L(x,y)+\log R(x,y)$$

左右作傅里叶变换（DFT）：

$$F_I(x,y)=F_L(x,y)+F_R(x,y)$$

接下来，设计一个滤波器 H，它能起到增益反射 R，减益照度 L 的效果。由于照度 L 在图像上变化很小且缓慢，故可以认为 L 的能量主要集中在低频部分。问题转化为设计一个滤波器 H，它增益高频，减益低频。

把这个滤波器作用在$\log I(x,y)$上，可得：

$$F_IH=F_LH-F_RH$$

这便是削减 L，强调 R 的结果。接下来，做一系列反变换即可恢复成图像（空域）：

$$\log I'(x,y)=\mathrm{IDFT}(F_IH)$$

$$I'(x,y)=\exp\log I'(x,y)$$

最后把I'的灰度值重新归到原先的动态范围上，即完成了对比度增强操作。

ImageZhuo 采用 Retinex 方法实现要求的细节增强需求，算法是在 4096 级灰度（原始数据）上操作的，结果仍保持 4096 级灰度范围。由于自行实现的二维 FFT 的效率在没有进行底层 C/C++实现、多线程并发、体系结构优化的情况下，效率仍然不高。所以为保证运算效率，Retinex 实现时调用了 numpy 的 FFT，而非自己实现的 FFT。

ImageZhuo 在进行同态滤波时设计的滤波器是一种调整过的高通的高斯滤波器。对于该滤波器的相关参数，提供了一组默认参数，并询问用户进一步修改，如下图所示：

关于该算法的更多细节，如算法选择的原因、滤波器的设计、参数的选择、效果的讨论等等，请见本报告的第六节：图像增强处理算法和结果分析。

均值滤波和中值滤波模块（function.smooth）

均值滤波和中值滤波是两种常用的基础的图像平滑方法。二者均使用一个 $n\times n$（n 为奇数）的滤波窗口在图像上滑动，将中心点像素按窗口内的像素进行一定策略计算后进行替换。前者使用窗口像素的平均值：lambda x: sum(x) / len(x)；后者使用窗口像素的中位数：lambda x: sorted(x)[(len(x) + 1) // 2 - 1]。

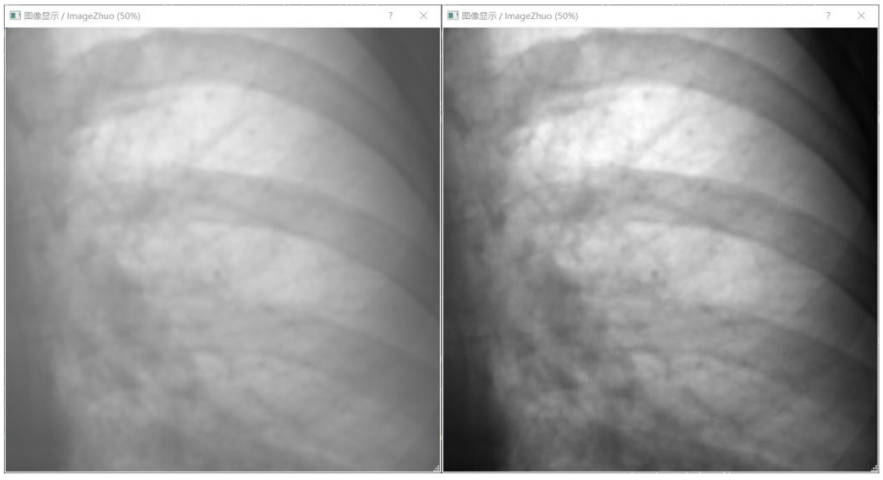
对于四周一圈像素，由于无法以它们为中心放下滤波窗口，ImageZhuo 采用的策略是放弃对这一圈像素的处理，保留原值。中值滤波和均值滤波的处理结果和讨论请见第五节的“中值滤波与均值滤波正确性验证”部分。

拉普拉斯锐化模块（function.sharpen）

拉普拉斯锐化的流程是，首先利用拉普拉斯算子对原图像进行滤波（类似于均值滤波和中值滤波的滑动窗口处理方式），得到梯度细节。然后将差分细节图像加到原始图像作为结果图像，即完成了锐化操作。根据差分阶数的不同，常用的有一阶拉普拉斯算子（左）和二阶拉普拉斯算子（右）：

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

ImageZhuo 在进行拉普拉斯锐化时，对四周一圈像素放弃处理。对于使用的拉普拉斯算子阶数，会询问用户。处理结果示意图如下，使用二阶拉普拉斯算子锐化 lung.raw 的一部分，左侧为处理前（默认窗宽窗位），右侧为处理后（WW/WL=2500/1250）：



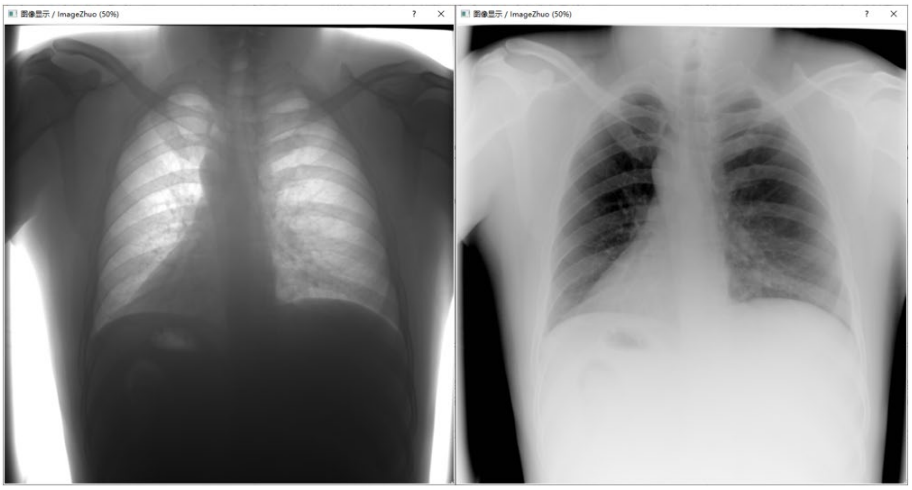
反相模块（function.reverse）

在进行 DR 阅片时，反相是很常用的功能。ImageZhuo 实现反相是按照如下公式替换原图的灰度值：

$$v' = \max - (v - \min)$$

其中v为原灰度值，min为原图像动态范围的最小灰度值，max为原图像动态范围的最大灰度值。一种工程上更常用的做法是在显示时使用反相的颜色查找表（LUT），避免对像素灰度值的修改。

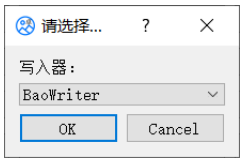
反相功能的示意图如下，以 lung.raw 为例，左侧为原始图像，右侧为反相后图像：



写入器模块（writer.*）

与读取器模块类似的，为支持多种不同类型约定格式的图像文件的保存（写入），可编写多种继承了基类_BaseWriter 的读取器模块，然后注册到_MetaInfo.py。其接收一个 MyImage 对象，暴露一个 save(path: str)方法用于被调用以将图像保存到 path 位置。

为“鲍老师格式”编写的写入器在 BaoWriter.py。首先按小端方式写入 4 个字节，为图像的宽；继续同理写入 4 个字节，为图像的高。然后将 MyImage.data 按顺序以 2 个字节为一个像素，按小段方式写入文件。最终，即获得了按照“鲍老师格式”保存的结果图像。保存图像时询问用户选择写入器的示意图如下：

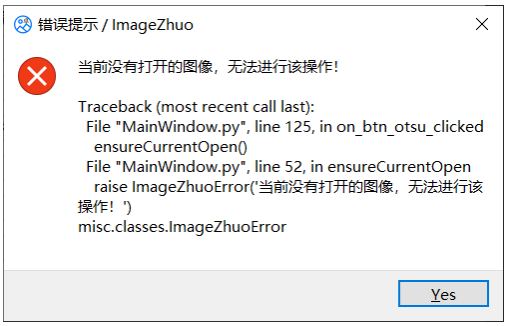


五、 软件测试

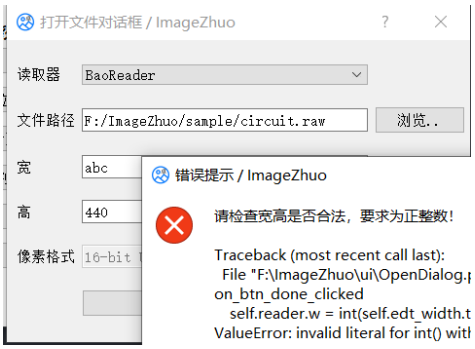
对异常情况的处理

ImageZhuo 自定义了异常类 misc.classes.ImageZhuoError，对于可预期的错误，均抛出该类型异常，进行警示与恢复，避免程序崩溃。对于不可预期的其他错误，进行警示，尽量避免程序崩溃。对于主要处理的几种可预期异常，经测试均正常处理，举例部分示意图如下：

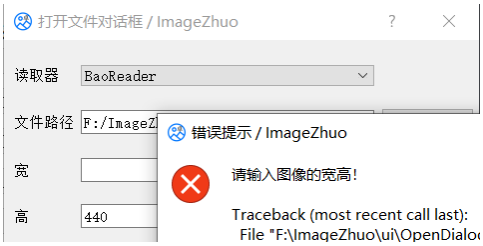
未打开图像就调用功能：



非法宽高：

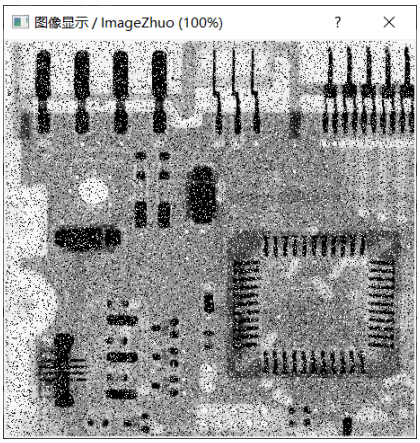


缺失宽高：

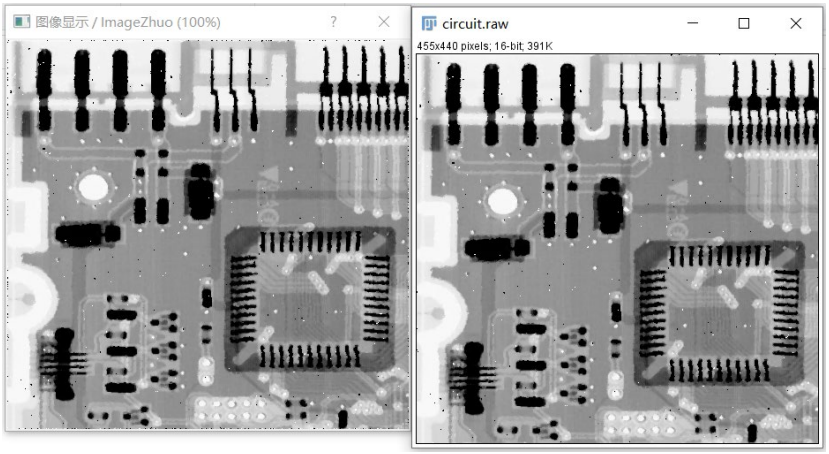


中值滤波与均值滤波正确性验证

本测试环节采用与参考软件 ImageJ 的对拍方法进行。ImageJ 是一款知名的常用的科研图像查看软件。对于这两种平滑方法，本环节使用笔者自行修改的一张符合“鲍老师格式”的著名的“椒盐噪声电路板”图像（sample/circuit.raw）作为测试图像。ImageJ 打开图像时，将 Offset to first image 设为 8，以跳过“鲍老师格式”头部的宽高部分。原始图像如下所示：

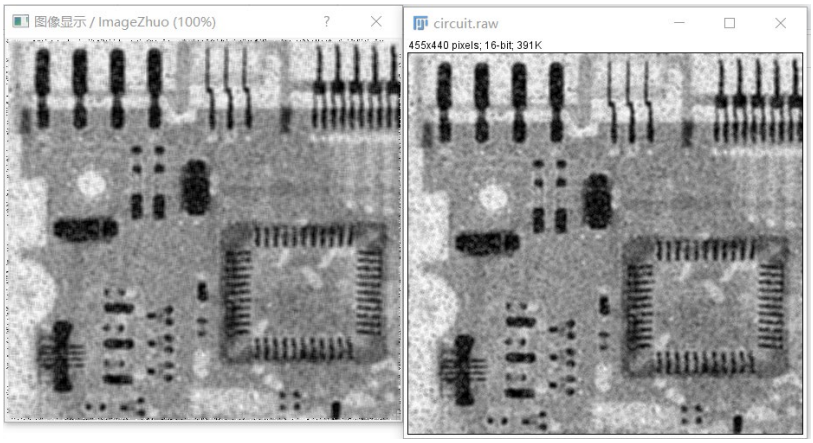


使用尺寸为 3×3 的中值滤波器处理，结果图如下。左侧为 ImageZhuo 的显示，右侧为 ImageJ 的显示：



去噪处理结果好，符合预期。观察残余噪声点位置，两张图像是符合的。由于 ImageZhuo 和 ImageJ 对边缘一圈像素的处理策略不同，故四周一圈的表现不同，亦符合预期。

使用尺寸为 5×5 的均值滤波器处理，结果图如下。左侧为 ImageZhuo 的显示，右侧为 ImageJ 的显示：



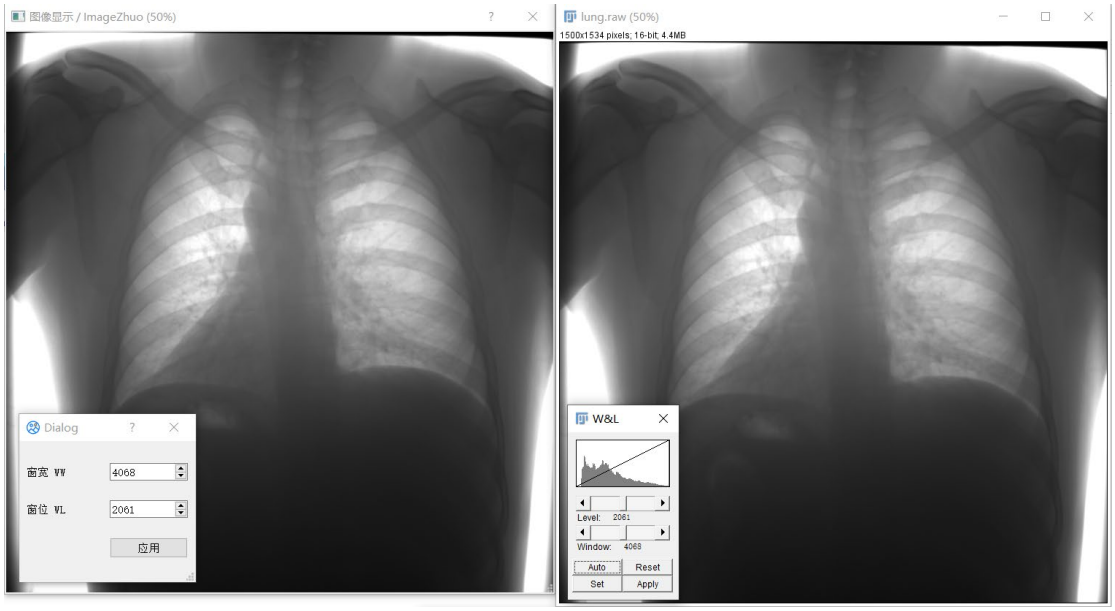
去噪处理结果较差，符合预期。观察两张图像，基本是一致的。由于 ImageZhuo 和 ImageJ 对边缘一圈像素的处理策略不同，故四周一圈的表现不同，亦符合预期。

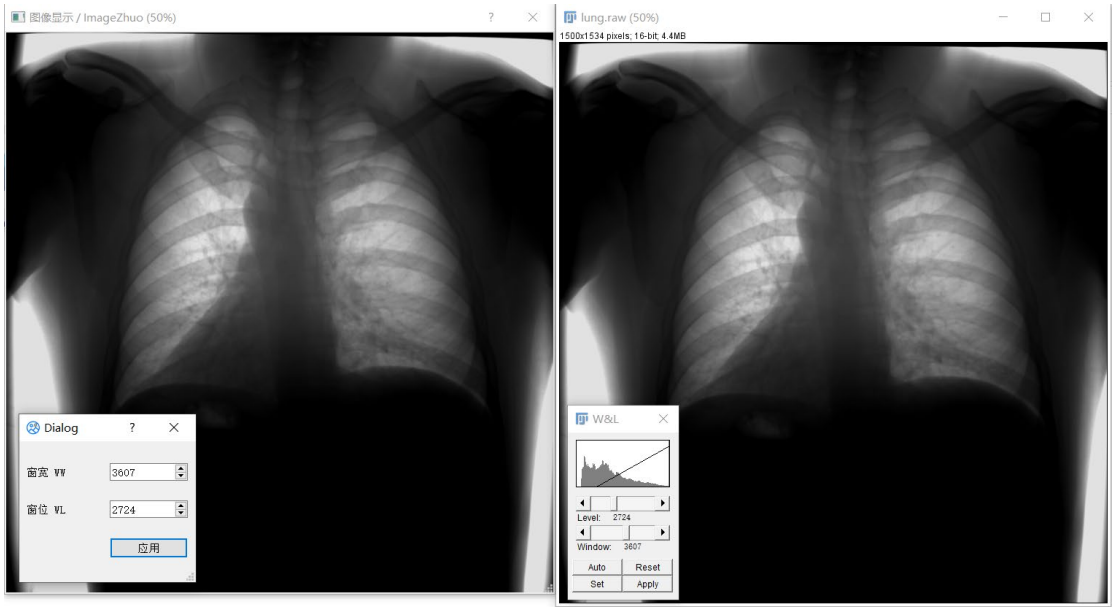
综上所述，ImageZhuo 的中值滤波与均值滤波功能的正确性得到了验证。

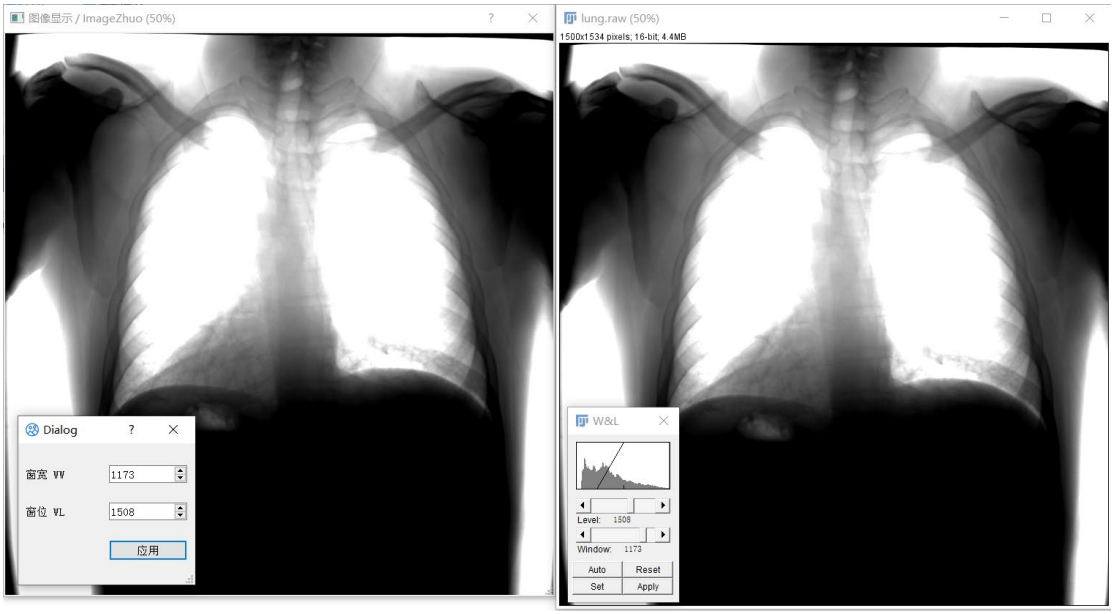
窗宽窗位调节正确性验证

本测试环节继续采用与参考软件 ImageJ 的对拍方法进行。使用 ImageJ 打开 lung.raw 图像，调节几组窗宽窗位，然后使用 ImageZhuo 调节到同样的窗宽窗位，对显示效果进行比较，即可验证窗宽窗位调节的正确性。

以下为三组实验结果图，左侧为 ImageZhuo 的设置和显示，右侧为 ImageJ 的设置和显示：


图像显示 / ImageZhuo (50%)
lung.raw (50%)
1500x1534 pixels; 16-bit; 4.4MB
Dialog
窗宽 WW
4068
窗位 WL
2061
应用
W&L
Level: 2061
Window: 4068
Auto
Reset
Set
Apply

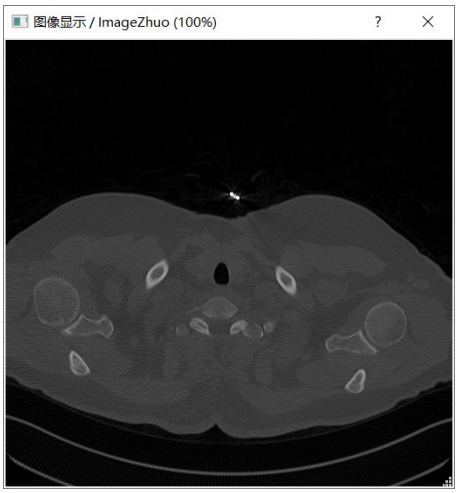

图像显示 / ImageZhuo (50%)
lung.raw (50%)
1500x1534 pixels; 16-bit; 4.4MB
Dialog
窗宽 WW
3607
窗位 WL
2724
应用
W&L
Level: 2724
Window: 3607
Auto
Reset
Set
Apply



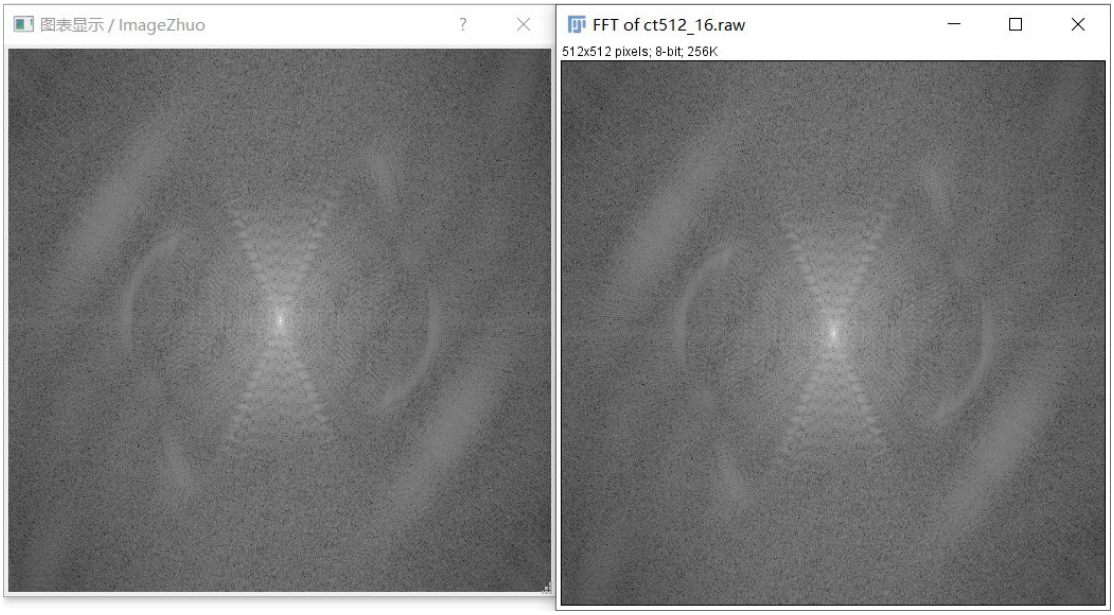
经实验，ImageZhuo 与 ImageJ 的窗宽窗位调节功能在一致的参数下能够获得相同的结果，ImageZhuo 的窗宽窗位调节模块正确性得到验证。

FFT 幅度频谱绘制正确性验证

由于自行实现的二维 FFT 的效率在没有进行底层 C/C++实现、多线程并发、体系结构优化的情况下，效率仍然不高。为此，本环节使用笔者自行修改的一张符合“鲍老师格式”的较小尺寸（512×512）的 CT 扫描图像（sample/ct512_16.raw）作为测试图像。原图是来源为 AAPM Low Dose 数据集（https://www.aapm.org/grandchallenge/lowdosect/ ）的 CT DICOM 图像。图像如下所示：



仍然采用与窗宽窗位调节正确性验证相同的与 ImageJ 对拍的测试方式，得到幅度频谱绘制结果如下，左侧为 ImageZhuo 的显示，右侧为 ImageJ 的显示：

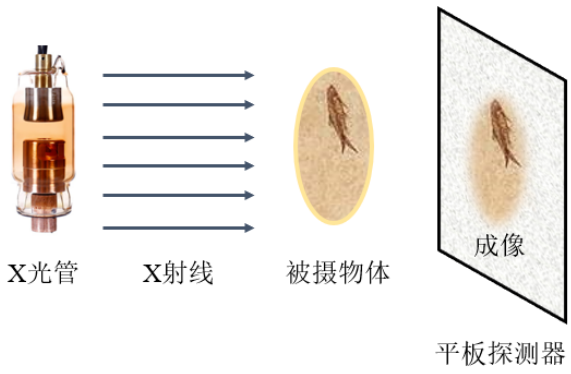


经实验，ImageZhuo 与 ImageJ 的 FFT 幅度频谱绘制功能能够获得相同的结果，Image Zhuo 的 FFT 幅度频谱绘制模块正确性得到验证。

六、 图像增强处理算法和结果分析

本节将细致讨论 ImageZhuo 为何选择 Retinex 算法实现图像细节增强需求，以及相关实现细节与参数设定，最后进行结果分析。正如第四节对 Retinex 算法模块进行的相关介绍，其能够尽量消除平滑变化的外界照度的影响，强调来自物体本身的细节信息。进一步，结合主要看图对象为 DR 影片的先验知识，Retinex 算法是合适的细节增强算法。

考虑 DR 的成像机理，如下图所示：



将 X 光线视为“外界照度”，被拍摄的人体骨骼、肌肉、组织等视为“光照到的物体”。对于相同的人体部位，对 X 线的吸收能力基本一致，且即使有变化，在空间上也是平滑变化的，这符合“照度在图像上变化很小且缓慢”的先验要求。而吸收能力发生急剧变化的位置，是不同人体部位对的交界交替处，这也正是我们需要增强的图像细节，符合“来自物体本身的信息主要集中在高频”的先验要求。因此，Retinex 算法适合对 DR 图像进行图像增强。

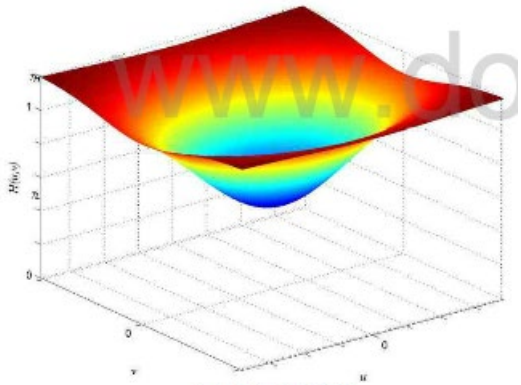
经过资料搜索，如下文献也使用 Retinex（同态滤波）方法对 X 线成像进行增强，这进一步证明了该算法选择的正确性和有效性：

[1]王彦臣,李树杰,黄廉卿.基于多尺度 Retinex 的数字 X 光图像增强方法研究[J].光学精密工程,2006(01):70-76.
[2]王攀峰,赵书俊,王双玲,马长征,李胜春,孙亮.一种基于 Retinex 原理的 DR 图像增强改进算法[J].中国体视学与图像分析,2020,25(01):57-64.

在进行 Retinex 算法处理时，设计的滤波器 H 的性能对处理结果好坏很重要。经查阅资料，一种调节过的高斯高通滤波器是常用的选择（据《数字图像处理》（冈萨雷斯）和部分中文文献）。其频域表达式是：

$$H(u,v) = (\gamma_H - \gamma_L)(1 - \exp[-c \cdot \left(\frac{D(u,v)}{D_0}\right)^2]) + \gamma_L$$

γ_H表示对高频的增益系数（大于等于 1），γ_L表示对低频的减益系数（小于 1），c是控制该函数坡度的参数，$D(u,v)$表示点(u,v)到滤波器中心的距离，D_0是截止频率。该滤波器的三维图示如下所示（摘自网络）：

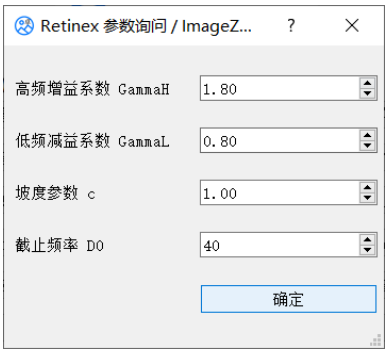

三维网格图

ImageZhuo 在进行 Retinex 处理时采用该滤波器。该滤波器有四个参数需要设定。

自适应的参数选择是困难的，因为不同用户所期待的增强程度不同，而且所打开的图像拍摄部位的不同也会带来增强需求的不同。因此，ImageZhuo 首先预置了一组经人工试验，在所给的三张图像（lumbar.raw、lung.raw、vertebra.raw）上都有相对较好的表现的参数，作为默认参数：

$$\gamma_H = 1.8, \gamma_L = 0.8, c = 1, D_0 = 40$$

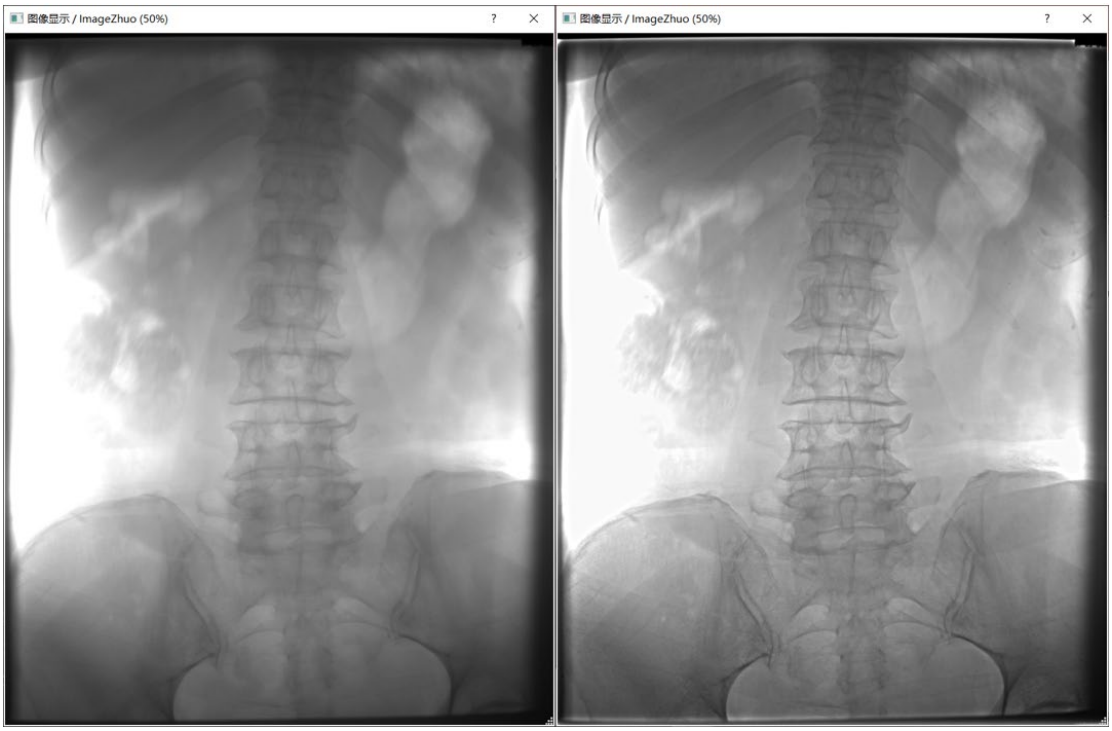
在执行 Retinex 功能时，会进一步询问用户自行对参数进行调优，示意图如下：

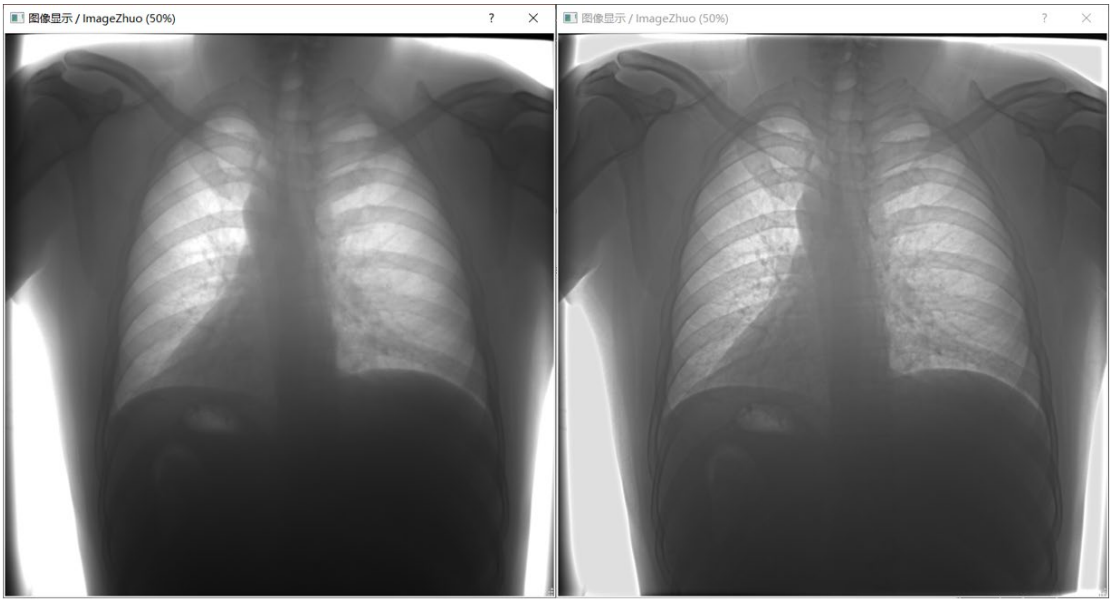


这里，笔者使用默认参数对三幅图片进行细节增强处理，处理结果前后对比图展示于下。需要说明的是，在执行 Retinex 算法后，由于低频被减弱，图像的整体灰度值会向下跌落。因此需要对窗宽窗位进行调节后再对比观察，才能达到公平。窗宽窗位的调节整体原则是依

据灰度分布直方图，尽量囊括大部灰度，削去异常值。

下面从上至下分别为 lumbar.raw、lung.raw、vertebra.raw 的处理结果前后对比，左侧为原图（WW/WL 分别为默认、默认、1800/800），右侧为处理后图像（WW/WL 分别为 1450/800、600/300、800/300）对于这些图像处理结果的 raw 图，以“鲍老师格式”保存在了打包的 result 文件夹内。

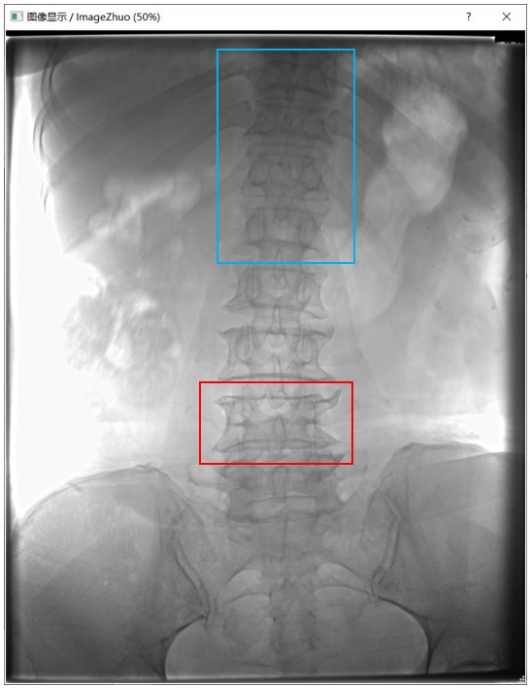




可见，Retinex 方法确实对图像的细节起到了增强的作用。尤其是 lumbar 图像的骨骼形状细节，取得了明显的提升；lung 图像的肺部内细节也获得了较好的增强；vertebra 的骨骼细节也有所增强。

由于在进行 Retinex 后为了观察调整了窗宽窗位，故可能有这样的疑虑：怎么知道细节增强究竟是 Retinex 的效果，还是调节窗口带来的对比度的上升呢？为此，笔者在这里再次补充一个实验。

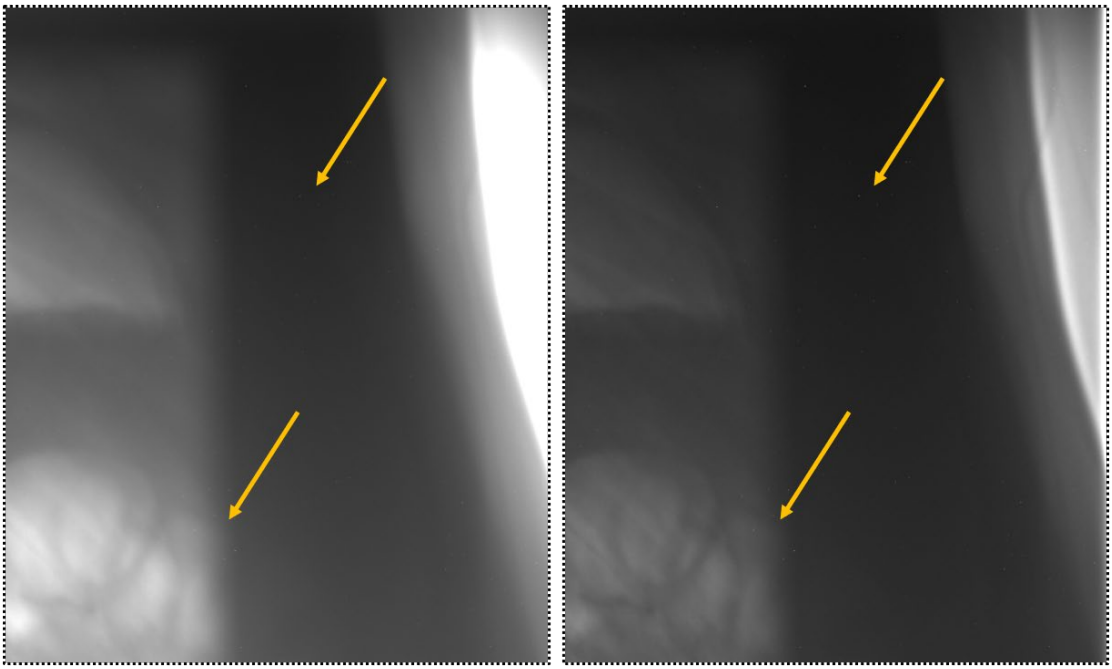
以 lumbar 图像为例，进行 Retinex 滤波并调窗后，图像如下图所示（不含方框）：



以图中框红框的骨节为参照，使用 ImageJ 打开原图像并调节窗宽窗位（不进行其他增强），直到参照骨节的显示效果与上图近似，此时参数为 WW/WL=2773/2450，图像为：

可见，尽管此时的红框对应部分显示效果近似，但诸如蓝框等其他部分的细节完全无法显现，图像整体观感显著不如 ImageZhuo 做 Retinex 处理后的结果图。这说明仅调节窗宽窗位无法实现一样效果的细节增强，验证了 Retinex 算法处理的有效性。

接下来，笔者将讨论 ImageZhuo 进行的 Retinex 处理能否满足“噪声没有明显放大”的处理要求。采用举例观察的方式说理。经检查，vertebra.raw 的右上角有一些量子涨落带来的白亮点噪声。将 Retinex 处理前后的同块区域放在一起比较，示意图如下（左边为处理前图像，右边为处理后图像，黑色虚线边框是为了便于观察添加的）：



观察黄色箭头所指示的两群 shot noise，可见噪声情况保持近似，没有明显增加，符合要求。

最后，对使用的滤波器 H 的四个参数做一些定性的说明，也就是调节参数的整体指导与经验法则：

参数	说明
高频增益系数γ_H	大于 1。越大的值越能够突出细节部分，更强力地增强图像，但也可能带来噪声的放大。
低频减益系数γ_L	小于 1。越小的值越减益照度的影响，也有助于更强力地突出细节，但可能影响图像的慢变的主体信息，或是使得图像整体灰度大幅下降。
坡度参数c	一般控制为 1 即可。用于调节滤波器的 cut-off 速度。更大的坡度会对低频裁剪地更坚决，但可能带来图像中不够平缓自然的过渡。
截止频率D_0	用于调节高-低频的分界准则。更大的截止频率能够删减更多的低频成分，但可能也使得高频信息损失。

七、 参考资料与其他

- Qt Documentation. https://doc.qt.io/
- 其余参考资料已在文内注明。

ImageZhuo 的入口为 ui/MainWindow.py，可使用根目录的 start.cmd 脚本快捷运行。

ImageZhuo 的相关代码预计将在课程作业结束后开源在 https://github.com/z0gSh1u/ImageZhuo 。

212138 卓旭 东南大学

2021-11-15