A very fast Implementation of 2D Iterative Reconstruction Algorithms

Peter Toft, Department of Mathematical Modelling,
Technical University of Denmark, DK-2800 Lyngby, Denmark, Email pto@imm.dtu.dk.
Jesper James Jensen, ØDS-Holding A/S, Kroghsgade 1,
DK-2100 Copenhagen E, Denmark, Email jjj@oedan.dk.

Abstract

One of the limitations of using iterative reconstruction methods in tomography is the slow performance compared with the direct reconstruction methods, such as Filtered Backprojection. In this paper we demonstrate a very fast implementation of most types of iterative reconstruction methods. The key idea of our method is to generate the huge system matrix only once, and store it using sparse matrix techniques. From the sparse matrix we can perform the matrix vector products very fast, which implies a major acceleration of the reconstruction algorithms. In this paper we demonstrate that iterative reconstruction algorithms can be implemented and run almost as fast as direct reconstruction algorithms. The method has been implemented in a software package that is available for free, providing reconstruction algorithms using ART, EM, and the Least Squares Conjugate Gradient Method.

I. Introduction

In the last 15 years the iterative reconstruction methods have gained much attention in the literature [1, 2]. Several methods have been very prominent, such as EM (Expectation Maximization) [3, 4, 5, 6, 7], ART (Algebraic Reconstruction Technique) [8, 1], and LSCG (Least Squares Conjugate Gradient) [9, 10].

These methods formulate the reconstruction problem as a linear set of equations

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} \;\Leftrightarrow\; b_i = \sum_{j=1}^{J-1} a_{i,j} x_j, \quad i = 1, 2, \ldots, I \qquad (1)$$

where $\boldsymbol{b}$ is an I-dimensional vector containing the known sinogram values wrapped into a vector, and $\boldsymbol{x}$ is a J-dimensional vector containing the unknown image to be reconstructed. Here $\boldsymbol{A}$ is the system matrix, which contains the weight factors between each of the image pixels and each of the values in the sinogram, corresponding to line orientations. Compared with Radon transform based direct reconstruction methods [11], the use of linear algebra has several advantages, such as easier incorporation of irregular geometries. The system matrix can model several real-world properties, such as finite, i.e., non-zero detector size and varying detector sensitivity. Furthermore regularization can easily be incorporated [12, 9] in order to affect the often ill-conditioned reconstruction problem.

One problem is the huge size of the system matrix. A 2D sinogram from, e.g., a GE Advance PET scanner contains $I = 281 * 336$ values, and reconstructed into a $J = 201 * 201$ grid, i.e., the system matrix has approximately 3.8 billion elements, requiring over 15 GBytes of memory, when using 4 bytes per matrix element. This is a large amount of memory, even looking some years into the future. Besides this aspect, it would not be wise to store all that data, due to the fact that approximately 99% of the matrix entries will be zeros. This knowledge should be incorporated into the reconstruction schemes.

Assuming that memory is not available for storing the full system matrix, one possibility is to compute the individual matrix elements in each iteration when needed. This can be done by using the Radon transform, e.g., [13] or other modelling schemes for the scanner. This approach is rather easily implemented and is viable and storage requirements are reduced to a minimum, only requiring memory for the sinogram ($\boldsymbol{b}$) and the current solution ($\boldsymbol{x}$), and perhaps some additional temporary variables of the same size or smaller, but no system matrix is stored in memory. It will be demonstrated that this implementation has a major drawback in speed, since the system matrix will be computed many times during an iterative reconstruction. Each time at the same high computational cost.

II. Accelerated 2D Iterative Reconstruction

Here a hybrid solution is investigated [14, 13] for accelerating the iterative reconstruction algorithms, but requiring as much memory as modern workstations are currently equipped with, or will be soon. The idea is to store the non-zero elements of the system matrix in the main memory using sparse matrix techniques. In this way the core of the reconstruction algorithms, highly based on matrix vector multiplications, can be accelerated significantly, and thereby solve one of the major drawbacks of the iterative methods.

It is proposed that the system matrix $\boldsymbol{A}$ is calculated one time only using all the modifications found for the actual scanner setup. If no specific scanner model is provided then the system matrix can be modelled and generated using the Radon transform or other simpler schemes. From the system matrix the very small values in the matrix can truncated to zero,

$$\tilde{a}_{i,j} = \begin{cases} a_{i,j} & \text{if } a_{i,j} > \gamma \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

where the threshold γ can be chosen to a certain fraction of the maximum matrix value, e.g., $\gamma = 0.05 \max_{i,j}\{a_{i,j}\}$. If γ is chosen sufficiently low, a good compromise between resolution and the sparseness of the matrix can be reached, and normally this does not alter the behaviour of the algorithms. Current work concerns the quantification of the truncation error.

The sparse structure of $\boldsymbol{A}$ can be exploited by only storing non-zero values in the fast memory. For a certain row, number i, all of the matrix elements are calculated, stored, and truncated using Eq. 2. Hereby the number of non-zero elements in the row, denoted by Z_i, will be much smaller than the image size $J = M^2$. The values of Z_i are stored in a simple, one dimensional vector. Two vectors of length Z_i, indexed by an integer z, can then be allocated and stored containing the non-zero matrix value a_z and the corresponding column index j_z. The procedure is repeated for all rows.

Assuming a nearest neighbour approximation with one pixel for each point along the integration lines and using 4 bytes for storing each of the vector elements, the total storage requirement is then reduced to approximately $8\sum_{i=1}^{I} z_i \approx 8IM$ bytes. In the example shown in the introduction approximately 100 MBytes memory is required. Assuming this amount of memory is present most iterative algorithms can be implemented from three basic operations: Matrix vector multiplication $\boldsymbol{A}\tilde{\boldsymbol{x}}$, scalar product between the i'th row of the system matrix and a vector $\boldsymbol{a}_i^T\tilde{\boldsymbol{x}}$, and finally multiplication with the transpose of the matrix $\boldsymbol{A}^T\tilde{\boldsymbol{b}}$.

In the following pseudo code (called Algorithms), the implementation of the matrix vector multiplication and the multiplication with the transpose of the system matrix are shown. A C++ style is used for comments and note that all indices here start at zero.

Algorithm 1 : $\boldsymbol{Ax}$

```
For i = 0 to I-1                 //For all rows
 sum = 0                         //Initialize
 Set a and j to correct row      //Use pointers
 For z=0 to Z(i)-1               //For row i
  sum=sum+a(z)*x(j(z))           //Increment sum
 End
 bt(i)=sum                       //Store value
End
```

End Algorithm

Algorithm 2 : $\boldsymbol{A}^T\boldsymbol{b}$

```
For j=0 to J-1                   //For all columns
 xb(j)=0                         //Initialize
End
For i=0 to I-1                   //For all rows
 Set a and j to correct row      //Using pointers
 For z=0 to Z(i)-1               //Compute sum
  xb(j(z))=xb(j(z))+a(z)*b(i)    //Update sum
 End
End
```

End Algorithm

III. Implemented Methods

A software package has been written in C including the proper structures for manipulating sparse matrices and vectors, along with an optimized code for computing matrix vector products, well suited for iterative reconstruction algorithms. In the package ART, EM, and LSCG are implemented both in a fast version using sparse matrix storage of the system matrix and in a slow version where the system matrix is not stored and needed matrix entries are computed in each step of the iterative algorithms. The software package is available for free, but protected by the GNU General Public License. The package is available at `http://eivind.imm.dtu.dk/staff/ptoft`.

ART: For a certain row i of the matrix (i depends on the iteration number k), the general iteration step incrementing the current solution $\boldsymbol{x}^{(k)}$ can, e.g., be found in [1]

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \frac{b_i - \boldsymbol{a}_i^T \boldsymbol{x}^{(k)}}{\boldsymbol{a}_i^T \boldsymbol{a}_i} \boldsymbol{a}_i \tag{3}$$

EM: The general iteration step of EM [5, 15] requires a forward projection, a backprojection, and two fast updates in each iteration.

$$\boldsymbol{b}^f = \boldsymbol{A}\boldsymbol{x}^{(k-1)} \tag{4}$$

$$b_i^r = \frac{b_i}{b_i^f} \tag{5}$$

$$\boldsymbol{x}^b = \boldsymbol{A}^T \boldsymbol{b}^r \tag{6}$$

$$x_j^{(k)} = \frac{x_j^{(k-1)} x_j^b}{s_j} \text{ where } s_j = \sum_{i=1}^{I} a_{i,j} \tag{7}$$

LSCG: The Least Squares Conjugate Gradient method requires some initialization [10]

$$\begin{aligned}
\boldsymbol{s}^{(0)} &= \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)} \\
\boldsymbol{r}^{(0)} &= \boldsymbol{p}^{(0)} = \boldsymbol{A}^T \boldsymbol{s}^{(0)} \\
\boldsymbol{q}^{(0)} &= \boldsymbol{A}\boldsymbol{p}^{(0)}
\end{aligned}$$

Then for each iteration the LSCG algorithm on the normal equations becomes

$$\begin{aligned}
\alpha &= \frac{\left(\boldsymbol{r}^{(k-1)}\right)^T \boldsymbol{r}^{(k-1)}}{\left(\boldsymbol{q}^{(k-1)}\right)^T \boldsymbol{q}^{(k-1)}} \\
\boldsymbol{x}^{(k)} &= \boldsymbol{x}^{(k-1)} + \alpha \boldsymbol{p}^{(k-1)} \\
\boldsymbol{s}^{(k)} &= \boldsymbol{s}^{(k-1)} - \alpha \boldsymbol{q}^{(k-1)} \\
\boldsymbol{r}^{(k)} &= \boldsymbol{A}^T \boldsymbol{s}^{(k)} \\
\beta &= \frac{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{r}^{(k)}}{\left(\boldsymbol{r}^{(k-1)}\right)^T \boldsymbol{r}^{(k-1)}} \\
\boldsymbol{p}^{(k)} &= \boldsymbol{r}^{(k)} + \beta \boldsymbol{p}^{(k-1)} \\
\boldsymbol{q}^{(k)} &= \boldsymbol{A}\boldsymbol{p}^{(k)}
\end{aligned}$$

For all three methods an initial value of the solution, i.e., $\boldsymbol{x}^{(0)}$ is needed. In the package an image found by, e.g., a fast direct method, can be supplied and used. If not provided, all of the initial values of the vector are initialized to a properly chosen constant.

A. *Interpolation Methods*

Several interpolation methods have been implemented for computing the system matrix. Furthermore it is possible to use sub-sampling in the Radon domain and average over the sub-windows in order to incorporate a model of a non-zero detector size. This will reduce aliasing problems.

- Nearest Neighbour interpolation based on the Radon transform.
- Linear interpolation based on the Radon transform.
- Analytical Radon transform of a square, i.e., the length through a quadratic pixel is used.
- The Radon transform of a sinc expansion in the image domain.

IV. Results

The program has been used on two types of machines. A Linux machine with a 120 MHz Pentium processor and an Onyx from SGI equipped with four 200 MHz R4400 processors, where the program was running on one processor.

A. *Example 1*

In the first example the (synthetic) sinogram has $125 * 101$ samples and the reconstructed image has $101*101$ samples. In Table 1 the reconstruction times on both machines are shown for the fast and the slow method as well as the ratio between the execution times (slow/fast).

Times are measured for ART, EM, and the LSCG-method, when EM and LSCG were running (arbitrarily) 20 iterations, and ART 20 full iterations, i.e., 20 times the number of rows (chosen randomly), which is $20*125*101$ iterations in Eq. 3. Note that all times only correspond to the actual iterations. For the fast versions of the iterative reconstruction algorithms, the time to generate the system matrix once should be added if changing the system matrix, e.g., when changing the sampling parameters of the reconstructed image.

For this example the system matrix was modelled using discrete Radon transformation with linear interpolation, where the threshold γ was chosen to zero, hence the slow and the fast methods give exactly the same results. Note that the large difference in speedup between ART and EM/LSCG is due to the implementation of the forward projection is more efficient than the multiplication with the transpose of the system matrix. The slow methods can be accelerated some by implementing multiplication with the transpose of the system matrix (adjoint operator) as a backprojection integral, but note that this implies that the approximation of the system matrix will be different in the forward and the backprojection part. The sparse system matrix for this transformation geometry required approximately 13 MBytes, and each iteration requires approximately one second.

Machine	Type	ART	EM	LSCG
Pentium	Fast	26 sec	17 sec	17 sec
	Slow	2218 sec	5776 sec	5250 sec
	Ratio	85	340	309
Onyx	Fast	16 sec	16 sec	16 sec
	Slow	1306 sec	2722 sec	2715 sec
	Ratio	82	170	170

Table 1 Time usage for 20 iterations of EM and LSCG. For ART the time is for 20 full iterations, i.e., $20*125*101$ of the iterations used in Eq. 3. The time measurements are for a sinogram with $125*101$ samples reconstructed into a $101*101$ samples image.

B. *Example 2*

A 2D sinogram corresponding to a human brain was obtained on a GE Advance PET scanner. The sinogram with $281*336$ samples is shown in Fig. 1,

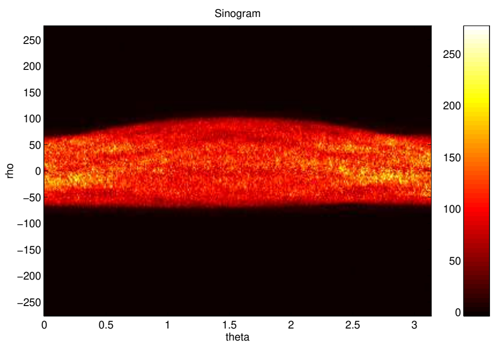

Figure 1 A PET sinogram of a human brain.

The sinogram is reconstructed into a image with $201*201$ samples. The system matrix has $94416*40401$ elements of which 0.62% are non-zero when modelling then system matrix using the Radon transform with linear interpolation and each element is averaged over a sub-sampled $3*3$ window in order to avoid aliasing problems. On the Onyx it required 22 minutes to generate the 189 MBytes sparse matrix, and each iteration of EM or LSCG required approximately 9 seconds in the fast version. After 10 iterations of EM the algorithm was stopped, and the reconstructed image is shown in Fig. 2. Using LSCG it appears that the best reconstructed result is obtained after 9 iterations which is shown in Fig. 3. For sake of comparison, a reconstructed image using Filtered Backprojection is shown in Fig. 4, and this reconstruction used in total 6 seconds on the Onyx.

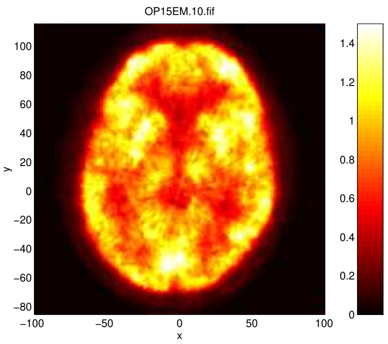

Figure 2 The reconstructed image after 10 iterations of EM.

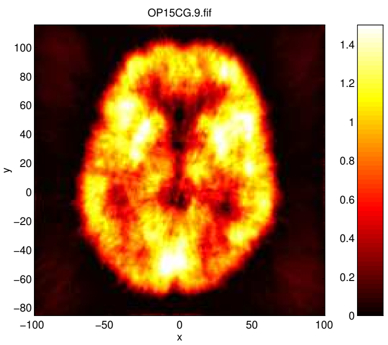

Figure 3 The reconstructed image after 9 iterations of LSCG.

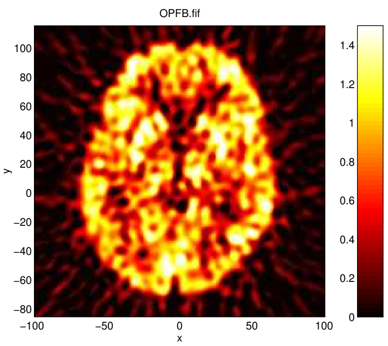

Figure 4 The reconstructed image using Filtered Backprojection using a Hann window with cutoff at half of the sampling frequency.

V. Conclusion

We have demonstrated a very fast implementation of iterative reconstruction comparable in speed with direct reconstruction methods. The implementation is based on storing of the system matrix in fast memory using sparse techniques. The approach is mainly applicable to 2D reconstruction, due to the requirements of a sufficient amount of memory, but in principle the method can also be applied to 3D reconstruction.

The idea is implemented in a software package which is available for free. In the package several direct reconstruction methods are also available, as well as regularization tools and several constraining methods.

The cost of the strategy is that a large amount of memory is required, but for ART we have demonstrated a speedup factor of approximately 80 and for EM and LSCG 170-340 depending on the machine, for a fixed transformation geometry and interpolation level. These factors could be somewhat moderated by a faster implementation of the multiplication with the inverse system matrix.

Acknowledgment

We thank associate professor Lars Kai Hansen for support and useful comments.

References

[1] Yair Censor. Finite Series-Expansion Reconstruction Methods. In *Proc. of the IEEE*, volume 71, pages 409–419, March 1983.

[2] Julia K. Older and Paul C. Johns. Matrix formulation of computed tomography reconstruction. *Physics in Medicine & Biology*, 38:1051–1064, 1993.

[3] L. A. Shepp and J. B. Krustal. Computerized tomography: The new medical x-ray technology. *Am. Math. Monthly*, 85:420–439, April 1978.

[4] L. A. Shepp Y. Vardi and L. Kaufman. A Statistical Model for Positron Emission Tomography. *Journal of the American Statistical Association*, 80(389):8–37, March 1985. The pages include comments and discussion by several authors.

[5] R. E. Carson and K. Lange. The EM parametric image reconstruction algorithm. *Journal of the American Statistical Association*, 389(80):20–25, March 1985.

[6] Linda Kaufman. Implementing and Accelerating the EM Algorithms for Positron Emission Tomography. *IEEE Trans. Med. Imag.*, MI-6(1):37–51, march 1987.

[7] John M. Ollinger. Maximum-Likelihood reconstruction of transmission images in emission computed tomography via the EM algorithm. *IEEE Trans. Med. Imag.*, 13(1):89–101, March 1994.

[8] Gabor T. Herman and Lorraine B. Meyer. Algebraric Reconstruction Techniques Can Be Made Computationally Efficient. *IEEE Trans. Med. Imag.*, 12(3):600–609, Sep. 1993.

[9] Linda Kaufman and Arnold Neumaier. Image Reconstruction Through Regularization by Envelope Guided Conjugate Gradients. Technical report, Bell Labs, 1993. Report 11274-94-14.

[10] John A. Scales. Iterative methods for large, sparse, inverse calculations. Samizdat Press. Colorado School of Mines, 76 Olcott Drive, White River Junction, Vermont 05001, April 1993. Lecture notes.

[11] S. R. Deans. *The Radon Transform and Some of Its Applications.* Krieger Publishing Company, Malabar, Florida, 2 edition, 1993. Previously John Wiley & Sons Inc., 1983.

[12] Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems.* Polyteknisk Forlag, Lyngby, Denmark, 1996. Doctoral Dissertation.

[13] Peter Toft. *The Radon Transform - Theory and Implementation.* PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, 1996.

[14] Tom R. Miller and Jerold W. Wallis. Fast maximum-likelihood reconstruction. *Journal of Nuclear Medicine*, 33(9):1710–1711, September 1992.

[15] G. T. Herman and D. Odhner. Performance evaluation of an iterative image reconstruction algorithm for positron emission tomography. *IEEE Trans. Med. Imag.*, 10:336–246, 1991.