A method based on vector type for sparse storage and quick access to projection matrix

YANG Juan, HOU Hui-ling, SHI Lang

(*Key Laboratory of Instrumentation Science & Dynamic Measurement (North University of China), Ministry of Education, Taiyuan 030051, China*)

Abstract: For sparse storage and quick access to projection matrix based on vector type, this paper proposes a method to solve the problems of the repetitive computation of projection coefficient, the large space occupation and low retrieval efficiency of projection matrix in iterative reconstruction algorithms, which calculates only once the projection coefficient and stores the data sparsely in binary format based on the variable size of library vector type. In the iterative reconstruction process, these binary files are accessed iteratively and the vector type is used to quickly obtain projection coefficients of each ray. The results of the experiments show that the method reduces the memory space occupation of the projection matrix and the computation of projection coefficient in iterative process, and accelerates the reconstruction speed.

Key words: projection matrix; sparse storage; quick access; vector type

CLD number: TP391 **Document code**: A

Article ID: 1674-8042(2015)01-0053-04 **doi**: 10.3969/j.issn.1674-8042.2015.01.010

Computed tomography (CT) technology is currently recognized as the best nondestructive testing (NDT) technology, and it is widely used in medical science and industry. CT reconstruction algorithms include analytic method and iterative method. The analytic method can reconstruct high quality images quickly, but it will have a bad result if the projection noise is high or the projection is incomplete. While because the iterative method does not need a complete projection, it could reconstruct a good image under these conditions. However, the iterative method has a large amount of computation and it takes a long time to meet the practical requirements. Furthermore, the projection matrix is a very important parameter in iterative method[1-2]. For example, in algebraic reconstruction technique (ART) process, the time to calculate projection matrix is almost 90% of the total reconstruction time. If the reconstructed image is, projection angle is, there are 256 rays in each angle and the coefficient is float type, thus, it needs at least 22.5 GB to store the projection matrix and it becomes hard to access what we need in ART process. Therefore, how to quickly and efficiently obtain the projection matrix is of great importance to iterative method.

Now many scholars based on Siddon algorithm put forward some methods to quickly compute projection coefficient[3-4]. These can speed up the calculation of projection matrix to some extent, but the calculation is still repetitive in each iteration, and using the fixed size array to store these data can not optimize memory efficiently.

We begin our discussion with an overview of ART algorithm, then particularly introduce how to compute projection coefficient, how to use vector type sparsely store and quickly access projection matrix. Finally, we demonstrate the efficiency of the proposed method through detailed experiments.

1 ART algorithm

In ART algorithm, one image is divided into $n \times n$

Received date: 2014-10-11

Foundation items: National Natural Science Foundation of China (No. 6171177)

Corresponding author: YANG Juan (yangjuanpinbo@163.com)

equal squares, and each square stands for a pixel. The iteration formula is

$$x_j^{(k+1)} = x_j^k + \lambda_k \frac{p_i - \sum_{j=1}^{N} w_{ij} x_j^{(k)}}{\sum_{i=1}^{N} w_{ij}^2} w_{ij}, \tag{1}$$

where i is the projection subscript, k denotes the iteration number and x_j is the image vector. We suppose that the projection line i passes through pixels j and w_{ij} represents the length of projection line through corresponding pixels. We name w as the system matrix and λ_k as the relaxation parameter, $0<\lambda_k<2$.

As we can see that w is used repetitively in ART iteration. Therefore, if we calculate w in each iteration, it saves some memory but costs lots of time to compute. Instead of this method, we compute and sparsely store only one time, then directly use it in each iteration.

2 Sparse storage and quick access to projection matrix

2.1 Library vector type

The library type vector is a C++ class template. A vector is a collection of objects of a single type, each of which has an associated integer index to access itself. Vector type can support variable size array and quickly access objects, as well as rapidly and efficiently add elements when program is running. Table 1 lists the commonly used vector operations[5].

Table 1 Vector operations

vector ⟨T⟩ v	Vector that holds objects of type T
v. empty()	Returns true if v is empty; otherwise returns false
v. size()	Returns number of elements in v
v. push_back(t)	Adds element with value t to end of v
v. clear()	Delete all the elements in v

2.2 Projection coefficient

To get projection coefficient, we need to compute the grid number and the corresponding length of projection line through pixels. As follows, we can use vector type instead of fixed-size array to define coefficients.

```
class Pointray
{ public:
      unsigned int num; //the grid number
      float length; }; //the length
vector⟨Pointray⟩ ray; //vector type
```

The computation steps of one ray's coefficients[6] are as follows.

1) Determine the coordinate system and the ray equation.

2) Compute the incident point $P(P_x, P_y)$ and exit point $Q(Q_x, Q_y)$ of the ray through object.

3) Define two vector classes as follows:

```
class Point
{ public: float j_x; float j_y; };
vector⟨Point⟩ jdx; vector⟨Point⟩ jdy;
```

While $P_x \leqslant x \leqslant Q_x$, $x=\lfloor P_x \rfloor+1$, get all the intersections, then do jdx. push_back(...).

While $P_x \leqslant y \leqslant Q_y$, $y=\lfloor P_y \rfloor+1$, get all the intersections, then do jdy. push_back(...).

Merge the two vectors, then sort these data from small to large according to j_x.

4) In a loop, compute the midpoint of two adjacent intersections, and then calculate the grid number and intersection length.

$$z_x = \frac{(x[i]+x[i+1])}{2}, \tag{2}$$

$$z_y = \frac{(y[i]+y[i+1])}{2}, \tag{3}$$

$$len = \frac{z_x}{\sqrt{z_x^2+z_y^2}}. \tag{4}$$

The corresponding operations are:

```
Pointray grid;
grid. num=floor(z_x)+n • floor(z_y);
grid. length=len;
ray. push_back(grid).
```

2.3 Sparse storage of projection matrix

Because the grid number is between 0 and $n-1$, thus, we can add these statements to distinguish coefficients among different rays.

The corresponding operations are:

```
grid. num=n×n;
grid. length=0;
```

```
    ray.push_back(grid);
```

If there are 180 projection angles, there will be n rays in each angle. After we calculate projection coefficients of n rays in one angle, we can store these coefficients in binary format to a txt file named as angle. The routine is as follows:

```
    for(m=0;m<ray.size(); m++)
        fwrite(&ray,sizeof(Pointray),1,fp);
```

2.4 Quick access to projection matrix

In reconstruction process, we read these txt files, then use vector to store all the coefficients in one angle. The routine is as follows:

```
fp=fopen(filef,"rb");
fseek(fp,0,SEEK_END);
len=ftell(fp);
w=new Pointray[len]; rewind(fp);
fread(w,sizeof(Pointray),len,fp); //read all the coefficients in one angle
```

Based on that the grid number is less than $n \times n$, we can integrally obtain coefficients of one ray and distinguish from other rays. The routine is as follows:

```
for(i=0;i<n;i++)
{if(ray.empty()==false)
        ray.clear();
    for(j=jj;j<len;j++)
    {if((w+j)->num<n×n)//if the grid number is less than n×n
    ray.push_back(*(w+j)); //get coefficients of one ray
        else
        { jj=j+1; break;} }
    reconstruction(); // ART reconstruction }
```

3 Experimental results and discussion

To clarify the effectiveness of the proposed method, the solid rocket engine is reconstructed, and then the memory and time used are compared with those of the method in Ref. [6]. The reference method is also called calculated on-the-fly, which means using Siddon algorithm to calculate one ray's coefficients and directly using them in reconstruction process, then calculating again before coefficients need to be used next time.

Visual Studio 2010 is used as developer kits. The test computer configuration is i5-3470 3.2 GHz, 4 GB double data rate (DDR) memory. The x-ray source is cone-beam and scans with a circular orbit. The detector is 1 920×1 536. The distance of X-ray source to the center of object is 1 060 mm and the center of detector is 140 mm to the center of object. The projection angle is 0°—359°, and the sample interval is 1°. We reconstruct the 770th slice by using ART algorithm. The reconstructed image is 256 × 256. Fig. 1 shows the reconstructed result.

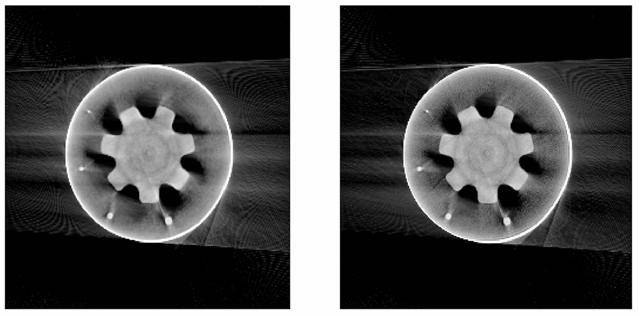
(a) One iteration (b) Two iterations

Fig. 1 Reconstructed solid rocket engine

It costs about 135.24 s to use Siddon algorithm to compute projection matrix and then use vector type to sparsely store the coefficients. After sparsely storing, the memory usage is 227 MB. It is much less than 22.5 GB. Table 2 lists the reconstructed time usage of the proposed method and that of the method in Ref. [6].

Table 2 Comparison of time usage

Method	Number of iterations	
	1	2
The proposed method	3.259	6.357
The method in Ref. [6]	158.329	316.455

The reconstructed time use of proposed method includes the time to quickly access projection matrix and the time that uses ART to reconstruct. Therefore, the proposed method costs about 138.499 s (135.24+3.259) in the first iteration, and it only costs about 3 s in the second iteration. However the method in Ref. [6] costs about 158.329 s to calculate coefficients and uses them in ART to reconstruct in each iteration. It can be seen that the proposed method could effectively reduce memory space occupation of storing projection matrix and time use of reconstruction.

4 Conclusion

This paper put forward an approach that uses vector type to sparsely store and quickly access projection matrix, which can effectively reduce the memory space occupation of projection matrix and iteration reconstruction time use. Finally, in VS2010 software platform-combined with ART algorithm the solid rocket engine is reconstructed. And the time and memory use are analyzed to demonstrate the effectiveness of the proposee approach. However, if parameters of capture device changed, the projection matrix needs to be calculated and stored again.

References

[1] ZENG Geng-sheng. Medical image reconstruction. Beijing: Higher Education Press, 2010: 21-125.

[2] JIANG Ming, WANG Ge. Convergence studies on iterative algorithms for image reconstruction. IEEE Transactions on Medical Imaging, 2003, 22(5): 569-579.

[3] ZHANG Shun-li, ZHANG Ding-hua, ZHAO Xin-bo. Research of fast image reconstruction on ART algorithm. Computer Engineering and Applications, 2006, 24: 1-4.

[4] ZHOU Bin, WANG Lian-tang, WANG Jun-jie, et al. Fast projection coefficient computation method in algebraic reconstruction technique. Computer Engineering and Applications, 2008, 44(25): 46-47.

[5] Lippman S B, Lajoie J, Moo B E. C++ Primer. 4th Edition. American: Addison Wesley Professional, 2005: 158-167.

[6] CHEN Hong-lei, HE Jian-feng, LIU Jun-qing. Projection matrix algorithm based on two-dimensional index. Computer Engineering, 2013, 39(2): 229-232.

一种基于容器的对投影矩阵稀疏存储与快速访问的方法

杨　娟，侯慧玲，石　浪

（中北大学 仪器科学与动态测试教育部重点实验室，山西 太原 030051）

摘　要： 针对迭代重建算法中投影系数的重复计算，以及投影矩阵存储占用空间大，检索效率低等问题，本文提出一种基于 vector 容器的投影矩阵稀疏存储与快速访问方法。该方法只计算一次投影系数，并利用容器的大小可变性将投影系数以二进制格式进行稀疏存储。在迭代重建过程中，循环访问这些二进制文件，并使用容器快速检索获得每一条射线的投影系数。实验证明，本文提出的方法有效地减少了投影矩阵占用的内存，减少了迭代过程中计算投影系数的运算量，加快了重建速度。

关键词： 投影矩阵；稀疏存储；快速访问；容器

引用格式： YANG Juan, HOU Hui-ling, SHI Lang. A method based on vector type for sparse storage and quick access to projection matrix. Journal of Measurement Science and Instrumentation, 2015, 6(1): 53-56. [doi: 10.3969/j.issn.1674-8042.2015.01.010]