数据读取管道：医学影像文件导入指南

Chen Zhang

2022.12.12

1 DICOM 协议相关库对数值问题的处理

Python生态中对 DICOM 文件的主流导入支持，主要是通过pydicom或者SimpleITK两个库得以完成，然而这两个库在数据导入的过程中在对数值的处理上存在区别。由于这两种库的官方文档中缺乏与其它同类型库的横向对比，而网上的技术分享文档中针对数值问题的细节也鲜有提及，因此在使用过程中有诸多注意点需要在数据导入过程中尤其注意。

1.1 数据维度及体素顺序

SimpleITK在进行单层 DICOM 数据导入时,使用GetArrayFromImage方法将ReadImage所获取的图像加载便能得到数值信息，但读入后的数值仍维持 d=3 的维度，其体素排列顺序为 zxy。如果确定读取的是单层 DICOM 文件，则可用第一个索引将数值取出，此时的数据ndim属性应为 2，体素排列顺序为 xy。

pydicom在进行单层 DICOM 数据导入时，使用pixel_array属性获取图像数值信息，读入后的数值维度属性 (ndim) 为 2，其体素排列顺序为 yx，且正好与SimpleITK的数值矩阵形成转置的关系。

1.2 数值处理精度

SimpleITK中(U)Int8，(U)Int16，(U)Int32等都是其内置数值类型，在数据读入时其数值精度较宽。而 pydicom 的数据加载完成后,一般为ndarray，(U)int16类型，且多数情况下为uint16类型。因此在处理 CT 数据时，常

常会出现最低 HU 值为 0 的情况，需要将数据类型强制转换为`int16`后，再在此基础上减去 1024，数值上才能与 SimpleITK 相对应。

此外，由于`uint16`的数值精度极限为 $2^{15} = 32768$，对于如 PET 模态这样数值范围较大的数据，用`pydicom`读取通常会导致因处理精度不足而造成的数值溢出。此外，数值溢出并不会引发`pydicom`的报错或警告信息，因此在未进行可视化前该问题很难被察觉。

1.3 其它细节

`informatics`里针对`SimpleITK`与`pydicom`标准不一致等问题，都做了相应的集成优化。基于 DICOM 序列的重建逻辑源码位于`rebuild`模块中。上述的数值处理逻辑可参考该脚本中`DcmSeries`类的初始化过程，重建数值标准以`SimpleITK`作为参考，体素顺序默认采用 zyx 的方式进行堆砌。

与`SimpleITK`相比，`pydicom`对中文路径支持友好，因此在该模块中预设了两套配置 (`SimpleITK_config`,`pydicom_config`) 可供采用。`informatics`的 DICOM 重建管道默认采用`SimpleITK_config`的配置来进行数据读入加载，但当处理的数据中存在大量非英文路径，且数据的数值范围较小且稳定时 (如 CT 图像)，可将配置更换为`pydicom_config`以作更好的适配。

另外，通过`pydicom`读取后的图层，其数值数据会以二进制缓存形式保存在字段`PixelData`中。用`get("PixelData")`获取图层的二进制缓存后，用`numpy`的`frombuffer`方法，指定`dtype`后即可将数据恢复为`numpy`的一维数组，再利用`reshape`方法，可以获得与`pixel_array`相同的结果。这里需要强调的是，`PixelData`的二进制缓存与`pixel_array`本质上等效。作者尝试将数值处理精度不足的图层，其对应`PixelData`取出并从缓存中进行重建，但最终效果也与产生溢出后的数值结果相同。因此，对于存在上述问题的数据，应考虑采用`SimpleITK`来作为数据加载引擎，并在加载时，指定精度满足要求的数据类型。

2 DICOM 在物理空间中的重构

2.1 像素空间与体素空间的映射

根据 DICOM 委员会 PS3.3 2022d 标准，图层像素与体素间的转换关系图层像素与体素间的转换关系如式1所示：

$$\begin{bmatrix} V_x \\ V_y \\ V_z \\ 1 \end{bmatrix} = \begin{bmatrix} X_x\Delta_i & Y_x\Delta_j & 0 & S_x \\ X_y\Delta_i & Y_x\Delta_j & 0 & S_y \\ X_z\Delta_i & Y_x\Delta_j & 0 & S_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_i \\ P_j \\ 0 \\ 1 \end{bmatrix} \tag{1}$$

上式中，$[X_x,\ X_y,\ X_z,\ Y_x,\ Y_y,\ Y_z]$ 对应着 Image Orientation Patient 标签 (“0020|0037”)，为体素平面两个基向量与像素平面三个基向量间的余弦值；$[\Delta_i,\ \Delta_j]$ 对应着 Image Spacing 标签 (“0028|0030”)，为体素平面两个指向行、列基向量的体素间距；$[S_x,\ S_y,\ S_z]$ 对应着 Image Position Patient 标签 (“0028|0032”)，为像素坐标原点对应着体素空间中的绝对坐标。

$[X_x,\ X_y,\ X_z,\ Y_x,\ Y_y,\ Y_z]$ 为无量纲量，而 $[\Delta_i,\ \Delta_j]$ 与 $[S_x,\ S_y,\ S_z]$ 在真实物理空间中则以毫米 (mm) 进行计量，因此根据式1中转换矩阵求解出来的体素坐标 $[V_x,\ V_y,\ V_z]$ 应当仍旧保持毫米 (mm) 的计量单位。

2.2 informatics 管道中的仿射矩阵

在`informatics`的 DICOM 构造管道中，利用仿射矩阵来代替转换矩阵。仿射矩阵采用式1的泛化形式：式2为标准来进行执行。该矩阵可以通过图像重建完成后图像的`affine`属性来进行访问。

$$\begin{bmatrix} V_x \\ V_y \\ V_z \\ 1 \end{bmatrix} = \begin{bmatrix} X_x\Delta_i & Y_x\Delta_j & Z_x\Delta_k & S_x \\ X_y\Delta_i & Y_x\Delta_j & Z_y\Delta_k & S_y \\ X_z\Delta_i & Y_x\Delta_j & Z_z\Delta_k & S_z{}' \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_i \\ P_j \\ P_k \\ 1 \end{bmatrix} \tag{2}$$

通常情况下，$[X_x,\ X_y,\ X_z]$ 和 $[Y_x,\ Y_y,\ Y_z]$ 会与像素空间中的行、列指向相重叠，即便存在坐标旋转，$[X_x,\ X_y,\ X_z]$ 与 $[Y_x,\ Y_y,\ Y_z]$ 的二范数通常也为 1(即互为线性无关的单位向量)，因此 Image Spacing(“0028|0030”) 可以互换为行、列的体素间距。然而更一般地，若存在坐标的缩放 (如带缩放的仿射变换)，则映射的构建不能简单以 Image Spacing(“0028|0030”) 为标准进行。

式2中，$[Z_x,\ Z_y,\ Z_z]$ 与 Δ_k 是通过已知信息间接计算而来。在确定 $[X_x,\ X_y,\ X_z]$ 和 $[Y_x,\ Y_y,\ Y_z]$ 正交的前提下，$[Z_x,\ Z_y,\ Z_z]$ 为 $[X_x,\ X_y,\ X_z]$ 和 $[Y_x,\ Y_y,\ Y_z]$ 的向量外积。Δ_k 则是通过层间距字段 (Spacing Between

Slices，“0018|0088”)，或者 Image Position Patient 标签 (“0028|0032”) 的最后一个元素的层间差计算取得。

与式1相区别的是，式2中的 S_z' 为首张图层所对应体素空间中原点坐标的 z 值。

2.3 仿射矩阵的数学性质与现实意义

线性代数中我们知道，矩阵事实上对应着把数据从一个空间呈现到另一个空间的映射规则。通过上述示例 (式2) 可以看出，仿射矩阵实际上是数据从像素空间到物理空间中的对应法则，其构成考虑了旋转、平移、以及拉伸等因素，是一种通用的仿射变换。

使用仿射矩阵的一大优势是，它是一种抽象的数学概念，与厂商、设备型号、数据导出方式、甚至是读入的库逻辑无关。一般我们关注的影像信息如 Spacing、Origin 等属性，都能够被仿射矩阵所包含 (例如向量 $[X_x\Delta_i,\ X_y\Delta_i,\ X_z\Delta_i]$ 的模长即第一个轴向上的体素间距)。有了仿射矩阵，就可以不再需要针对不同厂商、不同设备型号等因素写大量的判断语句去进行适配，因为各种设备型号等的预定义坐标体系、数值上的不同，最终都会反映在仿射矩阵上，因此基于仿射矩阵的管道会对数据有着始终如一地呈现。

使用仿射矩阵的另一大优势是能够简化计算和方便拓展。例如需要对图像进行重采样，需重新计算体素间距 Spacing 时，可以完全简化为仿射矩阵的缩放问题，再从被缩放后的仿射矩阵中重新提取体素间距；又例如需要对获取图像旋转角度等位相信息时，可以简化为仿射矩阵的奇异分解问题从而获取成像时的旋转矩阵。

3 NIfTI 协议文件的一致性处理

医学影像格式除 DICOM 标准外，另一套常用的标准是 NIfTI，其后缀常为“nii”或“nii.gz”。仿射矩阵在其它库中也有体现。如`nibabel`是`Python`中读取该文件的库，加载后的数据，能够通过`affine`属性访问其仿射矩阵。`SimpleITK`中也包含支持该格式的对应 IO。

NIfTI 由于考虑兼容性等原因，存在三种不同的构建方式。如果其原始信息中显示是通过`sform`构建，则其像素到物理空间中的对应关系便如式2一致。而当图像采用`qform`来进行构建时，其仿射矩阵则无法与真实的物理空

间发生对应，这里需要分情况来进行讨论：

- 存在qfac因子
- 不存在qfac因子

第一种情况是采用较新的qform标准，qfac的作用在于修正（左右手）坐标系，使数据的呈现保持一致，其仿射矩阵信息能包含到物理空间的旋转、平移等信息；而第二种情况对应于比较旧的标准，仅包含像致至体素的对应关系，到物理空间的旋转信息会产生丢失。需要指出的是，两种无论是哪种方式，其转换到的物理空间中的坐标相对设备而言的。如果复数个影像数据在采集过程中，设备存在不同的空间补偿设置，则采用qform的构建很难将这部分信息也包含进去。

在 0.0.5 版本之后informatics会以SimpleITK为基础，来实现与 DICOM 构建类似的数据访问逻辑。

