PVM Implementations of 3D Reconstruction Algorithms

C. LAURENT, F. PEYRIN and J.-M. CHASSERY

TIMC-IMAG Laboratory, URA CNRS D 1618
Institut Albert Bonniot, Domaine de la Merci, 38706 La Tronche cedex, FRANCE
E-mail : Christophe.Laurent@imag.fr

Abstract

The problem of reconstructing a 3D image from a set of its 2D conic projections from different angles of view arises in medical imaging for 3D tomography. 3D tomography requires voluminous data and long computation times. Parallel computing seems to be a good approach to manage this problem. In this study, we propose different reconstruction algorithms using PVM. We investigate two communication schemes; one using synchronous communications and other one using asynchronous communications. Experimented results are presented.

1 Introduction

Truly three dimensional tomography is a generalisation of conventional 2D tomography allowing the reconstruction of volumes (3D images). In X-Ray tomography, some prototypes using the rotation of a (or several) cone-beam X-Ray source(s) have been built [9][5]. Cone-beam geometry acquisition systems are also used in Simple Photon Emission Tomography (SPECT). In all these cases, the computional problem is to reconstruct a 3D image from a set of its 2D conic projections from different angles of view. The 2D reconstruction methods are not adapted and the problem has to be considered directly in 3D. These reconstruction algorithms involve large amount of data (536 Mbytes for a 512^3 image), and large computation time. For example, the reconstruction of a 128^3 image from about of 100 128^2 projections, using the extention of the classical filtered back-projection algorithm know as Feldkamp algorithm, requires about 10 hours on a SUN4 workstation.

Parallel computers seem to be a good solution to get over these long computation times and memory requirements. Different solutions have been so far proposed in the literature, some using dedicated parallel architectures, other classical parallel computers. Atkins [1] has proposed an architecture based on a transputer network for 3D PET tomography. Some analytic methods have been developped on MIMD parallel computers (NCUBE and IPSC), which use different topologies (hypercube[10], grid and torus) and communication schemes based on binary tree [3] or ring [2]. Algebraic and statistical methods have been developped on an IPSC (MIMD computer) for 3D PET [4], and on a MASPAR (SIMD computer) for 3D SPECT[8]. In this study, we are interested with the implementation of cone beam X-Ray reconstruction algorithms on MIMD parallel computers. We propose different solutions for the parallel implementations of the basic operators in tomography (projection,backprojection). We have chosen PVM for its flexibility and its portability. We present different algorithms with an evaluation of their complexity. Results on simulated data are presented.

2 Cone-Beam Reconstruction

The possible 3D reconstruction methods, from cone-beam acquisitions, may be classified into analytic methods, algebraic methods and statistical methods [6]. Analytic methods are based on a continous formulation of the problem, and on an inverse analytic formula expressing the 3D object $f(\vec{x})$ to be reconstructed in function of its 2D cone-beam projection $p_\tau(\vec{l})$. The conventional Feldkamp algorithm [7] is based on an approximated analytic inversion formula. It consists in the filtering and the backprojection of the 2D weighted projections. Algebraic methods are based on a discrete formulation of the problem which is equivalent to solve a large linear system $\mathcal{X} f = p$, where f represents the image, p the set of acquisitions and $\mathcal{X}$ the projection operator. Since for the reconstruction of a 128^3 image from 100 128^2 projections, the matrix size $\mathcal{X}$ is approximately $4.10^6 * 8.10^5$, it is scarcely generated on the computer. Conventionnal reconstruction methods such as ART or SIRT are iterative and may be interpreted as iterations of weighted backprojection of some differences between computed and measured data.

Thus, most of analytic and algebraic algorithms use the same basic operators: projection and backprojection. Since these operations may be considered as dual from each others, their implementation will be similar. The implementation of such operators, using a "voxel driven" method, requires three basic functions: address of the projection for each voxel, ponderation for each voxel, and a pixel interpolation model since data are discrete. For m 2D projections of size M^2 and a 3D image of size N^3 The complexities of the projection algorithms and the backprojection algorithms are $O(m.N^3)$. The schemes of the algorithms are given below.

```
        Projection algorithm                     Backprojection algorithm

for(l=0; l< m;l++)                          for(l=0; l< m;l++)
  Create(Projection);                         read(Projection);
  for(k=0; k<N; k++)                          for(k=0; k<N; k++)
  for(j=0; j<N; j++)                          for(j=0; j<N; j++)
  for(i=0; i<N; i++)                          for(i=0; i<N; i++)
    calculate pixel(p,q) from voxel(i,j,k)      calculate pixel(p,q) from voxel(i,j,k)
    Projection[p+M*q]=Image3D[i+j*N+k*N*N];     Image3D[i+j*N+k*N*N]=Projection[p+M*q];
  write(Projection);                        write(Image3D);
```

3 Parallelisation of basics operators

The target parallel machine, used in this study, is a workstation network supporting PVM. The network is composed of five SUN 4 workstations with two fastest workstations. The basic operators may be considered as "data parallel" algorithms. We present below the data partition.

3.1 The data partition

Each voxel of the 3D image is independant of the other voxel, its 2D projection depends of the geometric parameters and changes for each angle of view. Then we have chosen to decompose the 3D image, rather than the 2D projection, on the processor memory. We assume that the 3D image (N^3) is decomposed in PE subimages of size $N_{PE}.N^2$ with $PE.N_{PE} = N$, and that we have m 2D projections (M^2) distributed on PE processors, with m_{PE} M^2 images on each processor ($m = m_{PE}.PE$). To compute a local projection or a local backprojection on a processor, the 2D projection and a subimage from 3D image are required, on this processor. As the data are distributed on processor memory, and to minimize the communications cost, we must choose to communicate either the 3D subimages either the 2D projections. For instance, a global projection, computed on a ring of PE processors with a synchronous communication, will cost:

- $(PE-1)\frac{m.M^2}{PE}$ if the 2D projections are communicated,
- $(PE-1)\frac{N^3}{PE}$ if the 3D image are communicated.

The choice of communication scheme depends if $N^3 > m.M^2$. In this study, we assume that it is the case and we prefer to communicate 2D projections by the network rather than the 3D image.

3.2 Implementation of projection and backprojection on PVM

Since the implementation of projection and backprojection algorithms are similar, we only present projection algorithms. We use two communication schemes. The fisrt one uses "synchronous" communications, the second uses "asynchronous" communications. We use the second algorithm with a "Adaptative data partition". The two algorithms are presented below.

- The "synchronous" version is implemented like a "data parallel" model. The data are distributed on the processor memory, and we make alternately a communication phase and a calculation phase. Each processor must see all the 2D projections to compute the local projection with its 3D subimage (projections_seen,Proc_projection[ ] array of projections seen). When each 2D projection have seen the whole set of 3D subimages (projection.proc_seen), they may be written. The theoretical cost is $O(N_{PE}.N^2.m)$ for the calculations and $O((PE-1)\frac{m.M^2}{PE})$ for the communication of the 2D projections.
- On the workstation network, we have different types of workstations. On the presented synchronous version, the slowliest workstations penalize the fastest. We then implement an asynchronous version to take into account the workstations disparity. This implementation allows to mask partialy the communications by the calculations,and the total theoretical cost becomes $O(N_{PE}.N^2.m + \varepsilon.(PE-1)\frac{m.M^2}{PE})$ where ε is the percentage of communication not masked. The communications being asynchronous, if we have one processor by workstation, the processors, finishing fisrt, are on the fastest workstations. Then with this algorithm version, the different PVM processors are more independants.

- To improve the total execution time, there are two solutions. The first one consists to increase the number of virtuals processors on faster workstations. But these algorithms will be implemented on MIMD computers with only one process by processor. The second one consists in distributing the 3D image according to the speed of each workstation by using a "Adaptative data partition". For this purpose, after each execution, we evaluate the processor performance and their location to calculate the workstation speed. During the next execution, these informations are used to allocate the 3D image. All the processors determine their location and their theoretical speed up, and communicate them to one processor. This later calculates the partition of data for each processor and reply to them. After a few executions when the workstations network is only used for our applications, each workstation speed up will be defined and the data partition will be optimized. The theoretical cost is $O(N_{speedup}.N^2.m + \varepsilon.(PE-1)\frac{m.M^2}{PE})$ where $N_{speedup}$ is the size depanding of workstation speed up.

```
      synchronous communications

Create(projection);
projection.proc_seen=0;projections_seen=0;
while(projections_seen != m)
{  if(projection.proc_seen != PE)&&
     (Proc_projection[projection]==not_seen)
   {  Local_Projection(projection);
      Proc_projection[projection]=seen;
      projections_seen++;projection.proc_seen++;
   }
   if(projection.proc_seen == PE)
   {  write(projection);
      if(projection_seen < m)
      {  Creat(projection);
         projection.proc_seen=0; }
   }else{
      pvm_initsend(PvmDataDefault);
      pvm_pkfloat(projection);
      pvm_send(MyProc+1,projection);  }
   if(projection_seen!=m)&&(projection.proc_seen!=0)
   {  pvm_recv(MyProc-1,projection);
      pvm_upkfloat(projection);
   }
}
```

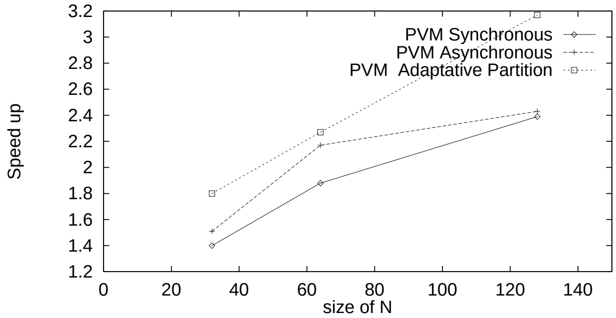
```
asynchronous communications

projections_seen=0;
while(projections_seen != m)
{  if(pvm_nrecv(MyProc-1,projection)
   {   pvm_upkfloat(projection); }
   else{
       if(projection_seen < m)
       { Create(projection);
         projection.proc_seen=0; }
   }
  if(projection.proc_seen != PE)&&
    (Proc_projection[projection]==not_seen)
    {   Local_Projection(projection);
        Proc_projection[projection]=seen;
        projections_seen++;projection.proc_seen++;
    }
    if(projection.proc_seen == PE)
    {   write(projection); }
    else
    {   pvm_initsend(PvmDataDefault);
        pvm_pkfloat(projection);
       pvm_send(MyProc+1,projection);
    }
 }
```

3.3 Some execution time for the projection and backprojection algorithms

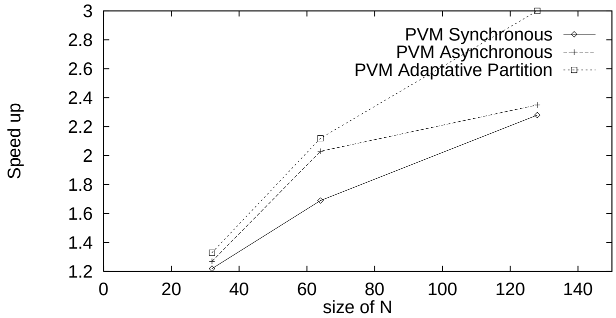
As an illustration we present in the next table, some computation times obtained on a network of workstations, for a N^3 image and 100 128^2 projections (pixels). We give the execution times obtained for the two operators (projection, backprojection) and for a complete analytic reconstruction algorithm (Feldkamp algorithm) which performs the backprojection of the filtered 2D projections. The filtering operation is implemented using the Fast Fourier Transform which is done when each processor reads the 2D projection before the calculations. We have tested each algorithm with the three implementations on PVM. For comparaison, the sequential time obtained on SUN4 workstationis given (the fastest workstation is used). The figure 1 presents the speed up of the different implementations

Algorithm	PE	Projection(sec)			Backprojection(sec)			Feldkamp(sec)		
N		32	64	128	32	64	128	32	64	128
SUN4	1	162	961	7639	118	923	7334	437	1565	10606
Synchronous communications	5	116	521	3202	97	547	3211	199	628	3395
Asynchronous communications	5	107	443	3142	93	455	3123	191	541	3216
Adaptative data partition	5	90	424	2412	89	436	2449	171	446	2593

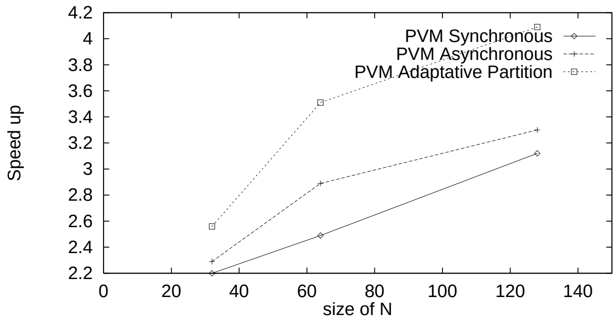
As the operators are dual, the projection and the backprojection implementations have similar computation times while the Feldkamp implementation is penalized by the filtering cost. The speed up increases with the size of 3D image for all implementations. We observe that the best speed up are obtained with the Feldkamp implementations because the sequential algorithm is more penalized by the filtering than the parallel algorithms.



(a) Projection Speed up



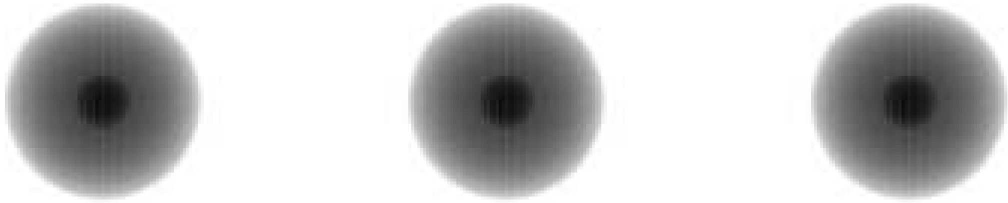
(b) Backprojection Speed up



(c) Feldkamp Speed up

Figure 1: Speed up of basics operators and Feldkamp implementations

The "asynchronous" algorithms won an average only 5 percent of speed up compared to "synchronous" algorithms because communication cost. The "Adaptative data partition" algorithms allow to optimize calculations and communications with this network of heterogeneous workstations. Then it won an average 20 percent of speed up compared to "synchronous" algorithms. The best effiency obtained is 84 percent for the Feldkamp implementation with the "Adaptative data partition" algorithms. The next illustration shows three projections of as simulated 3D image composed of two spheres.



(a) $\Theta = 0$ (b) $\Theta = 60$ (c) $\Theta = 120$

Figure 2: Projection of 3D image at different angles of view (Θ)

4 Conclusion

We have presented the implementation of reconstruction methods on a network supporting PVM. When using the synchronous communication mode, the experimented results are not in accordance with the theoretical complexities which should be in $O(m.N^3/PE)$. This may be related to the heterogeneity of the workstation network. The Feldkamp parallel implementations obtain the best speed up because the long calculation step. The asynchronous implementations allow to win 5 percent of speed up and that show the recovery of communications by calculations is not global. However, the different results show a speed up increase when using the "Adaptative data partition" algorithms. The best speed up obtained with this algorithms show the good parallelization of the 3D Reconstruction Algorithms.
Similar implementation of projection and backprojection operators will be tested on a CRAY T3D and on a PARAGON of Intel. Experimental 2D acquisitions from the Morphometre are used. The parallelisation of reconstructions algorithm will allow to reconstruct large images and to reduce computation time.

5 Acknowledgments

This work has been done within the french CNRS reseach group (GDR 134) TDSI (Traitement du signal et de l'image).

References

[1] M. S. Atkins, D. Murray, and R. Harrop. Use of transputers in a 3-D Positron Emission Tomograph. *IEEE Transactions on Medical Imaging*, 12(2):173–181, 1993.

[2] H.-P. Charles, J.-J. Li jjli, and S. Miguet. 3D image processing on distributed memory parallel computers. In *SPIE*, volume 1905, pages 379–390, 1993.

[3] C. M. Chen, S.-Y. Lee, and Z. H. Cho. A Parallel Implementation of 3-D CT Image Reconstruction on Hypercube Multiprocessor. *IEEE Transactions on Nuclear Science*, 37(3):1333–1346, 1990.

[4] C. M. Chen, S.-Y. Lee, and Z. H. Cho. Parallelisation of EM Algorithm for 3-D PET Image Reconstruction. *IEEE Transactions on Medical Imaging*, 10(4):513–522, 1991.

[5] D. Saint-Félix et al. A New system for 3D computerized X-RAY angiography: first in vivo result. In *Annual International Conference of the IEEE Engineering in Medecine and Biology Society*, pages 2051–2052, 1992.

[6] L. Garnero and F. Peyrin. Methodes de Reconstruction 3D en Tomographie X. Technical report, GDR TDSI CNRS, France, May 1993. Rapport de synthèse 93-01.

[7] G. T. Herman. *Image Reconstruction from Projections*. Academic Press, New York, 1980.

[8] M. I. Miller and C. S. Butler. 3-D maximun *a posteriori* Estimation for Single Photon Emission Computed Tomography on Massively-Parallel Computers. *IEEE Transactions on Medical Imaging*, 12(3):560–565, 1993.

[9] E.L. Ritman, J.H. Kinsey, R.A. Robb, L.D. Harris, and B.K. Gilbert. Physics and technical considerations in the design of the DSR. *J. of Roenhgenology*, 134:369–374, 1980.

[10] E. L. Zapata, I. Benavides, F. F. Rivera, J. D. Bruguera, and J. M. Carazo. Image Reconstrcution on Hypercube Computers. In *The 3rd Symposium on the Frontiers of Massively Parallel Computation*, pages 127–133, 1990.