



单位代码: 10006

学 号: 14171002

分 类 号: TP391.4

北京航空航天大学
B E I H A N G U N I V E R S I T Y

毕业设计(论文)

锥形束 CT 快速重建算法的技术研究

学 院 名 称	<u>仪器科学与光电工程学院</u>
专 业 名 称	<u>遥感科学与技术</u>
学 生 姓 名	<u>马煜华</u>
指 导 教 师	<u>周付根 教授</u>

2018 年 6 月

北京航空航天大学

本科生毕业设计（论文）任务书

I、毕业设计（论文）题目：

中文：锥形束 CT 快速重建算法的技术研究

英文：The Technical Research of Fast Reconstruction Algorithms of CBCT

II、毕业设计（论文）使用的原始资料（数据）及设计技术要求：

锥形束 CT (Cone Beam CT , CBCT)是一种医学成像技术。这种技术利用锥形 X 光射线对物体进行旋转采集得到多角度下的二维投影图，并结合重建算法，利用二维投影图进行三维重建，得到物体的三维断层图像。这种技术在牙科手术，血管造影，放射治疗等领域有着广泛的应用。相应的成像产品不仅受到国内外研究人员的广泛关注，而且很多巨头企业和初创公司也越来越多的开展这方面产品的研发。

在临床的应用中，CBCT 的快速重建算法是整个产品的核心。快速重建使得在临床应用中，医生能够更快地开展手术，开始放射治疗和进行血管造影等。

本课题针对现有的 CBCT 快速重建算法开展研究，旨在调研和实现至少两种重建算法，分析和比较这些算法的性能，包括重建时间，重建质量和抗噪能力，并争取能够有所改进。

原始资料包括多组三维 CT 图像和二维数字重建影像（DRR）的生成工具等，其中 DRR 的生成工具可以从三维 CT 图像中生成二维投影图，用于模拟旋转采集过程。

III、毕业设计（论文）工作内容：

- 1、阅读国内外参考文献，综述课题的背景、最新进展及应用；
- 2、学习有关 CBCT 的基础知识、软件工具的使用方法以及图像数据预处理方法；
- 3、研究和实现至少两种重建算法；
- 4、对比两种算法的重建效果，并争取进行改进；
- 5、总结实验数据，完成毕业论文的撰写。

IV、主要参考资料：

1. S Rit, M Vila Oliva et al. The Reconstruction Toolkit, an open-source cone-beam CT reconstruction toolkit based on the Insight Toolkit (ITK). [J]. Journal of Physics: Conference Series, 2014.
2. D Maret, N Telmon, OA Peters et al. Effect of voxel size on the accuracy of 3D reconstructions with cone beam CT. [J]. DentoMaxilloFacial Radiology, 2012.

仪器科学与光电工程 学院（系） 遥感科学与技术 专业类
141717 班

学生 马煜华

毕业设计（论文）时间： 2018 年 1 月 12 日至 2018 年 6 月 15 日

答辩时间： 2018 年 6 月 15 日

成 绩：

指导教师： 周付根

兼职教师或答疑教师（并指出所负责部分）：

_____系（教研室） 主任（签字）： _____

注：任务书应该附在已完成的毕业设计（论文）的首页

本人声明

我声明，本论文及其研究工作是由本人在导师指导下独立完成的，在完成论文时所利用的一切资料均已在参考文献中列出。

作者：马煜华

签字：

时间：2018 年 6 月



锥形束 CT 快速重建算法的技术研究

学 生：马煜华

指导老师：周付根

摘要

锥形束 CT 将射线以锥束状投射到物体上，并利用投影图对物体进行三维重建。这种成像方式在放射肿瘤治疗的摆位验证方面有重要应用。研究锥形束 CT 的快速重建方法对推动其应用具有重要意义。锥形束 CT 三维重建的常用算法有迭代类算法和反投影重建算法，其中迭代类算法虽然精度高但重建速度很慢，而反投影重建算法的重建精度能够满足使用需要，同时具有适中的重建速度。如何提升反投影速度是快速算法研究的重点。本文在介绍平行束、扇形束和锥形束重建的基础上，分析了 CT 重建的数学原理，着重分析锥形束 CT 重建的原理及 FDK 算法。随后对体素驱动、像素驱动、距离驱动三种反投影实现途径的 FDK 算法进行分析比较，编程实现了距离驱动的 FDK 算法，并对其边界点选取方案进行改进，在重建图像质量不变的情况下提高了重建速度。

关键词： 三维重建，锥形束 CT，反投影重建，FDK，距离驱动



Research on Technique of Fast Reconstruction of Cone-beam CT

Author: Yuhua Ma

Tutor: Fugen Zhou

Abstract

Cone beam computed tomography (or CBCT) projects cone-shape X-rays onto the objects and reconstructs the object using projected images. This imaging technique plays an important role in position verification of tumor radiotherapy. Studying the rapid imaging method of cone beam CT is of great significance for promoting its applications. The common algorithms for 3D reconstruction of cone beam CT include iterative reconstruction algorithm and back-projection reconstruction algorithm. Iterative reconstruction algorithm is high in accuracy, but the reconstruction is slow while the reconstruction accuracy of back-projection reconstruction algorithm can meet the needs of use while having a moderate reconstruction speed. How to improve the back-projection speed is the focus of fast algorithm research. This thesis analyzes the mathematical principle of CT reconstruction based on the introduction of parallel beam, fan beam and cone beam reconstruction. The principle of cone beam CT reconstruction and the FDK algorithm are emphatically analyzed. Then three back projection methods of FDK algorithm are discussed which are Voxel-driven method, Pixel-driven method and Distance-driven method. The Distance-driven method with no open source code is implemented using C++, based on which improvement is made in the selection of voxels' boundary points. The result shows that the reconstruction speed of optimized Distance-driven algorithm is improved while the image quality remains unchanged.

Key words: Three-dimensional reconstruction, CBCT, back-projection reconstruction, FDK, Distance-driven



目录

1	绪论.....	1
1.1	课题的背景与意义.....	1
1.2	国内外研究现状.....	2
1.2.1	解析算法.....	2
1.2.2	迭代算法.....	4
1.2.3	三维重建算法的 GPU 加速.....	5
1.3	锥形束 CT 近似重建算法存在的不足.....	5
1.4	研究方案.....	6
1.4.1	数据准备及工作环境.....	6
1.4.2	研究思路.....	7
1.4.3	研究方法.....	7
1.5	论文结构.....	8
2	CT 图像重建的数学原理.....	9
2.1	投影与反投影.....	9
2.1.1	投影 (Radon) 变换.....	9
2.1.2	反投影.....	10
2.2	中心切片定理.....	11
2.3	平行束与斜坡滤波.....	12
2.4	扇形束与余弦加权.....	14
2.5	本章小结.....	15
3	锥形束 CT 重建算法.....	17
3.1	锥形束 CT 几何模型及 Radon 变换.....	17
3.1.1	锥形束几何模型.....	17
3.1.2	Radon 变换.....	17
3.2	FDK 算法分析.....	18
3.3	反投影驱动方法.....	20



3.3.1 体素驱动方法(Voxel-driven)	20
3.3.2 像素驱动方法(Pixel-driven)	21
3.3.3 距离驱动方法(Distance-driven)	24
3.4 本章小结	25
4 反投影算法的实现	26
4.1 像素驱动与体素驱动算法	26
4.2 二维距离驱动算法	26
4.2.1 二维距离驱动算法的实现	26
4.2.2 算法性能分析	29
4.3 三维距离驱动算法	30
4.3.1 三维距离驱动算法实现	30
4.3.2 距离驱动算法的改进	35
4.3.3 距离驱动算法的 GPU 加速	37
4.4 本章小结	40
5 算法性能分析	42
5.1 重建速度	42
5.2 重建精度	44
5.3 算法鲁棒性	49
5.4 本章小结	49
结论	50
致谢	52
参考文献	53
附录	55
附录 A1. 仿真头模型及仿真数据的产生	55
附录 A2. 数字图像质量评估方法	57
附录 A3. 实验环境与重建参数	58
附录 B1. 投影数据	59



附录 B2. 反投影算法误差分布	61
附录 B3. 反投影算法鲁棒性测试	63
附录 B4. 反投影算法性能参数汇总	64

1 绪论

1.1 课题的背景与意义

X 射线由伦琴 (W. C. Roentgen) 于 1895 年发现, 自此学术界与工程界对这种射线的研究从未停歇。X 射线成像技术在临床医学领域的应用极大地促进了诊疗技术的发展。如今, 超声、CT 以及核磁 (MRI) 是医学中最为常见的影像学成像技术, 在临床中根据各自的成像特点, 在疾病的诊断方案的选择, 术后效果评估及定期伴随治疗等环节发挥着各自的优势。

计算机断层扫描 (CT) 成像技术由 X 射线成像技术发展而来, 能够很好地呈现人体内部软组织的影像。与 X 射线所成的透视图像不同, CT 的成像方式为断层成像, 其成像质量在密度分辨率和空间分辨率上与早期的 X 射线诊断技术相比有极大的提升^[1]。但 CT 成像亦存在诸多不足, 如空间分辨率存在各向异性, 以及因采集数据耗时较多而很难进行动态研究等。

随着科学技术的快速发展, 研究者们以二维 CT 为依据提出了基于平板探测器 (FPD) 的三维锥形束 CT (CBCT) 成像方式。CBCT 与传统 CT 的成像方法有极大的不同, 它将射线以锥束状投射到物体上, 可对物体某一部位进行整体成像, 从而解决了 CT 成像中空间分辨率各向异性的问题。此外, 由于采用了锥形束放射源和大面积 FPD 来采集物体各角度的投影数据, CBCT 大大提高了 X 射线的使用效率, 扫描速度也有了相应的提升。

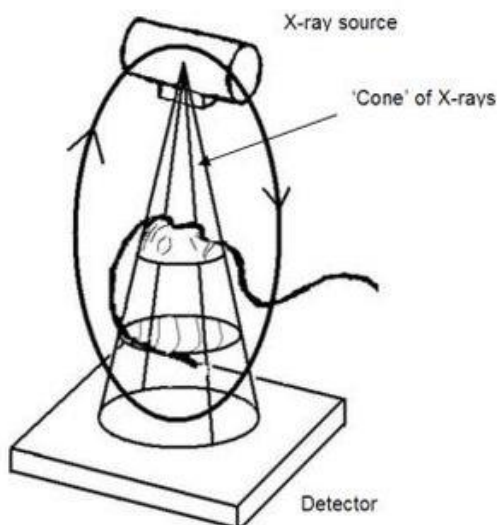


图 1.1 锥形束 CT 原理成像示意图



目前, CBCT 已逐步走向牙科疾病诊断、影像引导放射治疗、术中导航和植入计划等临床应用^[2]。尤其在呼吸运动跟踪成像方面, CBCT 具有重大的应用意义, 扫描时间的缩短使得运动器官如心脏等进行瞬时数据采集变得可能。随着 FPD 的不断发展, 其探测单元的数量逐渐增多, CBCT 经过一次旋转扫描得到的扫描数据越来越大, 重建速度也相应减慢^[3]。因此, 如何进一步提高重建速度成为锥形束 CT 三维重建算法的研究热点。

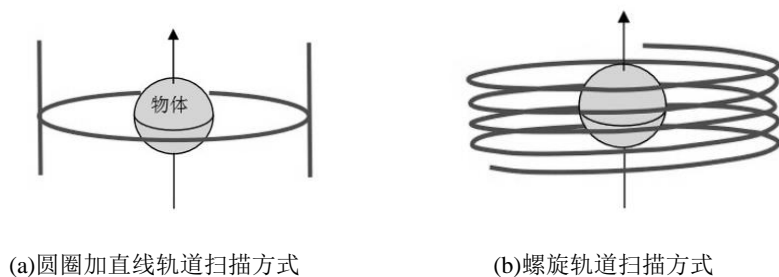
1.2 国内外研究现状

CBCT 三维重建算法目前分为解析和迭代两大类, 其中解析类算法又可以分为精确重建和近似重建。实际应用中快速的重建至关重要, 解析类算法重建速度较快因而应用广泛, 而迭代类算法迭代类算法走向临床还需要科研工作者的进一步努力, 迭代重建的原理使其比解析类算法需要更多的计算资源, 耗时巨大。近年来, 将并行度高的算法从 CPU 全部或部分移植到图形处理器 (GPU), 进行 CPU-GPU 协同计算从而大幅提升运行速度成为一种趋势。

1.2.1 解析算法

1、精确重建算法

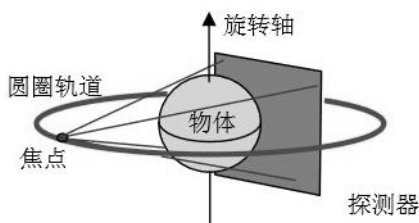
Radon 变换原理问世后, Kirillov 于 1961 年在 Radon 变换基础上推导出 N 维函数重建公式, 此方法因针对复数值投影数据而无法直接应用。数年后, Smith 提出锥形束投影数据的三维卷积公式, 首次将 Kirillov 的算法引入实空间。精确 CBCT 三维重建的条件由 Tuy 推导得出, 要求每一个与物体相交的平面都必须包含至少一个锥形束的焦点位置^[4]。Grangeat 推导出了通过径向一阶导数由锥形束投影数据向三维 Radon 变换的方法, 在三维 Radon 球面坐标系重排所采集坐标系的数据, 通过三维 Radon 数据可以通过逆变换重建。滤波反投影 (FBP) 由 Katsevich 于二十世纪初推导得出, 是效率极高的里程碑式算法。在 Katsevich 的工作基础上, Zou 和 Pan 实现了在滤波反投影基础上, 用更少的投影图重建满足精度要求的重建方法。此外, 早在 21 世纪初期已有完备的推导证明满足任意形状扫描轨迹的三维精确重建的可行性。

图 1.2 满足 Tuy 条件的轨道扫描方式示例^[4]

精确重建算法的应用并不普遍，其原因主要是 Tuy 条件和算法复杂度的限制。满足 Tuy 条件的轨道在制造工艺方面难度较大，且计算机的特性不方便进行精确重建算法的实现。

2、近似重建算法

近三十年来，在精确 CBCT 三维重建算法研究不断发展的同时，许多近似 CBCT 三维重建算法也不断涌现，并越来越多的出现在实际应用中。其中最著名的算法之一，是 1984 年 Feldkamp 等人提出的 FDK 算法^[5]。该算法可以通过扫描圆形轨迹来重建投影数据，圆形轨道扫描方式如图 1.3 所示。FDK 算法的主要思想是将圆锥束投影数据转换成多个扇形束投影数据，然后根据锥角进行余弦加权校正，通过逐行滤波处理校正投影数据。最后，通过三维反投影得到重构对象^[6]。由于该算法是由扇形束重建算法推导而来，当放射源锥角小于 10° 时，FDK 算法可重建出质量较好的 CT 图像。

图 1.3 不满足 Tuy 条件的圆圈轨道扫描方式^[4]

然而由于锥角较小这一要求，致使 FDK 算法在临床应用中有很大的限制。为了增大锥角和提高图像质量，许多学者对算法进行了改进；1999 年，Turbell 提出 P-FDK 算法，重排投影数据为三维平行束的投影数据，提高了重建速度；随后引出 T-FDK 算法，将截断后的投影数据进行重排重建，不但在计算方面更有效率，并且有效地减少了由于锥角增大而产生的伪影；2013 年，Guo 提出一种准确找到重建体素和投影像素之间的对应关系的方法，提高了重建结果的精度；2015 年，Jin 综合了最近邻插值和双线性插值的优点，对 FDK 算法中的插值方法进行了改进，有效的保证图像的边缘细节信息不缺失^[7]。

近似重建算法在 CBCT 重建算法中始终占据很重要的位置。与精确重建相比, 近似重建算法能够平衡医学应用领域中重建图像的质量和速度。当给出较小角度或适当角度时, 近似重建算法在图像分辨率、图像噪声和剂量效率等方面均胜过精确重建算法。

1.2.2 迭代算法

迭代算法基于离散模型, 然后在预置初始图像的数学迭代操作之后重建图像。即首先把图像离散化, 以体素作为最小单元, 然后利用解线性方程组的方法达到图像重建的目的, 图像重建迭代算法过程如图 1.4 所示。迭代重建算法基本分为两种, 代数重建法 (ART) 和统计迭代重建法 (SIR), 其中最为经典的当属 R.Gorden 等人提出的代数重建法和 P.Gilbert 提出的联合迭代重建法[2]。后来研究者们在前人工作的基础上发展了多种改进算法, 大致分为两类;

(1)在两种经典迭代算法基础上对其进行格式修正, 例如 Eggerment 等人将 Kacmarz 代数迭代算法改进为固定迭代块, 随后 Y.Censor与G.T.Herman 在此基础上将可变迭代块应用其中, 将经典迭代重建算法形成统一的框架。

(2) R.L.Kashyap和M.C.Mittal 提出了一种基于优化理论的统计迭代算法, 并将 CT 图像重建作为参数估计问题。

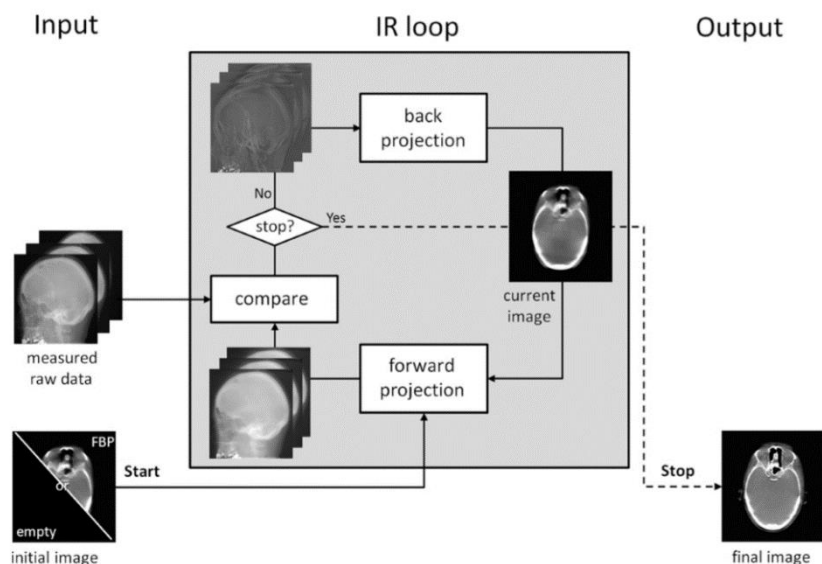


图 1.4 图像重建迭代算法过程示意图^[8]

代数迭代算法由以下三个步骤来实现: 投影、投影数据的修正和反投影, 其中投影与反投影占据重建总耗时的绝大部分^[9]。随着计算机的不断发展, 迭代算法的重建速度会有很大的提升, 许多提高迭代效率的算法也随之发展, 代数迭代重建算法已逐渐趋向



应用。

1.2.3 三维重建算法的 GPU 加速

医学图像的三维重建过程在微处理器上进行,设计微处理器的两个主要方向为多核方向和众核方向。多核方向尝试将串行程序移植到多个内核,众核方向更多的关注并行应用程序时的吞吐量。现代 GPU 的计算能力使其非常适合 CT 图像重建,除了加速重建之外,与传统 CPU 相比, GPU 的额外计算性能还可用于以多种方式提高图像质量。

近年来,将并行度高的算法从 CPU 全部或部分移植到 GPU,进行 CPU-GPU 协同计算从而大幅提升运行速度成为一种趋势。在进行算法加速时,要想实现最优性能,需要有效的数据并行算法以及针对 GPU 架构特性的优化。由于 CBCT 在 CPU 上进行重建的速度难以满足临床应用需要,许多学者基于三维重建过程的并行特性研究了 GPU 架构下的并行实现方法。

利用统一计算架构(CUDA)通用编程技术,可以将重建算法当做普通的多处理器任务在 GPU 中进行。针对这种体系结构,SHCL 提出了一种加速 FDK 算法^[11]。程杰等利用 CUDA 实现了 FDK 图像重建算法^[12],其整体加速达到 30 倍。李中华和周付根提出了一种基于 GPU 的 CT 图像^[13],利用 GPU 的并行计算能力和浮点运算能力,通过分开执行 CT 图像重建中的像素与几何计算部分,减少了重复运算次数。

1.3 锥形束 CT 近似重建算法存在的不足

当前锥形束 CT 重建的近似重建算法在锥角足够小的情况下能够获得满足要求的图像质量,但仍存在诸多问题。

首先是重建反投影过程的耗时巨大的问题。重建所使用的投影图数量与重建精度成正比关系,而反投影过程仅遍历体素的计算复杂度就为 $O(n^3)$,当投影密度增大时,重建时间呈指数倍增长。即相邻两幅投影图的投射角度差越小,所获得的投影图层数越多,计算投影地址和反投影权重所花时间越久,且对计算机内存的需求也大幅增加。这是影响实时诊断的主要因素。

此外,如何消除星状伪影也是研究的热点问题。由于 FFT 频域滤波对高频噪声的扩大,人造伪影一直是影响 FDK 重建精度的首要因素。通过 RamLak, Shepp-Logan 等高分辨率滤波的降噪操作能够一定程度的减轻伪影,但不能完全消除。针对 FDK 算法的伪影问题,学者们提出了多种改进策略。王革等提出一种由圆轨道推广到平面内任意封

闭曲线扫描轨道 General-FDK 算法, 在螺旋扫描下由于沿 z 方向具有均匀的误差分布, 能够有效减少伪影。CC-FDK(Concentric Circle FDK)算法将通过两个不同轨道半径扫描得到的重建结果进行综合处理, 提升重建质量。

1.4 研究方案

1.4.1 数据准备及工作环境

本论文利用三维 Shepp-Logan 模体生成投影数据, 通过 MATLAB 进行初期建模验证, 并在 Visual Studio 2015 编译环境下参考 Reconstruction Toolkit(RTK), 用 C++实现不同重建算法并进行分析比较。

Shepp-Logan 模体是由 Larry Shepp 和 Benjamin F. Logan 为其 1974 年的论文“头部的傅里叶重建”创建的标准测试图像, 见图 1.5。描述模体的函数被定义为 2×2 正方形内的 10 个椭圆的总和。Shepp-Logan 模体作为人类头部的模型, 在图像重建算法的开发和测试中被广泛使用。



图 1.5 Shepp-Logan 模体

RTK 是一个基于 C++语言的开源 CBCT 重建工具包, 由 RTK 联盟开发。RTK 中实现了常用的 CBCT 重建算法^[14]。

1.4.2 研究思路

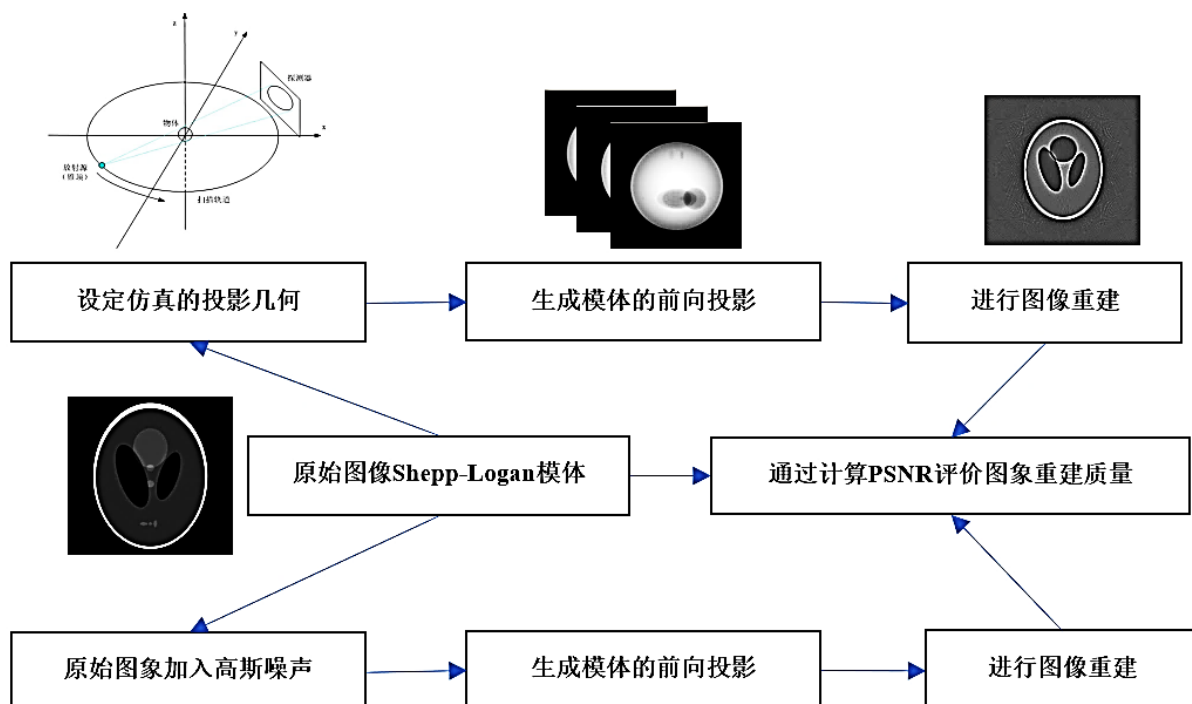


图 1.6 总体研究路线图

整个研究路线图如图 1.6 所示。在第一阶段，根据辐射源、模体、探测器和扫描形式的相对位置，建立仿真的射影几何参数。接着，确定采样间隔和投影矩阵，生成模体的前向投影数据集，作为三维重建算法的输入。之后，运用解析或迭代算法进行锥形束 CT 的三维图像重建。最后，基于 PSNR，评估重建图像的质量。为评价算法鲁棒性，在第二阶段，对原始图像加入高斯噪声，然后投影、重建并进行 PSNR 指标的计算。

1.4.3 研究方法

无论是解析类算法还是迭代类算法，前向投影和反投影占据重建总耗时的绝大部分，由于投影和反投影是互逆的运算，算法原理相近，研究反投影算法的耗时能够一定程度上反应对应的前向投影算法的耗时。本论文在讨论投影和反投影驱动方式的基础上，只对反投影算法进行实现和性能测试与分析，并尝试改进。

现有的反投影驱动方式有三种，如图 1.7。

体素驱动（Voxel-driven）是指将辐射源和体素中心的连线作为反投影的路径。像素驱动（Pixel-driven）是指将辐射源和探测器各像素单元中心的连线作为反投影的路径。距离驱动（Distance-driven）是指将体素和探测器单元的所有水平和垂直边界映射到一个公共平面上，将重叠区域的面积作为反投影的权重。

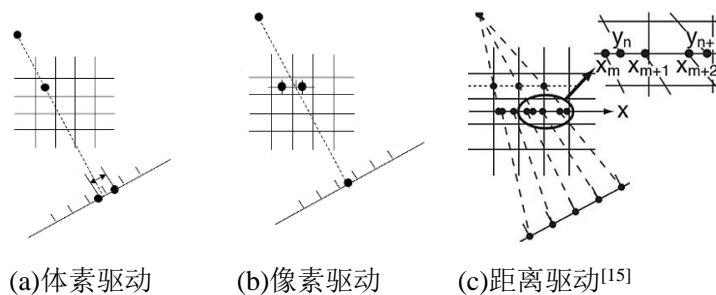
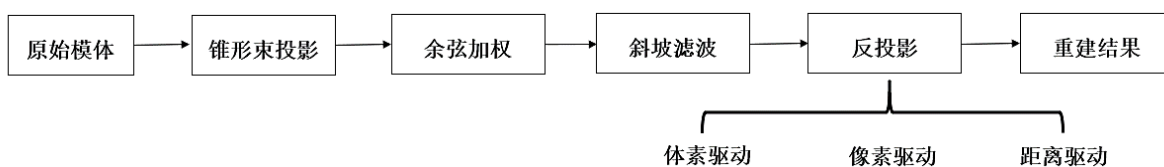


图 1.7 三种反投影方式示意图

实验中以最经典的 FDK 算法为实验框架,依次实现三种反投影算法并带入框架进行重建,实现思路如图 1.8 所示。

图 1.8 解析算法实现思路^{[16][17]}

1.5 论文结构

论文分为五部分。绪论部分介绍锥形束 CT 的优势和 CBCT 重建算法的国内外发展现状,随后阐述本文进行算法研究的方案设计;第二部分分析 CBCT 重建的数学原理,并阐述从平行束、扇形束到锥形束的算法推广过程。第三章详细阐述锥形束 CT 图像重建的数学原理,分析 FDK 重建算法并着重讨论三种反投影方式的原理和优缺点;第四章在第三章的基础上深入讨论距离驱动算法的实现,包括二维距离驱动算法的实现和性能分析、三维距离驱动算法的实现、当旋转轴沿与坐标轴平行方位时的算法优化以及针对此优化方法的 GPU 加速;第五章从重建速度、精度、鲁棒性三方面进行算法性能分析;最后对论文进行总结和展望。

2 CT 图像重建的数学原理

本章首先介绍投影的数学原理和反投影的步骤, 然后简要论证图像重建中最为重要的中心切片定理, 随后讨论二维重建中的平行束和扇形束重建的数学原理。

2.1 投影与反投影

2.1.1 Radon (投影)变换

设某一物理量在二维实欧氏空间中的分布函数为 $f(x, y)$, (x, y) 定义为有限二维平面上点的坐标。则 $f(x, y)$ 沿该平面上的直线 L 以 dl 为线元素增量进行积分的过程即为投影, 见公式(2.1):

$$p = \int_L f(x, y) dl \quad (2.1)$$

$f(x, y)$ 的投影函数 $g_\theta(R)$ 可用 R, θ 构成一个极坐标系统, 这个二维空间通常被称为 Radon 空间, (R, θ) 代表模体空间中所计算的密度函数的一个线积分值。

在 CT 中, 将 N 维图像转换为 $N-1$ 维线性积分的集合的过程称为投影 (前向投影)。例如生成模体的 X 射线图像的物理过程。经过对数转换后, X 射线图像在数学上被描述为该物体的线性衰减系数分布的线积分投影。当 X 射线穿过模体时, 射线行径内各体素的密度对探测器上的投影值均产生影响, 反投影时把投影值作为各射线行径内体素密度分布的一种度量, 将所有经过某一体素射线的投影值之和的平均值为该体素的密度值。

CBCT 扫描的投影过程可由线性方程(2.2)表示

$$Wf = p \quad (2.2)$$

其中 $p \in P$ 为投影数据, $f \in F$ 表示模体, $W: F \rightarrow P$ 为从模体变换为投影图的降维投影矩阵。实际数学模型中, 模体和投影图均为离散的数据, 三维模型中, f 通过在对立方体进行网格划分形成离散数据, 投影数据 p 由探测器像元分布进行离散数据的获取。则方程改写为

$$p_{m,\theta} = \sum_{n=1}^N W_{m\theta,ijk} f_{i,j,k}, m=1,2,\dots,M \quad (2.3)$$

其中 $W_{m\theta,ijk}$ 表示体素 $f_{i,j,k}$ 对宽度为 m 的探测器在 θ 旋转角度时的贡献。

2.1.2 反投影

反投影操作在滤波反投影和迭代重建中均有涉及，其具体实现方法对重建耗时和重建精度有显著影响，是重建算法的重点研究内容。

根据式(2.3)，反投影过程可表示为：

$$f_{i,j,k} = \sum_{m,\theta} W_{m\theta,ijk} p_{m,\theta} \quad (2.4)$$

反投影的过程即遍历体素 $f_{i,j,k}$ ，对于每一幅投影图 $p_{m,\theta}$ ，根据反投影权重 $W_{m\theta,ijk}$ 计算该投影图对该体素的反投影贡献值，并将所有投影图的计算结果进行累加。不同反投影驱动方法对反投影权重 $W_{m\theta,ijk}$ 的计算过程不同，目前有三类驱动方式：体素驱动、像素驱动和距离驱动方式。

二维模体的重建可以看作所有方向下各个点的投影累加过程，如图 2.1，

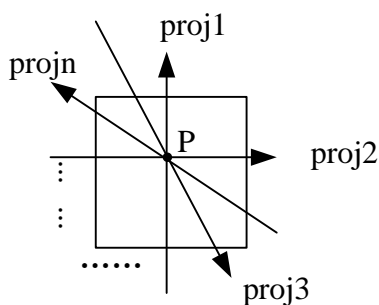


图 2.1^[6] 射线从不同角度穿过物体的情况

则 P 点的反投影重建结果就是对 $proj1, proj2, proj3, \dots, projn$ 求平均，其中 $proj1, proj2, proj3, \dots, projn$ 为通过 P 点的各射线行径投影值。

$$\begin{aligned} proj1 &= \int_{L1} f(x, y, z) dl \\ proj2 &= \int_{L2} f(x, y, z) dl \\ proj3 &= \int_{L3} f(x, y, z) dl \\ &\dots\dots \\ projn &= \int_{Ln} f(x, y, z) dl \end{aligned} \quad (2.5)$$

反投影过程实际上是一个直接将投影值沿射线行径“回抹”的过程，而非投影过程的逆过程，因此只靠反投影不能复原出模体体素的准确密度值。

2.2 中心切片定理

中心切片定理是二维图像解析类重建算法的基础，用来解决平行束射线源下的模体重建，其内容可以描述为：平行投影的一维傅里叶变换等同于原始物体的二维傅里叶变换的一个切片。这一过程可以用图 2.2 说明，二维函数 $f(x, y)$ 在 θ 视角下的平行投影数据 $p^P(\theta, t)$ 的傅里叶变换 F_t 等价于 $f(x, y)$ 的二维傅里叶变换 $F(\rho_x, \rho_y)$ 通过原点并与 ρ_x 成 θ 角的一个切片。

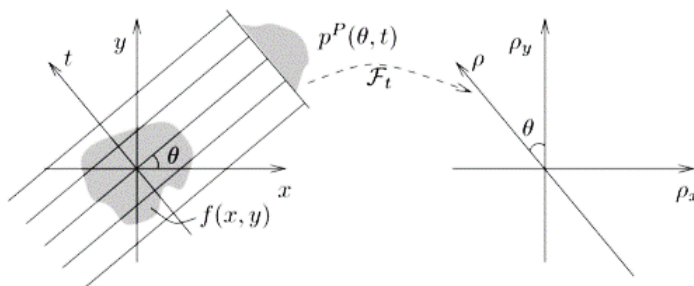


图 2.2^[17] 傅立叶中心切片定理示意图

以中心切片定理为依据的图像重建步骤为：首先取物体在各视角下的投影，然后对投影数据进行一维傅立叶变换，得出二维模体的傅立叶变换，最后进行傅立叶反变换即可还原该二维图像。

在坐标系下傅立叶变换值呈放射状分布，图 2.3 显示多个角度下的投影数据在频域内的分布情况，当投影数量为有限个时，低频部分分布密集，高频部分分布稀疏，需在傅里叶空间对空缺数据进行插值或补零，引入误差，这是中心切片定理的固有缺陷。

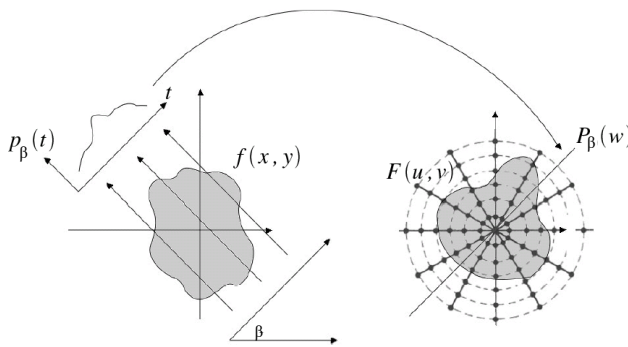


图2.3 傅立叶中心切片定理示意图

中心切片定理指出了从投影重建图像的可能性。具体做法如图 2.4 所示，将投影结

果由 Radon 空间转换到傅里叶空间利用中心切片定理进行处理,而后对傅里叶反变换后的投影数据进行反投影从而复原模体的密度值。

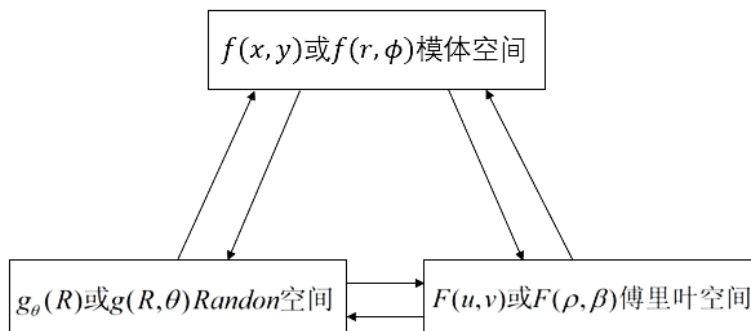


图2.4 模体空间、Radon空间与傅里叶空间的相互转换

2.3 平行束与斜坡滤波

由于中心切片定理存在的固有缺陷,在反投影过程中不可避免地会产生星状伪影 (Star-like artifacts), 因此在进行反投影重建之前需对频域的傅里叶变换结果进行插值, 使其密度均匀, 这一加权过程称为反投影前的滤波操作。简单起见, 可以用图 2.5 说明对 3×3 二维模体的平行投影结果进行反投影过程中星状伪影产生的原因:

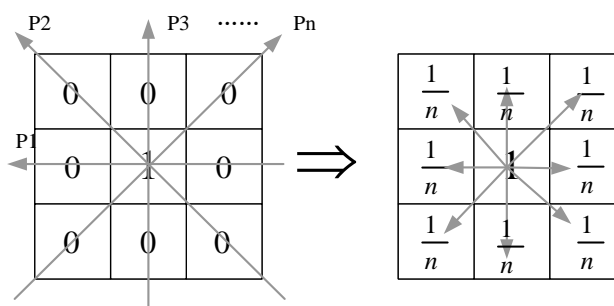


图2.5 反投影过程中星状伪影的产生

左图为 3×3 矩阵,除了中心外,其他元素的值都是 0,对其取 n 个方向的投影,且每个投影都通过矩阵中心。这样,测量得到的每一个投影值都为 1,即

$$P_1 = P_2 = P_3 = \dots = P_n = 1 \quad (2.6)$$

因为反投影过程是将投影均匀地回抹到投影所经过的各个点上,故每一个矩阵元素在各个方向的反投影值都是 $\frac{1}{n}$ 。各元素的反投影值计算如下:

$$P'_{center} = \frac{P_1 + P_2 + \dots + P_n}{n} = 1;$$

$$P'_{other} = \frac{P_i}{n} = \frac{1}{n}; \quad (i = 1, 2, 3 \dots n)$$
(2.7)

这里 P'_{center} 表示中心点的反投影值, P'_{other} 表示除中心点外其它的元素的反投影值, 反投影后所得图像如图 2.4 右图所示。可以发现, 原来不为 0 的元素经过反投影后变成 $\frac{1}{n}$, 这就是伪影产生的原因。注意到这种伪影以矩阵中心为中心向四周呈现向四周辐射状, 因此被称作星状伪影。

由傅里叶变换理论可知, 傅里叶空间的密度分布正比于 $1/\sqrt{\omega_x^2 + \omega_y^2}$, 因此斜坡滤波器可以在一定程度上消除星状伪影^[18]。

之所以说斜坡滤波器不能完全消除星状伪影, 是因为斜坡滤波是抑制低频分量, 提高信号的高频部分比例的滤波器。由于高频往往不包含有用信息反而富含噪声, 且滤波器的高频边缘在离散傅里叶逆变换过程中表现不佳, 因此斜坡滤波会引起重建结果中的噪声放大。实践中, 我们可以将斜坡滤波器与多个窗函数相结合, 对高频分量进行滤波。

$$H(w) = |w| W(w)$$
(2.8)

式(2.8)中, $W(w)$ 为带限窗函数。滤波反投影算法的理想滤波效果如图 2.6, τ 为采样间距。理想的带通滤波器是不存在的, 需要在现有降噪滤波器中选择在带宽和噪声抑制之间折中较好的滤波器。

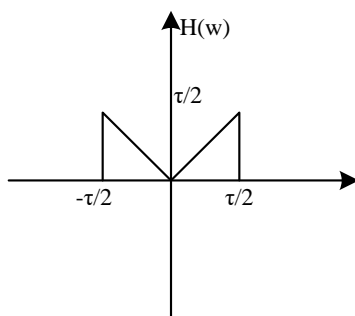
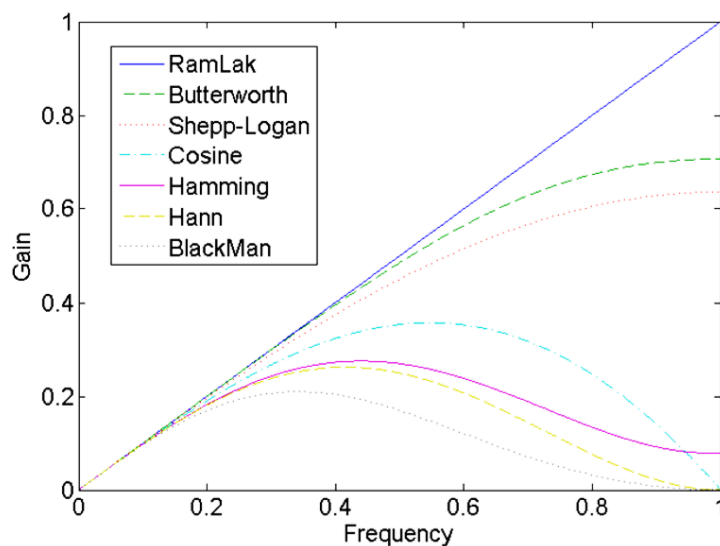


图2.6 滤波反投影算法的理想滤波器的频域响应

降噪滤波器分为高分辨率滤波器和低分辨率滤波器两类, 如图 2.7, 横轴表示从 0 (基频) 到 1 (最高频率) 的频率, 竖轴表示当前频率的增益 (乘数)。


图2.7 常见滤波器特性^[25]

RamLak, Butterworth 和 Shepp-Logan 是高分辨率滤波器, 滤波后分辨率降低的百分点低于噪声的减小百分点。第二类中有 Cosine, Hamming 和 Hann 滤波器, 这些滤波器的噪声减少比高分辨率滤波器更好, 但分辨率也大幅降低。CT 重建旨在创建高分辨率图像, 因此一般选择高分辨率滤波器, 如 Shepp-Logan 滤波器, 其噪声降低可达至少 10%, 而分辨率几乎是斜坡滤波图像的 96%。巴特沃斯滤波器对分辨率的影响更小, 同时仍能改善噪声性能。

2.4 扇形束与余弦加权

扇形束重建算法可由平行束重建算法推广得到, 扇形束与平行束的投影几何对比如图 2.8 所示。

通过把扇形束中互相平行的射线分在一组的形式对数据进行重组可将扇形束重建问题转化为平行束重建问题, 但在重组过程中涉及插值计算, 引入人为误差。工程实际操作是用对算法的积分做变量替换代替理论分析中对数据进行重组的操作。

$$\begin{aligned}\theta &= \gamma + \beta \\ s &= D \sin \gamma\end{aligned}\tag{2.9}$$

扇形束射线与平行束的几何关系如公式(2.9)。由微积分知识可知, 在做积分的变量替换时需要加入由偏导数得出的行列式, 即雅可比因子 $J(\gamma, \beta)$ 。由于在变量代换中的雅各比因子中含 $\cos \gamma$ 项, 因此这种变换操作被成为“余弦加权”。综上, 从平行光束图像重建算法到扇形束图像重建算法的推导流程图如图 2.9。

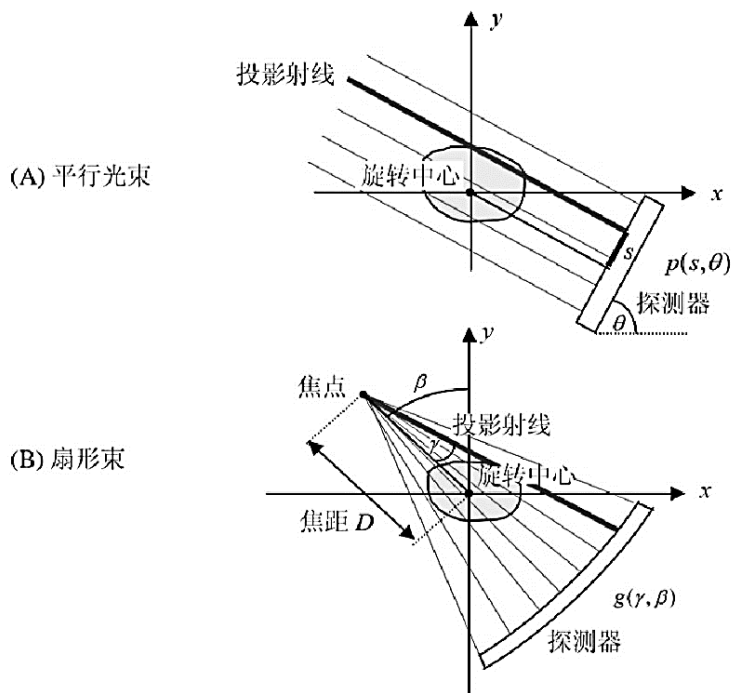
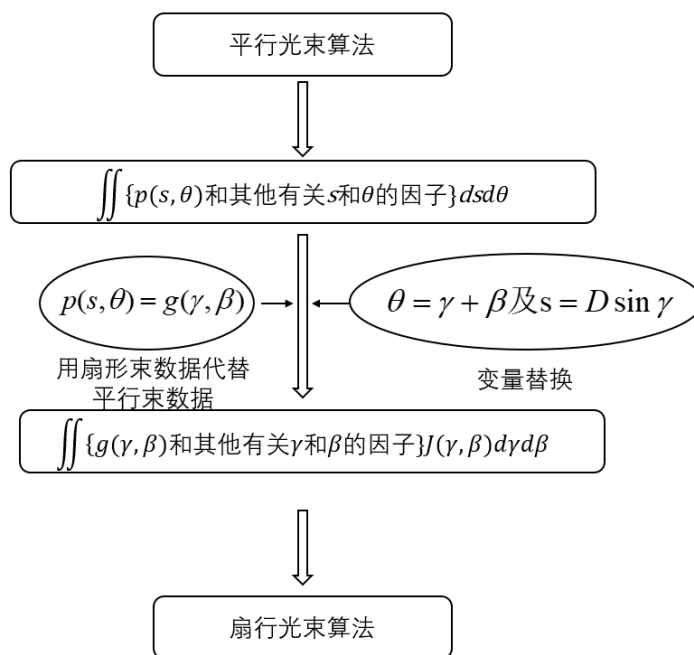

图2.8 平行束与扇形束的投影几何对比^[4]


图2.9 从平行束到扇形束的推导过程示意图

2.5 本章小结

本章介绍了 CT 图像重建的数学原理，从数学概念出发引出了基于 Randon 空间的投影和反投影过程，说明反投影是直接“回抹”的过程，不能复原模体。随后引出著名的中心切片定理，指出通过在傅里叶空间对投影做变换从而复原图像的可行性，但



由于投影数据在频域分布不均，重建结果存在星状伪影。接着，基于平行束介绍了通过斜坡滤波减轻星状伪影的方法以及抑制高频的降噪滤波器的使用。最后，根据扇形束与平行束的投影几何关系，通过变量替换实现平行束到扇束的算法推广。

3 锥形束 CT 重建算法

平行束和扇形束 CT 可以很好解决对模体中一个平面的成像，但对于立体空间成像，平行束和扇形束通过多个切片叠加的方式存在分辨率各向异性的问题，在沿切片叠加方向数据分布稀疏，插值后引入人为误差。CBCT 的辐射呈锥束状，一次成像可以获取三维空间的信息，而不是一个断层的信息，能够直接重建三维模体，因此锥形束 CT 在立体空间成像上优势明显。本章首先分析锥形束的几何模型以及 Radon 变换的数学基础，随后介绍锥形束重建常用的 FDK 算法实现步骤，并对第三步反投影的驱动方式进行原理阐述。

3.1 锥形束 CT 几何模型及 Radon 变换

3.1.1 锥形束几何模型

锥形束的几何模型如图 3.1 所示，图中 β 表示扫描角度， γ 表示扇角， κ 表示锥角。

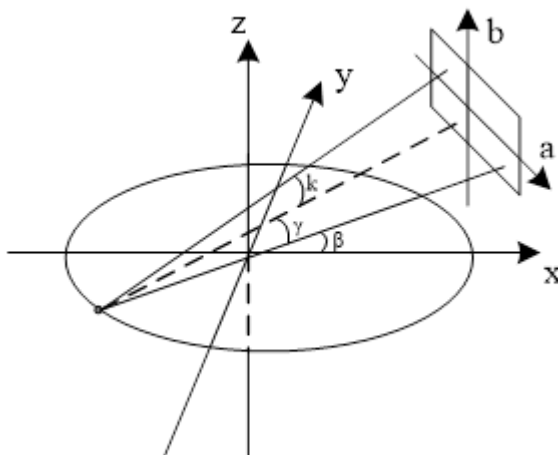


图3.1 锥形束扫描的几何模型

我们定义平面探测器 a-b 直角坐标系的原点为探测器的中心，a, b 分别对应投影的 x, y 方向。射线源点的沿圆轨道运动，并在不同的位置（即不同的 β 角）对物体成像，后文中用 $P(\beta, a, b)$ 表示投影值。探测器平面与扫描平面垂直，与 z 轴平行。

3.1.2 三维 Radon 变换

精确的三维重建算法是基于三维 Radon 变换的，一个物体区域内的平面积分对应一个三维 Radon 变换值，且一个三维空间中的矢量点可以唯一确定一个平面，因此可以用

Radon 值表示空间平面的信息。三维 Radon 变换的公式是

$$p(t, \theta, \varphi) = Rf(t, \Theta) = \iiint f(x, y, z) \delta(t - (x \cos \varphi \sin \theta + y \sin \varphi \sin \theta + z \cos \theta)) dx dy dz \quad (3.1)$$

式中 t 为模体中心到投影平面的距离, (θ, φ) 表示投影平面的朝向位置。Radon 变换理论表明只有获得 Radon 空间中所有点的 Radon 值才能完成模体的精确重建。

要想精确重建出原物体,必须满足所有穿过模体的平面与扫描轨道有交点,从而获得所有 Radon 值。根据这一要求,扫描轨道可以分为满足条件的完全轨道和不满足条件的不完全轨道。在圆扫描轨道中,与扫描平面平行的平面与扫描轨道没有交点,因此为不完全轨道。

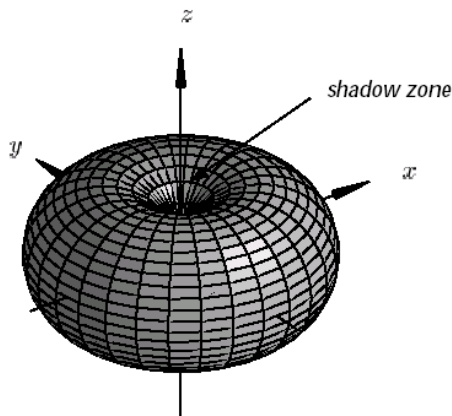


图3.2 圆轨道扫描所测量的Radon值在Radon空间的分布^[19]

图 3.2 中 shadow zone 表示缺失的 Radon 值。由于基于圆轨道的锥形束无法测量平面方向与 z 轴夹角小的平面的 Radon 值,所得投影值对应的有效 Radon 值在三维 Radon 空间中呈花环状分布。因为圆轨道不能测量完整的 Radon 值分布,所以只能近似重建原模体。

3.2 FDK 算法分析

针对圆轨道的近似重建算法是由 Feldkamp, Davis 和 Kress 在 1984 年提出的,该算法简称为 FDK 算法。由前文所述 Radon 变换原理可知,圆轨道的扫描方式由于不满足完全轨道条件,得到的重建结果存在不可消除的误差,但在锥角不太大 ($<10^\circ$) 的情况下,FDK 算法的重建结果误差可以控制在允许范围内。

FDK 算法是扇形束重建算法的推广,同样以滤波反投影算法为基础。利用 FDK 算法对平面探测器所获投影图进行重建时公式如下:

$$f_{fdk}(x, y, z) = \int_0^{2\pi} \frac{D_{sd}^2}{U^2} p''(\beta, a, b) d\beta \quad (3.2)$$

其中 D_{sd} 表示辐射源到探测器的距离, U 为辐射源到过重建点且与探测器平面平行的平面的距离, p'' 为经过加权滤波后的投影数据, a, b 分别为重建点投影到探测器上的交点与探测器中心连线沿 a 轴和 b 轴的长度。

FDK 算法的重建步骤分为三步:

第一步是对投影图进行余弦加权, 与扇形束重建不同的是, FDK 算法的加权值是锥角 κ 和扇角 γ 的余弦, 即

$$P'(\beta, a, b) = P(\beta, a, b) * \cos \gamma \cos \kappa \quad (3.3)$$

这里

$$\cos \gamma \cos \kappa = \frac{D_{sd}}{\sqrt{D_{sd}^2 + a^2}} \frac{\sqrt{D_{sd}^2 + a^2}}{\sqrt{D_{sd}^2 + a^2 + b^2}} = \frac{D_{sd}}{\sqrt{D_{sd}^2 + a^2 + b^2}} \quad (3.4)$$

第二步是对加权后的数据进行滤波, 虽然锥形束 CT 成像所获取的投影为二维投影, 但是在滤波方向上沿投影的水平方向逐行进行一维滤波, 因此锥形束重建的滤波器可以选择与扇形束重建相同的滤波器。

滤波的主要功能是消除由于线性插值方式进行的加权操作与实际投影过程不相符而产生的星状伪影。

FDK 算法的滤波过程可以看作是一个卷积操作, 如下

$$P''(\beta, a, b) = P'(\beta, a, b) \otimes g(a) \quad (3.5)$$

$g(a)$ 与扫描角度 β 和坐标 b 都独立, 仅与坐标 a 有关, 即只跟投影图的水平方向有关。 $g(a)$ 在频域内的形状呈斜坡状, 即 $g(a)$ 为前文提及的可以消除星状伪影的斜坡滤波器。斜坡滤波基于 FFT, 时间复杂度为 $O(n \log n)$

第三步是将滤波后的各幅投影图进行加权反向投影并对所有投影得出的反投影结果进行求和。反投影是指将经过加权滤波后的投影数据回抹到体素中的过程, 此过程涉及反投影地址求解、反投影权重计算等大量运算过程。反投影算子 (BP) 仅遍历体素的时

间复杂度就为 $O(n^3)$ ，是 FDK 算法中最耗时的环节，也是除滤波之外，有望提高图像质量的环节。由于计算能力有限，确切的 BP 只能用现代算法逼近。目前，BP 有三种不同的方法：像素驱动、体素驱动和距离驱动。

3.3 反投影驱动方法

3.3.1 体素驱动方法(Voxel-driven)

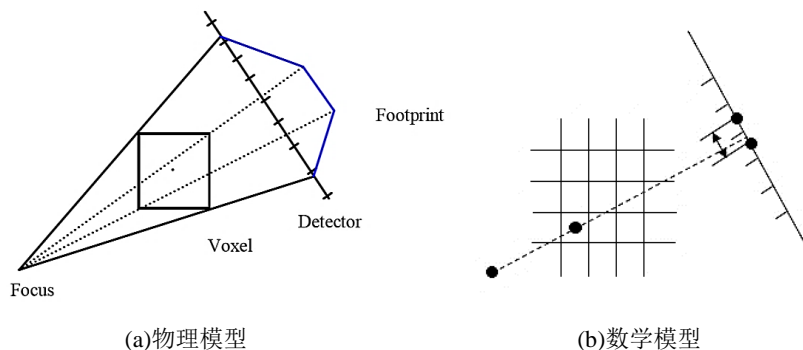


图3.3 体素驱动方法

医学图像重建的反投影过程最早使用体素驱动方法 (Herman 1980, Peters 1981, Zhuang 等 1994)，物理模型如图 3.3(a)，体素对穿过其中的射线衰减程度用该体素在探测器上的足迹(Footprint)表示。该投影方式对于每个体素都是独立的，因此可以直观获得适合 GPU 体系结构的并行实现。数学实现时，将放射源和模体中各体素中心的连线作为反向投影的路径，以射线与探测器交点处的插值结果作为反投影值。如图 3.3(b)所示。

体素驱动的反投影方法通常要求探测器阵列满足规则间隔，在计算射线与探测器平面的交点时具有较高的计算复杂度（如果信号接收端为平板探测器则采用分区计算，若接收端为弧形探测器则需进行弧度角的正切计算）。虽然体素驱动算法的实现可以通过专门处理电路解决 (Kotian 等 1995)，即适用于硬件设备，但是在通用微处理器中，由于算法复杂性，体素驱动的方法表现不佳。体素驱动的前向投影是反投影的逆过程，由于放大了高频噪声而很少在实际应用中被使用。(Zeng 和 Gullberg 1993, De Man 和 Basu 2002)。这些伪影可以通过使用更复杂的加权方案进行削减，但要以计算复杂性的增加为代价。

RTK 中提供了体素驱动方法的图像滤波器，可设置双线性插值/最近邻差值,在 CPU/GPU 上运行等参数。

3.3.2 像素驱动方法(Pixel-driven)

像素驱动是指将锥形射线束以各探测器像素边界进行离散化处理, 对于每幅投影图中的各像素, 遍历模体的体素, 计算当前像素对射线行径上体素的贡献值, 如图 3.4(a)。数学实现上将放射源和探测器各像素单元的中心连线作为反投影的路径, 如图 3.4(b), 这种方法将位于射线附近的各体素的衰减系数加权和作为每条线积分的值。

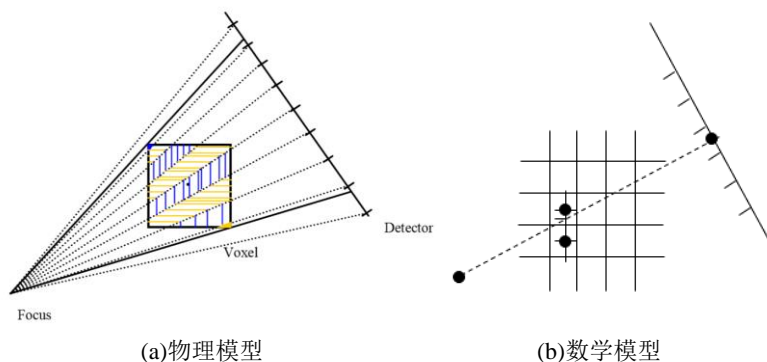


图3.4 像素驱动方法

像素驱动计算了射线路径与要重建的每个体素之间交点的长度, 并表示为

$$\mu_i = \frac{\sum_j l^{ij} p_j}{\sum_j l^{ij}} \quad (3.6)$$

其中 μ_i 是模体中索引值为 i 的体素 V_i 的衰减系数, p_j 是滤波反投影数据中索引值为 j 的像素 P_j 的像素值, l^{ij} 是各体素的加权因子, 其大小为 V_i 与连接放射源和 P_j 像素中心的射线的交线长度 (如果没有交线, 则 $l^{ij} = 0$)。对全部投影数据的各像素应用这一射线追踪过程, 步骤如下: 首先, 将所有 μ_i 值设置为 0。随后, 通过连接辐射源和 P_j 像素中心的连线 α^j 找到影响 P_j 的体素。接着, 通过 Siddon 等线积分近似方法计算 α^j 和模体中各体素的交线 (后文有进一步介绍)。随后, 对于 α^j 穿过的每一个体素 V_i , 在其当前衰减系数中间计算结果 μ_i 的基础上加上 $l^{ij} p_j$ 。最后, 对模体进行归一化处理, 即将每一个衰减系数 μ_i 除以 $\sum_j l^{ij}$ 。

像素驱动法正投影和反投影的精度因不同线积分近似法的选择而不同。现有的像素驱动方法有四种: Siddon 方法、Joseph 方法、RayCast 方法以及 Köhler 方法^[22]。Siddon 方法基于最近邻插值, 并以射线穿过各体素的交线长度作为各体素对投影值的贡献

(Herman 1980, Siddon 1985, Zeng and Gullberg 1993, Zhuang et al 1994) , 如图 3.5 所示。该方法计算射线穿过体素的长度, 与插值结果直接相乘并累加, 得到投影数据。

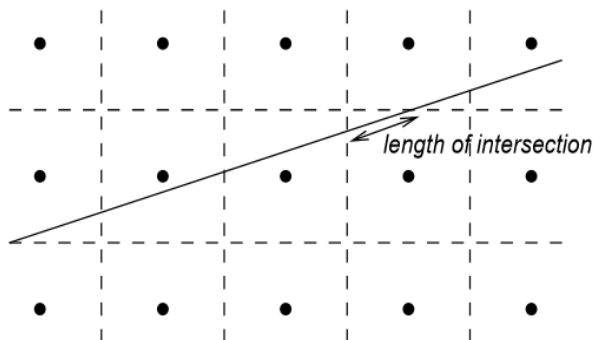


图3.5 Siddon方法原理示意图^[22]

Joseph 方法基于双线性插值, 计算射线的主方向, 基于双线性插值累加与主方向垂直的平面与射线交点处的衰减系数, 并在最后用与射线方向有关的系数对累加结果进行加权, 得到投影数据 (Joseph 1983, Zhuang et al 1994), 如图 3.6 所示。

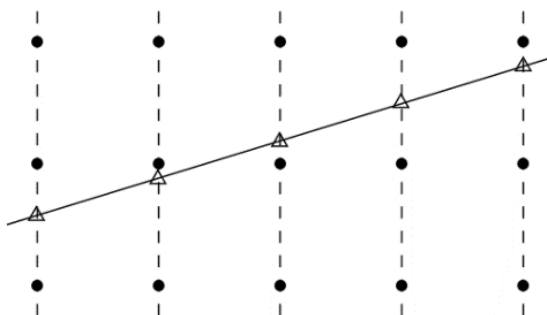
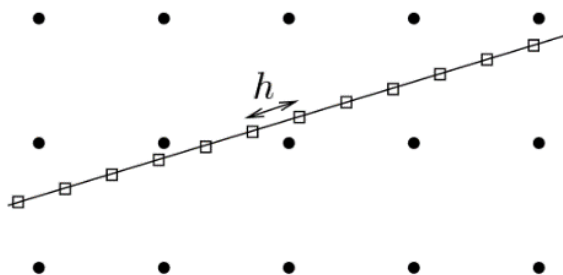
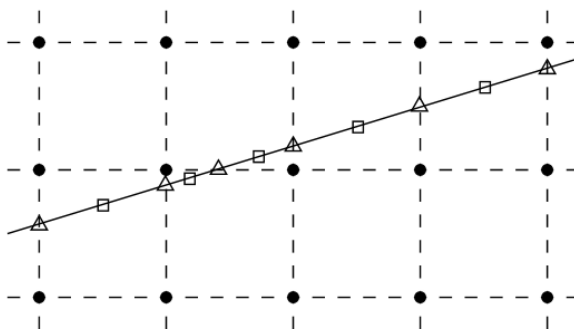


图3.6 Joseph方法原理示意图^[22]

RayCast 方法与 Joseph 方法的投影思路类似, 所不同的是 Joseph 方法只在射线主方向的各切片上设置采样点而 RayCast 方法在采样点之间保持固定的步长, 如图 3.7, 也就是说 Joseph 方法的采样是在各切片平面上进行双线性插值, 而 RayCast 方法中要求解的很可能是非切片平面内的一点, 即需要对三维空间内一点的三次线性插值。因此 RayCast 方法是采用三次线性插值的正向投影方式, 通常在 CUDA 中实现, 其中步长 h 决定了投影精度。

图3.7 RayCast方法原理示意图^[22]

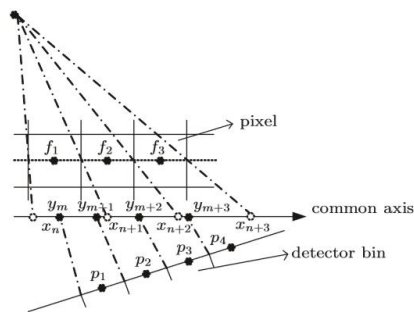
Köhler (2000)方法基于双线性插值的方法,如图 3.8,该方法利用辛普森积分公式简化了线积分过程。在两点之间的积分可以用端点处的函数值与中点处的函数值估计。该方法把物体平移,使得采样点位于体素的顶点处。

图3.8 Köhler方法原理示意图^[22]

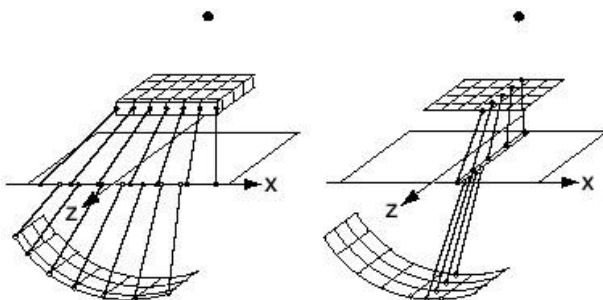
现有近似方法中由于 CudaRayCast 方法利用三次线性插值重建精度较高,且在 GPU 上运算耗时相对较小,因而应用广泛。

综上,像素驱动方法通常非常适用于投影,但往往会在反投影中引入伪影(De Man and Basu 2002,2004)。此外,像素驱动通常具有高度非顺序的存储器访问模式,并行性不佳,使得存储器带宽成为其性能的限制因素,在实际应用中影响数据采集和重建的总耗时。此方法遍历各射线,对射线行径上的体素进行投影和反投影相关运算,每条射线都涉及对各体素位置和反投影值信息的读写操作,因此除非特殊处理(详见 5.1),无法像体素驱动那样在对每幅投影图进行反投影时对各体素的运算进行多线程并行计算。

3.3.3 距离驱动方法(Distance-driven)



(a)二维边界映射过程



(b)三维边界映射过程

图3.9 距离驱动算法示意图^[26]

De Man 和 Basu 于 2003 年提出了一种新颖的投影和反投影方法，并称之为距离驱动方法，距离驱动的投影反投影通过将每个像素和检测器单元的边界映射到公共轴(二维重建，如图 3.9(a))或公共平面(三维重建，如图 3.9(b))上，两者重叠部分的长度(二维重建)或面积(三维重建)作为投影和反投影的权重。以二维为例，距离驱动的内核操作为：遍历所有交点，计算两个相邻交点之间的间隔长度，并使用这些长度来确定相应体素对相应探测器单元的贡献（前向投影），或探测器单元对相应体素的贡献（反向投影）。

这种做法的依据是体素和探测器单元满足双射的关系，各投影角度下，体素上的点被唯一映射到探测器单元的某点上，反之亦然。那么，无论公共轴或公共平面在何处，只要重叠长度或面积存在，则其归一化的值是恒定的，即投影、反投影的过程可以转化为对公共轴/平面上的数据加权重采样的过程。距离驱动算法的投影、反投影过程是对称且可逆的，因此不会在投影、反投影的过程中造成额外误差；另外，由于其具有与体素驱动相似的遍历顺序，对于一幅投影图，计算各体素的反投影获得值的过程相互独立，不存在写入冲突，易于使用并行方式实现。

距离驱动算法具有低运算成本和高度顺序的存储器访问模式，不会引入空域或频域



伪影。该方法适用于平行光束，扇形束和锥形束几何结构。距离驱动的方法可防止像素驱动投影和加速驱动反投影中产生的伪影。从计算耗时的角度来看，它优于像素驱动和光线驱动的方法，并且适用于硬件实现。距离驱动算法能够在具有分级存储器的计算机上利用高级缓存从而减小耗时。下文中将在带有平板探测器的三维锥束几何模型下比较三种驱动方式的速度精度鲁棒性等性能。

目前 RTK 中不提供距离驱动方法的开源代码，需要根据原理自行实现。

3.4 本章小结

本章首先介绍锥形束几何模型及三维 Radon 变换，随后针对有 Radon 值缺失的圆扫描轨道，介绍经典的 FDK 近似重建算法。FDK 算法的重建步骤为，(1)根据待重建体素位置的锥角和扇角，对投影图进行加权；(2)沿投影的水平方向对加权后的投影数据逐行进行一维滤波；(3)将滤波后的各幅投影图进行加权反投影并求和。本章最后针对最为耗时的反投影环节介绍了三种驱动方法：像素驱动、体素驱动和距离驱动，内容包括算法原理、优缺点及代码开源情况。



4 反投影算法的实现

在反投影的三种驱动方式中, 距离驱动算法较好的结合了体素驱动算法速度快和像素驱动算法精度高的特点, 具有高度顺序的存储器访问模式和低算法成本。目前距离驱动算法没有开源代码, 因此首先在 MATLAB 环境下实现二维距离驱动算法并进行简单优化, 之后在 VS2015 环境下实现三维距离驱动算法并优化, 并对 CPU 优化版本的代码进行 GPU 加速。

4.1 像素驱动与体素驱动算法

反投影时间跟重建模体的大小以及投影图数量有关, 实际应用中通常重建 $512 \times 512 \times 100$ 尺寸的模体, 使用投影图 500 到 700 张。仿真模拟时按照 RTK 自带的默认参数重建 $256 \times 256 \times 256$ 尺寸的模体, 使用投影图 180 张, 对反投影算法开源代码进行测试, 命令行操作和参数说明如下:

(1) 利用专门为 Shepp-Logan 模体设计的射线椭球插值图像滤波器 *RayEllipsoidIntersectionImageFilter* 进行投影图的仿真, 投影图像尺寸 $256 \times 256 \times 180$, 间距 2×2 。

(2) 使用 *rtkfdktwodweights*, *rtkramp* 工程中默认参数进行加权、滤波操作。

(3) 对反投影方式设置参数赋值, 获得 "*CudaRayCast*" "*VoxelBased*" 等不同反投影方式的重建结果和耗时。

(4) 重建模体尺寸 $256 \times 256 \times 256$, 间距 $2 \times 2 \times 2$ 。进行 PSNR 计算时(见附录 A2), 参考模体图和重建模体图的分辨率要相符, 即应使用 *rtkdrawgeometricphantom* 生成间距为 $2 \times 2 \times 2$ 的模体图。

4.2 二维距离驱动算法

4.2.1 二维距离驱动算法的实现

在 MATLAB 环境下实现二维距离驱动算法, 几何模型定义如图 4.1 所示, 以模体中心为原点建系, 放射源到模体中心的距离为 $d=200$, 体素大小为 128×128 个像素, 探测器和模体逆时针绕模体转动进行扇形束 CT360 度全扫描。

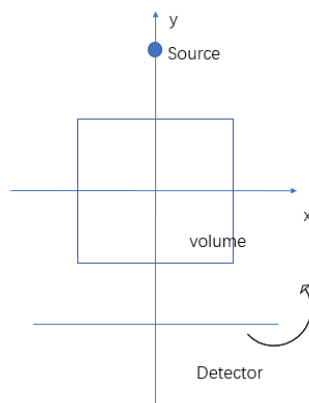
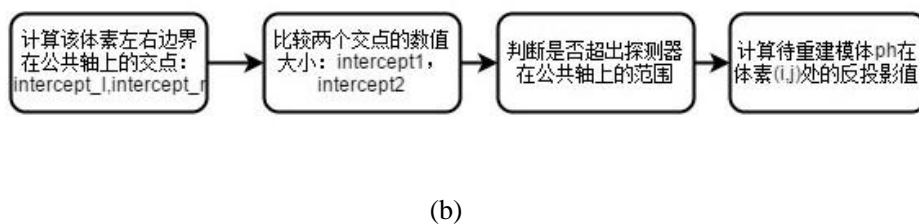
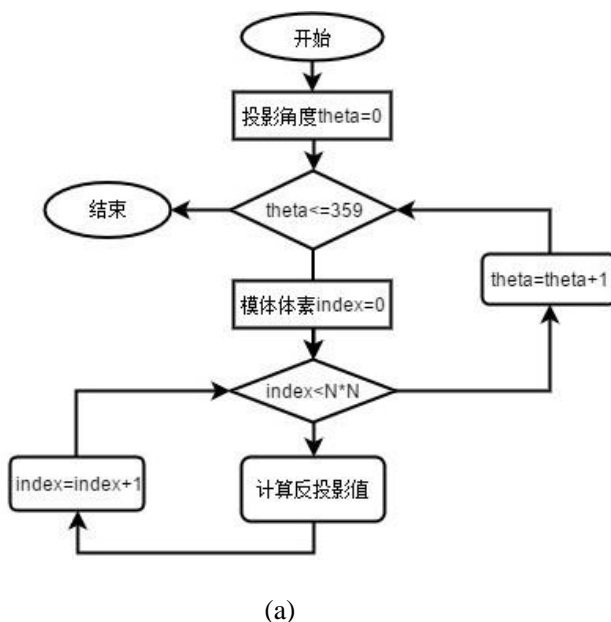


图4.1 二维算法实现的几何示意图1

基于此几何模型的加权滤波实现见 2.3, 2.4 节, 反投影算法流程如图 4.2 下所示:



(a)二维距离驱动算法的总体流程 (b) 反投影的详细过程

图4.2 二维距离驱动算法流程图

为方便计算边界点的投影位置, 设公共轴为 x 轴, 且假设探测器和源不动, 模体绕坐标原点顺时针转动。不同投影角度下边界点坐标由投影矩阵求解, 当模体顺时针旋转 θ 角度时, 旋转矩阵为

$$A = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \quad (4.1)$$

边界点投影点坐标由相似三角形求解，如图 4.3，

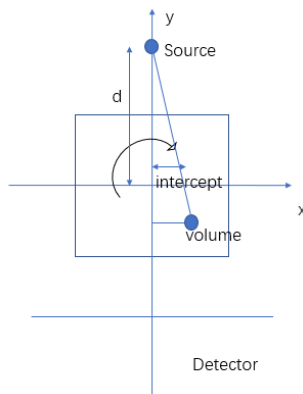


图4.3 二维算法实现的几何示意图2

取过体素中心且与 x 轴平行的直线与体素边界的交点作为距离驱动算法的边界点，如图 4.4 所示，可以发现，相邻体素的边界点重合，利用这一特质可减少边界点投影位置的计算量。

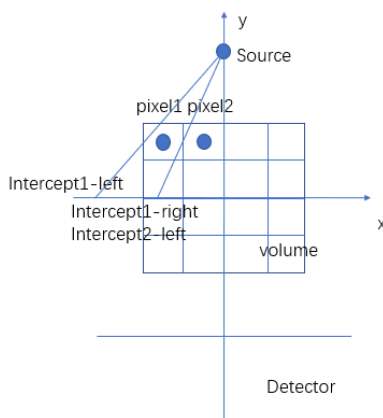


图4.4 二维算法实现的几何示意图3

MATLAB 中的扇形束 fanbeam 函数规定，当探测器类型选择“直线型”探测器时，默认各探测器单元与放射源连线与 x 轴的交点等间隔，且间隔为单位 1。因此探测器边界到公共轴 x 轴的投影位置即像元标签值加上探测器位置与标签值的偏移量 offset 在加减半个单位长度 0.5 即可。

值得注意的是，探测器的范围是有限的，为确保求解的投影位置在探测器的接受范围内，提高算法的鲁棒性，需要对边界点的投影位置进行出界判断。如图 4.5 蓝色矩形在长度方向覆盖的区域为探测器边界在公共轴上的投影范围，每对箭头表示某体素边界

投影在公共轴上的区域,两边界点投影都在探测器范围内的情况如第一行所示。如果有一侧出界,则将出界侧的投影位置更正为探测器边缘所对应的投影在公共轴上的坐标,如第二行所示。当两投影位置都不在探测器接收范围内时,说明当前体素对当前投影角度下的投影图没有贡献,跳出循环对下一个体素进行操作。

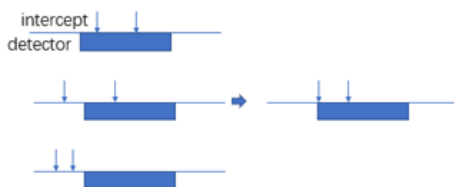


图4.5 二维算法实现的几何示意图4

体素对投影图的贡献与体素到公共轴沿 y 方向距离的平方成反比,与该体素边界点在公共轴上的有效投影范围和探测器单元在公共轴上的投影范围的比值成反比。

4.2.2 算法性能分析

重建效果如图 4.6(c)所示,图 4.6(a)为模体图,图 4.6(b)为利用 MATLAB 自带滤波反投影(FBP)算法重建的结果图。

重建时间方面,由于 MATLAB 调用机制和内存访问机制的特殊性,轻微的代码改动就可能对重建时间产生很大的影响。例如在编写距离驱动方法代码时,优化前后的时间相差 10 倍有余,从 10s 提升至 0.8s,而优化前后改动的仅仅是将矩阵乘法写开为表达式、减少矩阵创建、用数值代替变量等与数据调用相关的减少内存访问次数的编程技巧。在算法时间比较时无法控制调用机制、内存访问机制等变量相同,因此不做二维重建的时间比较。

比较图(b)(c)可以看出,距离驱动方法重建精度相近。分别测试 FBP 重建结果和二维距离驱动重建结果与原图间的误差,如图(d)(e)所示,两种算法在模体密度梯度较大的区域误差较大,如模体的边缘。距离驱动算法在模体上的误差分布范围较 FBP 算法大得多,初步分析这是由于 FBP 算法除加权滤波反投影操作外还自带预处理和后处理过程,而距离驱动算法没有加入这些减少伪影处理。由于 MATLAB 自带 FBP 算法没有开源代码,内部实现不透明,两种算法无法做到同意变量,深入分析比较没有意义。

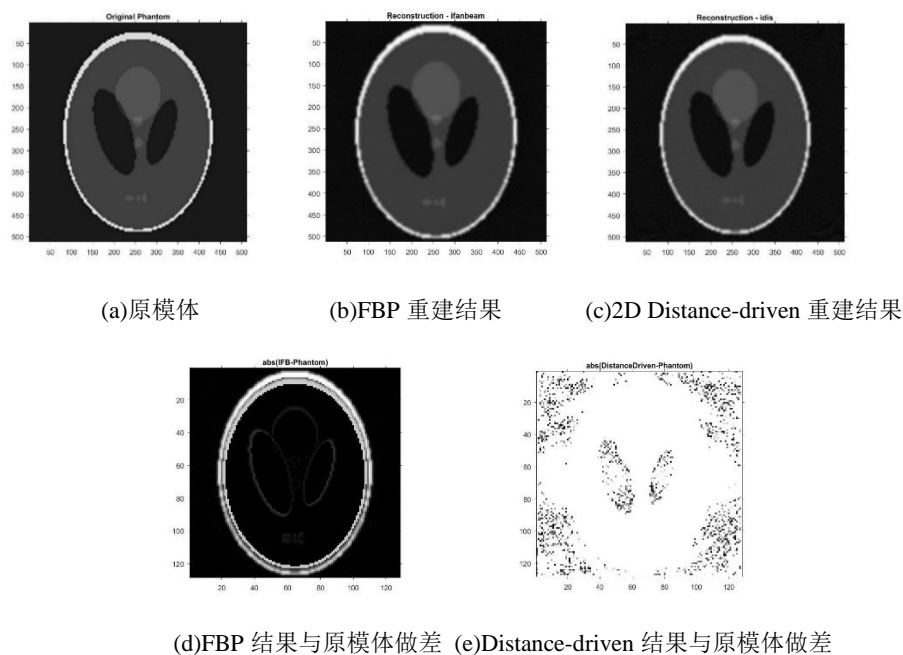


图4.6 二维算法重建结果比较

4.3 三维距离驱动算法

4.3.1 三维距离驱动算法实现

三维距离驱动算法利用 ITK 的管线机制,在 8 个线程下通过投影矩阵计算投影位置,并得出加权后的反投影值。

1、管线机制(pipeline)

在 VS 环境下利用 RTK 实现三维距离驱动算法。在 ITK 中提供一种特殊的数据传输与调用机制—pipeline,这种管线机制将数据对象(例如图像和网格)和处理对象结合在一起。pipeline 支持自动更新,当且仅当其输入或其内部状态发生变化时才会执行 filter。用 RTK 进行 FDK 算法框架搭建时,仿照 `rtk::FDKConeBeamReconstructionFilter`,将余弦加权的类 `rtk::FDKWeightProjectionFilter`、频域滤波的类 `rtk::FFTRampImageFilter` 与需要自己编写的实现距离驱动反投影算法的类 `DDBackProjectionImageFilter` 串联起来,即将前一步的结束接口设置为当前步的开始接口,将当前步的结束接口设置为下一步的开始接口,方便各类的对象之间的数据调用和传输。

2、多线程并行计算

由于设备存储器和主机间的数据访问耗时巨大,因此在未达设备内存容量上限的前

提下尽可能多的将投影图数据一次性搬运到设备存储器中，分组进行反投影计算。程序中设定每批处理 16 幅投影图，每批的计算结果输出值作为下一批开始计算的输入值。同时，为充分利用计算资源，采用多个线程并行执行，本次实验所用 CPU 为 4 核，有 8 个逻辑处理单元，因此设置线程数为 8 个。这样一来，每次执行代码时，会将所有体素平均分成 8 个区域进行重建，比单线程的速度提升 8 倍。

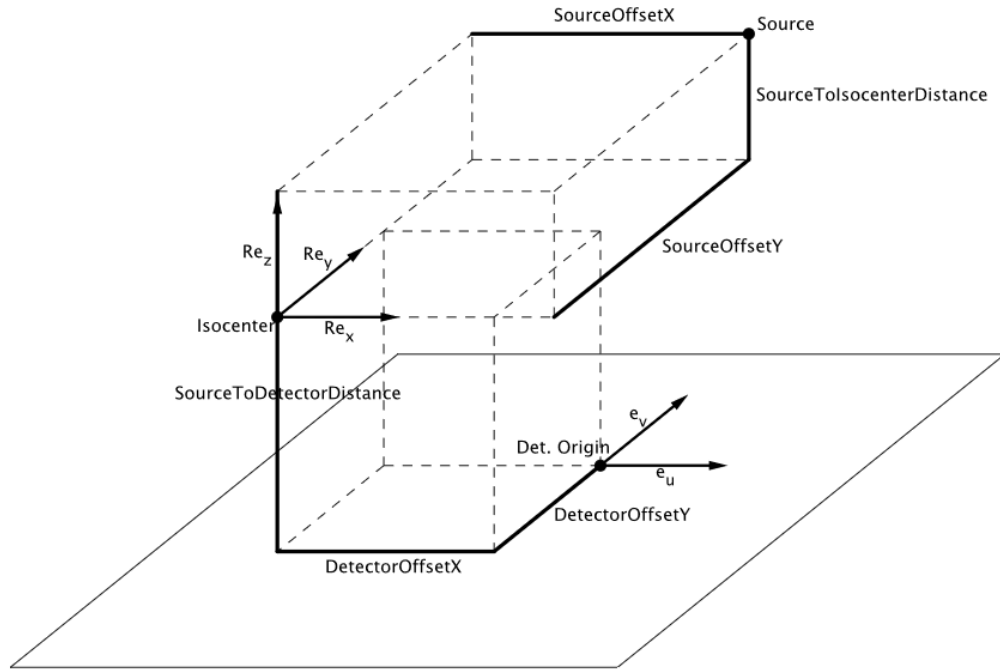
3、投影矩阵

投影矩阵常用于记录从 N 维到 N-1 维的降维操作，为方便进行矩阵运算，在数字图像处理中常使用齐次坐标将旋转平移等变换在同一矩阵中表示。三维重建中，体素的投影位置通过 3×4 大小的投影矩阵来计算。

根据 RTK 中对圆环轨道的射影几何的定义，用传统的 ZXY 坐标轴对应的三个欧拉角表示探测器的旋转方向，其中 *InPlaneAngle* 表示绕 z 轴沿 z 轴负方向看去逆时针旋转角度值，*OutOfPlaneAngle* 为绕 x 轴旋转角度，*GantryAngle* 为绕 y 轴旋转的角度。则旋转矩阵 M_R 如式*，假定探测器静止，模体相对于探测器旋转，因此三个欧拉角取负号。

$$\begin{aligned}
 M_R = & \begin{bmatrix} \cos(-InPlaneAngle) & -\sin(-InPlaneAngle) & 0 & 0 \\ \sin(-InPlaneAngle) & \cos(-InPlaneAngle) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 & \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-OutOfPlaneAngle) & -\sin(-OutOfPlaneAngle) & 0 \\ 0 & \sin(-OutOfPlaneAngle) & \cos(-OutOfPlaneAngle) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 & \times \begin{bmatrix} \cos(-GantryAngle) & 0 & \sin(-GantryAngle) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-GantryAngle) & 0 & \cos(-GantryAngle) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.2)
 \end{aligned}$$

辐射源、模体中心、探测器中心的相对位置由 X, Y 方向 4 个参数和 Z 方向两个参数确定，它们分别为 X, Y 方向的源相对模体的偏置 *SourceOffsetX* , *SourceOffsetY* , 探测器相对模体的偏置 *DetectorOffsetX* , *DetectorOffsetY* 以及 Z 方向的源到模体距离 *SourceToIsocenterDistance (SID)* , 源到探测器的距离 *SourceToDectorDistance (SDD)* 2 个参数确定，如图 4.7。


图4.7 三维重建射影几何^[20]

因此，将旋转和平移变换结合的投影矩阵 M_p 定义为

$$M_p = \begin{bmatrix} 1 & 0 & \text{SourceOffsetX} - \text{ProjectionOffsetX} \\ 0 & 1 & \text{SourceOffsetY} - \text{ProjectionOffsetY} \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} -\text{SourceToDetectorDistance} & 0 & 0 & 0 \\ 0 & -\text{SourceToDetectorDistance} & 0 & 0 \\ 0 & 0 & 1 & -\text{SourceToIsocenterDistance} \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & -\text{SourceOffsetX} \\ 0 & 1 & 0 & -\text{SourceOffsetY} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times M_R \quad (4.3)$$

除特殊声明，本文三维重建算法测试的射影几何取 4 个 X,Y 方向的偏置为零， $SID=1000mm$ ， $SDD=1536mm$ 。

4、反投影权重

对一个点源做投影、反投影运算（投影/反投影）后的结果呈移动不变的星状图案^[4]，称为点扩散函数(PSF)。PSF 是反投影值与真实值间的修正系数，可以证明，在二维成像时，投影/反投影的 PSF 为 $\frac{1}{r}$ ，三维的 PSF 为 $\frac{1}{r^2}$ ， r 表示点源到模体中心的距离。

在齐次坐标下，位于(X, Y, Z)的三维点可以通过以下映射关系投影到二维像点(u, v)

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = M_p \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (4.4)$$

其中 M_p 是对应的投影矩阵, w 为与反投影权重相关的常数。对于体素 (X, Y, Z) 其投影结果的齐次表达为 $[uw, vw, w]$, 表示 (X, Y, Z) 的投影射线与平面 $z = w$ 的交点, w 为不影响投影结果的高维信息, 可取任意非零常数。为方便计算反投影权重 $\frac{1}{r^2}$, 对投影矩阵做相应的等比例变换, 使模体中心 $(0, 0, 0, 1)$ 的投影结果为 $(0, 0, 1)$, 即反投影权重为 $\frac{1}{r^2} = 1$, 则对任一体素 (X, Y, Z) , 利用变换后的投影矩阵可以求出其投影值为 $[\frac{uw'}{w'}, \frac{vw'}{w'}]$, 反投影权重为 $\frac{1}{w'^2}$ 。

5、投影值的计算

距离驱动算法的另一核心环节是计算重叠区域对应的投影值。

二维算法中取像素左右边的中点作为边界点, 并判断两点在投影后的位置顺序。在三维算法中, 取公共平面中各投影位置在 u 方向的最大最小值, 以及 v 方向的最大最小值, 即投影位置的外接矩形作为计算反投影权重的参数。在二维算法实现时需要考虑的是否出界的问题, 在三维算法中同样需要进行特殊的处理, 原理大同小异。

此外, 按照理论分析反投影时权重, 二维算法中所使用的重叠长度在三维算法中应推广为重叠面积。不过需要注意的是, 当交点投影在同一直线上时, 重叠面积为 0, 但并不意味着体素此时对投影的贡献为零, 如图 4.8, 此时应改用重叠长度计算反投影权重, 在程序中加入布尔变量 `flagu`, `flagv` 记录沿 u , v 方向的退化情况。

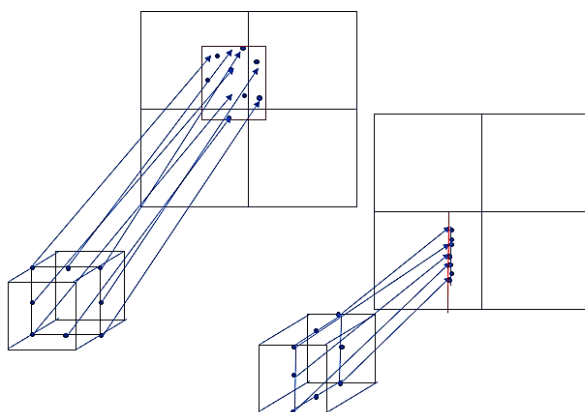


图4.8 三维距离驱动算法几何示意图1

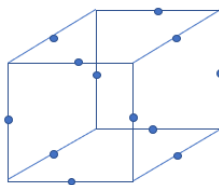
三维距离驱动算法伪代码为:

```

for projs
    for voxel
        for border
            compute borders' projection
            update min & max u & v value
        calculate BP weights
        BP
    end
end
end

```

按照文献中的建议,选取各体素中 12 条楞边的中点作为边界点,如图 4.9。



12个楞边中点为各体素的边界点

图 4.9 三维距离驱动算法几何示意图 2

在求解体素边界点的投影点时,将处在位置 (X,Y,Z) 的三维点投影到二维像点 (u,v) 满足的映射关系由矩阵表达式展开为三个方程进行编程求解,即

$$\begin{aligned}
 wu &= \text{matrix}[0][0] \times X + \text{matrix}[0][1] \times Y + \text{matrix}[0][2] \times Z + \text{matrix}[0][3]; \\
 wv &= \text{matrix}[1][0] \times X + \text{matrix}[1][1] \times Y + \text{matrix}[1][2] \times Z + \text{matrix}[1][3]; \\
 w &= \text{matrix}[2][0] \times X + \text{matrix}[2][1] \times Y + \text{matrix}[2][2] \times Z + \text{matrix}[2][3];
 \end{aligned}
 \quad (4.5)$$

利用方程三将 w 值带入方程一、方程二,从而求解出二维像点坐标 (u,v) 。在获得

投影矩阵前, RTK 会对矩阵进行归一化操作, 使投影图的中心点的反投影权重为 1, 因此, 在计算各体素反投影权重是, 相应的权重为

$$BPWeight = \frac{1}{w^2 \times \sum area} \quad (4.6)$$

式中 w 为边界点中心的体素三维坐标带入映射关系方程所得的 w 值, $area$ 为边界投影点的外接矩形与探测器各单元的重叠面积, 探测器单元的面积为单位 1。 $\sum area$ 即外接矩形的面积。

4.3.2 距离驱动算法的改进

RTK 中自带体素驱动算法 VoxelBased 的实现, 利用迭代器遍历整个体素实现反投影。通过几何分析可知, 在 RTK 3D 圆轨道投影几何坐标系下, 在探测器的旋转轴平行于 Y 轴或 X 轴时, 投影矩阵在 $matrix[1][0], matrix[2][0]$ 或 $matrix[1][1], matrix[2][1]$ 处分别为零, 如图 4.10。这时不同边界点的投影位置有紧密联系, 可以利用这层联系减少计算量。以探测器绕 Y 轴旋转为例, 当体素沿 Y 轴方向遍历的时候, 投影点 v 坐标不变, u 坐标等间隔增大 $u += Y \times du$ 。因此在这种情况下, 抛弃 ITK 迭代器, 换一种遍历体素的方法, 可以显著降低矩阵乘法的次数, 对于每一个投影角度, 遍历体素的伪代码如下:

```
For k
  For i
    j=0
    计算v方向投影位置
    for j
      计算u方向投影位置
      v方向投影点位置不变, u方向投影点位置 + du
      反投影
    end j
  end i
end k
```

在边界点选取时, 同样可以利用 $j++$ 时的优秀特质, 减少需计算的边界点数量, 选取体素中心 xoz 平面内的 4 个棱边中点和 4 个面心, 对于四个面心处的点, 可通过只变换 y 坐标的方式获得前后面 8 个楞边中点。因此通过 8 个点即可获得 12 个楞边中点的投影位置信息, 显著减少计算量。

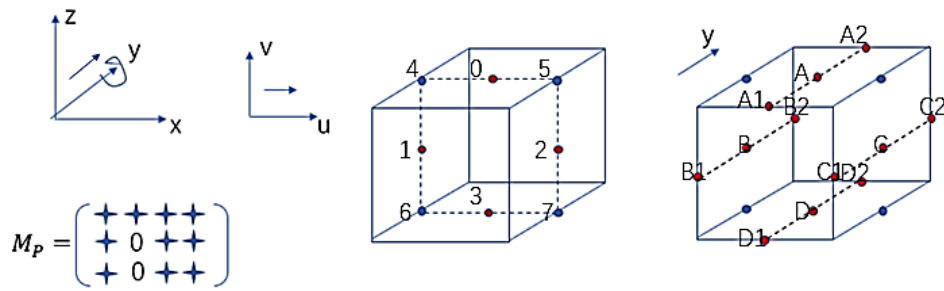


图 4.10 三维距离驱动算法几何示意图 3

根据绕 Y 轴旋转时投影位置变换的规律对 8 个点的投影位置进行分组计算，0-3 号为第一组，4-7 号为第二组，八个点的位置与编号的对应关系如图 4.10 所示。因为第一组选点 A-D 与对应的棱边中点 A1,A2-D1,D2 仅 y 坐标变化，所以各面心在 u 方向的投影位置与对应楞边中点在 u 方向的投影位置相同，在 v 方向上与对应点相差 $\pm du$ ，du 的推导过程如下：

$$wu = M_p(1,:) \frac{Y}{Z}, \quad w(u+du) = M_p(1,:) \frac{Y+1}{Z}, \quad w = M_p(3,:) \frac{Y}{Z} \Rightarrow du = \frac{M_p(1,2)}{w} \quad (4.7)$$

首先遍历 Y 方向的循环外层计算 8 个点在 v 方向的投影位置极值 maxv, minv，即 12 个边界点的投影位置 v 方向极值。然后将 8 个点分两组计算出 12 个边界点 u 方向投影位置的极值，代码如下：

```
for (unsigned int index = 0; index < 4; index++)
{
    tw = BorderP[index][0] + float(j + 0.5) * BorderP[index][2];
    minu = minu < tw ? minu : tw;
    maxu = maxu > tw ? maxu : tw;

    tw = BorderP[index][0] + float(j - 0.5) * BorderP[index][2];
    minu = minu < tw ? minu : tw;
    maxu = maxu > tw ? maxu : tw;
}
for (unsigned int index = 4; index < 7; index++)
{
    tw = BorderP[index][0] + j * BorderP[index][2];
    minu = minu < tw ? minu : tw;
    maxu = maxu > tw ? maxu : tw;
}
```

在大规模并行计算时使用单线程是极为耗时的行为。为充分利用 CPU 的计算能力，缩短重建时间，选用多线程运行程序。ITK 中的线程分割方式，是按所设置的线程数将

重建的体素分割成若干份,每个线程处理一份。当分割的份数超出逻辑处理单元的个数时,则每个线程处理完一份后,再处理本批次的另一份,直到执行完毕。在 4 核 8 逻辑处理单元的 CPU 环境下,每次最多同时反投影时的线程数设置处理 8 份数据,因此本文将线程数设置为微处理器的处理上限 8。

表 4.1 为距离驱动方法优化前后的重建时间比较,Distance-driven Opti.表示用 8 点代替 12 点的优化(Optimized)方法。经优化后,算法耗时缩减为原来的 10 倍,提速明显,但由于在 CPU 上以 DEBUG 编译模式运行,耗时不能令人满意,需要进一步提速。

表 4.1 距离驱动方法优化前后的重建时间比较

方法	驱动方式	硬件环境	线程数	重建一幅投影图的耗时	PSNR/dB
Distance-driven	距离驱动	CPU	8	15s	31.01
Distance-driven Opti.	距离驱动	CPU	8	1.5s	31.01

4.3.3 距离驱动算法的 GPU 加速

CPU 与 GPU 有着完全不同的设计理念,CPU 是面向延迟的设计,其算术逻辑单元(ALU)能处理各种各样的任务,相反,GPU 的每个 ALU 只能实现单一的功能。一个形象的比喻,CPU 的 ALU 就像科学家,每个都是万能的,但数量少,GPU 的 ALU 则像流水线上的工人,每个人干的活都是一样的,但数量庞大。GPU 是面向吞吐量的设计,一个 GPU 可以有数百个内核而 CPU 通常不超过四个内核和八个逻辑处理单元。CPU 和 GPU 的架构如图 4.11 所示:

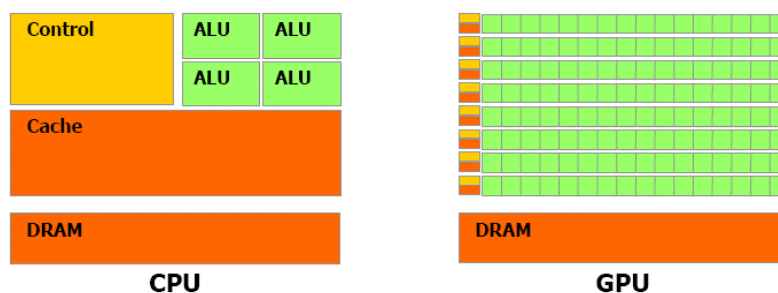


图 4.11 CPU 与 GPU 的架构比较^[10]

三维重建算法的 GPU 加速,除对算法本身并行性的需求高之外,对 GPU 计算资源的充分利用是提速的关键,需将全局存储器(Global Memory)、纹理存储器(Texture Memory)、共享存储器(Shared Memory)、常数存储器(Constant Memory)、线程私有的寄存器(Register)、块(Block)的划分配置以及线程同步(Thread Synchronization)与算法特性紧密结合。

通用计算图形处理器(GPGPU)核心思想是每个线程干同样的事情,用不同的数据。距离驱动算法中每个线程的工作为计算线程索引(Index)对应的体素(voxel)的反投影值。由于距离驱动算法与像素驱动算法除反投影实现细节外,整体框架相同。因此,距离驱动算法的 CUDA 版本参照 RTK 中像素驱动算法的 GPU 加速进行实现。下面对于实现技巧做简要阐述:

1、共享内存的使用

共享存储器被一个线程块中所有线程共用,常用于保存公用计数器和线程块中公用的结果等。在距离驱动算法中,沿 y 方向的各体素,其边界点投影后在 v 方向的投影位置极值 $\min v(j), \max v(j)$ 与 $j=0$ 时结果相同

$$\min v(j) = \min v(0), \max v(j) = \max v(0), j = 0, 1, 2, \dots, N-1 \quad (4.8)$$

各线程块中的线程在计算其体素的反投影值和面积时要调用 $\max v, \min v$ 各 4 次,因此将这两个调用频繁的公用结果存入共享内存中。

用 $j = 0, \dots, N-1$ 表示当前线程块处理的体素标签, j_0, j_1, \dots, j_7 表示第 j 个体素的 8 个边界点,几何关系见图 4.8。 j_0, \dots, j_3 可计算出 8 个棱边中点对应的 u 方向投影位置

$$u(j_{i_1}) = u(0_i) + (j + 0.5) \times du, u(j_{i_2}) = u(0_i) + (j - 0.5) \times du, i = 0, \dots, 3 \quad (4.9)$$

其余 4 个棱边中点的 u 方向投影位置为

$$u(j_i) = u(0_i) + j \times du, i = 4, \dots, 7 \quad (4.10)$$

du 计算见式 (4.7)。因此 $j=0$ 时 8 个点 $0_0, 0_1, \dots, 0_7$ 对应的投影位置信息 u, v, du 也存入到共享内存中。

2、线程块与线程同步

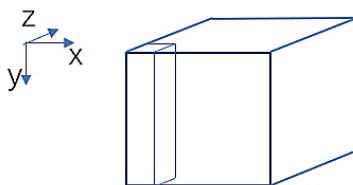


图 4.12 线程块示意图

线程块提供了线程同步的可能性,根据距离驱动优化算法中沿 $j++$ 方向遍历体素的特点,设置三维线程块深度为 $1 \times 256 \times 1$,如图 4.12。

线程同步是为了防止因线程间执行速度差异导致运算出错,对于每个线程块中的线程需进行两次线程同步操作,如图 4.13:第一次同步设置在体素 $j=0$ 完成对 $\text{minv}, \text{maxv}, \text{BorderP}[8][3]$ 的运算之后,再进行后续各体素的投影位置和反投影值计算。第二次同步设置在处理下一幅投影图之前,因为各线程处理完一幅投影图的时间有先有后,如果 $j=0$ 所在线程率先完成运算,进入下一次循环,则会在处理新的投影图时更新 $\text{minv}, \text{maxv}, \text{BorderP}[8][3]$ 等数据,而停留在上一轮的线程还需要读取 minv, maxv 的值,而此时两个极值已经不是上一幅投影图对应的投影位置,而是下一轮更新后的数值,所以必须在处理新投影图之前设置一次线程同步。

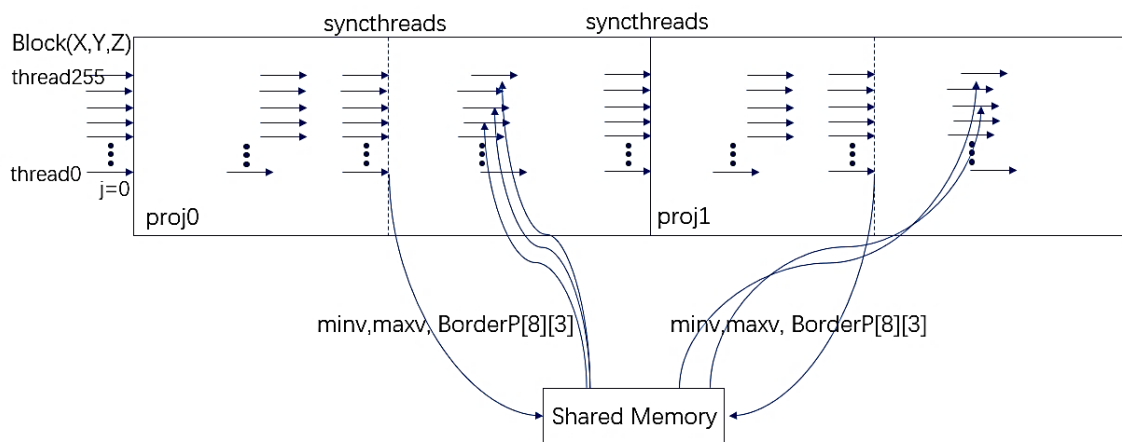


图 4.13 线程同步示意图

3、纹理内存的使用

在距离驱动算法中,要频繁的调用投影数据根据重叠区域信息计算反投影值,因此将投影数据存入到纹理存储器中,其优势在于将临近的数据存入缓存,提高方位物理位置相近的数据的速度。

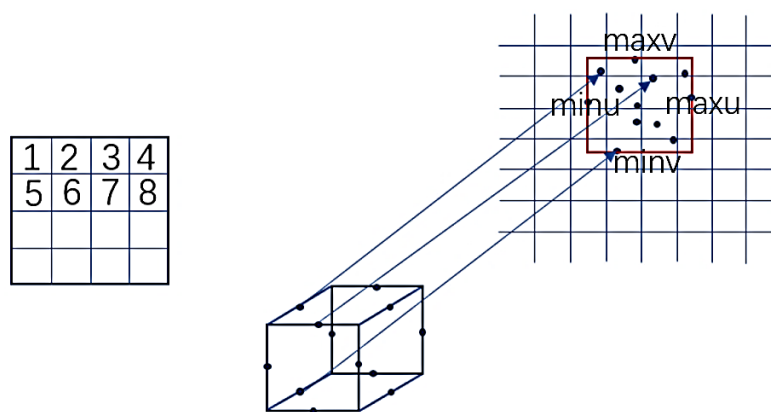


图 4.14 纹理存储在获得投影值时的优势

一个很直观的例子，图 4.14 的矩阵中，1、5 这两个数据在行优先的线性存储中并不具有相邻的物理地址，因此连续索引这些数据效率是低下的；但是纹理存储可以在读取其中某个数据时将临近的值载入缓存，这样下次访问时，则可以直接命中缓存，减少对 Global Memory 的访问，从而提高效率。

表 4.2 呈现了 GPU 加速前后的重建时间比较，Voxel-based 为基于体素驱动的 CPU 多线程反投影方法，CudaBackProjection 为该方法的 GPU 版本，Distance-driven Opti. 为基于 8 点的距离驱动优化算法，CudaDistance-driven Opti. 为该方法的 GPU 版本。经 GPU 加速后的体素驱动和距离驱动算法耗时均大幅减小。

表 4.2 反投影算法在 GPU 加速前后的重建时间比较

方 法	驱动方式	硬件环境	重建一幅投影图的耗时
Voxel-based	体素驱动	CPU	16.7ms
CudaBackProjection	体素驱动	GPU	6.5ms
Distance-driven Opti.	距离驱动	CPU	1.5s
CudaDistance-driven Opti.	距离驱动	GPU	97ms

4.4 本章小结

本章对三种驱动方式的实现进行说明，重点阐述距离驱动算法的实现。

像素驱动和体素驱动利用 RTK 中的类进行实现。

二维距离驱动算法在自定义的几何模型下，以 x 轴为公共轴，对每幅投影图，遍历模体体素计算反投影值，并将所有投影在同一体素位置的计算结果累加。边界点的投影坐标由相似三角形求解，且利用相邻体素边界点重合的特质可以减少投影位置计算量。为确保投影位置在探测器接受范围内，对投影位置做出界判断，最终根据反投影权重进



行模体重建。

三维距离驱动算法在 RTK 几何模型下，以探测器为公共平面，对各体素的 12 条棱边的中点，利用投影矩阵计算投影位置，随后根据反投影权重进行重建。当探测器旋转轴平行于 X/Y 轴时，投影点的位置有规律可循，利用这一特质调整遍历方式，并将选点方式改为 8 点，优化后重建速度提升 10 倍。

为增加驱动方式间的可比性，对距离驱动进行 GPU 加速，并于 Cuda 版本下的体素驱动、像素驱动进行算法比较。距离驱动的 Cuda 版本根据算法特性使用了共享内存和纹理内存，加速比为 15 倍。

5 算法性能分析

本章在 FDK 框架下比较体素驱动、像素驱动和距离驱动算法的重建速度、精度及算法的鲁棒性，并分析讨论算法并行化难易程度和各类重建伪影的产生原因。

Windows 系统下对各反投影算法进行测试，原模体与三种反投影驱动方式重建结果如图 5.1 所示。图中展示的是轴状位第 136 幅切片，窗位为 1，窗宽 0.2。

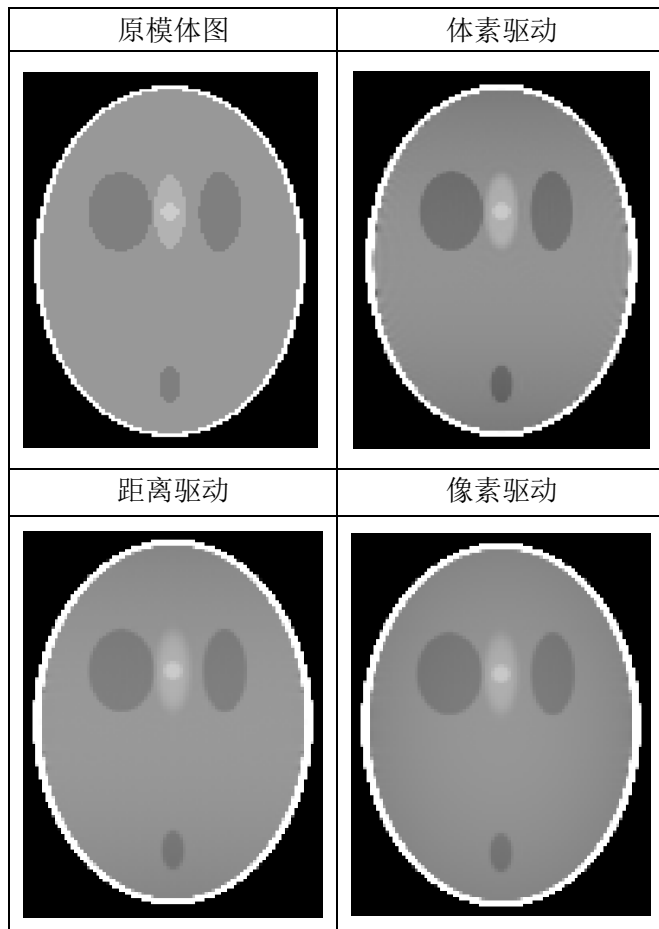


图 5.1 原模体及不同反投影方式重建结果展示

5.1 重建速度

算法的时间比较要建立在硬件环境相同的前提下，如表 5.1。像素驱动的原理导致很难进行并行化处理，在 CPU 下无法使用多线程并行计算，运行效率低于体素驱动。即使在 GPU 上运行，由于其算法的复杂度极高，依然比体素驱动慢 7.7 倍。

距离驱动算法经 GPU 加速后耗时缩减 15 倍，重建速度介于像素驱动和距离驱动之间。

在 GPU 加速方面，三种驱动方式的并行化难易程度各异，有不同的处理技巧。

表 5.1 重建时间比较

方法	驱动方式	硬件环境	重建一幅投影图的耗时
Voxel-based	体素驱动	CPU	16.7ms
CudaBackProjection	体素驱动	GPU	6.5ms
Distance-driven Opti.	距离驱动	CPU	1.5s
Cuda Distance-driven Opti.	距离驱动	GPU	97ms
CudaRayCast	像素驱动	GPU	129ms

体素驱动方法由于其低复杂度,被广泛用于执行 CBCT 前向和后向投影。当算法从 CPU 转换到 GPU 时,由于并发线程以分散的方式将数据写入 GPU 存储器中,这种操作可能会导致线程间干扰(线程竞争)问题。由于散射操作(Scatter Operations)的性质,体素驱动的反投影很容易被 GPU 加速,但其前向投影的实现是非并行的。论文[2]中通过在不同的投影角度使用不同的线程平面方向的方式,利用锥束几何来缓解线程干扰问题。

射线驱动方式不适合在 GPU 上执行,因为一个体素 V_i 的衰减系数 μ_i 求解受多个射线 α^j 影响,如果对体素的衰减系数求解做并行化处理,则需要同时访问多个射线对 μ_i 进行修正,而 GPU 不支持对同一内存位置的并发写入访问。

为了解决并发写入的问题,希望核函数能够一次性计算出体素的衰减系数,从而实现体素级并行化。[31]中给出了判断 V_i 是否被 α^j 穿过以及交线长度的计算方法,根据公式(3.6)分母的求和过程需要对所有 j 进行求解,对所有射线进行计算从而求出所有穿过 V_i 的交线长度。

虽然上述处理方式解决了并发写入的问题,但繁琐的交线计算仍然需要顺序执行。GPU 处理器不适宜执行大型串行计算,对于复杂的串行计算, GPU 通常会直接终止程序运行,因此需要减少需要计算的射线数量。论文[32]提出有效射线的选择策略,其主要思想是找到位于 V_i “阴影”中的投影图像素,经过体素八个角点的射线在投影图中形成“阴影”,只对此“阴影”范围内的射线进行计算。公式(3.6)变为:

$$\mu_i = \frac{\sum_{j \in \Omega^i} l^{ij} p_j}{\sum_{j \in \Omega^i} l^{ij}} \quad (5.1)$$

式中 Ω^i 表示位于 V_i 形成的“阴影”区域内的射线。

距离驱动与体素驱动的框架类似,各体素重建时核函数运行时间较长,需要针对算法细节进行优化。如何进一步提升 GPU 加速比是进一步研究工作的关键。

5.2 重建精度

重建图像中 CT 数与物质真实吸收系数间的差异称为伪影。对仿真投影图进行重建时不同部位的伪影都不同原因造成。

分辨力不足导致的伪影如图 5.2, (a)为投影数量不够导致的伪影, (b)为探测器分辨率不够(间隔大)导致的伪影。

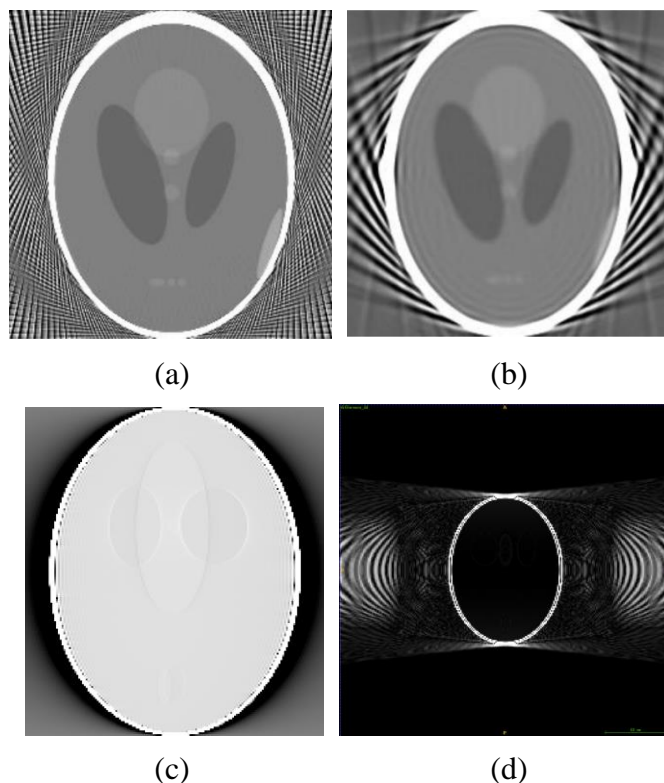


图 5.2 伪影示例

图(c)为经过加权滤波操作后的第 90 幅投影图, 将窗位设为 0, 窗宽设为 0.02 后, 可以清楚看到在密度对比度高的边缘位置形成与边缘线相切的黑白条纹。由于源和探测器单元不是理想的点, 使测得的原始数据和理想条件下对模体的投影数据存在差异, 这种伪影称为幂指数的边缘梯度效应 (Exponential edge gradient effect^[36]), 通过提高采样密度能够减轻此类伪影, 论文[36]中提出了几种补偿伪影的方法, 但由于奈奎斯特采样定理的限制, 无法完全消除伪影。

投影数据的空间分布是有限的, 因此其频带无限。但在实际运算时, 频带不能非周期无限长, 需截断高于 $1/2d$ 的频谱 (d 为采样间隔), 从而造成频谱混叠, 如图 5.3。频域滤波导致的伪影如图(d), 是在模体外围分布的条纹。

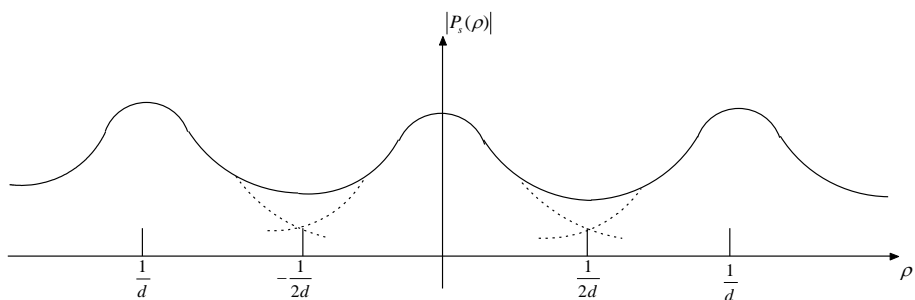


图 5.3 阐明投影定理的频谱混叠

通过给斜坡滤波函数加窗能够有效抑制频谱混叠,用于加窗的滤波函数有 $R-L$ 滤波器, $S-L$ 滤波器等,它们的离散表示如图 5.4。由于高频信号中有用信息较少,对高频的截断基本不会丢失重建模体所需的信息。

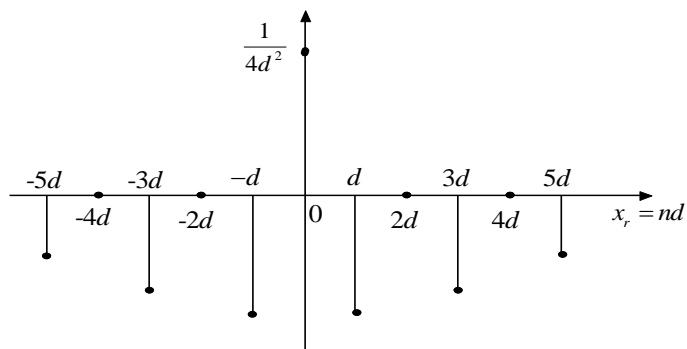
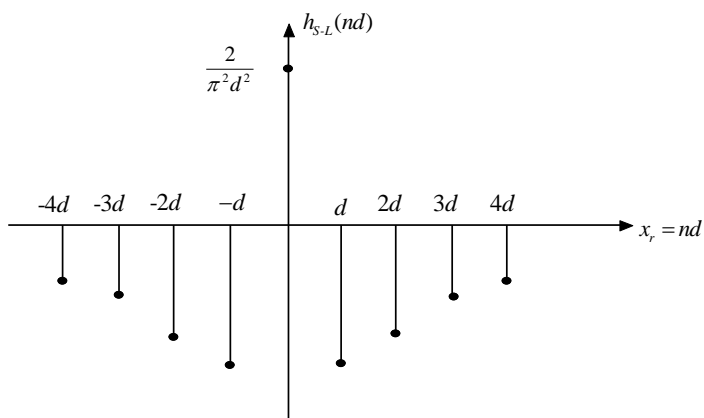

(a) $R-L$ 滤波函数离散表示

(b) $S-L$ 滤波函数离散表示

图 5.4 滤波函数

图像重建质量的需求与医疗设备生产商的技术水平相关,相关技术的突破会改变客户对图像质量可接受性的要求,但就目前而言,如何减少伪影仍是医学图像重建的研究重点。

表 5.2 重建精度比较

反投影方式	体素驱动		像素驱动	距离驱动	
	Voxel-based	CudaVoxel-based	CudaRayCast	Distance-driven Opti.	CudaDistance-driven Opti.
PSNR/dB	30.51	30.54	31.46	31.01	31.27

表 5.2 中展现了各驱动方式的 PSNR 值，观察同一种驱动方式的不同版本可发现，Cuda 版本的重建精度 PSNR 比 CPU 高 0.03 左右，这是因为 Cuda 版本中将投影图存储在纹理内存中，其纹理拾取的硬件差值方式类似近似的双线性插值，优于 CPU 版本中使用的最近邻插值。

比较不同驱动方式的重建精度可得，体素驱动的重建质量最低，距离驱动重建结果稍好，像素驱动重建质量最高。像素驱动是用时间换取与实际物理过程十分接近的投影反投影权重的三次线性插值方法，重建精度最高不难理解。距离驱动算法重建质量优于体素驱动的原因在于，其利用重叠长度或面积作为反投影权重的方法比体素驱动中利用线性插值的方式计算反投影数值的方法更加符合实际的物理投影过程，换言之，距离驱动算法从理论上来说不会在反投影阶段因算法原理本身而引入误差，即没有人造伪影。

如图 5.5 所示，以一维平行束投影时的情形为例， s_i 为辐射源第 i 个单元的中心， d_i 为探测器第 i 个单元的中心， s_{ij}, d_{ij} 为相应的边界点投影位置。将投影过程等同于对探测器单元进行填色的过程，而经各体素衰减后的射线强度等同于上色的涂料。距离驱动算法根据边界的位置，将 s_1 的 $\frac{s_{12}-d_{11}}{s_{12}-s_{11}}$ 和 s_2 的 $\frac{d_{12}-s_{21}}{s_{22}-s_{21}}$ 填入 d_1 中，此过程完全符合实际的物理投影情况。而体素驱动算法利用线性插值作为反投影权重，即体素 s_2 根据 d_1, s_2, d_2 间的位置关系将其 $\frac{d_2-s_2}{d_2-d_1}$ 赋给探测器单元 d_1 ，则与实际物理过程填入的 $\frac{d_{12}-s_{21}}{s_{22}-s_{21}}$ 相差一个红色区域，从而造成人为引入的误差。

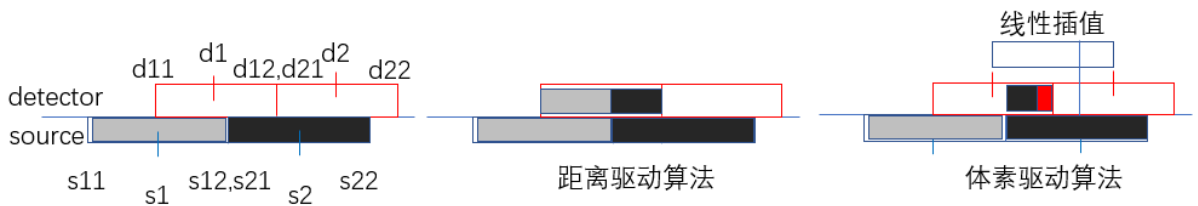


图 5.5 算法性能差异示意图 1

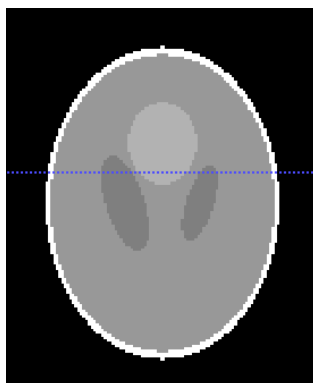


图 5.6 截线位置示意图

取三种驱动方式所得重建图像与原模体在(1,122,140)至(256,122,140)。体素坐标下的密度值进行分析，截线段如图 5.6 所示，得曲线图 5.7:

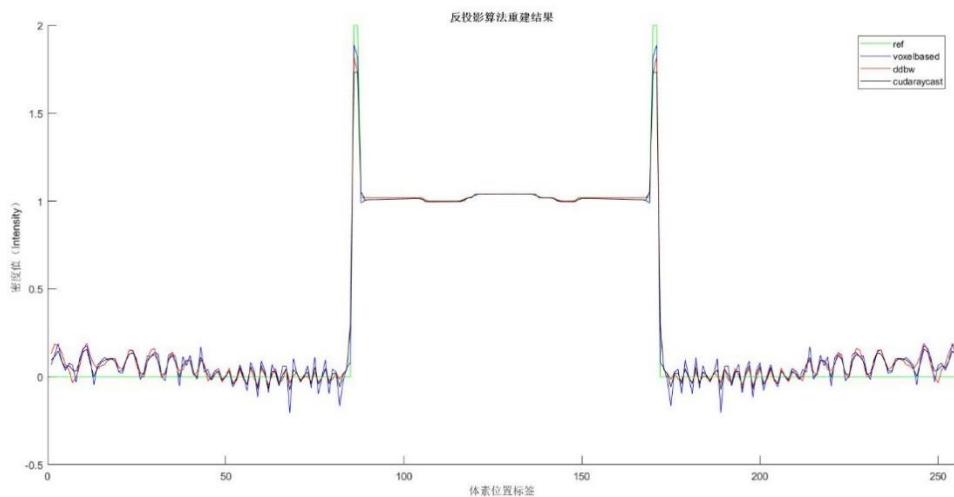


图 5.7 重建结果与原模体的截线段数据比较 1

其中，绿色曲线 ref 为原模体数据，蓝色曲线 voxelbased 为体素驱动数据，红色 ddbw 为距离驱动，黑色 cudaraycast 为像素驱动。下面阐述曲线各部分细节的规律，并分析可能的原因。

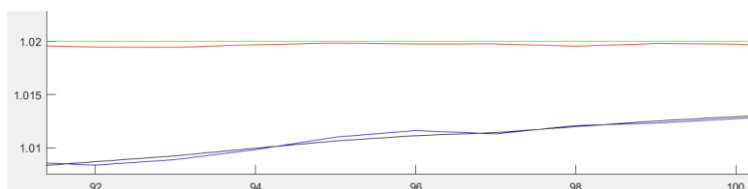


图 5.8 重建结果与原模体的截线段数据比较 2

如图 5.8，在密度值平稳的区域，距离驱动重建结果非常接近真实值，体素驱动和像素驱动稍差。这是因为距离驱动的数学模型更接近真实物理投影，而线性插值会引入人

为误差，如前文所述。

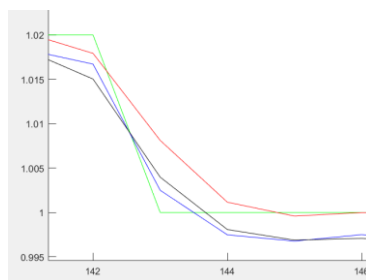


图 5.9 重建结果与原模体的截线段数据比较 3

如图 5.9，在密度值变化斜率大的时候，如模体的轮廓处，距离驱动算法惯性较大，重建结果偏差大，体素驱动和像素驱动较为准确。这种误差是仪器本身的空间分辨率引起的，同样以填色过程做类比，探测器单元接收的信息是经过相邻模体衰减后的射线强度的叠加，衰减系数变化大的区域，填色结果与该区域的密度极值偏差大，回抹的过程中误差就大，即图 5.9 中红色曲线所呈现的惯性大的现象。体素驱动和像素驱动使用线性插值的方式虽然与实际物理投影过程有偏差，但在密度值变化梯度大的区域反而比实际探测器接收到的能量更能反应出衰减系数的分布情况，相当于通过反投影算法，对探测器空间分辨率不足造成的误差做了一定程度的补偿，如图 5.10。

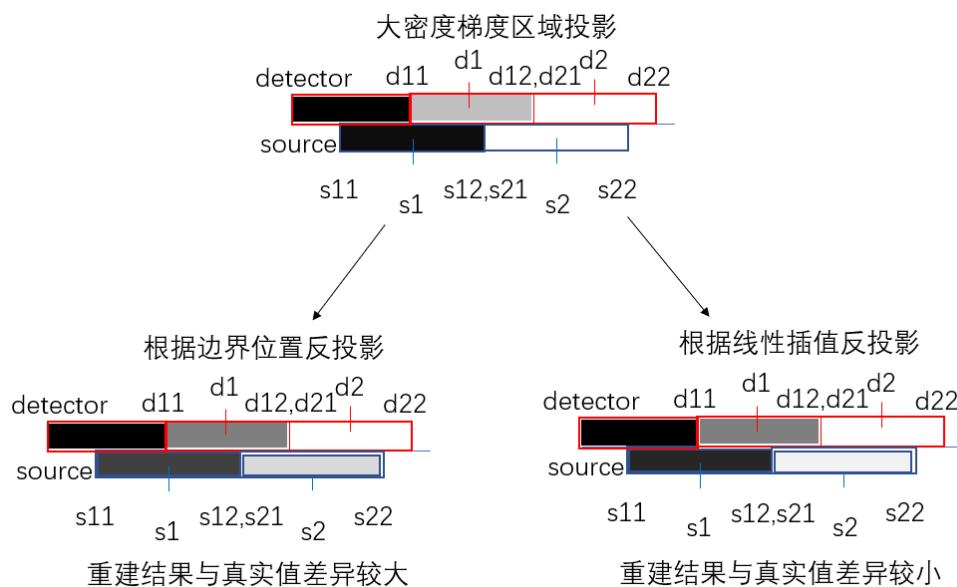


图 5.10 算法性能差异示意图 2

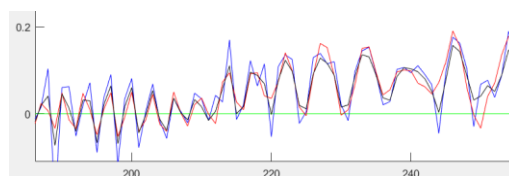


图 5.11 重建结果与原模体的截线段数据比较 4



对于所有重建算法，在重建模体外围区域，如图 5.11，重建结果呈震荡式误差分布，这是由于 FFT 频域变换时对高频信息做截断处理的缘故。

重建图象的误差分布图详见附录 B2。

5.3 算法鲁棒性

加性高斯白噪声可以建模为 $I=I_0+N$ ，其中 I 是加噪后的图像， I_0 是无噪声的原图像， N 是正太分布随机变量，均值为 μ ，方差为 σ^2 ， $N \sim N(\mu, \sigma^2)$ 。噪声与像素强度无关。设均值为 0，标准偏差为 1。

表 5.3 鲁棒性比较

方法	驱动方式	PSNR/dB	加入噪声后 PSNR/dB	PSNR 变换情况
CudaBackProjection	体素驱动	30.54	30.072	0.468
CudaDistance-driven Opti.	距离驱动	31.27	31.16	0.11
CudaRayCast	像素驱动	31.46	31.305	0.155

表 5.3 为三种驱动方式的鲁棒性比较，实验结果表明，距离驱动算法鲁棒性明显优于体素驱动算法，像素驱动算法鲁棒性介于两者之间。重建算法的鲁棒性测试结果图详见附录 B3。

5.4 本章小结

本章对三种驱动方式下 FDK 重建算法在重建速度、重建精度和鲁棒性方面的性能进行了分析。重点分析了三种驱动方式的计算复杂度和并行化难易程度，以及不同重建伪影的产生原因。

综合来看，与像素驱动相比，距离驱动的重建精度下降 0.6%，重建时间提升 24%，加入噪声后的重建质量下降程度减少 0.14%。与体素驱动相比，距离驱动的重建耗时增加 14 倍，重建精度增加 1.16%，加入噪声后的重建质量下降程度减少 1.15%。因此距离驱动的 FDK 算法是重建精度较高，重建速度适中，鲁棒性最优的驱动方式。



结论

论文首先阐述 CT 图像重建的数学原理,引出投影和反投影的定义,重点分析了傅立叶中心切片定理并引入频域滤波的概念,在此基础上讨论二维反投影重建的原理。平行束反投影前需进行频域滤波消除星状伪影,而扇形束通过余弦加权能够变换成平行束重建问题。

论文接下来讨论了锥形束 CT 的重建算法,根据 Radon 变换,FDK 重建算法因不满足完全轨道条件只可近似重建,在锥角小于 10° 时重建结果可接受。FDK 算法中耗时最大的反投影环节,反投影有体素驱动、像素驱动和距离驱动三种驱动方式。体素驱动重建速度快,但由于通过双线性插值求解投影值的过程与实际投影物理过程存在差异,体素驱动存在人造伪影。像素驱动算法根据物理投影模型建立,重建精度最高,但计算量大,且各体素的重建难以并行化,从而难以充分利用设备的计算能力。距离驱动算法将体素和探测器的边界点投影到公共平面上,既能满足与物理投影接近的投影模型,又可保证体素重建的相互独立。

论文涉及各驱动算法的实现方法,重点阐述距离驱动算法从二维到三维的实现过程以及 GPU 加速的实现要点。在距离驱动算法研究中发现,当旋转轴与坐标轴 X 或 Y 平行时,利用投影矩阵的优良特质,可以在图像质量不变的情况下大幅减小矩阵乘法运算量,且这种距离驱动框架可对进一步探究选点的个数、几何位置等对重建速度和精度的影响提供方便。

论文还对三种反投影算法的图像重建速度和质量进行了比较分析。体素驱动算法重建速度快,精度低。像素驱动算法重建速度慢,精度高。距离驱动算法的速度和精度介于体素驱动和像素驱动之间。耗时上体素驱动最快,距离驱动次之,像素驱动最慢。距离驱动算法的优化方法使重建速度提升约 10 倍,GPU 加速可进一步提升 15 倍耗时。对投影加入高斯噪声,测得体素驱动鲁棒性低于像素驱动,距离驱动对噪声的鲁棒性最好。

虽然距离驱动的优化算法提速明显,但实际临床检测中不能达到旋转轴与 X 或 Y 坐标轴的绝对平行,因此如何提升距离驱动优化算法的鲁棒性,或寻找补偿方式的数学模型,也是今后研究的一个方向。GPU 加速对不同硬件环境有不同的最优配置,如何自适应得到这些配置参数是今后研究的方向。此外,GPU 的强大并行计算能力使得时间约束减少,可以尝试更复杂的算法计算图像,边界点的选取对重建速度精度的影响以及背后



的数学原理是今后研究的方向。



致谢

作为编程小白,选择这个毕设题目的主要目的之一是锻炼代码能力。正如自己所料,在整个毕设过程中上手很慢,以至进度缓慢不能不减轻原定的任务量。

毕设过程中学到了太多太多,在这里首先感谢我的导师周付根教授对我的鼓励、引导和包容,一个毕设题目就如同一个小的科研项目,从思考要做什么、要怎么做到做成什么样子,周付根老师一直引导我体会解决问题的思路和方法。虽然最终没能完成老师期望的目标着实遗憾,但这过程中收获慢慢,相信是今后研究生阶段继续深造的一块重要垫脚石。

其次,十分感谢毕设过程中郭斌师兄对我的帮助,无论我何时何地请教他何种问题,他都耐心高效的为我解答,师兄分析问题和解决问题的迅捷果断让我羡慕和向往,是我今后长期学习和努力的榜样。

另外,要特别感谢同样大四在读的本科生周宇韬同学对我的帮助和鼓励,尤其在 GPU 加速的优化方法方面给我提供了很多指导性意见。周宇韬同学在完成自己毕设任务之余,力所能及的帮助身边的同学,他过硬的实力和乐于助人的品德得到大家的一致称赞。



参考文献

- [1]温俊海, 王俊贤, 程敬之. X 线断层摄影方法及其新进展[J]. 北京生物医学工程, 1999, 18(2).
- [2] Yi D, Yu G, Xiang X, et al. GPU accelerated voxel-driven forward projection for iterative reconstruction of cone-beam CT[J]. Biomedical Engineering Online, 2017, 16(1):2.
- [3]成旭. 基于小波变换的锥束 CT 快速三维重建算法研究[D]. 山东大学, 2017.
- [4]曾更生. 医学图像重建[M]. 北京: 高等教育出版社, 2010.
- [5] Kress J W, Feldkamp L A, Davis L C. Practical cone-beam algorithm[J]. Josa A, 1984, 1(6):612-619.
- [6]蒋路帆. 锥形束 CT 重建算法研究[D]. 北京: 北京航空航天大学, 2007.
- [7] 韩民, 成旭, 李登旺. 基于小波变换的多分辨率锥束 CT 图像快速三维重建算法[J]. 电子与信息学报, 2017, 39(10):2437-2441.
- [8] Beister M, Kolditz D, Kalender W A. Iterative reconstruction methods in X-ray CT[J]. Physica medica : PM : an international journal devoted to the applications of physics to medicine and biology : official journal of the Italian Association of Biomedical Physics (AIFB), 2012, 28(2):94-108.
- [9] D Maret, N Telmon, OA Peters et al. Effect of voxel size on the accuracy of 3D reconstructions with cone beam CT. [J]. DentoMaxilloFacial Radiology, 2012.
- [10] CUDA C Programming Guide.
<https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>
- [11] Scherl H, Keck B, Kowarschik M, et al. Fast GPU-Based CT Reconstruction using the Common Unified Device Architecture (CUDA)[C]. 2007 IEEE Nuclear Science Symposium Conference Record:4464-4466.
- [12] 程杰, 罗守华. 基于 GPU 统一计算设备架构的 Micro CT 图像重建[J/OL]. 中国科技论文在线精品论文, 2010, 3(8):819-824.
- [13] 李忠华, 周付根, 白相志. 一种基于 GPU 的体积 CT 快速重建算法[A]. 生物医学工程杂志, 2011, 28(2):238-242.
- [14] Rit S, Oliva M V, Brousmiche S, et al. The Reconstruction Toolkit (RTK), an open-source cone-beam CT reconstruction toolkit based on the Insight Toolkit (ITK)[J]. 2014, 489(1):108-108.
- [15] De M B, Basu S. Distance-driven projection and backprojection in three dimensions.[J]. Physics in Medicine & Biology, 2004, 49(11):2463-75.
- [16] Reconstruction[EB/OL].
<http://oftankonyv.reak.bme.hu/tiki-index.php?page=Reconstruction>.
- [17] Turbell H. Cone-beam reconstruction using filtered backprojection [J]. Linköping University, 2001.
- [18] 杨民, 路宏年, 黄朝志. 用查找表和极坐标反投影法实现计算机断层扫描快速重建. 兵工学报, 2004 25(4):476-479
- [19] Stefan H, Michael L, Michael U. Filter design for filtered back-projection guided by the interpolation model. Proceedings of the SPIE International Symposium on Medical Imaging. Image Processing, 2002 4684(II):806-813
- [20] RTK 3D circular projection geometry. <http://www.openrtk.org/Doxygen/DocGeo3D.html>
- [21] Errata for Principles of Computerized Tomographic Imaging.
<http://www.slaney.org/pct/pct-errata.html>
- [22] Forward Projection through Voxel Volumes.
<http://www.isy.liu.se/cvl/edu/TSBB31/download/TurbellPhDCh5.pdf>
- [23] The ITK Software Guide. <https://itk.org/ITKSoftwareGuide/html/>
- [24] Liu R, Fu L, Man B D, et al. GPU-based Branchless Distance-Driven Projection and Backprojection[J].



IEEE Transactions on Computational Imaging, 2017, PP(99):1-1.

[25] Gábor Jakab, Attila Rácz, Kálmán Nagy. High Quality Cone-beam CT Reconstruction on the GPU. ReserchGate uploaded by Attila Rácz on 25 March 2015.

[26] Chen Jian-Lin, Li Lei, Wang Lin-Yuan et al. Fast parallel algorithm for three-dimensional distance-driven model in iterative computed tomography reconstruction. Chin. Phys. B Vol. 24, No. 2 (2015) 028703

[27] Kontos D. Iterative CT reconstruction with small pixel size: distance-driven forward projector versus Joseph's[C]// SPIE Medical Imaging. International Society for Optics and Photonics, 2015.

[28] Miao C, Liu B, Xu Q, et al. An improved distance-driven method for projection and backprojection.[J]. Journal of X-ray science and technology, 2014, 22(1):1-18.

[29] Choi S, Kim Y, Lee H, et al. Imaging characteristics of distance-driven method in a prototype cone-beam computed tomography (CBCT)[C]// SPIE Medical Imaging. 2016:97832U.

[30] 陈建林, 李磊, 王林元, et al. Fast parallel algorithm for three-dimensional distance-driven model in iterative computed tomography reconstruction[J]. Chinese Physics B, 2015, 24(2):513-520.

[31] Kay, T. L., Kajiya, J. T. Ray tracing complex scenes. ACM SIGGRAPH Conference, 269-278 (1986). DOI: 10.1145/15886.15916

[32] Park H G, Shin Y G, Lee H. A Fully GPU-Based Ray-Driven Backprojector via a Ray-Culling Scheme with Voxel-Level Parallelization for Cone-Beam CT Reconstruction[J]. Technology in Cancer Research & Treatment, 2015, 14(6):709-20.

[33] GB/T 7714 高上凯. 医学成像系统[M]. 清华大学出版社, 2010.

[34] Schulze R, Heil U, Groß D, et al. Artefacts in CBCT: a review[J]. Dento Maxillo Facial Radiology, 2011, 40(5):265-73.

[35] 王学礼. 医用 CT 图像常见伪影成因及矫正分析[J]. CT 理论与应用研究, 2004, 13(3):1-8.

[36] Joseph PM, Spital RD. The exponential edge-gradient effect in x-ray computed tomography.[J]. Physics in Medicine and Biology, 1981, 26(3):473-87.

附录

附录 A1. 仿真头模型及仿真数据的产生

在研究从投影重建图像的算法时，为了较为客观的评价重建算法的有效性，人们常选用公认的 Shepp Logan(S-L)头模型作为研究对象。该模型由 10 个位置、大小、方向、密度各异的椭圆组成，图 A1.1 是 S-L 头模型的灰度显示图像。

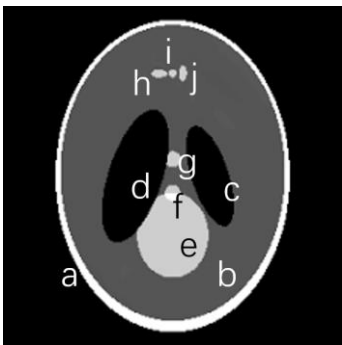


图 A1.1 S-L 头模型

表 A1 给出了 S-L 头模型中 10 个椭圆的中心坐标和密度值。

表 A1 头模型中的椭圆参数

序号	中心坐标	密度
a	(0,0)	2.0(a,b 之间)
b	(0,-0.0184)	1.02(b 内)
c	(0.22,0)	1.0
d	(-0.22,0)	1.0
e	(0,0.35)	1.03
f	(0,0.1)	1.04
g	(0,-0.1)	1.04
h	(-0.08,-0.605)	1.03
i	(0,-0.605)	1.03
j	(0.06,-0.605)	1.03

对原模体 S-L，设置窗位 0.1，窗宽 0.2，取轴状位第 136 层切片，矢状位第 126 层切片，冠状位第 112 层切片。如图 A1.2 所示：

用 S-L 头模型进行计算机仿真研究的主要好处之一是可以获得该模型投影数据的解

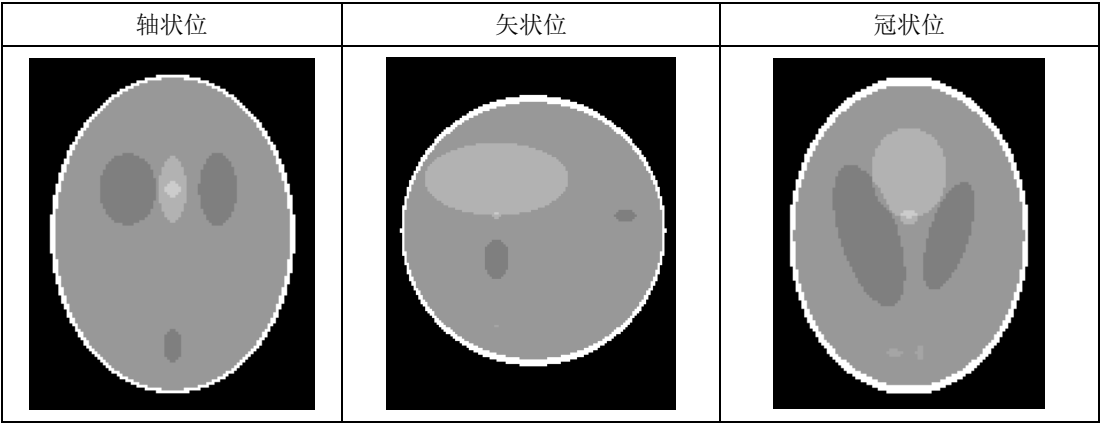


图 A1.2

析表达式。

在 RTK 中提供了求解射线与椭球交点的类 `rtkRayEllipsoidIntersectionImageFilter`，结合椭球密度分布信息，即可求出准确的仿真投影图。

附录 A2. 数字图像质量评估方法

本论文中用到的部分图像评价性能的参数有像素误差(Error Per Pixel)、均方误差(Mean Square Error, MSE)、峰值信噪比 (Peak Signal to Noise Ratio, PSNR)、QI 值和时间(Time)等。

一般地, 在样本量一定时, 评价一个点估计的好坏标准使用的指标总是点估计与参数真值的距离的函数, 最常用的函数是距离的差的绝对值求和以及距离差的平方求和, 由于估计量具有随机性, 可以对该函数求期望, 像素误差和均方误差的定义如下:

$$ErrorPerPixel = \frac{1}{mn} \sum_{i=1}^n \sum_{j=1}^m |K(i, j) - I(i, j)| \quad \text{式(A1)}$$

$$MSE = \frac{1}{mn} \sum_{i=1}^n \sum_{j=1}^m \|K(i, j) - I(i, j)\|^2 \quad \text{式(A2)}$$

峰值信噪比是最为常用的衡量图像失真或噪声水平的指标。定义为,

$$PSNR = 10 \log_{10} \left(\frac{MAX^2}{MSE} \right) \quad \text{式(A3)}$$

其中 MAX 表示图像灰度的最大值, 本论文中取 MAX=255。

2 个图像之间 PSNR 值越大, 则越相似。普遍基准为 30dB, 30dB 以下的图像劣化较为明显。此外 QI 值的定义为,

$$QI = \frac{MAX - ErrorPerPixel}{MAX} \quad \text{式(A4)}$$



附录 A3. 实验环境与重建参数

本文中测试的实验环境如表 A3.1 所示。

表 A3.1 实验环境参数

环境参数	值
处理器	Intel(R) Core(TM) i7-7700HQ CPU @2.80GHz
系统类型	Windows10, 64 位操作系统, 基于 x64 的处理器
CPU 核心数量	内核 4 个, 逻辑处理器 8 个
GPU 型号	GeForce GTX 1050 Ti
全局内存	4094MB
一个线程块中的共享内存	49152Byte
常数内存	65536Byte
一个线程块中的最大线程数	1024
显存频率	1.62GHz
CUDA 核心数量	768

重建的输入输出相关参数如表 A3.2 所示

表 A3.2 重建输入输出相关参数

重建参数	值
原始模体	Shepp-Logan
生成仿真投影图的类	RayEllipsoidIntersectionImageFilter
投影图参数	尺寸 256*256 间距 2 全扫描方式得 180 幅投影图
反投影参数	尺寸 256*256*256 间距 2
射影几何参数	SDD=1536mm, SID=1000mm

附录 B1. 投影数据

模体的投影数据如图 B1.1:

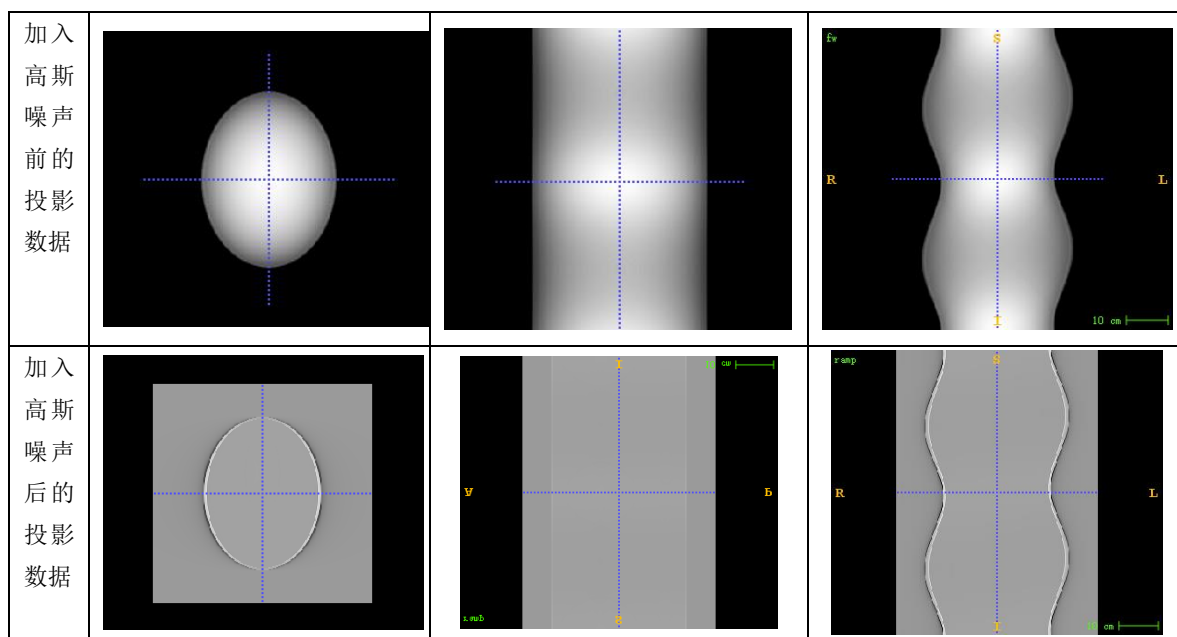


图 B1.1 模体的投影数据

数据说明: 该组数据为由 `rtksimulatedgeometry` 与 `rtkdrawgeometricphantom` 联合生成的 S-L 模体完全扫描数据, 扫描范围为 $0-360^\circ$, 每隔一度生成一幅投影图, 详细参数如表 B1

表 B1 投影图参数

投影数量	扫描范围	源点到探测器的距离	源点到探测器的距离	投影大小
360	$0-360^\circ$	1000mm	1536mm	256×256

从滤波后的投影图中已经能够看到原模体的诸多信息, 图 B1.2 为滤波后投影图及对应的模体切面图比对。(a)为第 91 幅投影, (b)为轴状面第 141 层, (c)为冠状面第 112 层。(d)为第 124 幅投影, (e)为矢状面第 121 层。

以距离驱动的 FDK 算法为例, 模体经前向投影、加权滤波至反投影各阶段的灰度直方图分布如图 B1.3

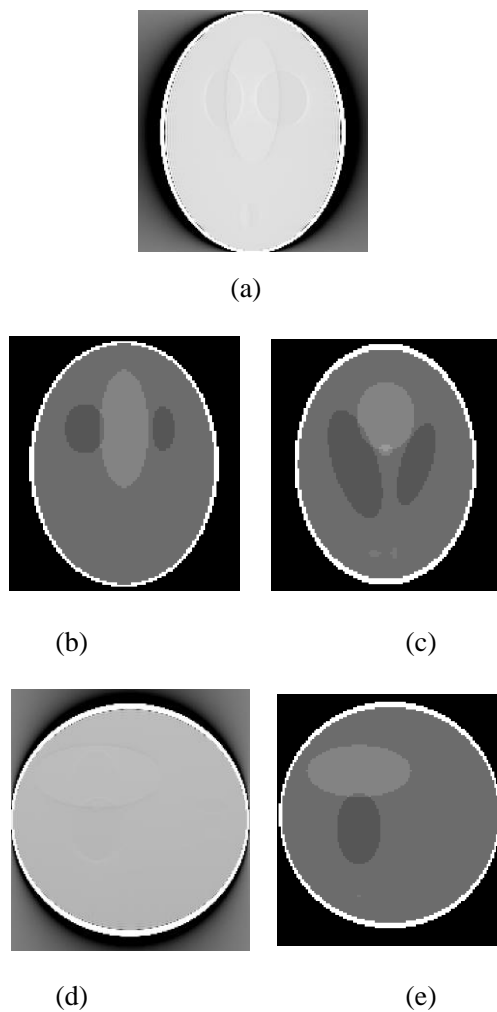


图 B1.2

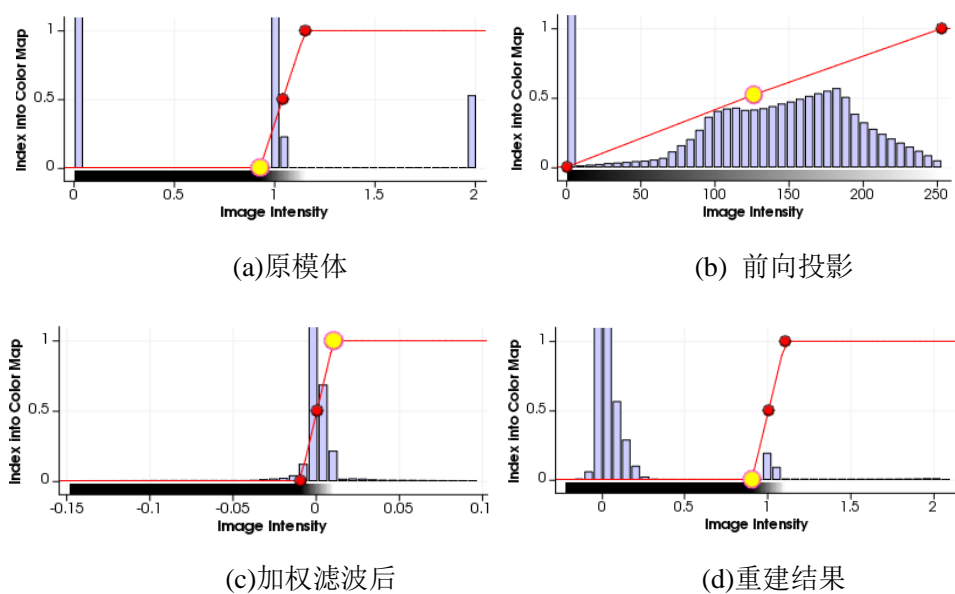


图 B1.3 各阶段的灰度直方图分布

附录 B2. 反投影算法误差分布

将重建结果与原模体做差，窗位 0.1，窗宽 0.2，取轴状位第 136 层切片，矢状位第 126 层切片，冠状位第 112 层切片进行结果展示。随后将做差结果进行两两做差，取轴状位第 136 片进行比较。

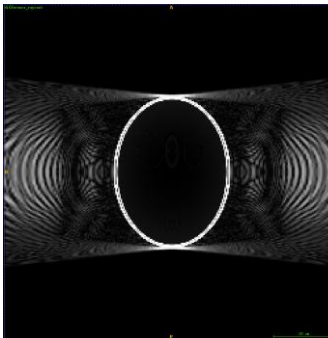
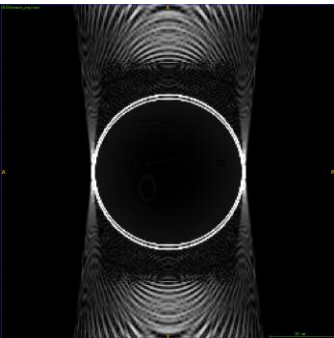
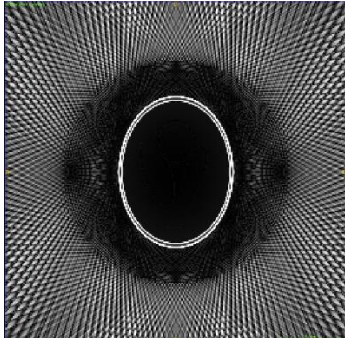
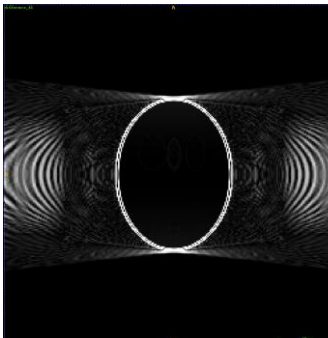
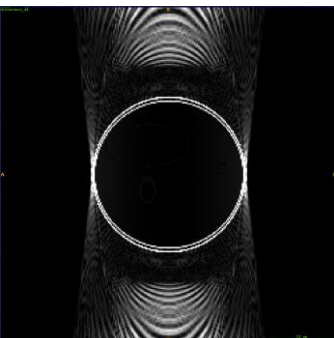
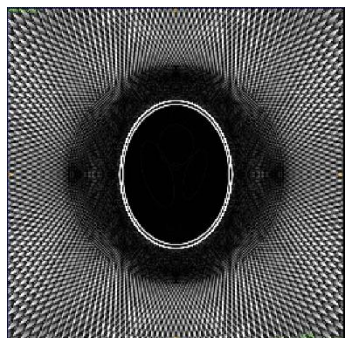
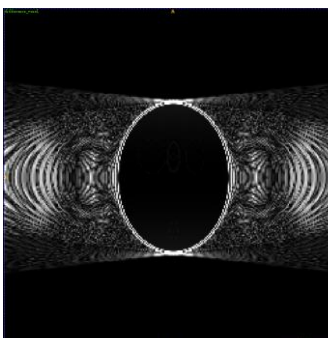
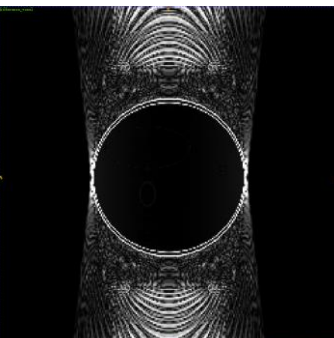
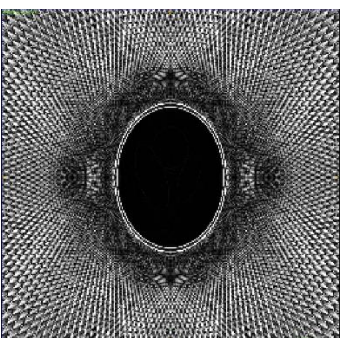
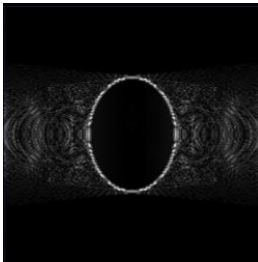
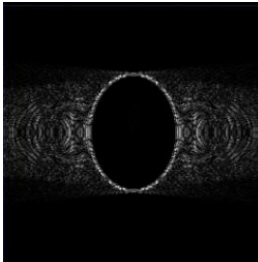
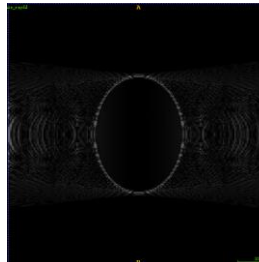
	轴状位	矢状位	冠状位
像素驱动			
距离驱动			
体素驱动			
差值做差进行比较	体素与像素驱动 	体素与距离驱动 	像素与距离驱动 

图 B2 反投影算法误差分布

从图 B2 中可以看出，误差集中分布在密度变化率高的边缘位置，同时四周有明显的



放射状伪影分布。前者由算法处理导致，对于密度变化梯度大的位置在运算量不变的情况下相对稀疏的数据很难重建出精确解。后者是受 FFT 频域变换时高频噪声影响的缘故，这部分伪影可以通过合理设计加权与滤波等预处理和后处理工作减轻，但是无法被完全消除。

此外，通过对做差结果进一步两两做差，可以发现像素驱动与距离驱动算法的误差两两做差后差异小，说明两者精度相近。而体素驱动与像素、距离驱动的误差比较时，差值较大，分布明显，说明体素驱动精度低于像素、距离驱动。

附录 B3. 反投影算法鲁棒性测试

将三种驱动方式算法在加入高斯白噪声前后的重建结果与原模体做差, 窗位 0.1, 窗宽 0.2, 取轴状位第 136 层切片进行结果展示。从图 B3 可以看出, 体素驱动鲁棒性较差, 距离驱动和像素驱动鲁棒性相近, 且均明显好于体素驱动加噪后的重建结果。

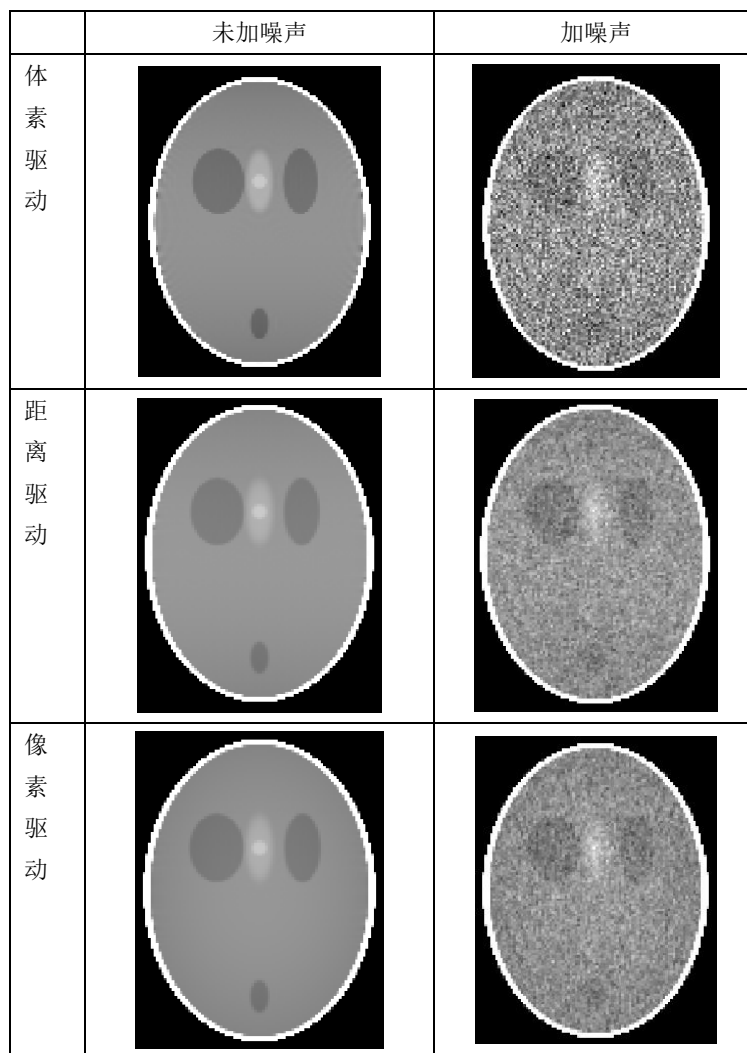


图 B3 重建算法的鲁棒性测试



附录 B4. 反投影算法性能参数汇总

表 B4 反投影算法性能

方 法	方法描述	硬件环境	编译模式	线程数
Voxel-based	体素驱动	CPU	Release	8
Distance-driven	基于 12 个边界点的距离驱动	CPU	Debug	8
Distance-driven Opti.	基于 8 个边界点的距离驱动	CPU	Debug	8
CudaRayCast	像素驱动	GPU	Release	-
CudaBackProjection	体素驱动	GPU	Release	-
CudaDistance-driven Opti.	基于 8 个点的距离驱动	GPU	Release	-

表 B4 续

方 法	重建一幅投影图的耗时	反投影时间	PSNR/dB	加入噪声后 PSNR	PSNR 变化
Voxel-based	16.7ms	3.01s	30.51	30.048	0.462
Distance-driven	15s	2700s	31.01	-	-
Distance-driven Opti.	1.5s	270s	31.01	-	-
CudaRayCast	129ms	23.22s	31.46	31.305	0.155
CudaBackProjection	6.5ms	1.17s	30.54	30.072	0.468
CudaDistance-driven Opti.	16.7ms	5.76s	31.27	31.16	0.11