CT 系统参数标定及成像

摘要

本文在充分了解了 CT 系统工作原理的前提下，设计了对 CT 系统相关参数的标定方法，并利用标定后的 CT 系统对未知介质进行了图像重建与测量，最后分析了参数标定的准确性与稳定性，在此基础上改进了标定模板，对 CT 系统的优化提出了一些建议。

对于 CT 系统参数的标定，我们分成三个部分来考虑，即要得出 CT 系统旋转中心的位置、探测器单元之间的距离以及该 CT 系统使用的 X 射线的 180 个方向。为此我们根据附件二的数据得出了标准模板的扫描图像，并找到其与模板上均匀介质图形的内在联系，进而求出探测器单元的距离为 0.2761 mm。进一步，通过比较探测器单元接收到的投影与原像的长度之比，可以得出旋转角的变化趋势，即从 29.4056° 到 208.3340°，以 0.9996° 为间隔等间距变化。最后，为了确定旋转中心的位置，我们运用几何关系同时结合旋转方向以及模板的初始位置，得出 CT 系统的旋转中心位置为 $(-8.8439, -6.6130)$（单位：mm），以正方形托盘中心为原点，以椭圆长轴为 y 轴，短轴为 x 轴。

针对图像重建的问题，我们联系数学中关于二维图像的 Radon 变换来解决这次的问题，并简要证明了 CT 系统成像中的扫描操作实际上是对于样品二维吸收率分布的某一方向的投影。结合中心切片定理，得到 Radon 逆变换的算法步骤，并且使用相应的 MATLAB 函数及前面标定的参数，得到一个图像重建的通用方法。使用该方法，我们较为准确地重建了附件 2 和附件 5 中未知样品的吸收率分布，并进一步得到了要求的 10 个点的吸收率。具体的吸收率可见附录的表格 5.2.4。

对于设计新模板的问题，我们首先分析了之前处理模板中遇到的一些困难和缺陷，即处理边界时交叉的部分会导致边界提取较大的误差。基于这一点，我们设计了空心圆的样品模板，规避了外围边界交叉的问题，能够更简单快速地标定相应参数，优化了边界提取效果，从而达到预期的效果。

最后，本文分析了现有模型的缺陷，并提出了进一步改进的方向。

关键字： 图像重建　Radon 变换　中心切片定理　三次样条插值

一、 问题重述

本题要求我们解决的问题大致可以概括为：

(1) 对现有的 CT 系统的有关参数，如旋转中心位置，探测器单元距离以及 X 射线方向等参数进行标定。标定的方法是根据其对一个较为标准的模板的测量数据来求出对应参数。

(2) 用已经明确参数的 CT 系统来对给出的未知介质进行测量。对所得的扫描结果进行图像重建，得出两种未知介质的相关性质，位置、几何形状和吸收率等信息，并给出指定点的吸收强度。

(3) 分析上述 CT 系统参数标定的精度以及稳定性，同时改进模板，以提高标定精度及稳定性。

二、 问题分析

我们先对材料中叙述的 CT 的实现原理进行了程序模拟，以便能得到一个较为直观的认识。程序模拟的结果表明，如果我们的旋转中心使用椭圆中心，每次逆时针方向旋转 1°，所得的扫描结果如图 1 所示。同时为了直观对比，也导出实际的结果，如图 2。

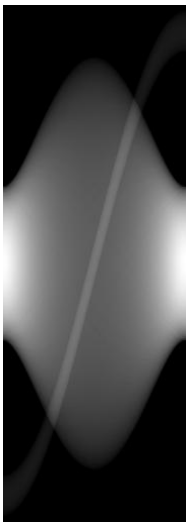

图 **1**　无偏差扫描结果灰度图

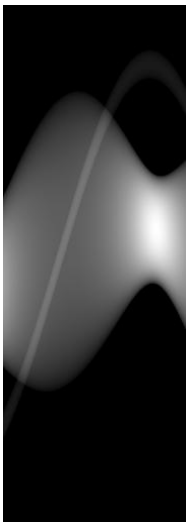

图 **2**　实际扫描结果灰度图

问题一要求我们根据标准均匀介质模板的测量数据来求出 CT 系统的三个参数，即旋转中心，探测器单元距离以及 X 射线方向。

首先，根据附件二，我们得出了测量数据对应的扫描结果的大致图形轮廓，在图形中找到椭圆长轴对应的像，以及其对应的探测器单元。由于已知椭圆长轴的实际长度，

以及探测器单元在探测器平面上是等距分布的，那么根据长轴的像对应的探测器单元个数，就可以得到探测器单元之间的距离。对于旋转中心，考虑模板中相对特殊的两个点，即圆心以及椭圆中心，同时考虑它们与旋转中心连接的线段长度。由于平行入射的 X 射线垂直于探测器平面，那么这相当于将正方形托盘上的二维待测介质投影到探测器平面上，那么也会将上述两条线段投影在探测器平面上。由于在前面已经得出了探测器单元之间的距离，所以可以由投影所在的探测器单元区间知道投影的长度。由于旋转中心在旋转过程中在探测器面板上的投影位置保持不变，而投影的长度却与旋转角度具有正弦关系，通过附件二中测量数据对应的扫描结果，我们可以得到圆心与椭圆中心的投影轨迹，进行正弦函数拟合后可以得到两个正弦函数的振幅。有投影的定义可以知道这两个振幅与圆心和椭圆中心到旋转中心的线段长度对应相等。分别以圆心以及椭圆中心为圆心，以这两个长度作为半径作圆，两圆交点便可能是旋转中心的位置。再结合旋转方向为逆时针以及正弦函数的初相位，便可唯一确定旋转中心的位置。对于 X 射线的 180 个方向，我们根据上面得出的圆心以及椭圆中心的轨迹对应点之间距离与两点连线之距离的比值，此即为旋转角的余弦值，用反余弦函数得出旋转过的角度，就知道了 X 射线的方向。

对于问题二和问题三，所给的数据是该 CT 系统对于两个未知样本的扫描结果，我们需要借助上题中标定的参数，将扫描结果进行图像重建，得到未知样本的在正方形托盘中的位置、几何形状和吸收率等信息。分析可得，这个图像重建过程是借助二维图像在各个方向上的一维投影来进行重建的。联系到数学上的 Radon 变换，有一些比较成熟的图像重建方法可供我们参考。

三、 模型假设

根据材料和上述分析，我们做出以下假设：

(1) CT 系统的探测器平面与 X 射线方向完全垂直。

(2) X 射线认为是平行入射的，没有误差。

(3) 测量出的 X 射线的吸收仅与样品的内部的吸收率有关，忽略折射散射等的影响。

四、 符号说明

符号	意义
$g(\rho,\theta)$	在 θ 方向上的投影
$f(x,y)$	二维图像或吸收率二维分布
$r(x,y)$	重建的图像或计算得吸收率二维分布

$G(\omega,\theta)$	$g(\rho,\theta)$ 关于 ρ 的 Fourier 变换
$F(u,v)$	原图像的二维 Fourier 变换

五、 建立模型与求解

5.1 CT 系统的参数标定

5.1.1 扫描结果的处理

附录二的每一列均记录了一个角度对应的吸收强度分布情况。对应的大拱形图样与小拱形图样分别由匀质的椭圆与圆产生，其边界的即为椭圆与圆投影的边界。三种典型的分布如图 3 所示。

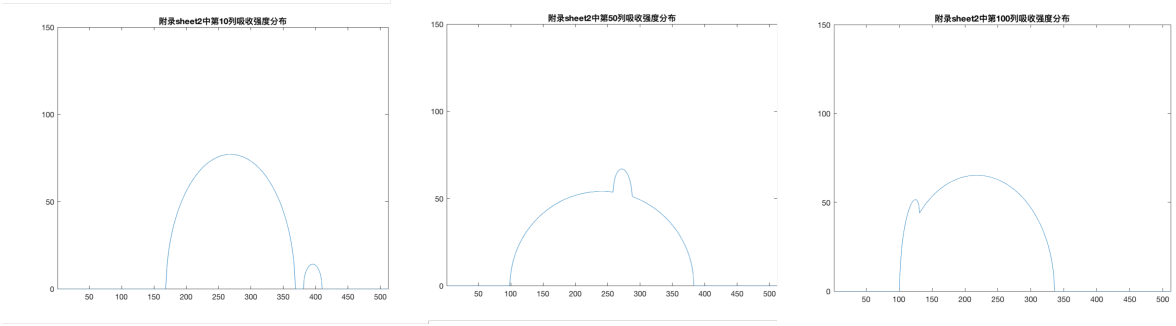

图 3　吸收强度典型分布图

观察到在边界处，图形的梯度较大。利用 MATLAB 中的 gradient 函数，求出列方向上的梯度如图 4 所示。

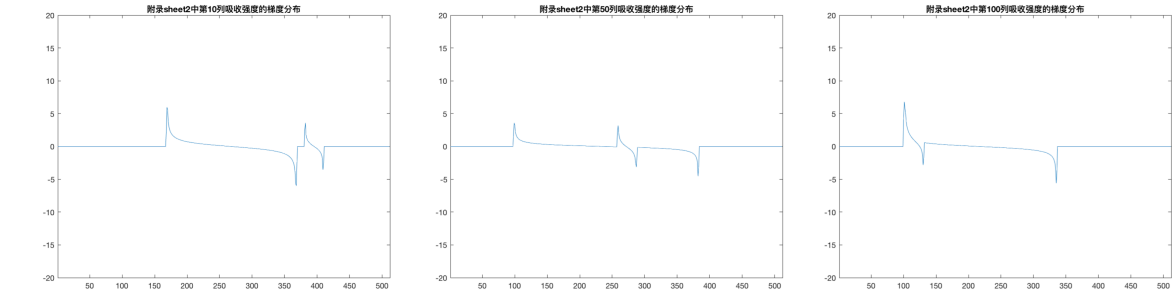

图 4　列方向梯度分布图

则波峰与波谷所对应的点即为投影的边界点，且较大的波峰、波谷与较小的波峰、波谷一一对应着两个图样的上边界与下边界。求出波峰和波谷（MATLAB 中 findpeaks）。由于该函数会提取出所有的波峰，而其中部分波峰是由于两个图样相聚较近而产生的小

峰值的干扰，编写 FindProperPeaks 寻找出较大的波峰，作为对应列的图样的边界点。由此，我们提取出了四条边界线，如图 5 所示。

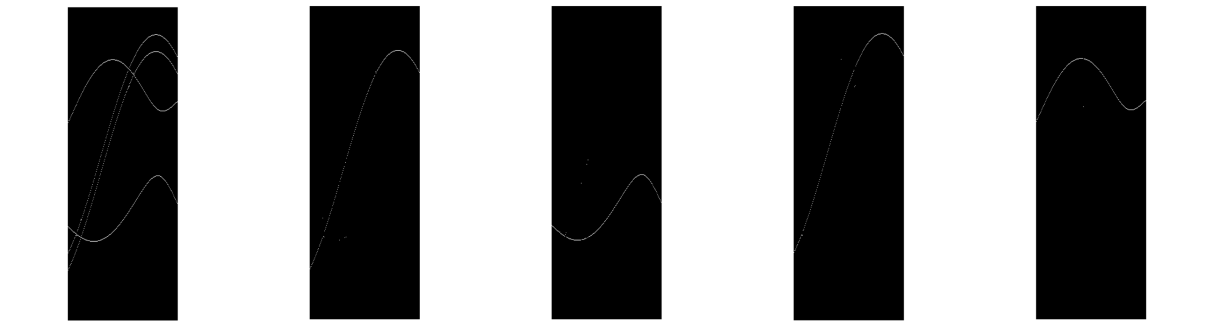

图 5　边界提取结果

观察到提取出的边界仍有少数点是偏离的。这是因为 FindProperPeaks 通过找出波峰的最大值、波峰的第二大值和波谷的最大值、波谷的第二大值来作为将求出的点分类到某一条曲线上的依据，而在边界曲线交叉的时候，会产生波峰波谷相消的现象，使该判断出现问题，需要对曲线进行进一步的降噪。去除偏离点后，用三次样条插值 [2] 实现对缺少点的补全，可以求圆心、椭圆中心的轨迹如图 6 所示。

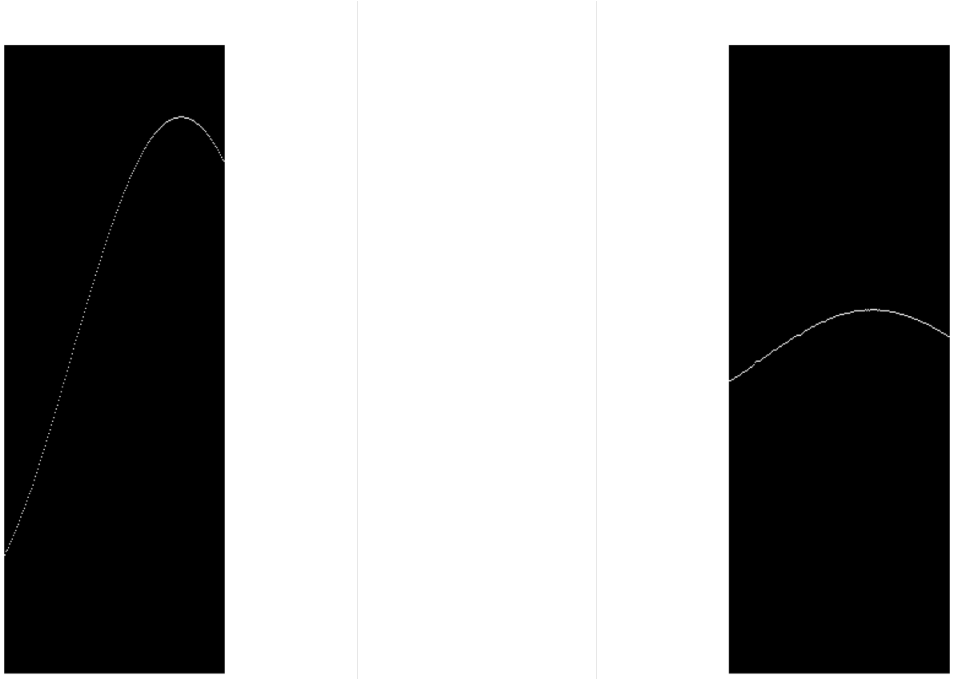

图 6　降噪后的圆心和椭圆中心轨迹

5.1.2 180 个方向的确定

以探测器平面为参照系，则圆心、椭圆中心与旋转中心旋转过程可用图 7 表示。

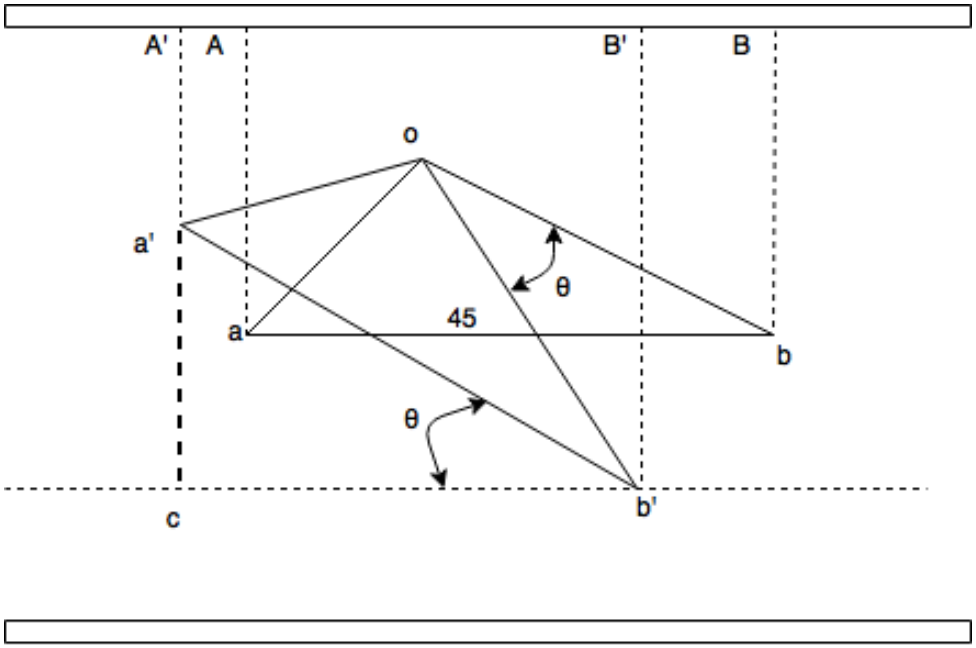

图 7　旋转过程示意图

其中 a 为椭圆中心，o 为旋转中心，b 为圆心，当 oab 旋转 θ 角成为 $oa'b$’ 时，由 $\triangle a'cb'$ 的几何关系知：$\cos\theta = cb'/a'b'$。其中 $cb' = A'B'$ 为椭圆中心到圆心的投影距离，可由之前获得的轨迹读出。$a'b'$ 为模板上椭圆中心到圆心的距离，为 45 mm。由此，根据 $\theta = \arccos(cb'/a'b')$ 可求出 180 个方向。所得数据图如图 8 所示。

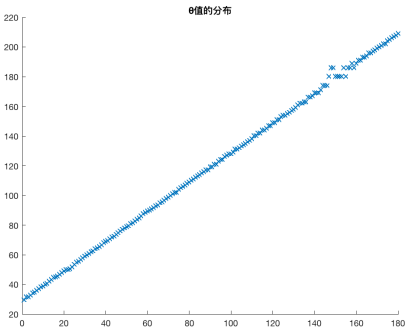

图 8

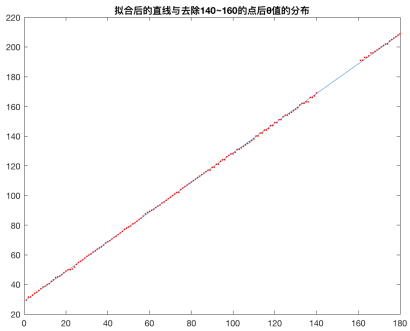

图 9

观察到整体趋势呈线性，但在 140 160 处线性度较差。对 θ 关于 cb' 求微分，有

$$d\theta = \frac{-1}{\sqrt{(1-(\frac{cb'}{a'b'})^2)}} d(cb') \tag{1}$$

可见 cb' 接近 $a'b'$ 时，θ 对于 cb' 较敏感，较小的 cb' 波动会造成较大的 θ 波动。去掉 140 至 160 处的点，对数据进行线性拟合，如图 9 所示。

$$y = 0.9996x + 28.729, R^2 = 0.9999 \tag{2}$$

可以认为角度是线性变化的，每次转过的角度为 0.9996 度。

5.1.3 探测单元的距离的确定

我们首先根据附件二的吸收强度数据，得出扫描结果的大致图形轮廓，如图 2。X 射线所成像的最长部分应为椭圆长轴长度（此时 X 射线方向与长轴垂直）。从附件二中我们可以读出这个像所处的探测器单位元区间（即对应的探测器单位元数目），与此同时，从图 2 中可知椭圆长轴的实际长度。根据探测器单位元在探测器平面上等距分布，我们就可以得出探测器单位元之间的实际距离。

最终计算得，探测器单位之间得距离为 0.2761 mm。

5.1.4 旋转中心的确定

椭圆中心 O_2 与圆心 O_1 均绕转动中心 C 做半径为 r_2 与 r_1 的圆周运动（如图 10）。若以圆心为参照系，旋转中心绕 O_1 做半径为 r_1 的圆周运动；若以椭圆中心 O_2 为参照系，旋转中心绕 O_2 做半径为 r_2 的圆周运动。旋转中心必然满足同时在圆 O_2、O_1 上，因此旋转中心一定为两圆两交点中的一个。（如图 11）

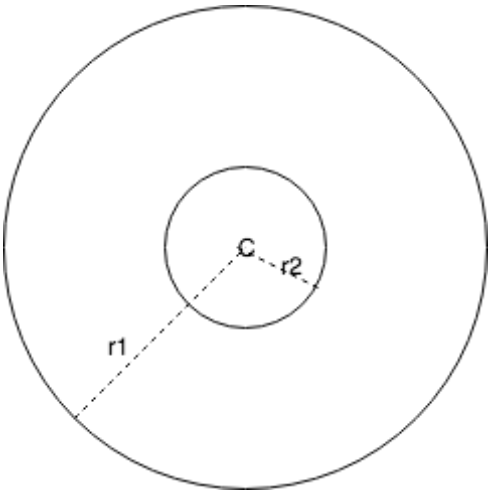

图 10

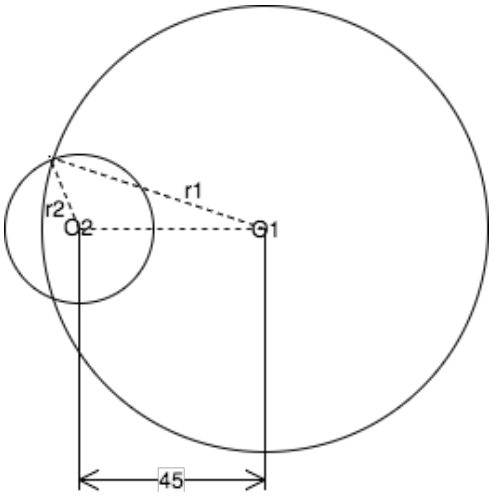

图 11

利用图 6 圆心和椭圆中心的轨迹可以求得 r_1 与 r_2。由之前得到的 180 个位置可知，探测器每次旋转 0.9996 度，共旋转了 180 次，可以认为图 6 中的轨迹为正弦函数的半个周期。因此可以求出两轨迹的幅值，即为半径 r_1、r_2。以椭圆中心 O_2 为原点，O_2、O_1 连线沿 O_2 至 O_1 方向为 x 轴，解出候选旋转点坐标为 $(-8.8439, -6.6130), (-8.8439, 6.6130)$。

若以像素作为单位，则约为 $(-32,-24)$ 与 $(-32,24)$。结合图 6 中圆心和椭圆中心的轨迹和图 9，两个正弦轨迹的初相约为-29.4 度。因此可以判断出旋转中心的位置为 $(-8.8439,-6.6130)$ 或用像素表示为 $(-32,-24)$。其位置如图 12。

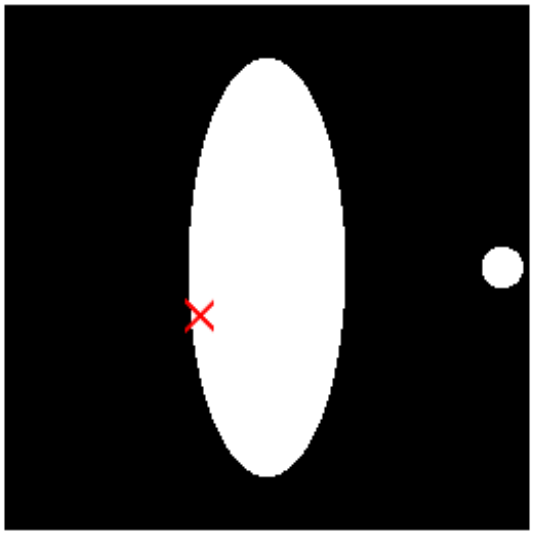

图 12　旋转中心

5.1.5 计算结果的验证

自此，我们得到了该 CT 系统进行扫描的几个基本参数：方向，探测器距离，旋转中心。可以使用这些参数对模板进行模拟扫描，将扫描结果与实际的结果作对比，以便验证我们的结果是正确的。加入参数的模拟扫描结果见图 13，实际的扫描结果见 14。

可以看出，加入参数的扫描结果与实际结果基本一致，证明我们所得到的参数的正确性。

5.2 从二维图像的一维投影进行图像重建

5.2.1 Radon 变换与 CT 系统成像

CT 系统成像的数学推导　根据材料，CT 系统在多个方向上，使用 X 射线对样品进行穿透，之后在每个方向上计算 X 射线的吸收程度，得到一个一维的扫描结果。

然而，由于我们仅能考察最终的探测器收到的信息，因此无法对这个方向上的吸收率分布有所认识，得到的扫描结果是一个累积结果。接下来我们将证明，这个吸收率的

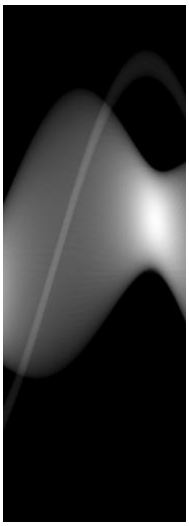

图 **13**　加入参数的模拟扫描结果

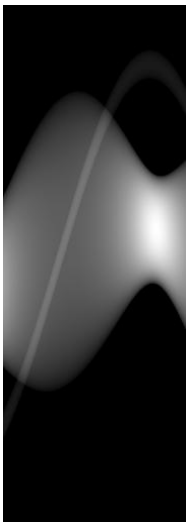

图 **14**　实际的扫描结果

累积结果是对于该样品吸收率分布在该方向上的积分。

X 射线在穿透厚度为 x_0 时，最终的射线强度为

$$I = I_0 \times e^{-\mu x_0} \tag{3}$$

由此，我们可以得到穿过不均匀材料的射线强度：

$$I = I_0 \times e^{-\int_0^{x_0} \mu(x)dx} \tag{4}$$

其中 $\mu(x)$ 为该材料的吸收率分布。这里可以看到，如果我们按照最终的吸收结果所推得的吸收率 μ_0，将是 $\mu(x)$ 在入射方向上的积分。

$$\mu_0 = \int_0^{x_0} \mu(x)dx \tag{5}$$

即证得在入射方向上吸收率的累积效果就是在该方向上的积分。

Radon 变换的描述　此外，我们再考察 Radon 变换。直角坐标系中的一条直线可以用其法线描述为 $x\cos\theta + y\sin\theta = \rho$。当 θ 保持一定时，表现的是一簇平行的曲线。对二维图像 $f(x,y)$ 的 Radon 变换是得到一个关于 ρ 和 θ 表达式，记为 $g(\rho,\theta)$。$g(\rho,\theta)$，即投影剖面上的任意一点 (ρ_j,θ_k) 的值由 $f(x,y)$ 沿直线 $x\cos\theta_k + y\sin\theta_k = \rho_j$ 的积分确定。即可得到 $g(\rho,\theta)$ 的表达式 [1]：

$$g(\rho,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy \tag{6}$$

在该表达式上，使用了冲激函数的特性，保证了这个积分仅在当前直线上进行，也就是沿该方向上的投影。

Radon 变换与 CT 系统成像 由上，我们能够比较清楚地发现所给材料中的 CT 系统成像实际上使用的就是 Radon 变换。样品吸收率的二维分布对应 Radon 变换中的二维图像 $f(x,y)$，所得的扫描结果对应 Radon 变换中的投影结果。因此，我们之后将样品吸收率的二维分布记为 $f(x,y)$，它与二维图像是等价的。

5.2.2 中心切片定理和滤波反投影

通过查阅资料，我们找到了一个与 Radon 变换十分相关的定理，中心切片定理（也叫做 Fourier 切片定理）。该定理表明，一个投影的 Fourier 变换，是得到该投影的区域的二维 Fourier 变换在该方向上的切片。即 [3]

$$G(\omega,\theta) = [F(u,v)]_{u=\omega\cos\theta, v=\omega\sin\theta} = F(\omega\cos\theta, \omega\sin\theta) \tag{7}$$

通过这个定理，我们可以比较方便地在频域内进行工作，并且推导出频域内的原二维图像（即 $f(x,y)$）的表达式：

$$f(x,y) = \int_0^{\pi} \left[\int_{-\infty}^{+\infty} |\omega| G(\omega,\theta) e^{j2\pi\omega\rho} d\omega \right] \tag{8}$$

观察到，在计算任意一点 (x,y) 的图像值（也就是吸收率）的时候，内部表达式计算的其实是一个加了滤波函数的 Fourier 逆变换。

这样，我们可以得到基本的求解 Radon 逆变换的步骤：

(1) 计算 180 个方向的投影的一维 Fourier 变换。

(2) 用滤波函数 $|\omega|$ 乘以步骤 1 所得结果。同时，为了得到更好的效果，在这一步一般还需加入一个窗口函数的处理。

(3) 得到第 2 步导致的每个滤波后的变换的一维 Fourier 逆变换。

(4) 对第 3 步得到的所有一维逆变换离散求和，得到 $f(x,y)$。

由于我们的数据均为离散的，因此 Fourier 变换与逆变换均使用 FFT 算法实现。

5.2.3 基于 Radon 逆变换的图像重建

我们使用 MATLAB 中的 iradon 函数进行 Radon 逆变换。它接受四个参数，投影数据，旋转角度，重建图像的内插方法和上一部分所述的滤波器。旋转角度和初相位在前面的篇幅中已经得到。为了避免重建图像输出到矩阵之外，我们对投影数据进行了扩展，相当于在 512 个探测器单元的两侧各加了 128 个检测不到吸收的探测器。这样就能完整地得到 Radon 逆变换之后地图像。

之后，我们对处理模板投影数据得到的图像进行二次标定。标定长度放缩比，灰度放缩比，以及位置偏移量。这样就得到了针对该 CT 系统的图像重建通用方法，可以进行进一步的工作。

5.2.4 最终结果

图 15，图 16分别是使用上述通用方法重建的切片图。注意，由于有的样品的相对吸收率大于 1，无法直接转化为灰度图，均进行了归一化处理。因此这里的不同样品之间的吸收率无法通过图像本身进行比较，准确的吸收率结果保存在其他文件中。

图 **15**　附件二的重建

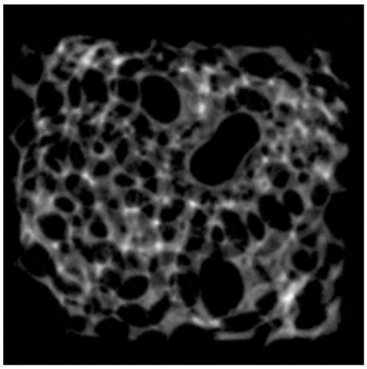

图 **16**　附件五的重建

另外，问题二和问题三要求的两个未知样品 10 个点的吸收率列入表 5.2.4。

位置		吸收率	
x	y	附件 3 的样品	附件 5 的样品
10	18	0	0.048977
34.5	25	0.99913	2.817
43.5	33	0	7.2033
45	75.5	1.2006	0
48.5	55.5	1.0686	0.17114
50	75.5	1.5055	3.3366
56	76.5	1.3077	6.1834
65.5	37	0	0
79.5	18	0	7.3756
98.5	43.5	0	0.047672

5.3 参数标定的精度和稳定性

利用问题一中获得的参数标定对模板进行复原。图 17，图 18，图 19分别展示了复原后的图像，原图以及两图相减后“剩余”的图像。

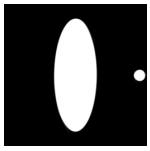

图 17　复原图像

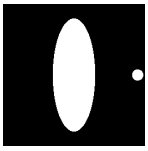

图 18　原图

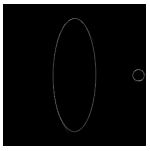

图 19　相减

可见复原整体是理想的，但在边缘处会出现残余。定义如下指标来定量衡量精确性：

$$\eta = \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}|f(x,y)-r(x,y)|dxdy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)dxdy} \tag{9}$$

其直观上的意义是两者相差的剩余部分的吸收率之和占原图的吸收率之和的的百分比，计算的到 $\eta = 0.3474\%$。该复原较为理想。

观察到复原后的图像边缘明显比原图光滑，这是因为附件 2 的信息量更大，复原过程中会自动地补全粗糙的边界。从复原图与原图相差的剩余部分明显可以看出这样的边界，这是 η 值产生的原因之一。另外的误差可能来源于图像边界提取。在边界处理时，去掉了一些偏离曲线的点。这些点事实上也是边界点，但被分类函数错误地分到了其它曲线上。若能改善边界的特征，则可以提高本部分的精度。此外由于数据只取 4 位有效数字，复原过程中也会产生截断误差。另外由于数据的离散性，在旋转等操作中，也会造成一定的误差。

“一个数学模型成为稳定的，是指即使这个模型不完全准确，由其导出的结果也是正确的。”[4] 在对这个 CT 系统标定过程中，我们在测量角度和距离的时候都用了较多的近似，也意味着建立的模型是不完全准确的，但是从图 13 和图 14 中根据参数复原和原图的比较中可以看到，我们最终得到的是基本正确的结果，也印证了这个模型是稳定的。

5.4 设计新模板

我们根据之前处理模板投影数据的经验，发现了以下几个产生误差的原因：

(1) 对于椭圆与圆组合的模板，其扫描图像由于在某些角度有交叉的部分，相互干扰，使得直接分离图像变得较为困难。

(2) 边界提取上，由于交叉点的存在，通过梯度峰值提取出的边界中不理想以及偏移的像素点会较多。

(3) 对提取后的边界降噪处理时，由于偏移的像素点会较多，且偏移量不明显，会去掉较多的数据点。

同时，为了合理标定参数，我们又必须要求有下面的几个点来标定所需的参数：

(1) 通过线段或者点的多个方向的投影来确定旋转的角度。

(2) 通过至少两个点的投影所形成的正弦函数曲线的振幅确定旋转的位置。

(3) 通过至少一个长度对应关系确定探测器单元之间的距离。

因此，为了规避以上的问题，并且同样能标定出足够的参数，我们设计了新的空心圆模板，如图 20 所示。

对其扫描图像进行模拟得到图 21。利用之前边界提取的算法，提取出四条边界如图 22，图 23，图 24，图 25 所示：

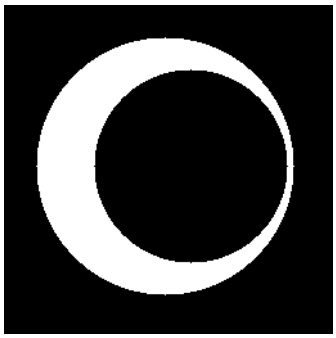

图 **20**　空心圆的模板

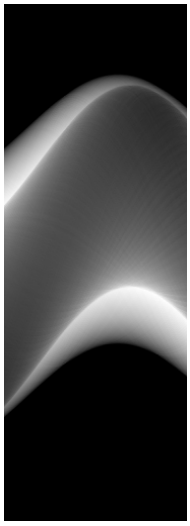

图 **21**　新模板的扫描结果

图 **22**　边界 **1**

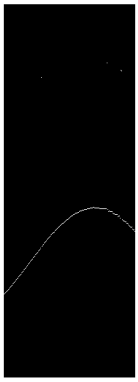

图 **23**　边界 **2**

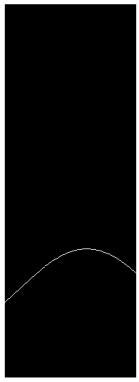

图 **24**　边界 **3**

图 **25**　边界 **4**

可见，由于新的空心圆模板的两条正弦轨迹并不会相交产生干扰，基于梯度极值点

的边界判别效果会提高，尤其体现在外侧正弦曲线组上（图 22 图和 24）。而内侧空心圆部分产生的正弦曲线组尽管有一些噪点，但这些噪点具有数量少且远离趋势线的特点，有利于进一步降噪效率的提高。

事实上，鉴于外侧正弦曲线组边界提取几乎无噪点，可以直接基于该曲线，在不删减点的情况下利用三角函数的相关特性进行 180 个方位的计算，提高这部分的精度。而内侧空心圆部分产生的正弦曲线组则可以在降噪插值后计算内侧空心圆圆心的轨迹曲线，在与大圆圆心轨迹曲线对比后用之前提到的方法求出旋转中心的位置与探测器单元之间的距离。

六、 模型的评价及改进

6.1 对于模型的评价

针对问题一，我们通过观察数据的扫描图像以及寻找图形之间的几何关系，完成了 CT 系统相关参数的标定。这其中运用了很多几何以及三角函数的知识，将一个原本需要处理大量数据的问题简洁明了地转化成直观形象的问题，并且给出了合理的解释与解答。

针对问题二与问题三，联想到图像重建的问题，我们找到数学中关于二维图像的 Radon 变换来解决这次的问题，并简要证明了 CT 系统成像中的扫描操作实际上是对于样品二维吸收率分布的某一方向的投影。结合中心切片定理，得到 Radon 逆变换的算法步骤，使用该方法，我们较为准确且较为简化地重建了附件 2 和附件 5 中未知样品的吸收率分布，并进一步得到了要求的 10 个点的吸收率。图像重建的结果也表明使用 Radon 变换的方法效果极佳。

针对问题四，我们首先分析了之前处理模板中遇到的一些困难和缺陷，即处理边界时交叉的部分会导致边界提取较大的误差。基于这一点，我们设计了空心圆的样品模板，规避了外围边界交叉的问题，能够更简单快速地标定相应参数，从而达到更好的效果。

6.2 对于模型的改进

本文中的模型也具有一定的局限性，比如现实中很难保证 X 射线始终与探测器平面垂直，X 射线的强度各处始终相同、各个探测器单元的敏感程度完全相同，实际待测对象对 X 存在折射和反射现象，扫描图像存在噪声等。这些问题可以通过对不同区域分别分析，对不同探测器单元赋以权重，对扫描图像降噪等方法进行进一步解决，本文不再赘述。

七、 参考文献

[1] Rafael C. Gonzalez, 数字图像处理 [M]. 北京: 电子工业出版社, 2014. 104-111

[2] John H. Mathews, Kurtis D. Fink. 数值方法 [M]. 北京: 电子工业出版社, 2010. 154-164

[3] 高上凯. 医学成像系统 [M]. 北京: 清华大学出版社有限公司, 2000. 46-50

[4] Meerschaert, M. M. 数学建模与方法分析 [M]. 北京: 机械工业出版社, 2015. 9

附录 A

1.1 apps.py

在本项目中的常用函数的编写。

```
from xlrd import open_workbook
from xlrd import XL_CELL_TEXT, XL_CELL_NUMBER, XL_CELL_DATE, XL_CELL_BOOLEAN
import numpy as np
import math
from PIL import Image

# 将 xls 文件中的工作表转化为 numpy 的数组对象
def sheet_to_array(
      filename,
      sheet_number,
      first_col=0,
      last_col=None,
      header=True):
   DEBUG = False
   # sheet
   book = open_workbook(filename)
   sheet0 = book.sheet_by_index(sheet_number)
   rows = sheet0.nrows
   # cols
   if not last_col:
      last_col = sheet0.ncols
   if first_col > last_col:
      raise Exception("First column must be smaller than last column!")
   cols = [col for col in range(first_col, last_col + 1)]
   # rows
   skip = 0
   if header:
      skip = 1
   data = np.empty([len(cols), rows - skip])

   for row in range(skip, sheet0.nrows):
      row_values = sheet0.row(row)
      for col, cell in enumerate(row_values):
         if DEBUG and row < 2:
            print(row, col, cell.ctype, cell.value, '\n')
         if col in cols and cell.ctype == XL_CELL_NUMBER:
            data[col - first_col, row - skip] = cell.value
   return np.transpose(data)

# 矩阵对象和图像对象的相互转换
def mat2img(mat):
```

```
    return Image.fromarray((mat / np.amax(mat) * 255).astype(np.uint8))

def img2mat(img):
    return np.array(img)

# 找到数组中相同值的中间坐标
def find_mid_index(value, li):
    first_index = li.index(value)
    last_index = first_index + 1
    n = len(li)
    while(li[last_index] == value and last_index < n):
        last_index += 1
    return int((last_index + first_index) / 2)
```

1.2 init.py

导入附件中的数据，并进行初步的处理。

```
import apps
import numpy as np

# 读入工作表文件的各个文件
FILE_NAME = './A_attachment.xls'

sheet_01 = apps.sheet_to_array(
    FILE_NAME,
    0,
    first_col=0,
    last_col=255,
    header=False)
sheet_02 = apps.sheet_to_array(
    FILE_NAME,
    1,
    first_col=0,
    last_col=179,
    header=False)
sheet_03 = apps.sheet_to_array(
    FILE_NAME,
    2,
    first_col=0,
    last_col=179,
    header=False)
sheet_04 = apps.sheet_to_array(
    FILE_NAME,
    3,
    first_col=0,
```

```
    last_col=1,
    header=False)
sheet_05 = apps.sheet_to_array(
    FILE_NAME,
    4,
    first_col=0,
    last_col=179,
    header=False)

# 将数据存入可交换数据格式便于之后的使用
np.savetxt('outputs/sheet_02.csv', sheet_02, delimiter=',')
np.savetxt('outputs/sheet_03.csv', sheet_03, delimiter=',')
np.savetxt('outputs/sheet_05.csv', sheet_05, delimiter=',')
```

1.3 std.py

模拟过程，对模板进行无偏差的 CT 成像。

```
# from init import sheet_01
import numpy as np
import apps
from PIL import Image

# 定义分辨率

sheet_01 = np.loadtxt('inputs/new_model_02.csv', delimiter=',')

res = 10

# 定义画板
board = np.zeros([180, 512 * res])

# 定义偏离值
offset_x = -24
offset_y = 32
sq = np.zeros([512, 512])
sq[128+offset_x:offset_x+256+128, 128+offset_y:offset_y+256+128] = sheet_01
tmp_img = apps.mat2img(sq).resize([512 * res, 512 * res])

# 模拟扫描
phase = 29.4056213110678
angle = 0.9996
tmp_img = tmp_img.rotate(-90 + phase)
for j in range(180):
  tmp_img = tmp_img.rotate(angle)
  tmp_mat = apps.img2mat(tmp_img)
```

```
  board[j] = tmp_mat.sum(1)

img_cat = np.transpose(board)[64*res:448*res, :]

img = apps.mat2img(img_cat).resize([180, 512])
img.save('outputs/new_model_02.png')
np.savetxt('outputs/new_model_02.csv', apps.img2mat(img), delimiter=',')
```

1.4 ellipse.py

对模板所成实际的像中的椭圆部分进行处理，得到长轴和短轴的长度数据和坐标数据。

```
from init import sheet_02
import numpy as np
import apps
from PIL import Image

# 找到椭圆的投影
vexs = []
for i in range(180):
  tmp = []
  for j in range(1, 512):
    if (sheet_02[j, i] != 0 and sheet_02[j-1, i] == 0):
      tmp.append(j)
    elif (sheet_02[j, i] == 0 and sheet_02[j-1, i] != 0):
      tmp.append(j-1)
  if (len(tmp)>2):
    if (tmp[1] - tmp[0] > tmp[3] - tmp[2]):
      tmp.remove(tmp[3])
      tmp.remove(tmp[2])
    else:
      tmp.remove(tmp[0])
      tmp.remove(tmp[0])
  vexs.append(list(tmp))

# 计算每个投影的最大长度
length = []
for item in vexs:
  length.append((item[1] - item[0]))

# 计算长轴与短轴的坐标和数值
max_axis = max(length)
min_axis = min(length)
```

```
max_index = apps.find_mid_index(max_axis, length)
min_index = apps.find_mid_index(min_axis, length)

print(max_axis, min_axis)
print(max_index, min_index)
```

1.5 angles.py

根据几何关系对 180 个方向的角度进行标定。

```
import numpy as np
import apps
from PIL import Image
from math import acos, pi

# 取得放缩比
RATIO = ((288 / 80) + (107 / 30)) / 2

# 读入处理好的圆心及椭圆中心的轨迹
cir_data = np.loadtxt('inputs/SplinedRound.csv', delimiter=',')
ell_data = np.loadtxt('inputs/SplinedEllipse.csv', delimiter=',')

# 对轨迹进行坐标化，存入两个一维数组中
cir_index = []
ell_index = []
for i in range(180):
  cir_index.append(list(cir_data[:, i]).index(max(cir_data[:, i])))
  ell_index.append(list(ell_data[:, i]).index(max(ell_data[:, i])))

# 计算轨迹差值
diff = (np.array(cir_index) - np.array(ell_index))

# 计算180个方向的角度，并加入修正因子
angles = []
no = []
yes = []
for i in range(180):
  cos_value = diff[i] / 45 / RATIO / 1.01085271318
  no.append(cos_value * 1.01085271318)
  yes.append(cos_value)
  angle_tmp = acos(cos_value) / pi * 180
  if (i == 0):
    angles.append(angle_tmp)
  elif (angle_tmp < angles[i-1]):
    angles.append(360 - angle_tmp)
  else:
```

```
    angles.append(angle_tmp)

# 存储角度数据
np.savetxt('outputs/angles.csv', angles, delimiter=',')
np.savetxt('outputs/no.csv', no, delimiter=',')
np.savetxt('outputs/yes.csv', yes, delimiter=',')
```

1.6 amp_acquire.m

```
%本程序求出圆心轨迹与椭圆中心轨迹的振幅，在此基础上求解旋转中心的位置
clear all
I1 = csvread('SplinedEllipse.csv');
I2 = csvread('SplinedRound.csv');

y1 =[];
x1 =[];
y2 =[];
x2 =[];

for i=1:180
      y1 = [y1;find(I1(:,i)==1)];
      x1 = [x1;i];
end

for i=1:180
      y2 = [y2;find(I2(:,i)==1)];
      x2 = [x2;i];
end

k1 = find(y1==min(y1));
k2 = find(y2==min(y2));

c1 = max((y1(1)+y1(180))/2-y1(k1));
c2 = max((y2(1)+y2(180))/2-y2(k2));

Ratio = 1.01085271318*(((288/80) + (107/30))/2);
radius1 = c1/Ratio;
radius2 = c2/Ratio;
%解方程
syms x y
S1=x^2+y^2-radius1^2;
S2=(x-45)^2+y^2-radius2^2;
[x,y]=solve(S1,S2);

%以现实中的距离表示
```

```
vpa(x)
vpa(y)
%以像素表示
round(Ratio*vpa(x))
round(Ratio*vpa(y))
```

1.7 edge_acquire.m

```
%本程序判断并获得附录2图像的边界点，store中1、2、3、4列分别存储4条边界线上点的坐标。
clear all;
%读取附录2中的图像
I = xlsread('A2.xlsx');
%获取图像的梯度信息
[FX,FY] = gradient(I);

store = zeros(180,4);

%获取边界信息
for i=1:180
store(i,:) = GetPeaks(FY,i);
end

Image = zeros(512,180);
%获取圆边界
for i=1:180
   Image(store(i,1),i) = 0;
   Image(store(i,2),i) = 1;
   Image(store(i,3),i) = 0;
   Image(store(i,4),i) = 1;
end

%获取椭圆边界
%for i=1:180
   %Image(store(i,1),i) = 1;
   %Image(store(i,2),i) = 0;
   %Image(store(i,3),i) = 1;
   %Image(store(i,4),i) = 0;
%end

%显示获得的图形
figure;
imshow(Image);


%导出圆心、椭圆中心轨迹坐标
```

```
csvwrite('Image_round.csv',Image);
%csvwrite('Image_ellipse.csv',Image);
```

1.8 FindProperPeaks.m

```
%FindProperPeaks函数筛选出边界点位置并返回它们
function location = FindProperPeaks(pks,locs)
   a = [-pks';locs'];
   a =a';
   aa = sortrows(a,1);
   location(1) = aa(1,2);
   location(2) = aa(2,2);
end
```

1.9 FlteringFunc.m

```
%本程序找出数据点连线的极值点（几何意义为偏离趋势线的点）进行第一次降噪
function FlteredI = FlteringFunc(I)

store = zeros(180,1);
for i=1:180
   store(i) = find(I(:,i)==1);
end

[pks,locs] = findpeaks(store);
locs1 = locs;

[pks,locs] = findpeaks(-1*store);
locs2 = locs;

locs = [locs1;locs2];

I(:,locs) = zeros();
FlteredI = I;
end
```

1.10 getParas.m

```
M = csvread('outputs/sheet_02.csv');

M = [zeros(128, 180); M ;zeros(128, 180)];
```

```
phase = 29.4056213110678;
angle = 0.9996;
im_I = iradon(M, [phase:angle:angle*179 + phase], 'cubic', 'Hamming');
imwrite(im_I, 'outputs/im_I.png')

im_B = im2bw(im_I,0.2);

% Get the ratio_de
ratio_de = sum(im_B) / sum(im_I);

% Get the index
cols = sum(im_B');
flag = 0;
for i = 1:length(cols)
   if flag == 0 && (cols(i)>0)
      index_x = i;
      index_1 = i;
      flag = 1;
   else if flag == 1 && (cols(i) == 0)
         index_x = (index_x + i - 1) / 2;
         index_2 = i;
         break;
      end
   end
end

rows = sum(im_B);
flag = 0;
for i = 1:length(rows)
   if flag == 0 && (rows(i)>0)
      index_y = i;
      flag = 1;
   else if flag == 1 && (rows(i) == 0)
         index_y = (index_y + i - 1) / 2;
         break;
      end
   end
end

ratio_len = (index_2 - index_1) / 80;
x = int16(index_x - 50 * ratio_len):int16(index_x + 50 * ratio_len);
y = int16(index_y - 50 * ratio_len):int16(index_y + 50 * ratio_len);
```

1.11 GetPeaks.m

```
%GetPeaks可以获取每一列中的较大的四个极值点位置，返回store储存这些位置
%这些位置即为图像的边界信息
function store = GetPeaks(fy,i)

fy_negative = -1*fy;

[pks,locs] = findpeaks(fy(:,i));
location = FindProperPeaks(pks,locs);
store(1)=location(1);
store(2)=location(2);

[pks,locs] = findpeaks(fy_negative(:,i));
location = FindProperPeaks(pks,locs);
store(3)=location(1);
store(4)=location(2);

end
```

1.12 getResult.m

```
run getParas.m

M = csvread('outputs/sheet_05.csv');
M = [zeros(128, 180); M ;zeros(128, 180)];

phase = 29.4056213110678;
im_I = iradon(M, [phase:0.9996:0.9996*179 + phase], 'cubic', 'Hamming');
im_Res = im_I(x, y) * ratio_de;
im_Res = im_Res .* (im_Res>0.05);
im_Res = imresize(im_Res, [256 256]);
%csvwrite('outputs/result_5.csv', im_Res)
imshow(im_Res/max(max(im_Res)))
```

1.13 interpolation_of_round_ellipse.m

```
%本程序对edge_acquire中获得的边缘进行两次降噪，并利用三次样条插值补全降噪后噪点处的数据
%本程序产生记录模板圆的圆心部分轨迹和椭圆形部分中心轨迹的csv文件
%为SplinedRound.csv与SplinedEllipse.csv。

%FlteringFunc(I)函数找出梯度较大的点（偏离轨迹较远）并去除
%Image_ellipse1.csv、Image_ellipse2.csv、Image_round1.csv、
%Image_round1.csv分别存储每一条边界线上点的位置

I = csvread('Image_ellipse2.csv');
```

```
ImageUp = FlteringFunc(I);

I = csvread('Image_ellipse1.csv');
ImageDown = FlteringFunc(I);

I = ImageUp + ImageDown;

%上下边际线靠的太近时需要删除，第二次降噪
for i = 1:180
   if sum(I(:,i)) == 2
     x=find(I(:,i)==1);
     if x(2)-x(1) <= 10
       I(:,i)=zeros();
     end
   end
end
%合成图像若某一列仅有一个点，无法获得圆心（或椭圆中心轨迹），需要删除
for i=1:180
   if sum(I(:,i)) ~= 2
      I(:,i)=zeros();
   end
end

ImageAve = zeros(512,180);
for i=1:180
   if sum(I(:,i)) == 2
      x=find(I(:,i)==1)
      ave = round(0.5*(x(1)+x(2))); %四舍五入
      ImageAve(ave,i) = 1;
   end
end

clear x
clear y
y =[];
x =[];
for i=1:180
   if sum(ImageAve(:,i)) ==1
      y = [y;find(ImageAve(:,i)==1)];
      x = [x;i];
   end
end
%三次样条插值，yy记录每一列的圆心（或椭圆中心）的纵坐标
yy = spline(x,y,[1:180]);
yy = round(yy)

ImageSplined = zeros(512,180);
```

```
for i=1:180
ImageSplined(yy(i),i)=1;
end

figure;
imshow(ImageSplined);
csvwrite('SplinedEllipse.csv',ImageSplined);
%圆心轨迹时写入文件
%csvwrite('SplinedRound.csv',ImageSplined);
```

附录 B

2.1 arccos 数据值表

注：下表为 180 个方向的 arccos 计算值，前行为修正前的原始值，后行为加入修正因子的修正值。

1	2	3	4	5	6	7	8	9	10
0.88062	0.86202	0.86202	0.84961	0.83721	0.83101	0.81860	0.80620	0.79380	0.78760
0.87117	0.85276	0.85276	0.84049	0.82822	0.82209	0.80982	0.79755	0.78528	0.77914
11	12	13	14	15	16	17	18	19	20
0.77519	0.76899	0.75039	0.73798	0.71938	0.71318	0.70698	0.69457	0.67597	0.66357
0.76687	0.76074	0.74233	0.73006	0.71166	0.70552	0.69939	0.68712	0.66871	0.65644
21	22	23	24	25	26	27	28	29	30
0.65116	0.65116	0.64496	0.62636	0.60155	0.58295	0.57054	0.55814	0.53953	0.52093
0.64417	0.64417	0.63804	0.61963	0.59509	0.57669	0.56442	0.55215	0.53374	0.51534
31	32	33	34	35	36	37	38	39	40
0.50853	0.49612	0.47752	0.46512	0.44651	0.43411	0.42171	0.40310	0.37829	0.36589
0.50307	0.49080	0.47239	0.46012	0.44172	0.42945	0.41718	0.39877	0.37423	0.36196
31	32	33	34	35	36	37	38	39	40
0.50853	0.49612	0.47752	0.46512	0.44651	0.43411	0.42171	0.40310	0.37829	0.36589
0.50307	0.49080	0.47239	0.46012	0.44172	0.42945	0.41718	0.39877	0.37423	0.36196
41	42	43	44	45	46	47	48	49	50
0.35349	0.33488	0.31628	0.30388	0.28527	0.26667	0.24806	0.22326	0.21085	0.19845
0.34969	0.33129	0.31288	0.30061	0.28221	0.26380	0.24540	0.22086	0.20859	0.19632
51	52	53	54	55	56	57	58	59	60
0.18605	0.16124	0.14884	0.13023	0.11163	0.09302	0.06822	0.04341	0.02481	0.01240
0.18405	0.15951	0.14724	0.12883	0.11043	0.09202	0.06748	0.04294	0.02454	0.01227
61	62	63	64	65	66	67	68	69	70
0.00000	-0.01240	-0.03101	-0.04961	-0.06202	-0.08682	-0.09922	-0.11783	-0.13643	-0.15504
0.00000	-0.01227	-0.03067	-0.04908	-0.06135	-0.08589	-0.09816	-0.11656	-0.13497	-0.15337
71	72	73	74	75	76	77	78	79	80

-0.17364	-0.18605	-0.20465	-0.21705	-0.24186	-0.25426	-0.27287	-0.29147	-0.31008	-0.32248
-0.17178	-0.18405	-0.20245	-0.21472	-0.23926	-0.25153	-0.26994	-0.28834	-0.30675	-0.31902
81	82	83	84	85	86	87	88	89	90
-0.34109	-0.35969	-0.37209	-0.39070	-0.40930	-0.42171	-0.44031	-0.45271	-0.46512	-0.48372
-0.33742	-0.35583	-0.36810	-0.38650	-0.40491	-0.41718	-0.43558	-0.44785	-0.46012	-0.47853
91	92	93	94	95	96	97	98	99	100
-0.49612	-0.51473	-0.52713	-0.54574	-0.55814	-0.57054	-0.58915	-0.60775	-0.62016	-0.62636
-0.49080	-0.50920	-0.52147	-0.53988	-0.55215	-0.56442	-0.58282	-0.60123	-0.61350	-0.61963
101	102	103	104	105	106	107	108	109	110
-0.63876	-0.65736	-0.66977	-0.68217	-0.69457	-0.70698	-0.71938	-0.73178	-0.74419	-0.75659
-0.63190	-0.65031	-0.66258	-0.67485	-0.68712	-0.69939	-0.71166	-0.72393	-0.73620	-0.74847
111	112	113	114	115	116	117	118	119	120
-0.76899	-0.77519	-0.79380	-0.80000	-0.81860	-0.81860	-0.83101	-0.84341	-0.84961	-0.86202
-0.76074	-0.76687	-0.78528	-0.79141	-0.80982	-0.80982	-0.82209	-0.83436	-0.84049	-0.85276
121	122	123	124	125	126	127	128	129	130
-0.86822	-0.88062	-0.88682	-0.89922	-0.90543	-0.91163	-0.91783	-0.92403	-0.93023	-0.93643
-0.85890	-0.87117	-0.87730	-0.88957	-0.89571	-0.90184	-0.90798	-0.91411	-0.92025	-0.92638
131	132	133	134	135	136	137	138	139	140
-0.94264	-0.95504	-0.96124	-0.96124	-0.96744	-0.96744	-0.97984	-0.97984	-0.98605	-0.99225
-0.93252	-0.94479	-0.95092	-0.95092	-0.95706	-0.95706	-0.96933	-0.96933	-0.97546	-0.98160
141	142	143	144	145	146	147	148	149	150
-0.99225	-0.99225	-0.99845	-1.00465	-1.00465	-1.00465	-1.01085	-1.00465	-1.00465	-1.01085
-0.98160	-0.98160	-0.98773	-0.99387	-0.99387	-0.99387	-1.00000	-0.99387	-0.99387	-1.00000
151	152	153	154	155	156	157	158	159	160
-1.01085	-1.01085	-1.01085	-1.00465	-1.01085	-1.00465	-1.00465	-0.99845	-1.00465	-0.99845
-1.00000	-1.00000	-1.00000	-0.99387	-1.00000	-0.99387	-0.99387	-0.98773	-0.99387	-0.98773
161	162	163	164	165	166	167	168	169	170
-0.99225	-0.99225	-0.98605	-0.98605	-0.97984	-0.97364	-0.97364	-0.96744	-0.96124	-0.95504
-0.98160	-0.98160	-0.97546	-0.97546	-0.96933	-0.96319	-0.96319	-0.95706	-0.95092	-0.94479
171	172	173	174	175	176	177	178	179	180
-0.94884	-0.94264	-0.93643	-0.93643	-0.92403	-0.91783	-0.90543	-0.89922	-0.89302	-0.88682
-0.93865	-0.93252	-0.92638	-0.92638	-0.91411	-0.90798	-0.89571	-0.88957	-0.88344	-0.87730