一种射束与像素的快速遍历和求交算法

张顺利[1),2)] 张定华[1)] 赵歆波[1)] 黄魁东[1)]

[1)](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)

[2)](咸阳师范学院图形图像处理研究所,咸阳 712000)

摘 要 针对 ART(algebraic reconstruction technique)算法重建速度慢的问题,提出了一种射束与像素的快速遍历和求交算法。该算法通过一个距离参数来确定射束穿过的像素索引并计算出射束覆盖像素的面积,距离参数采用增量计算,因而运算效率很高。利用该算法在图像重建过程中实时计算权因子,不但节省了大量的内存空间,而且大大提高了图像重建的速度。实验结果表明,提出的算法非常有效,与传统方法相比取得了 17 倍以上的重建加速比。

关键词 图像重建 ART 算法 遍历 求交 权因子

中图法分类号: TP391.72 **文献标识码**: A **文章编号**: 1006-8961(2009)10-1961-05

A Fast Traversal and Intersection Algorithm of Ray Beam-pixels

ZHANG Shun-li[1),2)], ZHANG Ding-hua[1)], ZHAO Xin-bo[1)], HUANG Kui-dong[1)]

[1)](*Key Laboratory of Contemporary Design and Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University, Xi'an* 710072)

[2)](*Institute of Graphics and Image Processing, Xianyang Normal University, Xianyang* 712000)

Abstract To solve the problem of slow reconstruction speed of algebraic reconstruction technique, this paper presents a fast traversal and intersection algorithm of ray beam-pixels. The algorithm determines the pixel index traversed by ray beam and calculates area overlapped by the ray beam through a distance parameter. The distance parameter can be calculated with an incremental method, thus the operation is of high efficiency. Applying this algorithm to calculate the weight coefficients on real-time during the process of reconstruction, not only large amount of memory is saved, but also the speed of image reconstruction is improved greatly. The experimental result shows that the algorithm is very effective and the reconstruction speed is improved about 17 times compared with the conventional method.

Keywords image reconstruction, algebraic reconstruction technique(ART), traversal, intersection, weight coefficient

1 引 言

图像重建是由物体在不同角度下的射线投影来获取物体横截面图像的过程,被广泛应用于工业无损检测和医学图像处理中。目前图像重建的方法大致分为两类:变换法和迭代法[1]。变换法的特点是重建速度快,对完全投影数据能获得很好的重建质量,缺点是对投影数据的完备性要求较高。实际应用中往往由于客观原因无法检测或很难检测到完全的投影数据。迭代法中以 ART[2] 算法为代表,该算法适合于不完全投影数据的图像重建,缺点是重建速度慢,从而限制了其在医学和工业领域的应用。

近年来,众多学者对提高 ART 算法的重建速度

基金项目:国家自然科学基金项目(50375126);陕西省自然科学基金项目(2009JQ8017);陕西省教育厅专项基金项目(07JK425,09JK810)
收稿日期:2008-07-09;**改回日期**:2008-09-11

第一作者简介:张顺利(1973~),男,副教授。西北工业大学航空宇航制造工程专业博士研究生。主要研究方向为工业 CT、计算机图形图像处理。E-mail:slmmzhang@sina.com

进行了大量的研究,并取得了一定的加速效果。文献[3]通过 SIMD(single instruction multiple data)并行处理技术实现了 ART 算法的快速重建,取得了很好的重建加速比。文献[4]~[5]通过改变迭代过程中投影数据的访问次序来提高重建速度,但其加速效果不是很明显。文献[6]对迭代算式中的松弛因子进行了研究,表明选择合适的松弛因子能够减少迭代次数、提高重建速度,但该方法并不能改变单次迭代所需的时间,这对于需要经过多次迭代才能收敛的情况往往不能收到良好的加速效果。

在 ART 算法的迭代过程中,权因子的计算成为制约重建速度的关键因素,而上述方法均没有涉及权因子的计算。文献[7]提出了一种内存优化技术实现了 ART 算法的快速重建,但该方法只考虑了简化权因子的计算,要实现高精度的图像重建,必须精确计算权因子。针对上述问题,提出一种射束与像素的快速遍历和求交算法,利用该算法在迭代过程中实时、准确地计算权因子,实现了 ART 算法的快速、高精度重建,并通过实验对本文算法的有效性进行了验证。

2 ART 算法原理

首先将一个 $n \times n$ 的正方形网格叠加在欲重建的未知图像 $f(x,y)$ 上,如图 1 所示。在每个网格(像素)内 $f(x,y)$ 是常量,用 f_j 表示第 j 个像素内的常数值,$N = n \times n$ 为像素总数。射束宽度为 τ(常取 τ 等于像素宽 δ),它覆盖像素的一部分面积,这部分面积与像素面积 δ^2 之比定义为权因子。

图 1 中,第 j 个像素被第 i 条射束所覆盖的面积为 $S_{\Delta ABC}$,则权因子 $w_{ij} = S_{\Delta ABC}/\delta^2$,反映了第 j 个像素对第 i 条射束投影的贡献。记第 i 条射束的投影为 p_i,则有:

$$\sum_{j=1}^{N} w_{ij} f_j = p_i \qquad i = 1,2,\cdots,M \tag{1}$$

式中,M 为投影总数。将式(1)写成矩阵形式为

$$\boldsymbol{RF} = \boldsymbol{P} \tag{2}$$

式中,$\boldsymbol{P}$ 为 M 维投影数据向量,$\boldsymbol{F}$ 为 N 维图像向量,$\boldsymbol{R}$ 为 $M \times N$ 维权因子矩阵。为了获得高质量的图像,M 和 N 通常都很大,且对于每条射束来说,只与很少的像素相交,因此 $\boldsymbol{R}$ 是一个大型稀疏矩阵,故很难用常规的矩阵理论来求解,实际中都采用迭代法。其过程为:

(1)给定图像的初值 $\boldsymbol{F}^{(0)}$,一般取为零向量。

(2) $$f_j^{(k+1)} = f_j^{(k)} + \lambda \frac{p_i - \sum_{n=1}^{N} w_{in} f_n^{(k)}}{\sum_{n=1}^{N} w_{in}^2} w_{ij} \tag{3}$$

式中,k 为迭代序号,$1 \leqslant i \leqslant M$,$1 \leqslant j \leqslant N$,$\lambda$ 为松弛因子($0 < \lambda < 2$)。当使用完所有的投影数据后,就实现了一次完整的迭代。重复步骤(2),直到满足一定的收敛条件。

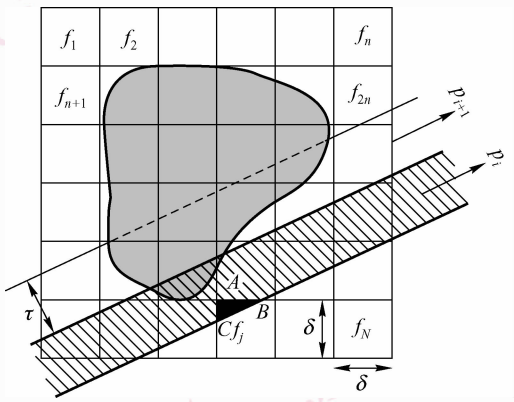


图 1 ART 算法重建模型

Fig. 1 Reconstruction model of ART algorithm

由式(3)可以看出,ART 算法在迭代过程中要反复用到权因子 w_{ij},因而能否快速获取 w_{ij} 成为决定 ART 算法重建速度的关键。传统方法是根据像素布置和射束的几何结构预先计算出所有的 w_{ij},然后将其保存在硬盘中,迭代时再调出。但一方面,由于 w_{ij} 的数目非常庞大,普通计算机很难存储;另一方面,频繁的访问硬盘会大大降低重建速度。因此,应该寻求一种在迭代过程中快速实时计算 w_{ij} 的方法,其实质是要计算出射束穿过的所有像素索引及面积,即射束与像素的遍历和求交。

3 射束与像素的遍历和求交算法

3.1 相关定义

图 2 为一条宽为 δ 的射束穿过重建区域(像素宽为 δ)时的情形。区域中心在坐标原点 O,像素索引按从左到右、从上到下的顺序进行编号,依次为 $0,1,\cdots,n^2-1$。为了讨论方便,假定射束的斜率 k 满足 $0 < k \leqslant 1$。其上边界直线为 l_1,对应的方程为 $y = kx + b_1$;下边界直线为 l_2,对应的方程为 $y = kx + b_2$。

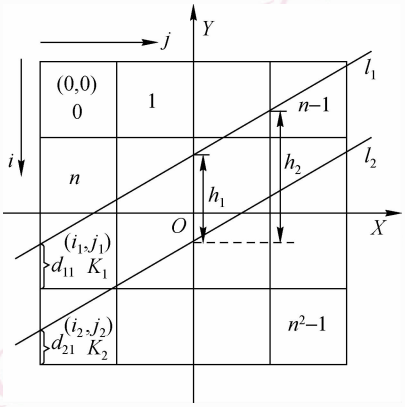


图 2　重建区域示意图

Fig. 2　Sketch map of reconstruction region

如图 2 所示,用 h_1 表示上下边界线之间的纵向距离,且有 $h_1 = b_1 - b_2 = \delta\sqrt{1+k^2}$;用 h_2 表示上下边界线在一列中的最大纵向距离,显然,$h_2 = h_1 + k\delta = \delta(\sqrt{1+k^2}+k)$。给出以下命题。

命题 1　射束内的像素至少与一条边界线相交。

证明　用反证法。假设射束内的一个像素不与任意一条边界线相交,此时该像素将完全被射束所覆盖,由此可推算出射束在一列的最大纵向距离至少为 $\delta(1+2k)$,显然 $\delta(1+2k) > h_2$,与实际不符,因而假设不成立。

命题 2　射束在一列中至少覆盖 2 个像素,最多覆盖 4 个像素。

证明　由于 $h_1 > \delta$ 且 $h_2 < 3\delta$ 成立,即纵向距离 h_1 大于一个像素的高度,最大纵向距离 h_2 小于 3 个像素的高度,因此射束在一列将至少覆盖 2 个像素,最多覆盖 4 个像素。

由命题 1 可知,射束与像素的遍历和求交问题可以转化为其 2 条边界线 l_1 和 l_2 与像素的遍历和求交。

3.2　直线与像素的遍历和求交

如图 3 所示,假设直线穿过某个像素 K,该像素左下角顶点为 P。定义距离参数 d 为直线到点 P 的 Y 向距离,如图 3 中的 d_1 所示。显然,若直线沿 X 轴正向变化一个像素 δ,则距离参数将从 d_1 变化到 d_2,且有 $d_2 = d_1 + k\delta$,据此可以计算出下一个像素的距离参数。下面具体讨论如何根据距离参数来进行直线与像素的遍历和求交。

由图 3 可以看出,取决于 d_2,直线穿过像素 K 时不外乎以下 3 种情形:

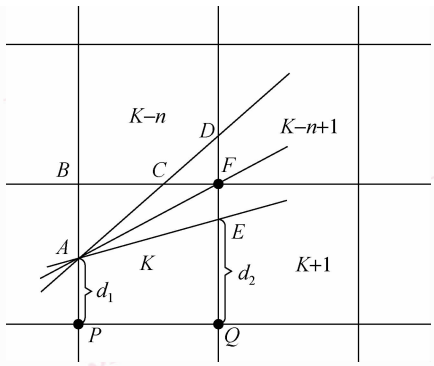


图 3　直线穿过像素 K 的情形

Fig. 3　Situations of a line passing through pixel K

(1)若 $d_2 > \delta$,如直线 AD 所示,直线将依次遍历像素 K 和其上方像素 $K-n$,下一个要遍历的像素是右上方像素 $K-n+1$,对应的距离参数为 $d_2 - \delta$。如果 AD 为上边界线,则像素 K 中 ΔABC 不在射束内,故像素面积减去 $S_{\Delta ABC}$,像素 $K-n$ 被射束覆盖的面积为 $S_{\Delta CFD}$;相反,如果 AD 为下边界线,则像素 K 中 ΔABC 被射束覆盖,像素 $K-n$ 中 ΔCFD 不在射束内,对应的像素面积将减去 $S_{\Delta CFD}$。其中 $S_{\Delta ABC} = (\delta - d_1)^2/2k$,$S_{\Delta CFD} = (d_2 - \delta)^2/2k$。

(2)若 $d_2 = \delta$,如直线 AF 所示,直线将只遍历像素 K,下一个要遍历的像素是右上方像素 $K-n+1$,对应的距离参数为 0。如果 AF 为上边界线,则像素 K 中 ΔABF 不在射束内,故像素面积减去 $S_{\Delta ABF}$;相反,如果 AF 为下边界线,则像素 K 中 ΔABF 被射束覆盖,面积为 $S_{\Delta ABF}$。其中 $S_{\Delta ABF} = \delta(\delta - d_1)/2$。

(3)若 $d_2 < \delta$,如直线 AE 所示,直线将只遍历像素 K,下一个要遍历的像素是右方像素 $K+1$,对应的距离参数为 d_2。如果 AE 为上边界线,则像素 K 被射束覆盖的部分为四边形 $AEQP$,其面积为 $\delta(d_1 + d_2)/2$;相反,如果 AE 为下边界线,则像素 K 中的四边形 $ABFE$ 被射束覆盖,面积为 $\delta(2\delta - d_1 - d_2)/2$。

根据以上分析,可以得到射束穿过像素时,满足以下性质:

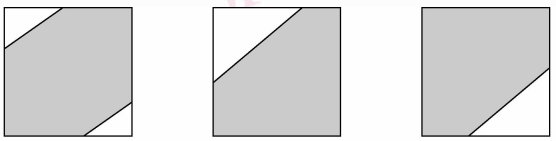
性质 1　如果射束的一条边界线以上述(2)或(3)的方式穿过某像素,则不可能与另一条边界线相交。

此时,射束穿过像素的情况是确定的,其面积可以按照(2)或(3)中的方法来计算。

性质 2　如果射束的一条边界线以上述(1)的

方式穿过某像素,则有可能与另一条边界线也相交。

图 4 给出了性质 2 所描述的射束穿过像素的几种可能情况。可见,由像素面积减去 1 个或 2 个三角形的面积即可得到射束覆盖像素的面积。为此,定义 2 个初值为 FALSE 的布尔型变量 *flag_up* 和 *flag_down* 来分别标记像素的左上角和右下角是否被射束所覆盖。若变量值为 TRUE,则表示没有被射束覆盖,此时需要减去三角形的面积。



(a) 两个角不被覆盖 (b) 左上角不被覆盖 (c) 右下角不被覆盖

图 4 性质 2 所描述的射束穿过像素的几种情况

Fig. 4 Situations of pixel traversed by ray beam described by property 2

由上面的分析可知,通过两条边界线的距离参数,完全可以确定射束所覆盖的像素索引及其面积,而且距离参数可通过增量方式计算。

3.3 算法描述

根据以上讨论,首先计算出射束的两条边界线 l_1 和 l_2 穿过的起始像素索引及距离参数,分别记为 K_1,K_2,d_{11} 和 d_{21},如图 2 所示。并假定起始像素在重建区域的同一列,然后沿 X 轴正向每步移动一个像素单位 δ,得到新的距离参数 d_{12} 和 d_{22},再按照第 3.2 节中的讨论计算射束穿过像素的索引及面积。重复该过程,直至遇到重建区域的边界。由命题 2 可知,该方法每步将遍历 2 ~ 4 个像素。为了在遍历过程中判断像素是否越界,需要计算像素所在的行标 i 和列标 j,其中 $0 \leqslant i,j < n$。算法具体步骤如下:

第 1 步 初始化。分别计算起始像素索引 K_1,K_2 及对应的距离参数 d_{11} 和 d_{21},像素 K_1,K_2 的坐标分别为 (i_1,j_1) 和 (i_2,j_2),且满足 $j_1 = j_2$。令 $delt = k\delta$,*flag_up* = FALSE,*flag_down* = FALSE。

第 2 步 $d_{12} \leftarrow d_{11} + delt$,$d_{22} \leftarrow d_{21} + delt$。

第 3 步 计算上边界线 l_1 与像素 K_1 的相交情况。

如果 $d_{12} > \delta$,则遍历像素 $K_1 - n$,面积为 $(d_{12} - \delta)^2/2k$,令 *flag_up* = TRUE,并记 $S_1 = (\delta - d_{11})^2/2k$,其中 S_1 为像素 K_1 要减去左上角的三角形面积,$d_{11} \leftarrow d_{12} - \delta$,$i_1 \leftarrow i_1 - 1$,$j_1 \leftarrow j_1 + 1$;如果 $d_{12} = \delta$,则遍历像素 K_1,面积为 $\delta(d_{11} + \delta)/2$,$d_{11} \leftarrow 0$,$i_1 \leftarrow i_1 - 1$,$j_1 \leftarrow j_1 + 1$,$K_1 \leftarrow K_1 - n + 1$;如果 $d_{12} < \delta$,则遍历像素 K_1,面积为 $\delta(d_{11} + d_{12})/2$,$d_{11} \leftarrow d_{12}$,$j_1 \leftarrow j_1 + 1$,$K_1 \leftarrow K_1 + 1$。

第 4 步 计算下边界线 l_2 与像素 K_2 的相交情况。

如果 $d_{22} > \delta$,则遍历像素 K_2,面积为 $(\delta - d_{21})^2/2k$,令 *flag_down* = TRUE,并记 $S_2 = (d_{22} - \delta)^2/2k$,其中 S_2 为像素 $K_2 - n$ 要减去右下角的三角形面积,$d_{21} \leftarrow d_{22} - \delta$,$i_2 \leftarrow i_2 - 1$,$j_2 \leftarrow j_2 + 1$;如果 $d_{22} = \delta$,则遍历像素 K_2,面积为 $\delta(\delta - d_{21})/2$,$d_{21} \leftarrow 0$,$i_2 \leftarrow i_2 - 1$,$j_2 \leftarrow j_2 + 1$,$K_2 \leftarrow K_2 - n + 1$;如果 $d_{22} < \delta$,则遍历像素 K_2,面积为 $\delta(2\delta - d_{21} - d_{22})/2$,$d_{21} \leftarrow d_{22}$,$j_2 \leftarrow j_2 + 1$,$K_2 \leftarrow K_2 + 1$。

第 5 步 计算像素 K_1 左上角及像素 $K_2 - n$ 右下角不被射束覆盖时的情况。

如果满足 *flag_up* = TRUE,*flag_down* = TRUE,且 K_1 和 $K_2 - n$ 相等,如图 4(a)所示,则遍历像素 K_1,面积为 $\delta^2 - S_1 - S_2$,$K_1 \leftarrow K_1 - n + 1$,$K_2 \leftarrow K_2 - n + 1$;如果满足 *flag_up* = TRUE,*flag_down* = TRUE,且 K_1 和 $K_2 - n$ 不相等,则约定首先遍历像素 K_1,面积为 $\delta^2 - S_1$,$K_1 \leftarrow K_1 - n + 1$,然后遍历像素 $K_2 - n$,面积为 $\delta^2 - S_2$,$K_2 \leftarrow K_2 - n + 1$;如果 *flag_up* = TRUE,*flag_down* = FALSE,如图 4(b)所示,则遍历像素 K_1,面积为 $\delta^2 - S_1$,$K_1 \leftarrow K_1 - n + 1$;如果 *flag_down* = TRUE,*flag_up* = FALSE,如图 4(c)所示,则遍历像素 $K_2 - n$,面积为 $\delta^2 - S_2$,$K_2 \leftarrow K_2 - n + 1$。然后重新令 *flag_up* = FALSE,*flag_down* = FALSE。

第 6 步 如果 $i_1 \geqslant 0$,$i_2 \geqslant 0$,$j_1 < n$,$j_2 < n$,则转第 2 步;否则结束。

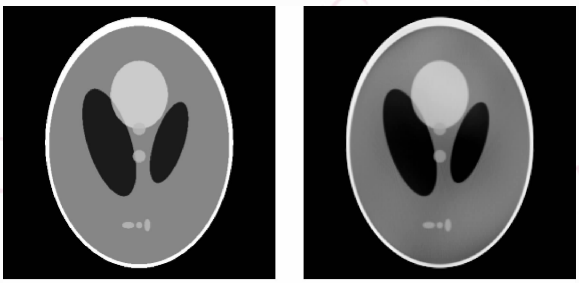
需要说明的是,实际中两条边界线的起始像素可能不在同一列,即 $j_1 \neq j_2$,此时必然有 $j_1 < j_2$,可以先让上边界线 l_1 进行遍历,直至 $j_1 = j_2$,然后再按上述算法进行。算法在执行过程中上边界线 l_1 可能会提前越界而终止,此时下边界线 l_2 按上述方法继续遍历,直至遇到重建区域的边界而结束。

对于一个 $n \times n$ 的重建区域,由于一条射束覆盖的像素总数不超过 $4n$,因此只需开辟 2 个大小为 $4n$ 的数组来分别记录由上述步骤所确定的像素索引及覆盖面积,再由覆盖面积除以像素面积 δ^2 即可得到权因子。与传统方法相比,本文算法节省了大量的存储空间。

上述分析和算法仅适用于 $0 < k \leqslant 1$ 的情形,对于其他情况的计算,可根据对称性得到。

4　实验结果及分析

为了验证算法的有效性,分别使用本文算法和传统方法对 ART 算法进行重建。重建模型为通用的 Shepp-Logan 模型[8],如图 5(a)所示,重建图像尺寸为 256 × 256,取 180 个投影角度,每个投影角下的射束为 256,$\lambda = 0.25$。测试计算机配置为 PⅣ 3.0 GHz 处理器、512 MB 内存,在 Windows XP 操作系统下用VC ++6.0实现了本文算法。表 1 给出了 2 种方法迭代一次的时间比较,图 5(b)给出了迭代三次后的重建图像。



(a) 原始模型　　(b) 重建图像

图 5　原始模型及迭代三次后的重建图像

Fig. 5　Original phantom and reconstructed image after 3 iterations

表 1　迭代一次所需时间对比

Tab. 1　Comparison of time costs for one iteration

方法	获取权因子	重建耗时(s)	总体耗时(s)
本文算法	1.282	0.610	1.891
传统方法	32.859	0.610	33.469

由表 1 可见,在一次迭代过程中,采用传统方法从硬盘文件中获取权因子耗时 32.859 s,为实际重建耗时的 53 倍多。利用本文算法在迭代过程中实时计算权因子仅耗时约 1.282 s,为实际重建耗时的 2 倍多。与传统方法相比,本文算法在获取权因子部分取得了 25 倍以上的加速比,总体重建加速比达到 17 倍以上,可见本文算法对重建速度的提升是非常显著的。由图 5(b)可以看出,经三次迭代后已经得到了满意的重建图像。

5　结　论

在 ART 算法的重建过程中,权因子的计算成为制约重建速度的瓶颈。针对平行束扫描方式提出了一种快速的射束与像素的遍历和求交算法,在迭代过程中实时计算权因子,不但节省了大量的内存空间,而且大大提高了 ART 算法的重建速度。与传统方法相比,在没有进行任何代码优化的情况下取得了 17 倍以上的重建加速比。这对于实现航空、航天等大型零部件的无损检测具有重要的意义。在今后的研究工作中,将对本文算法进行优化,并结合并行处理技术等来进一步提高 ART 算法的重建速度。

参考文献(References)

1　Sun Xiao-an, Chen Shu-zhen, Wu Zhi-bin, *et al*. Optimal method in image reconstruction [J]. Journal of Image and Graphics, 1999, **4A**(2): 105-109. [孙晓安,陈淑珍,吴志斌等. 图像重建中的最优化方法[J]. 中国图象图形学报, 1999, **4A**(2): 105-109.]

2　Klaus Muller. Fast and Accurate Three-dimensional Reconstruction from Cone-beam Projection Data Using Algebraic Methods [D]. OH, USA: Ohio State University, 1998.

3　Liu Yuan, Zhang Ding-hua, Zhao Xin-bo, *et al*. A rapid parellel ART based on SIMD technology [J]. Journal of Image and Graphics, 2007, **12**(1): 73-77. [刘远,张定华,赵歆波等. 一种基于 SIMD 技术的快速并行代数重建算法[J]. 中国图象图形学报, 2007, **12**(1): 73-77.]

4　Klaus Mueller, Roni Yagel. The weighted distance scheme: A globally optimizing projection ordering method for ART [J]. IEEE Transactions on Medical Imaging, 1997, **16**(2): 1-14.

5　Guan H, Gordon R. A projection access order for speedy convergence of ART: A multilevel scheme for computed tomography [J]. Physics in Medicine and Bidogy, 1994, **39**(1): 2005-2022.

6　Herman G, Meyer L. Algebraic reconstruction can be made computationally efficient [J]. IEEE Transactions on Medical Imaging, 1993, **12**(3): 600-609.

7　Qin Zhong-yuan, Mou Xuan-qin, Wang Ping, *et al*. A novel algebraic reconstruction technique of memory optimization and its fast implementation [J]. Acta Electronica Sinica, 2003, **31**(9): 1327-1329. [秦中元,牟轩沁,王平等. 一种内存优化的代数重建算法及其快速实现[J]. 电子学报, 2003, **31**(9): 1327-1329.]

8　Shepp L A, Logan B F. The Fourier reconstruction of a head section [J]. IEEE Transactions on Nuclear Science, 1974, **21**(1): 21-43.