# 第二次编程作业

黄磊　计 702　2022E8013282156

## A. Introduction

本次实验中，了解并介绍了 python 中的图像处理库 Pillow，使用其处理 16 位.tif 图像。不仅回顾了图像分析和计算机视觉中的一些基本操作，如图像的显示、读写，还有一些直方图计算、区域框选以及图像分割等分析手段。另外，接触并熟悉生物图像分析工具 Icy 的基本操作，并使用其对图像进行分割。

## B. Code Execution

对应 Python 以及使用的各个库版本见文件夹中 requirements.txt，内容如下：
imageio==2.22.0
matplotlib==3.6.2
numpy==1.23.3
Pillow==9.4.0

### Question2

2.1 运行文件 Question2.py ，其中 show_16bit_img() 函数用于将 16 位图像转换为 8 位便于 PIL 可视化，实际上并不保存该 8 位图像；draw_intensity() 函数用于统计该 16 位灰度图的像素值并绘制强度直方图。
2.2 运行文件 ROI_axon01.py ，其中 ROI_Crop() 函数用于剪裁图像。为了便于观察，当给定剪裁范围时候，会在原图中用矩形框标出该范围并给出剪裁后的图片以作对比。对比图也会保存在文件中。之后，保存剪裁图片时候，可以选择是否进行验证，以便检查 16 位图像是否无损保存。

## C. Result Section

### Question 1: Basic concepts of image analysis and computer vision

Pillow 是 python 的第三方图像处理库。由于原有图像处理库 PI L (Python Image Library) 更新缓慢，无法满足 Python3 的需求，因此社区志愿者开发了 Pillow 库作为更新和替换。Pillow 功能更加多样，用于图像的基本处理。

### 1.1 Pillow 所能处理的图像格式

Pillow 几乎能够处理任何格式的图像文件，比如 "jpeg"，"png"，"bmp"，"gif"，"ppm"，"tiff" 等等。

### 1.2 Pillow 对图像的基本处理

首先在 Python 中导入包：

```
from PIL import Image
```

使用 open() 函数读取图像，创建一个图像对象：

```
im = Image.open(fp,mode="r") # fp 为所读取图像的文件路径，包含文件名，例如 "/User/Dc/1.jpg"
```

使用 show() 函数显示图像：

```
im.show()
```

使用 save() 保存图像：

```
im.save(fp, format=None) # fp 为文件路径，包含文件名，例如 "/User/Dc/1.jpg", format 为可选的
```

重写格式，默认为 None。

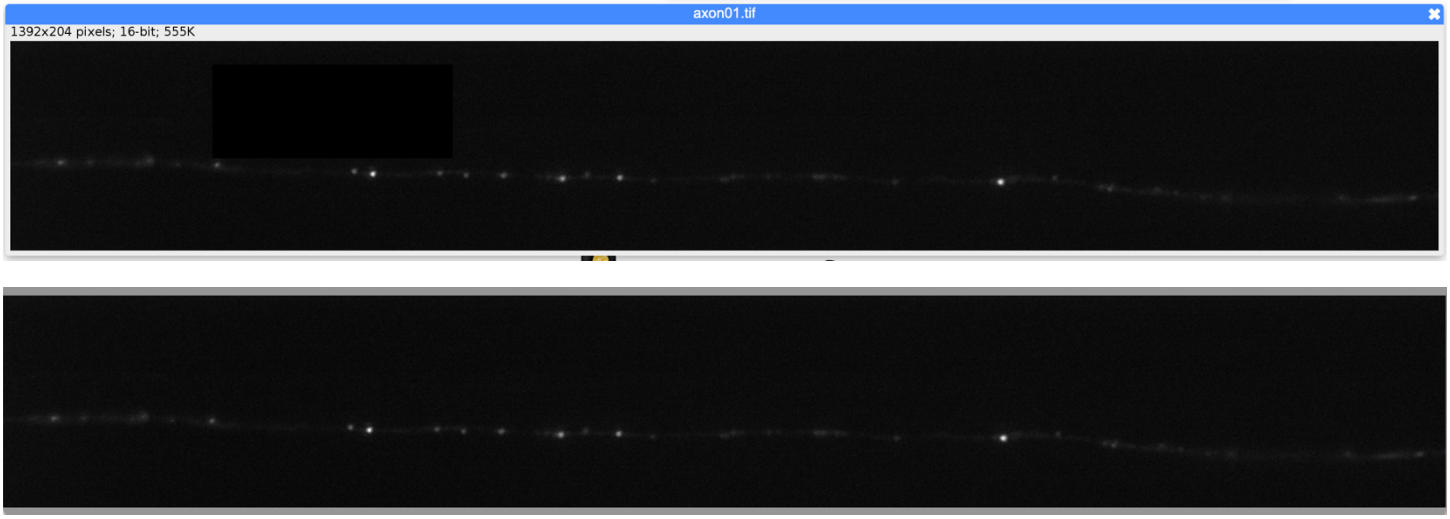
### 1.3 Pillow 所提供的功能

Pillow 除了基本的图像读取写入等操作之外，提供了丰富的图像处理：
图像识别：得到图像的具体信息，例如尺寸、压缩类型等；
区域操作：图像剪切、粘贴、合并、子矩形操作、波段分割合并等；
几何变换：调整大小、旋转、翻转等操作；
颜色变幻：在不同颜色空间中进行转换；
图像增强：包含多种滤波器、像素点操作、对比度/亮度/色彩平衡等操作；
图像序列：读取序列图像并进行操作；
此外，还包括一些具体操作，在此不一一详细列举.

## Question 2: Input/output of a static greyscale image

2.1 读取图像

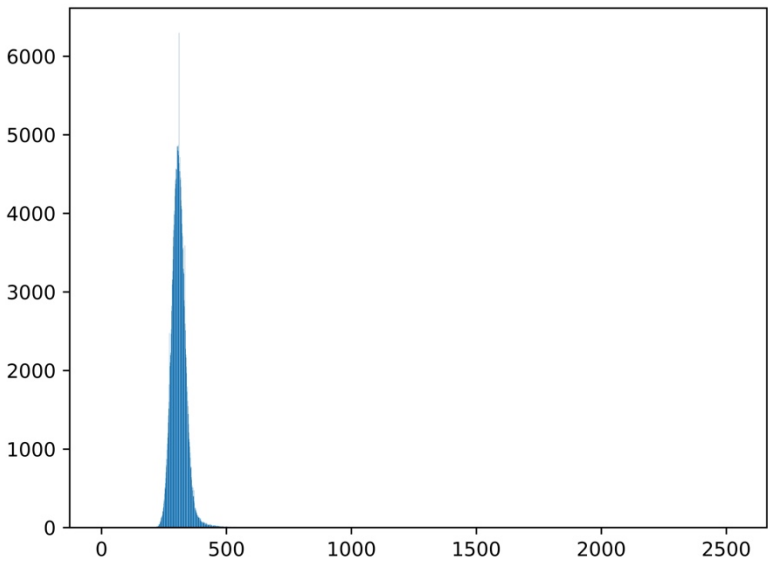
使用 PIL 的 Image 模块读取图像为数组，此时打印可知其为 uint16 图像；由于 PIL 不支持 16 位图像显示，为了正常显示，之后进行标准化转换，并转换成 uint8 图像以显示。下图中，上图为 ImageJ 读取结果，下图为 PIL 读取显示结果：




从上图可知，显示结果正确。

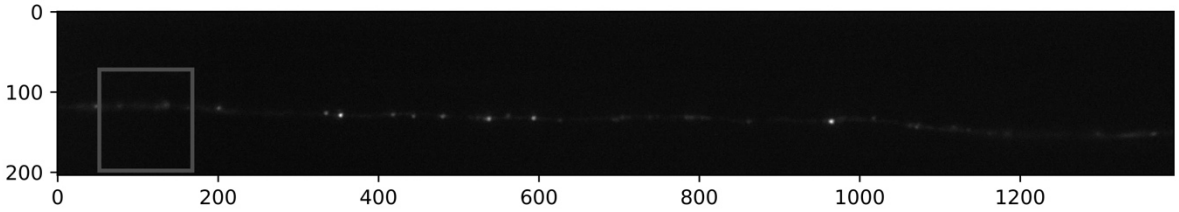
绘制强度直方图时，需要对 16 位原图的像素进行统计。由于 16 位灰度图理论上含有 0～$2^{16}$ 共 65536 个灰度值。若将所有的灰度值都显示在一张坐标轴中，显示效果十分差。因此我选择显示灰度值范围为：0～max _pixel 而非全段灰度值。

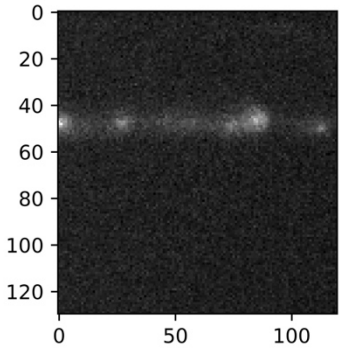
实验中发现，全图最大灰度值为 2535 ，远小于 65536，因此该方法比较合理。另外，为了使得灰度直方图更加清晰，选用 dpi=1200 进行图像保存（该参数可以自选调整）。具体结果见文件内附图。



## 2.2 剪裁图像

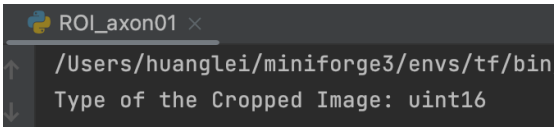
使用 Image 自带的 crop 函数进行裁剪。为了便于比对查看，当给定区域矩形框的时候，会在原图对应的八位显示图像中用矩形框选出来，同时会将所剪裁的部分提取出来（如下图所示）：



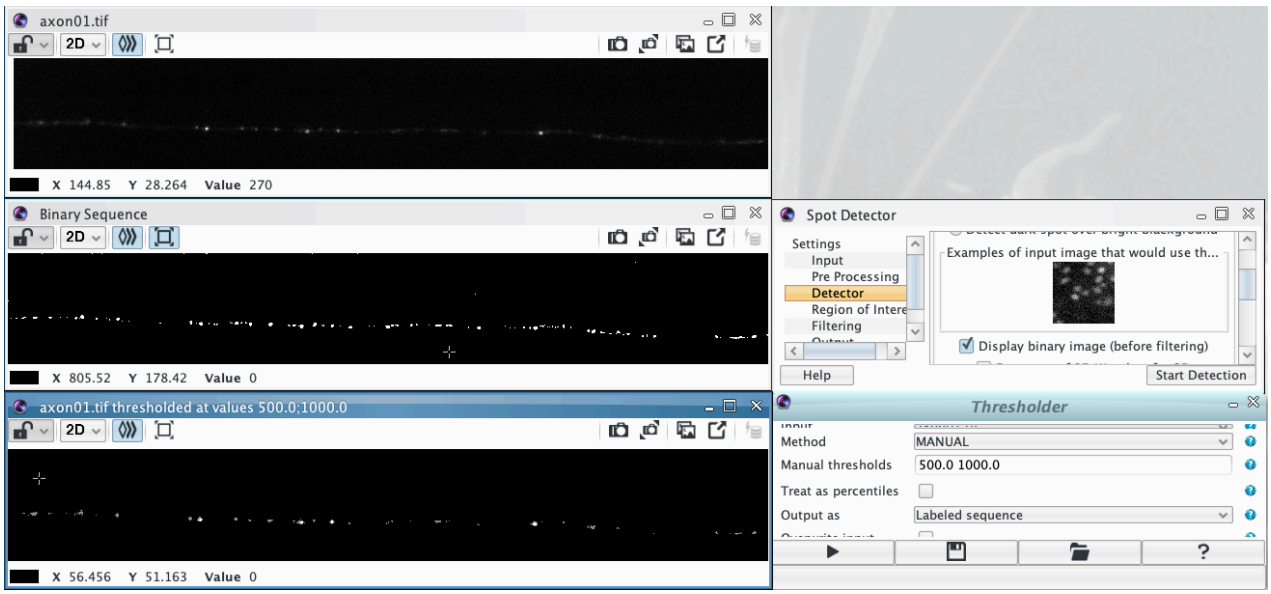

由于 16 位图像不能直接显示，因此该处显示的是变换后的 8 位图像，在保存的时候应该保存原 16 位图像。文件命名格式为“Cropped_ROI_x1_y1_x2_y2.tif”，其中后四个参数为框选矩形对角线的坐标点，这样能够保证每次框选出来的区域都对应一张图片进行保存。

为了验证成功对该剪裁后的 16 位图像无损保存，在 ROI_Crop() 给了一个验证可选项。其将保存后的剪裁图像再次进行读取，输出其图像类型。可以看出，该 16 位图像成功被保存。




**Question 3: Image segmentation using ICY-Spot Detector**

图片 axon01.tif 的前后背景相差不大，导致 segmentation 比较困难。初次接触 Icy，网络上也没有太多他的教程，导致用起来有一些不熟悉。尝试了自动最佳阈值的方法，效果很差，暂时不贴图了；另外尝试了粒子检测的方法（中），以及手动阈值的方法（下），效果还不错：



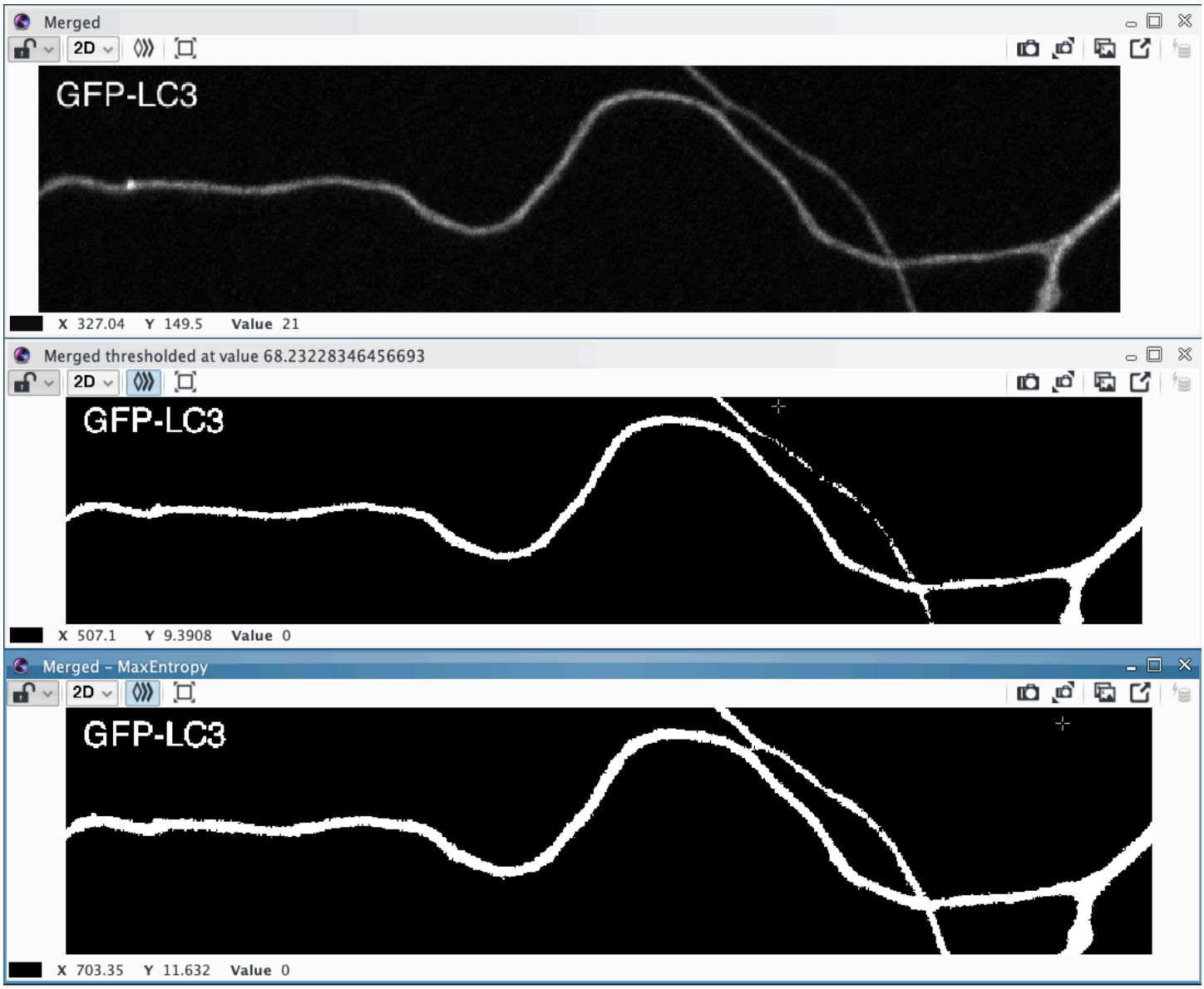
另外，本来想定义高斯核进行滤波再进行局域阈值分析，但在 Icy 上未找到相关操作，例如采取核：

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

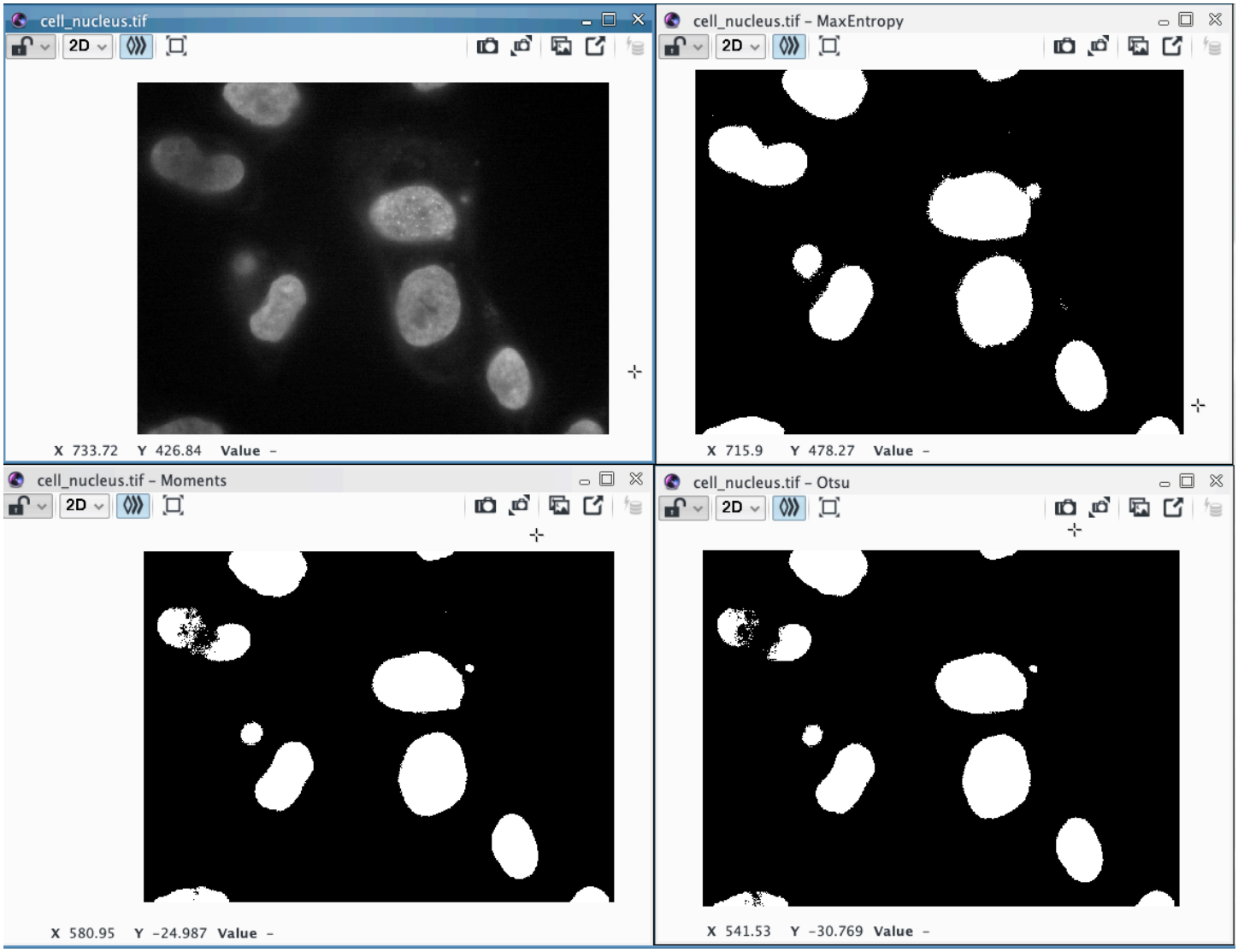
可以突出亮点部分，有助于找到亮点中心位置，进而优化阈值分析的结果。

**Question 4: (EXTRA CREDIT: 10 points): Retrieving image**

图片 axon02.tif 的前景和背景区分非常明显，因此使用阈值方法就能很好地进行分割：




另外一张图片 cell_nucleus.tif 也是如此，但是其包含了更多信息。经过 segmentation 后的图片丢失了一定隐含的动态信息，例如左上角的细胞，有可能是在变形，也有可能是在分裂。Moments 和 Otsu 方法将其彻底分开，一定意义上可以看做效果不好。另外，对于中间的大细胞，原图中可以隐隐约约看见有一个巨大的外膜包裹（不确定是不是细胞分裂后的细胞核形成），segmentation 无法表达这种关系，直接分解为两个细胞。




## D. Summary and Discussion

在本次实验中，回顾了 Python 常用图像处理库 PIL 的常用功能以及基本操作，并使用 Pillow 对 16 位图像进行读取、显示和剪裁操作，并对 16 位图像进行无损保存。另外，了解并熟悉生物图像处理工具 ImageJ 和 Icy 的操作。

通过本次实验，我对 Pillow 图像处理库了解更加深刻，更重要的是对 16 位图像的操作更加熟悉，在实验过程中，不仅熟悉了 Pillow 库，更查阅资料了解了 tifffile、opencv，imageio 等，了解 Pillow 不支持 16 位图像的显示，必须要转成 8 位图像；在保存中，也需要使用 imageio 进行无损写入。同时，我感觉到 Icy 使用起来比较困难，相关教程也非常少，入门门槛有些高，这些软件大都集成化、专业化，因此入门较难；相比起来，自己写代码实现一些简单的功能显得灵活一些。另外，非常期待上手深度学习方法进行相关数据处理实验。