上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

课程报告

CT 成像的扫描与重建算法实现

生物医学图像处理（2）课程作业

518021910971 裴奕博

目录

1 项目背景 **2**
1.1 CT 重建算法 2
1.2 数学基础 2

2 项目实现 **3**
2.1 项目总体实现 3
2.2 Radon 正变换的实现 3
2.3 Radon 逆变换的实现 4
2.4 图形化界面的实现 4

3 项目结果和评价 **6**
3.1 可视化结果 6
3.2 运算速度 7

4 感想与展望 **7**

1 项目背景

1.1 CT 重建算法

自 20 世纪 70 年代被发明以来，X 射线计算机断层成像（CT）在医学影像检查中扮演了越来越重要的地位。在 CT 成像的流程中，将图像进行数字化、投影和重建的算法非常重要。CT 成像算法的好坏，会直接影响到 CT 的成像质量。

1.2 数学基础

中心切片定理和滤波反投影算法的提出，是 CT 成像和重建算法的数学基础。

中心切片定理可以表述如下：

定理 1 (中心切片定理).

$$P(\omega,\theta) = F(\omega\cos\theta, \omega\sin\theta)$$

其中 P 是投影函数 $p(l,\theta)$ 的傅里叶变换，$F(u,v)$ 是图像的二维傅里叶变换。

该定理说明，某断层在角度为 θ 时得到的平行投影的一维傅里叶变换，等于图像二维傅里叶变换过原点的一个垂直切片，且切片的方向也为 θ 角。[2]

通过中心切片定理，我们可以得到原 CT 图像的投影图，再经过重建算法就可以得到 CT 重建图像。重建算法主要分为两大类：滤波反投影算法和反卷积反投影算法。本次的重建算法属于滤波反投影算法，可以用公式表示如下：[1]

定理 2 (滤波反投影算法).

$$f(x,y) = \int_0^{\pi} |\omega| P(\omega,\theta) \exp(j2\pi\omega l)|_{l=x\cos\theta+y\sin\theta}$$

其中 $P(\omega,\theta)$ 是通过中心切片定理得到的投影图的傅里叶变换。

2 项目实现

2.1 项目总体实现

项目的整体流程比较简单，如下图：

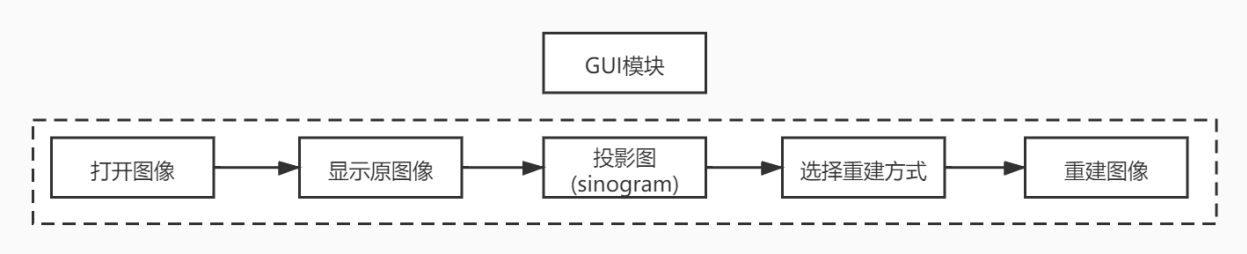

图 1: 项目流程图

所有程序均采用 Python 实现，图形化界面使用 PySide2 图形化框架实现。图像处理与重建的部分使用 numpy 实现，所使用的的依赖包可见 requirements.txt 文件。

2.2 Radon 正变换的实现

Radon 正变换的实现方法如下图：

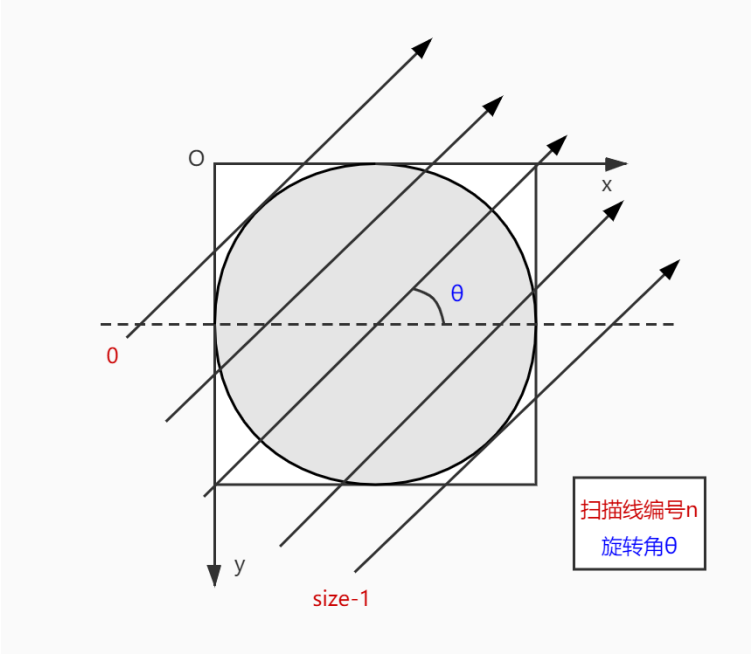

图 2: Radon 正变换实现

其中最大的正方形表示待扫描的图像。我们默认扫描的区域范围用灰色部分显示。默认的扫描线

方向如图中虚线所示，编号从上到下为 $[0, size-1]$。扫描线与默认方向之间的夹角为 θ。设扫描点的坐标为 (x, y), 图像中心点坐标为 (c, c)，变换后的扫描点坐标为 $P'(x', y')$，使用齐次变换矩阵，有

$$P' = R(P - C) + C \tag{1}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x-c \\ y-c \\ 0 \end{bmatrix} + \begin{bmatrix} c \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & c(1-\cos\theta-\sin\theta) \\ -\sin\theta & \cos\theta & c(1-\cos\theta+\sin\theta) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

记上式为

$$P' = R\prime P \tag{3}$$

即通过左乘 R 即可完成变换。由于 Numpy 对矩阵运算作了优化，因此直接进行矩阵运算相比逐项计算大大提升了速度。

最后将每条扫描线上的值相加，就可以得到其中一个 θ 角所对应的扫描值，将其显示出来即可得到正弦图，上述所有过程均使用 Numpy 的矩阵运算来实现。

2.3 Radon 逆变换的实现

Radon 逆变换的实现比较简单，可分为三个部分：频域滤波、插值重建和后处理。

- 频域滤波部分的 FFT 算法直接调用 Scipy 中的 fft，ifft 函数实现，部分滤波器也采用 Numpy 的相关滤波器函数实现。
- 插值重建部分采用了 Scipy 中的 interp1d 进行一维插值，默认使用线性插值法。
- 后处理部分：由于扫描时只扫描了中间圆形的部分，因此需要对圆形之外的区域置 0，同时也要对负值进行一些处理。

2.4 图形化界面的实现

图形化界面采用了 PySide2（PyQt）框架实现。UI 文件储存在 ui 文件夹下，由两个页面组成，一个主页面和一个滤波反投影算法的参数选择界面。

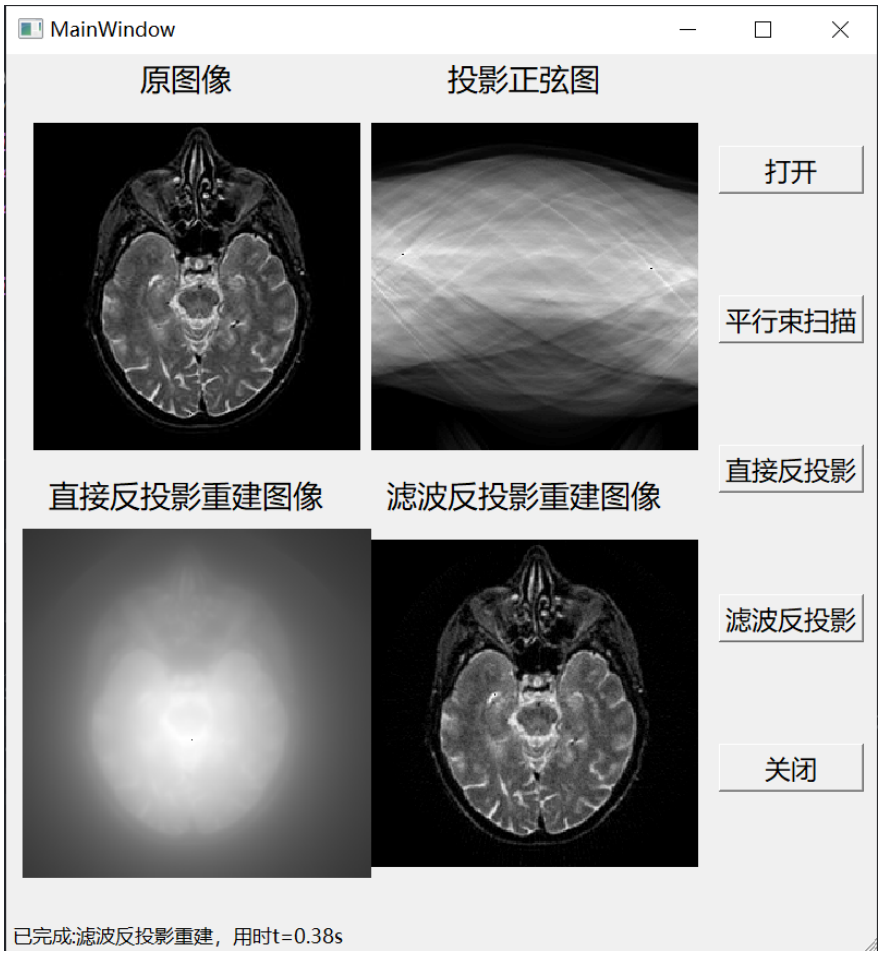

图 3: 主界面

本次实现的图形化界面具有以下几个特点：

- 可以自行选择本地文件打开。
- 使用了多线程操作。图像处理等耗时的操作运行在另一个线程下，但是在计算过程中将页面上所有交互按钮禁用，防止用户进行误操作。
- 底部有状态栏，在运行过程中可以给予提示，运行结束后可以显示进行的操作和所消耗的时间。
- 在用户误操作时可以跳出错误提示。

3 项目结果和评价

3.1 可视化结果

运行 visualize.py，可以查看可视化结果如下：图4是 shepp-logan 模型的可视化结果，图5是使用 CT 图像 test.jpg 的可视化结果，两者均为 256*26 尺寸。每张结果图中，上面一行是本次实现的算法结果，下面一行是 skimage 库中 radon 和 iradon 变换实现的结果。

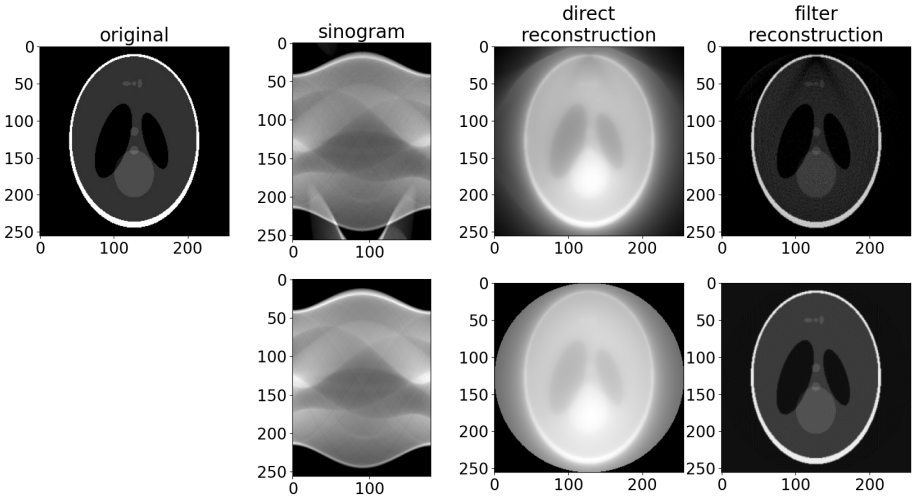

图 4: shepp-logan 模型重建结果

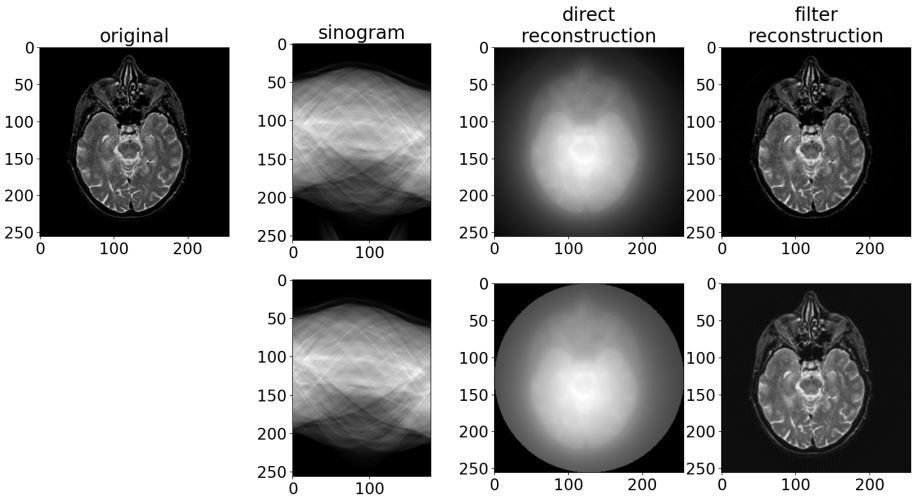

图 5: CT 图像重建结果

可以看到，无论是使用 shepp-logan 模型，还是使用一般的 CT 图像，本算法都取得了不错的效果，而且对于重建图像的周边部分进行了后处理，避免了 skimage 库算法中出现扫描区域边界重建之后颜色断层的结果。

3.2 运算速度

除了图像本身的质量以外，我还测试了本实现方法在不同图像大小情况下的运算速度，结果如下表1 可以看到，算法的速度在可接受范围内，但是与优化过的第三方库相比还有一定的差距。

表 1: 算法速度对比（时间：秒）

file	method	radon	iradon(direct)	iradon(filter)
res\shepp-logan1024.jpg	my algorithm	15.82	17.22	17.09
	skimage	8.53	5.84	5.82
res\shepp-logan256.jpg	my algorithm	2.59	1	0.96
	skimage	0.51	0.3	0.3
res\shepp-logan512.jpg	my algorithm	5.95	4.08	4.18
	skimage	2.07	1.44	1.61

4 感想与展望

本次大作业我实现了一个简单的 CT 扫描和重建算法。并通过一些辅助代码进行了可视化和计时测试，完成了一个比较完整的流程。然而现在实现的算法还存在以下几点不足：

- 运算速度相比第三方库的算法还有一定差距，需要进一步优化
- 只实现了平行束扫描和重建的方法，扫描速度较慢，没有涉及扇形束和锥形束。
- 对于异常值和结果的后处理部分比较简单粗暴，没有找到更好的解决方法。

最后，感谢助教和老师在本次大作业过程中给予的帮助。

参考文献

[1] Avinash C Kak, Malcolm Slaney, and Ge Wang. Principles of computerized tomographic imaging, 2002.

[2] GR Ramesh, N Srinivasa, and K Rajgopal. An algorithm for computing the discrete radon transform with some applications. In *Fourth IEEE Region 10 International Conference TENCON*, pages 78–81. IEEE, 1989.