基于 Shepp-Logan 头模型的锥束 CT 仿真投影快速计算

张顺利[1,2] 张定华[1] 李小林[2] 刘敏娜[2]
(西北工业大学现代设计与集成制造技术教育部重点实验室 西安 710072)[1]
(咸阳师范学院图形图像处理研究所 咸阳 712000)[2]

摘 要 针对三维 Shepp-Logan 头模型投影仿真,提出了一种快速并行投影计算方法。首先依次计算三维射线与各椭球的交点,然后对交点序列进行排序,用排序后的交点序列来确定射线穿过模型的区域编号及长度,最后将每个区域内的投影累加得到射线的投影值。在此基础上,将计算任务分解为4个独立的子任务,通过多线程技术在多核平台上实现了锥束 CT 仿真投影的快速并行计算。实验结果表明,该方法非常有效,在四核平台上取得了约3.5倍的加速比;图像重建结果验证了该方法生成的投影数据是准确的。

关键词 锥束 CT,Shepp-Logan 头模型,投影仿真,多线程,图像重建
中图法分类号 TP391 **文献标识码** A

Fast Calculation of Simulation Projection for Cone-beam CT Based on Shepp-Logan Head Phantom

ZHANG Shun-li[1,2] ZHANG Ding-hua[1] LI Xiao-lin[2] LIU Min-na[2]
(Key Laboratory of Contemporary Design & Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University, Xi'an 710072, China)[1]
(Institute of Graphics and Image Processing, Xianyang Normal University, Xianyang 712000, China)[2]

Abstract Aiming at the problem of projection simulation of 3D Shepp-Logan head phantom, this paper proposed a fast parallel method to calculate simulation projection. Firstly, the intersection points of 3D ray and each ellipsoid were calculated in turn. Then, the sequence of intersection points was sorted, by which the region number of phantom and length of intersection between the ray and the region were determined. Finally, the projection of the ray was obtained by summing the projection of each region. On this basis, we decomposed the computing task into four independent subtasks and realized fast parallel calculation of simulation projection of cone-beam CT on multi-core platform by multi-thread technique. Experimental result shows that the proposed method is very effective, can get a speedup about 3.5 times on quad cores platform. The accuracy of the projection data generated by the proposed method was verified by results of image reconstruction.

Keywords Cone-beam CT, Shepp-Logan head phantom, Projection simulation, Multi-thread, Image reconstruction

1 引言

CT(Computed Tomography)技术是计算机技术与放射探测技术相结合而形成的一种先进的成像技术,被广泛应用于医学诊断和工业无损检测等领域。锥束 CT 具有扫描速度快、空间分辨率各向同性等优点,因而成为当前 CT 研究的热点[1,2]。CT 技术的核心是由投影数据来重建图像,利用锥束 CT 进行图像重建的第一步便是获取投影数据。实际中,通过 CT 扫描系统获取投影数据代价高昂,且投影数据不可避免地存在噪声和射束硬化等现象,这对重建算法性能的有效性验证造成了一定的影响。为此,通常采用计算机仿真的方法来获取理想投影数据。Shepp-Logan 头模型是计算机仿真中采用的经典模型[3],被广泛应用于图像重建仿真实验和算法性能评测。

在感兴趣区域图像重建课题的研究过程中,为了对不同扫描方式下图像重建算法的性能进行研究,需要获取大量的仿真投影数据,这对投影计算的效率提出了更高的要求。文献[4]虽然介绍了利用 Shepp-Logan 头模型生成投影数据的方法,但没有考虑计算效率问题。本文针对 Shepp-Logan 头模型的投影仿真,提出了一种有效的投影计算方法,并通过多核加速实现了其快速并行计算,最后通过仿真实验对文中方法的有效性进行了验证。

到稿日期:2011-06-03 返修日期:2011-09-16 本文受中国博士后科学基金项目(20110491690),陕西省教育厅专项基金(11JK1030),咸阳师范学院引进人才项目(10XSYK102,09XSYK206)资助。
张顺利(1973-),男,博士,副教授,主要研究方向为工业 CT、计算机图形图像处理,E-mail:slmmzhang@sina.com;张定华(1958-),男,博士,教授,博士生导师,主要研究方向为 CAD/CAM、工业 CT 和计算机图形图像处理等;李小林(1976-),男,硕士,讲师,主要研究方向为计算机应用技术;刘敏娜(1981-),女,硕士,讲师,主要研究方向为计算机应用技术。

2 投影仿真原理

当强度为 I_0 的射线穿过厚度为 l、衰减系数为 μ 的均匀物体时，会产生光电效应、康普顿效应等物理过程，射线强度将衰减至 I，且服从 Beer-Lambert 衰减定律[5]，即

$$\mu l=\ln(I_0/I) \tag{1}$$

称 μl 为射线投影。对于非均匀物体，设 $\mu(x,y)$ 为衰减系数分布函数，则沿某一路径 L 的射线投影定义为

$$\int_L \mu(x,y)\mathrm{d}l = \ln(I_0/I) \tag{2}$$

由此可见，射线投影是物体的衰减系数分布函数 $\mu(x,y)$ 沿射线路径 L 的线积分，即 Radon 变换。改变射线的位置和方向，可得到各个方向、不同位置上的投影值。CT 理论的核心是如何由这一系列投影值来反求衰减系数分布函数 $\mu(x,y)$。

根据上述投影定义，当射线穿过由多种不同材质构成的物体时，如图 1 所示，投影值可由射线在每个物体内的投影累加得到，即

$$(l_{ab}+l_{ef})\mu_a+(l_{bc}+l_{de})\mu_b+l_{cd}\mu_c=\ln(I_0/I) \tag{3}$$

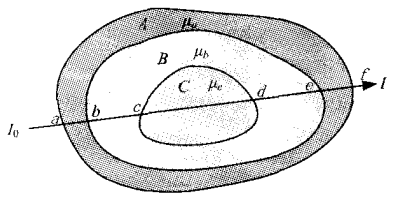


图 1 射线穿过不同物体时的投影示意图

推广到一般情形，射线穿过 n 个不同区域时的投影可定义为：

$$\sum_{i=1}^{n} l_i\mu_i=\ln(I_0/I) \tag{4}$$

式中，l_i 和 μ_i 分别为射线穿过第 i 个区域的长度和线性衰减系数值。由此可见，投影仿真的关键是确定射线穿过的不同区域及其长度。由于线性衰减系数 μ 近似正比于物体的密度 ρ，因此在 CT 仿真中也可采用物体的密度 ρ 来计算投影。

3 仿真投影计算

目前，仿真投影计算的方法通常有解析法、矢量法和离散法等[6-9]。解析法中广泛采用 Shepp-Logan 头模型，该模型由多个不同位置、大小、方向和密度各异的椭球组成。Shepp-Logan 头模型的具体参数设置如表 1 所列。

表 1 三维 Shepp-Logan 头模型参数

N_i	x_0	y_0	z_0	a	b	c	θ	ρ
1	0.0	0.0	0.0	0.69	0.9	0.92	0.0	2.0
2	0.0	0.0	−0.0184	0.6624	0.88	0.874	0.0	1.0
3	−0.22	−0.25	0.0	0.41	0.21	0.16	72.0	0.3
4	0.22	−0.25	0.0	0.31	0.22	0.11	−72.0	0.3
5	0.0	−0.25	0.35	0.21	0.35	0.25	0.0	1.5
6	0.0	−0.25	0.1	0.046	0.046	0.046	0.0	1.2
7	−0.08	−0.25	−0.605	0.046	0.02	0.023	0.0	1.5
8	0.06	−0.25	−0.605	0.046	0.02	0.023	90.0	1.5
9	0.06	0.625	−0.105	0.056	0.1	0.04	90.0	1.3
10	0.0	0.625	0.1	0.056	0.1	0.056	0.0	1.3
11	0.0	−0.25	−0.1	0.046	0.046	0.046	0.0	1.2
12	0.0	−0.25	−0.605	0.023	0.023	0.023	0.0	1.5

其中，(x_0,y_0,z_0) 为椭球中心坐标，a、b、c 分别为椭球在 X 轴、Y 轴和 Z 轴方向的半轴长，θ 为椭球绕 Y 轴逆时针旋转的角度，ρ 为椭球内部密度。

3.1 模型密度区域划分

根据 Shepp-Logan 头模型中椭球之间的几何拓扑关系来定义不同密度区域。由表 1 可以看出，模型中的椭球满足以下拓扑关系：$N_1\supset N_2\supset N_i$，$3\leqslant i\leqslant 12$；$N_3$ 与 N_5 相交，N_5 与 N_6 相交，N_3 与 N_{11} 相交。为了便于计算，将头模型划分为 15 个互不重叠的密度区域 U_j，$1\leqslant j\leqslant 15$。每个区域定义为

$$U_1=\{(x,y,z)\mid (x,y,z)\in N_1,(x,y,z)\notin N_2\}$$

$$U_2=\{(x,y,z)\mid (x,y,z)\in N_2,(x,y,z)\notin N_i,3\leqslant i\leqslant 12\}$$

$$U_3=\{(x,y,z)\mid (x,y,z)\in N_3,(x,y,z)\notin N_5,(x,y,z)\notin N_{11}\}$$

$$U_5=\{(x,y,z)\mid (x,y,z)\in N_5,(x,y,z)\notin N_3,(x,y,z)\notin N_6\}$$

$$U_6=\{(x,y,z)\mid (x,y,z)\in N_6,(x,y,z)\notin N_5\}$$

$$U_j=\{(x,y,z)\mid (x,y,z)\in N_j,j=4,7,8,9,10,12\}$$

$$U_{11}=\{(x,y,z)\mid (x,y,z)\in N_{11},(x,y,z)\notin N_3\}$$

$$U_{13}=\{(x,y,z)\mid (x,y,z)\in N_3,(x,y,z)\in N_5\}$$

$$U_{14}=\{(x,y,z)\mid (x,y,z)\in N_5,(x,y,z)\in N_6\}$$

$$U_{15}=\{(x,y,z)\mid (x,y,z)\in N_3,(x,y,z)\in N_{11}\}$$

对于两个椭球相交区域，定义其密度为两个椭球密度的平均值。即 U_{13}、U_{14} 和 U_{15} 的密度分别为 0.9、1.4 和 0.75。

3.2 射线与模型求交

密度区域划分后，当射线穿过模型时，需要确定射线穿过不同区域的长度以及区域编号。本文采用的方法是，首先计算射线与每个椭球的交点，得到交点序列，然后将其排序，再根据排序后的交点序列来计算射线穿过的区域长度以及区域编号。下面考虑射线与一个椭球体的求交。

设射线源坐标为 $S(Sx,Sy,Sz)$，射线在探测器上的坐标为 $E(Ex,Ey,Ez)$。根据上述计算方法，射线方程用参数式来表示，即

$$\begin{aligned} x(t)&=Sx+(Ex-Sx)\times t\\ y(t)&=Sy+(Ey-Sy)\times t \quad 0\leqslant t\leqslant 1\\ z(t)&=Sz+(Ez-Sz)\times t \end{aligned} \tag{5}$$

椭球经过平移、旋转后的方程可表示为

$$\frac{((z-z_0)\sin\theta+(x-x_0)\cos\theta)^2}{a^2}+\frac{(y-y_0)^2}{b^2}+\frac{((z-z_0)\cos\theta-(x-x_0)\sin\theta)^2}{c^2}=1 \tag{6}$$

联立式(5)和式(6)，得到关于参数 t 的一元二次方程，即

$$pt^2+qt+r=0 \tag{7}$$

式中，

$$p=(bc(\Delta x\cos\theta+\Delta z\sin\theta))^2+(ac\Delta y)^2+(ab(\Delta z\cos\theta-\Delta x\sin\theta))^2$$

$$q=2(bc)^2(\Delta x\cos\theta+\Delta z\sin\theta)((Sz-z_0)\sin\theta+(Sx-x_0)\cos\theta)+(ac)^2\Delta y(Sy-y_0)+(ab)^2(\Delta z\cos\theta-\Delta x\sin\theta)((Sz-z_0)\cos\theta+(x_0-Sx)\sin\theta)$$

$$r=b^2c^2((Sz-z_0)\sin\theta+(Sx-x_0)\cos\theta)^2+a^2c^2(Sy-y_0)^2+a^2b^2((Sz-z_0)\cos\theta+(x_0-Sx)\sin\theta)^2-a^2b^2c^2$$

$$\Delta x=Ex-Sx,\Delta y=Ey-Sy,\Delta z=Ez-Sz$$

如果射线穿过某个椭球，则必然与椭球有两个不同的交点，因此当式(7)的判别式 $q^2-4pr>0$ 时，利用求根公式得到关于 t 的两个解。

按照上述方法，依次计算射线与每个椭球的交点，得到交点序列 $t_1,t_2,\cdots,t_{M-1},t_M$，将其保存在数组 $array[i]$ 中。然后对数组 $array[i]$ 由小到大进行排序，则射线穿过第 i 个区域的长度 l_i 的计算公式为

$$l_i=(array[i+1]-array[i])\times|SE| \tag{8}$$

式中，$|SE|=\sqrt{\Delta x^2+\Delta y^2+\Delta z^2}$。穿过第 i 个区域的区域编号可通过该区域两个边界点的中点坐标来确定。

令 $t=(array[i+1]+array[i])/2$，将 t 代入式(5)，得到中点坐标 $(x(t),y(t),z(t))$，然后根据 3.1 节中区域的定义，确定区域编号，进而得到其密度值，由此可计算出射线在该区域内的投影值。将射线经过的所有区域投影相加，即可得到该射线最终的投影值。

由于椭球 1 包含其他椭球，因此射线如果不与椭球 1 相交，则一定不会与其他椭球相交。根据这一点，在实际计算过程中，可以先计算射线与椭球 1 的相交情况。如果射线与椭球 1 相交，则依次计算射线与其他椭球的相交情况；否则，投影值为 0。这样可以减少不必要的计算，下面给出计算一条射线仿真投影的步骤。

Step1　计算射线与椭球 1 的相交情况。如果相交，则将两个交点保存在 $array[i]$ 中；否则投影值为 0，结束。

Step2　依次计算射线与其他椭球的交点，并将交点保存在 $array[i]$ 中。

Step3　对数组 $array[i]$ 由小到大进行排序。

Step4　由 $array[i]$ 计算射线穿过不同区域的长度 l_i 以及区域的密度值 p_i。

Step5　由 l_i 和 p_i 根据式(4)得到射线的仿真投影，结束。

3.3　多核并行计算

在锥束 CT 投影仿真计算中，投影角度数通常为 360 以上，探测器阵列为 512×512、1024×1024 或更高，需要计算的射线投影数达千万条以上，因而计算量非常大。要提高投影计算速度，一种可行的方法是并行计算。近年来，随着计算机硬件技术的发展，多核计算机已经成为主流配置，多核 CPU 为并行计算提供了新的途径[10,11]。在多核平台上利用多线程技术能够实现真正意义上的并行计算。

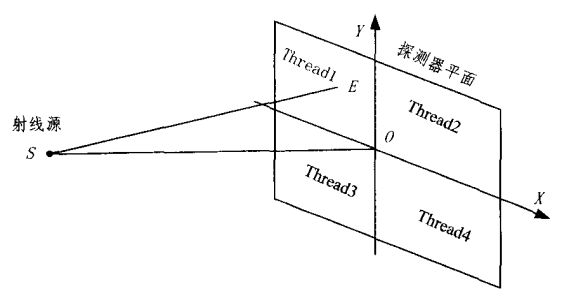


图 2　任务分解示意图

实现多线程的关键是充分挖掘任务的并行性，并且将计算负载尽可能均衡地分配到各内核上。在投影仿真计算中，各条射线之间相互独立，具备了良好的并行性，且不涉及复杂的同步操作。为了实现负载平衡问题，从投影数据出发，根据处理器内核个数，将探测器平面等分为几部分，相应的计算任务也分解为几个子任务，每个子任务通过一个线程来实现。本文后面的讨论以四核处理器为例，将计算任务等分为 4 个子任务，并通过 4 个线程来实现，如图 2 所示。

任务分解后，每个线程依次完成所有角度下投影数据的计算。设探测器阵列为 $N\times N$，投影角度数为 K，开辟数组 $detect[N/2][N/2]$ 来存放每个角度下射线的终点坐标，$projection[K][N][N]$ 来保存投影数据。线程的创建采用 Visual C++提供的 API 函数 CreateThread()，在主线程中分别创建 4 个线程。为了实现所有线程计算完后保存投影数据，必须要对 4 个线程进行同步。这里利用 CreateEvent()函数来创建事件对象 *event*，利用 WaitForSingleObject()函数等待该事件对象变为“有信号状态”，从而实现多个线程的同步。为此需要定义一个全局变量 *threadcompleted*，用来记录完成线程的个数，当一个线程完成计算任务后其值加 1，当 *threadcompleted* 为 4 时设置事件对象 *event* 为“有信号状态”。下面给出主线程的伪代码。

```
threadcompleted=0;//初始化完成线程个数
event = CreateEvent(NULL,FALSE,FALSE,NULL);//创建事件对象并初始化为“无信号状态”
CreateThread(NULL,0,Thread1,NULL,0,NULL);
CreateThread(NULL,0,Thread2,NULL,0,NULL);
CreateThread(NULL,0,Thread3,NULL,0,NULL);
CreateThread(NULL,0,Thread4,NULL,0,NULL);
if (WAIT_OBJECT_0==WaitForSingleObject(event,INFINITE))
  //等待事件对象为“有信号状态”
Save projection[n][i][j] to files;//保存所有投影数据
```

下面给出子线程函数 Thread1()的伪代码：

```
DWORD WINAPI Thread1(LPVOID param)
{
  initialize X-ray source S;
  initialize Detector plane detect[i][j];
  for(n=0;n<nScan;n++) //投影角度循环
  {
    for(i=0;i<N/2;i++) //探测器行标
    for(j=0;j<N/2;j++) //探测器列标
    Calculate projection data projection[n][i][j]; //计算该射线的投影
    Rotate X-ray source S; //旋转射线源
    Rotate Detector plane detect[i][j];//旋转探测器平面
  }
    threadcompleted++;//完成线程数加 1
    if ( threadcompleted == 4) SetEvent(event);
    return 0;
}
```

4　实验及结果分析

为了验证所提方法的有效性，分别采用单线程、双线程和四线程来实现本文方法。锥束 CT 扫描方式为单圆轨迹，扫描半径为 900mm，旋转中心距探测器平面 300mm，探测器阵列为 512×512，像元尺寸为 0.256mm，投影数为 360 幅，投影

角度采样间隔为1°，射线源的初始位置为(0，－900，0)，探测器平面在Y轴上的初始位置为(0，300，0)，测试计算机配置为4核Intel酷睿I5-760 2.8GHz CPU、4GB内存，开发工具为Visual C++6.0。表2给出了不同线程下的仿真投影计算时间对比。

表2　不同线程的计算时间比较

线程数	单线程	双线程	四线程
时间(s)	310.235	158.813	87.922
加速比	1.0	1.95	3.53

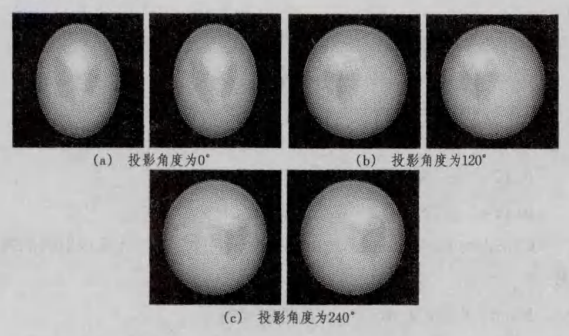
由表2可以看出，采用四线程并行加速后，取得了约3.5倍的加速比，加速效果良好。

为了进一步验证文中方法的效率，采用本文方法和基于体素模型的离散法来计算仿真投影，其中前向投影算法分别采用经典的Siddon算法[12]和文献[13]中提出的三维射线与体素的快速遍历和求交算法。实验中均采用单线程计算，其中投影数取90幅，投影角度采样间隔为4°，探测器分辨率分别取128×128、256×256和512×512。表3给出了几种方法的计算时间比较。

表3　几种方法投影计算时间比较(s)

探测器分辨率	128×128	256×256	512×512
Siddon算法	49.082	492.046	4645.124
文献[13]方法	2.797	27.954	263.453
本文方法	4.875	19.509	77.656

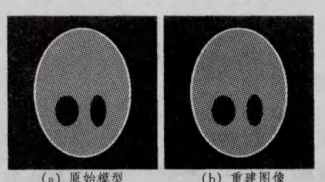
由表3可以看出，随着投影分辨率的提高，本文方法的加速效果显著提升。与文献[13]中方法相比，在探测器分辨率为512×512时，取得了3倍以上的加速比；与Siddon算法相比，更是取得了近60倍的加速比。由此可见，本文方法非常有效，能够满足感兴趣区域图像重建研究中不同扫描方式下投影数据的快速生成。图3给出了几种方法生成的不同角度下的投影图像。



(a) 投影角度为0°　(b) 投影角度为120°　(c) 投影角度为240°

图3　不同角度下的投影图像

图3中左侧一列为本文方法生成的投影图像，右侧一列为离散法生成的相同角度下的投影图像。由图3可见，两种方法生成的投影图像在视觉上几乎完全相同。



(a) 原始模型　(b) 重建图像

图4　中心剖面图像对比

为了验证本文方法的正确性，利用上述投影数据进行三维图像重建，重建算法采用锥束ART算法，置图像初值为0，松弛因子λ取0.25，迭代次数为3次。图4给出了Shepp-Logan头模型XOY平面的中心剖面图像及相应的重建图像对比。

由图4可以看出，重建图像与原始模型用肉眼几乎无法区分，这表明本文投影计算方法所生成的投影数据是准确的。

结束语　本文针对Shepp-Logan头模型提出了一种仿真投影计算方法，其大大提高了锥束CT投影仿真效率，通过多核加速进一步实现了仿真投影的快速并行计算，在四核平台上取得了约3.5倍的加速比。本文的研究为后续感兴趣区域图像重建研究奠定了基础。此外，本文提出的方法适用于任意扫描轨迹下的投影数据计算。利用Shepp-Logan头模型生成仿真投影的优点是精度高，存在的主要问题是，一旦椭球的几何拓扑结构发生变化，就必须重新设计投影算法。因而如何设计一种通用的仿真投影计算方法，使其能够适应模型中椭球几何拓扑结构的变化，是今后的研究内容。

参考文献

[1] Yan Guo-rui, Tian Jie, Zhu Shou-ping, et al. Fast cone-beam CT image reconstruction using GPU hardware [J]. Journal of X-Ray Science and Technology, 2008, 16: 225-234

[2] Wang G, Yu H Y, DeMan B. An outlook on x-ray CT research and development [J]. Medical Physics, 2007, 35: 1051-1064

[3] Yu Heng-yong, Zhao Shi-ying, Wang Ge. A Differentiable Shepp-Logan Phantom and Its Applications in Exact Cone-Beam CT [J]. Physics in Medicine and Biology, 2005, 50(11): 5583-5595

[4] 孙丰荣，刘泽，李艳玲，等. 基于模型的CT三维医学图像重建仿真[J]. 系统仿真学报，2006，18(3)：781-784

[5] Kak A, Slaney M. Principles of computerized tomographic imaging [M]. New York: IEEE Press, 1988

[6] 黄建林，吕东辉. 图像重建中X射线投影模拟的常用方法[J]. 上海大学学报：自然科学版，2006，12(3)：228-233

[7] Tang Shao-jie, Yu Heng-yong, Yan Hao, et al. X-ray projection simulation based on physical imaging model [J]. Journal of X-Ray Science and Technology, 2006, 14(3): 177-189

[8] Wulff J, Ubrich F, Zink K. Monte Carlo simulation of an x-ray volume imaging cone beam CT unit [J]. Med. Phys., 2009, 36: 127-136

[9] Merbach J M. Simulation of X-ray projections for experimental 3D tomography [R]. Linkoping University, Linkoping, Sweden: Image Processing Lab, Dept of Electrical Engineering, 1996

[10] 汪少敏，赵猛，朱振博，等. 基于多核处理器并发计算软件构架设计与实现[J]. 计算机科学，2008，35(7)：283-285

[11] Zhang ZM, Liang YZ, Xu QS. Multi-core computing: A novel accelerating method for chemometrics calculation [J]. Chemometrics and Intelligent Laboratory Systems, 2009, 96(1): 94-97

[12] Siddon R L. Fast calculation of the exact radiological path for three-dimensional CT array [J]. Med. Phys., 1985, 12: 252-255

[13] 张顺利，张定华，黄魁东，等. 锥束ART算法快速图像重建[J]. 仪器仪表学报，2009，30(4)：885-892