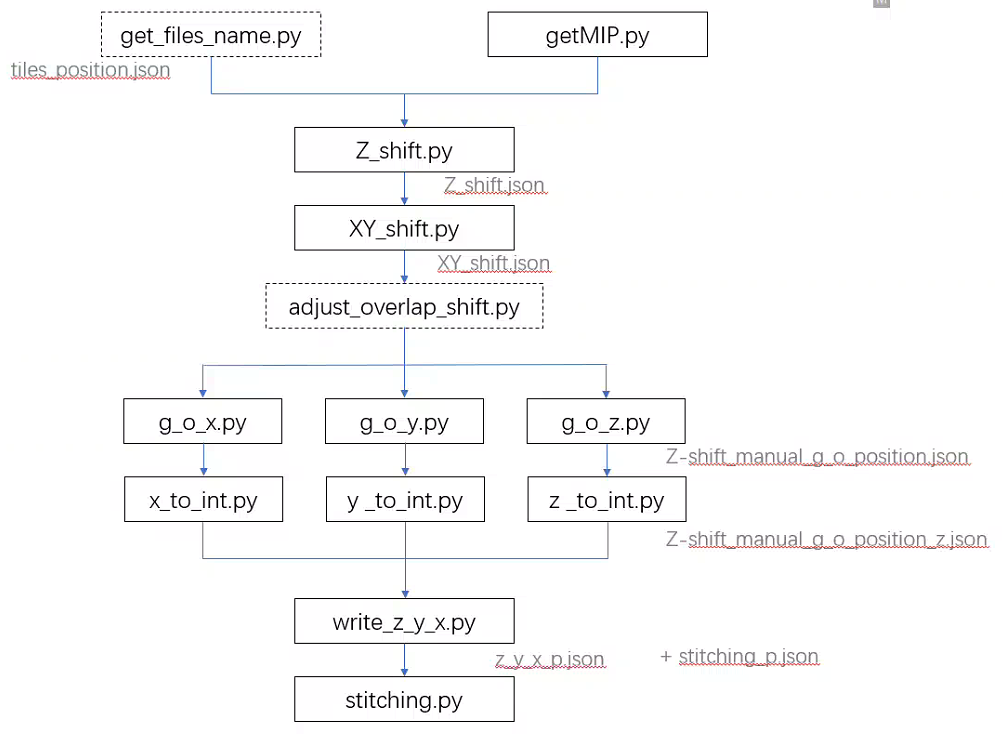
Stitching工作流程图如下所示，各个步骤得到的json文件格式参考在para文件夹中：



# a\_get\_files\_name

得到每个tiles的相对位置关系，以json文件 “tiles\_position.json” 保存，如果一个位置上没有对应的tile，则用 ”None” 表示。Json文件中还保存之后操作的其他变量信息。

"input\_folder": 后续计算偏移量所需的MIP图像所在的文件夹

"z\_result\_file": 表示计算Z方向shift的json文件的保存路径

"xy\_result\_file": 表示计算XY方向shift的json文件的保存路径

"locations": 表示所有tiles的相对位置关系

"x\_length": 表示每个tile每张slice在X方向上的pixel长度

"y\_length": 表示每个tile每张slice在Y方向上的pixel长度

"shift\_x\_P": 表示在左右关系设置的相邻tiles重叠区域的初始pixel大小

"shift\_y\_P": 表示在上下关系设置的相邻tiles重叠区域的初始pixel大小

"shift\_x\_P\_d": 表示在左右关系设置的相邻tiles overlap的变化范围，

"shift\_y\_P\_d": 表示在左右关系设置的相邻tiles shift的变化范围,

"shift\_x\_P\_d\_": 表示在上下关系设置的相邻tiles overlap的变化范围，

"shift\_y\_P\_d\_": 表示在上下关系设置的相邻tiles shift的变化范围,

"shift\_z\_P\_d\_": 表示相邻tiles 设置的Z shift的变化范围,

"thread\_num": 运行程序所设置的线程数

## get\_files\_name.py

根据成像系统得到的所有tiles的原始坐标信息，得到保存所有tiles相对位置关系的json文件。根据不同成像系统得到的不同原始坐标信息，程序需要略作修改。该程序以如下所示坐标信息为例，其中FieldX表示tile编号，PosX, PosY分别表示X、Y方向的物理坐标。

# <Tile FieldX="1" FieldY="0" PosX="0.0271577952" PosY="0.0306397943" />

## name\_position.py

根据上述成像系统得到的所有tiles的原始坐标信息，画出相对位置坐标所对应的散点图。

# b\_get\_MIP

得到每一个tiles的MIP（最大强度投影）

## getMIP.py

该程序有7个函数，前3个函数分别得到每个tileXY、YZ、XZ平面的MIP图像，后4个函数得到overlap区域的MIP图像。

XY\_MIP\_function (input, output)

YZ\_MIP\_function (input, output)

XZ\_MIP\_function (input, output)

YZ\_projection\_MIP\_L (input, output)

YZ\_projection\_MIP\_R (input, output)

XZ\_projection\_MIP\_U (input, output)

XZ\_projection\_MIP\_D (input, output)

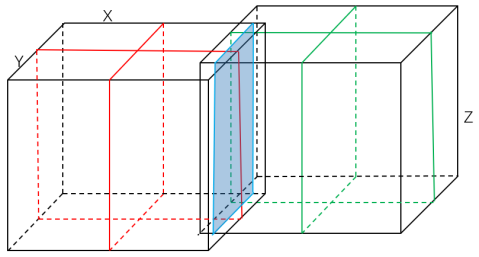


图1 相邻tiles overlap区域的MIP图像示意图

## getMIP\_One\_folder.py

成像之后所有tiles的数据放在1个文件夹里时，得到所有tiles的MIP图像。

# c\_calculate\_shift

计算相邻tiles之间的XYZ方向的shift。

## Z\_shift.py

输入1.1得到的 “tiles\_position.json”，计算得到相邻tiles的Z\_shift，结果保存在 “Z\_shift.json” 文件中。

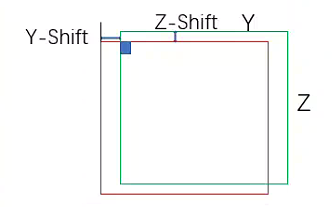


图2 相邻图像overlap区域MIP图像计算Z shift的示意图

## XY\_shift.py

计算得到相邻tiles对应MIP图像或考虑Z shift后的单张平面在XY方向的偏移量。

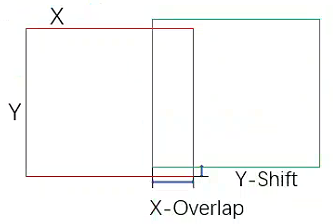


图3 相邻tiles计算overlap和shift的示意图

## 3.3 adjust\_overlap\_shift.py

1. 检测需要修正的数据，如果一个 tile A 上下左右的参数有一组对应 tile B 计算正确，且在后续做global optimization的时候，正确参数对应 tile B 坐标正确，则可以利用此正确参数和tile B坐标，得到该 tile A 的坐标；
2. 自动修正：如果从上到下做部分MIP（比如0-30, 30-60…）计算正确范围shift的平均值，如果不能得到正确范围的解，就使用周围同一行附近的overlap和shift平均值。
3. 自动修正后，仍不能得到正确范围的解的tiles，存放在 “need\_manual.json” 文件中，需要人工修正。

# d\_global\_optimization

对计算得到的shift做一个全局最优，使得到每个tiles的对应平面自洽

## g\_o\_x.py

## g\_o\_y.py

## 4.3 g\_o\_z.py

# e\_float\_to\_integer

使每一个对应平面变成整数，并得到每一个tiles的全局坐标。

## 5.1 x\_to\_int.py

## 5.2 y\_to\_int.py

## 5.3 z\_to\_int.py

## 5.4 write\_z\_y\_x.py

以 “z\_y\_x\_p.json” 文件保存每一个tile的全局坐标，以及每一个tile的XY方向pixel长度。

# f\_stitching

基于像素融合的方法拼接所有的tiles，以 “stitching\_p.json” 文件记录需要拼接的图像路径、拼接后结果的保存路径、拼接前Z方向的最大平面数、并行计算允许的线程数。

# 需要注意的问题

* 多线程跑程序，如果程序出错，不会出现报错信息，需要先改为单线程程序，根据报错信息，debug程序；