MEDITOMO: an high performance software for SPECT imaging

L. Carracciuolo - L. Antonelli -
L. D'Amore - A. Murli

RT-ICAR-NA-2007-1 **01-2006**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

MEDITOMO: an high performance software for SPECT imaging

L. Carracciuolo[1] - L. Antonelli[1] -
L. D'Amore[2] - A. Murli[2]

Rapporto Tecnico N.:
RT-ICAR-NA-2007-01

Data:
01-2006

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, Via P. Castellino 111, 80131 Napoli
[2] Università degli Studi di Napoli Federico II, Napoli

MEDITOMO:[*] an high performance software package for 3D SPECT imaging

L. Antonelli[2] - L. Carracciuolo[2] - L. D'Amore[1] - A. Murli[1]

1. University of Naples *Federico II*, Naples, ITALY

2. Institute of High Performance Computing and Networking, ICAR-CNR, Naples, ITALY

Abstract

We describe the parallel software library, named MEDITOMO, designed for analysis of MEDIcal images obtained by SPECT (*Single Photon Emission Computed Tomography*) TOMOgraphic systems. The library has been developed within a national research project carried out in collaboration with the Clinic Physiopathology Department of University of Florence, the Physics Department of University of Genoa and the Institute of High Performance Computing and Networking of CNR in Naples. MEDITOMO is the core library of the PSE (Problem Solving Environment) MEDIGRID, oriented to medical imaging applications, that will be employed in a grid infrastructure involving such departments.

1 The SPECT imaging model equation

The role of accurate and efficient investigation and diagnosis in the management of all disease is unquestionable. In last decades, central to the diagnostic process are imaging techniques, medical imaging not only provides for diagnosis but also serves to assist the treatment of malignant disease. Due to the huge amount of data, the exploitation of high performance environments both for the storage requirements and for their efficient elaboration, is mandatory. In particular, here we are concerned with the SPECT imaging, mathematical description of a SPECT analysis is based on the *Blurred Radon Transform* [12]:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(\mathbf{x},z')h(s-\mathbf{x}\cdot\theta,\mathbf{x}\cdot\theta^{\perp},z-z')d\mathbf{x}dz' = g(s,\phi,z) \qquad (1)$$

[*]This work is partially supported by the FIRB Grid.it RBNE01KNFP national project (2003-2005) within the activities of the WP9 *Grid enables scientific libraries* and by national PRIN (2000-2002; 2002-2004 e 2004-2006) projects.

where $\mathbf{x} = \{x, y\}$, $\theta = \{\cos\phi, sin\phi\}$ and $\theta^{\perp} = \{-\sin\phi, \cos\phi\}$. If $t = \mathbf{x} \cdot \theta^{\perp}$ is the distance between the source and the detector, coordinates $\{s, t, z\}$ refer to the plane $\{s, t\}$ which is rotated, with respect to the plane $\{x, y\}$, by the angle θ. $h(s, t, z)$ is the *Point Spread Function* (PSF) describing the *blurring* process, while $g(s, \phi, z)$ represents the data acquired by the gamma camera at the angle ϕ and $u(x, y, z)$ refers to the radioisotope distribution.
Following [12] we assume:

$$h(s, t, z) = h_0(s, t)h_0(z, t) \quad ,$$

where:

$$h_0(a, b) = \frac{1}{\sigma(b)\sqrt{2\pi}} e^{-\frac{a^2}{2\sigma^2(b)}} \quad ,$$

and

$$\sigma(b) = \sigma_{min} - \frac{\sigma_{max} - \sigma_{min}}{2r'}(b - r') \quad , \tag{2}$$

σ_{min}, σ_{max} and r' depends on the acquisition system ([1]). Equation (1) is said to be the *fully 3D* model. If the PSF is separable:

$$h(s, t, z) = E(s, t)A(z) \quad , \tag{3}$$

then:

$$\int_{-\infty}^{\infty} A(z - z')dz' \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(\mathrm{x}, z')h(s - \mathrm{x} \cdot \theta, \mathrm{x} \cdot \theta^{\perp})d\mathrm{x} = g(s, \phi, z) \quad . \tag{4}$$

Equation (4) is said to be the *2D+1* model [12].
Both the (1) and (4) can be written as:

$$g = Ku, \tag{5}$$

where K is the integral operator.

The mathematics of computed tomography involves the reconstruction of the unknown function u (usually a 3D surface) from its planar projections (g) acquired at different values of ϕ . More precisely, we deal with the inverse problem:

$$u = K^{-1}g \quad , \tag{6}$$

which is a well known inverse and ill posed problem [10]. Reconstruction methods available in literature can be largely classified into two groups, namely those based on *Filtered Back Projection* (FBP) and iterative methods. For a long time there was much debate as to the relative superiority of one or another and in particular whether one of the two was in some sense superior. Today, this debate has largely subsided with the inevitable conclusion that each method has its advantages, it being important to tailor the reconstruction algorithms to the

[1]Some details about values of σ_{min}, σ_{max} and r can be found in Section **??**.

scanner design and to the physics of the imaging modality.
Both FBP and iterative methods can be related to regularization approaches. Basic idea of regularization is to find the solution u of (6) as:

$$u^* = arg\ \underset{u}{min} F(u) = arg\ \underset{u}{min} \{\| Ku - g \|_2\} \quad , \tag{7}$$

where $\| \cdot \|_2$ is the norm defined on $L^2(\Omega)$, $\Omega \subset \Re^3$ is an open subset of $\Re^3$ and is the domain of the u and g functions.
If the noise affecting data is assumed to have a poisson distribution, solution is find as:

$$u^* = \arg\max_u p\,(g|u) \quad , \tag{8}$$

where $p\,(g|u)$ is the conditional probability[2] of g given u [26].

Both in (7) and in (8) we introduce as regularization functional the Total Variation seminorm, defined as:

$$TV\,(u) = \int |\nabla u|\ dxdydz = \int\int\int \sqrt{u_x^2 + u_y^2 + u_z^2} dxdydz \quad , \tag{9}$$

where u_x, u_y e u_z are partial derivatives of $u\,(x, y, z)$ with respect to x, y and z. Then:

$$u^* = arg\ \underset{u}{min} F(u) = arg\ \underset{u}{min} \left\{\| Ku - g \|_2^2 + \alpha TV(u)\right\} \tag{10}$$

and:

$$u^* = \arg\max_u \{\log p\,(g|u) + TV(u)\} \quad . \tag{11}$$

In this paper we describe the design and development of the software library, named MEDITOMO, collecting parallel algorithms implementing the TV-based regularization for solving the SPECT imaging problem in high performance computing environment. The paper is organized as follows, we start from the description of the discretization of the problem in Section 2 and in Section 3 the basic algorithms underlying the parallel software are described: the Fixed Point, the Conjugate Gradient and the Expectation Maximization; in Section 4 the analysis of the computational complexity of the algorithms is carried out and in Section 5 how the parallelism has been introduced is discussed; in Section 6 main features of MEDITOMO are shown and, finally, in Section **??** some numerical results are shown.

2 Discretization

2.1 Discretization of K

Let:

$$s \in [-r, r] \quad , \quad z \in [-R, R] \quad , \quad \phi \in [0, 2\pi] \quad , \tag{12}$$

[2]The maximum of $p\,(g|u)$ can equivalently be found as the maximum of $\log p\,(g|u)$

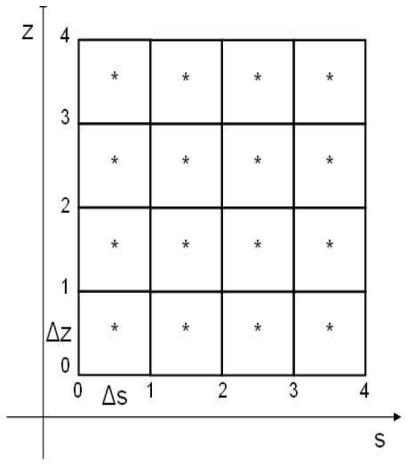


Figure 1: *A 4 × 4 grid for the discretization of $g(s, \phi_m, z)$. (*) means the center of a cell.*

and ([3]):

$$\begin{aligned} \phi_m &= m\Delta\phi, \quad \Delta\phi = \tfrac{2\pi}{M-1}, \quad m = 0, \dots, M \\ s_k &= -r + k\Delta_s, \quad \Delta s = \tfrac{2r}{J-1}, \quad k = 0, \dots, J-1 \\ z_l &= -R + l\Delta z, \quad \Delta z = \tfrac{2R}{L-1}, \quad l = 0, \dots, L-1 \end{aligned}$$

be the discretization of domain Ω.

Let g^{kml} indicate the value of $g(s, \phi, z)$ at (s_k, z_l, ϕ_m). Note that, at each m, and thus at each ϕ_m, we have a 2D mesh made of $J \times L$ boxes, as shown in figure 1. The centers C_{kml} of each square box have coordinates $(s_k + \frac{\Delta s}{2}, \phi_m, z_l + \frac{\Delta z}{2})$. Function u is discretized, by the same way, on a 3D mesh of size $N \times N \times L'$. We discretize the integrals in (1) using a midpoint quadrature formula which use the values of g at C_{kml}. Then, it is:

$$g^{kml} = \sum_{l'=0}^{L-1} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} k_{mkl}^{ijl'} \cdot u^{ijl'} \quad , \tag{13}$$

$$k = 0, \dots, J-1; \ m = 0, \dots, M-1; \ l = 0, \dots, L-1.$$

Equations (13) lead to the following linear system

$$[g] = [K]_3[u] \quad , \quad [g] \in \Re^{J \times M \times L} \quad , \quad [u] \in \Re^{N \times N \times L'} \tag{14}$$

where $[u]$ is unknown.

[3]The values of J, M, L depend on the acquisition system. Typical values of M are 120, 60, and of J and L are $2^l, l = 7, 8$.

Definition 2.1 *Matrix* $[K]_3 \in \Re^{((J\times M)\times L\times(N\times N)\times L')}$ *is called the* 3D projector. *Elements of* $[K]_3$ *are denoted by* k_{mkl}^{ijl}. $[K]_3$ *is a block matrix.*

$[K]_3$, is sparse with about 95% of zeros, then a compact storage technique was used.
Discretization of (4), leads to the following linear system:

$$g^{kml} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} E_{km}^{ij} \sum_{l'=0}^{L'-1} A_{l'}^{l} \cdot u^{ijl'} \quad , \tag{15}$$

where

$$k = 0,\dots,J-1;\ m = 0,\dots,M;\ l = 0,\dots,L-1.$$

whose matrix formulation is:

$$[g] = [K]_3[u] \quad , \quad [K]_3 = [E]_2 \otimes [A]_1 \quad , \quad \text{where} \quad [E]_2 \in \Re^{((J\times M)\times(N\times N))} \tag{16}$$

$$\text{and} \quad [A]_1 \in \Re^{L\times L'}, \tag{17}$$

where:

$$[E]_2 = \begin{pmatrix} E_{0,0}^{0,0} & \dots & E_{0,0}^{0,N-1} & \dots & E_{0,M-1}^{0,0} & \dots & E_{0,M-1}^{0,N-1} \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ E_{0,0}^{N-1,0} & \dots & E_{0,0}^{N-1,N-1} & \dots & E_{0,M-1}^{N-1,0} & \dots & E_{0,M-1}^{N-1,N-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ E_{J-1,0}^{0,0} & \dots & E_{J-1,0}^{0,N-1} & \dots & E_{J-1,M-1}^{0,0} & \dots & E_{J-1,M-1}^{0,N-1} \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ E_{J-1,0}^{N-1,0} & \dots & E_{J-1,0}^{N-1,N-1} & \dots & E_{J-1,M-1}^{N-1,0} & \dots & E_{J-1,M-1}^{N-1,N-1} \end{pmatrix} , \tag{18}$$

$[A]_1$ is a Toeplitz matrix:

$$[A]_1 = \begin{pmatrix} \alpha_0 & \dots & \alpha_{L'-1} \\ \vdots & \ddots & \vdots \\ \alpha_{L-1} & \dots & \alpha_0 \end{pmatrix} \quad , \tag{19}$$

where:

$$\alpha_l = (\Delta z)^2 \int_{\frac{-1}{2}}^{\frac{1}{2}} \int_{\frac{-1}{2}}^{\frac{1}{2}} A[\Delta z(\zeta - \zeta' + l)] d\zeta d\zeta' \quad , \tag{20}$$

Definition 2.2 $[E]_2 \in \Re^{(J\times M)\times(N\times N)}$ *is called* 2D projector. $[E]_2$ *is a block matrix.*

Using the commutative property of the Kronecker product in (16) it follows:

$$[g] = [K]_3[u] = [E]_2 \otimes [A]_1[u] = [A]_1 \otimes [E]_2[u]$$

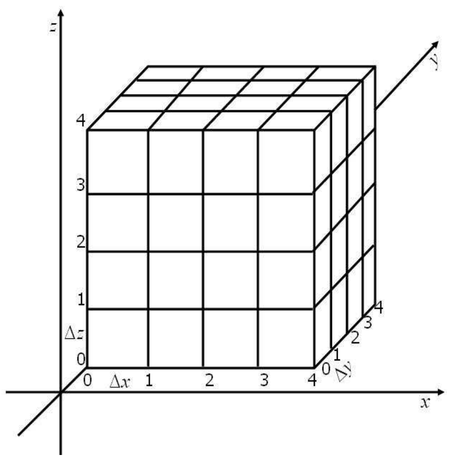


Figure 2: *A* 4×4 *grid used for the discretization of* $u(x, y, z)$

or:

$$\begin{aligned} [f] &= [E]_2[u] \quad , \\ [g] &= [A]_1 \otimes [f] \quad . \end{aligned} \tag{21}$$
$$[u] \in \Re^{(N\times N)\times L'}, \quad [f] \in \Re^{(J\times M)\times L'}, \quad [g] \in \Re^{(J\times M)\times L}$$

2.2 Discretization of TV

The Euler Lagrange equations associated to the problem (10) are:

$$\begin{cases} K^*(Ku - z) + \alpha \cdot \mathfrak{L}u = 0 & , \quad (x, y) \in \Omega \\ \frac{\partial u}{\partial n} = 0 & , \quad (x, y) \in \partial\Omega \end{cases} \tag{22}$$

where $\mathfrak{L}u = div\left(\frac{\nabla u}{|\nabla u|}\right)$
$\mathfrak{L}$ is discretized, on the same mesh of u and g (see fig. 2), using a 7-point stencil (figura 3) [23], with step $h = \Delta x = \Delta y = \Delta z$. We approximate first and second partial derivatives of $u\,(x, y, z)$ using the central finite differences:

$$\begin{aligned} \frac{\partial u}{\partial x} &\approx \frac{u_{i+1,j,l'} - u_{i-1,j,l'}}{2h} \quad , \\ \frac{\partial u}{\partial y} &\approx \frac{u_{i,j+1,l'} - u_{i,j-1,l'}}{2h} \quad , \\ \frac{\partial u}{\partial z} &\approx \frac{u_{i,j,l'+1} - u_{i,j,l'-1}}{2h} \quad , \end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 u}{\partial x^2} &\approx \frac{u_{i+1,j,l'} + u_{i-1,j,l'} - 2u_{i,j,l'}}{h^2} \quad ,\\
\frac{\partial^2 u}{\partial y^2} &\approx \frac{u_{i,j+1,l'} + u_{i,j-1,l'} - 2u_{i,j,l'}}{h^2} \quad ,\\
\frac{\partial^2 u}{\partial z^2} &\approx \frac{u_{i,j,l'+1} + u_{i,j,l'-1} - 2u_{i,j,l'}}{h^2} \quad .
\end{aligned}$$

Discretization of $\mathfrak{L}$, leads to :

$$L([u]) = B_h^T (D_h([u]))^{-1} B_h \quad ,$$

where D_h is a block diagonal matrix:

$$D_h = \begin{bmatrix} D_x([u]) & 0 & 0 \\ 0 & D_y([u]) & 0 \\ 0 & 0 & D_z([u]) \end{bmatrix} \quad ,$$

$B_h[u]$ is the matrix where each row contains just two not zero elements equal to 1. $D_x([u])$, $D_y([u])$, $D_z([u])$ are diagonal matrices of size $(N-1) \times N \times L'$, $N \times (N-1) \times L'$, $N \times N \times (L'-1)$ respectively.

Finally, discretization of (22) leads to the following non linear system:

$$([K^*]_3[K]_3 - \alpha L([u]))[u] = [K^*]_3[g] \quad , \tag{23}$$

Equation (23) represents the discrete problem corresponding to the TV-based regularization problem describing the 3D SPECT imaging in case of additive gaussian noise. In the following we describe discretization of the same problem in case of a poisson noise.

2.3 Discretization of (8)

We perform the discretization of Ω using the grid shown in fig. (2). Let u_s denote the values of u at the center of the s-th voxel, where $s = 1, \ldots, S$. Consider the function $\mathfrak{J}$ defined as:

$$\mathfrak{J} : (i, j, l') \mapsto s = (l'-1) * N^2 + (j-1) * N + i.$$

which maps (i, j, l') onto s, $s = 1, \ldots, S$. By the same way, we consider the function $g(s, \phi, z)$ and suppose that the gamma camera acquires T values of g. Let denote by $g_t, t = 1, \ldots, T$ such values and:

$$u = \begin{pmatrix} u_1 \\ \vdots \\ u_S \end{pmatrix} \quad , \tag{24}$$

$$g = \begin{pmatrix} g_1 \\ \vdots \\ g_T \end{pmatrix} \quad . \tag{25}$$

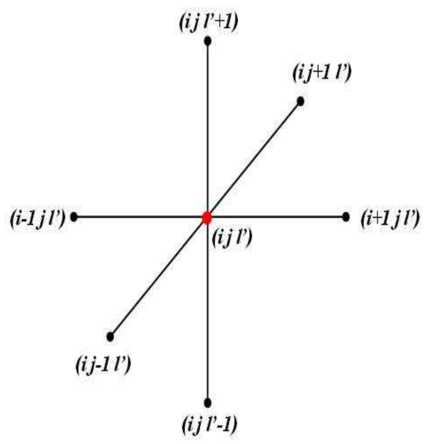


Figure 3: *The 7-point stencil used for discretization of* $\mathfrak{L}$

The difference between the discretization technique used in section 2.1 and this one is that in the first case function u is discretized at the points (i, j, l'), while now the values of u are computed at the centers of the voxels.

Let k_{ts} be the probability that the radio isotope radiation emitted from the generic voxel $s = 1, \ldots, S$ will be detected from the t-th $(t = \ \ldots, T)$ detector, then:

$$g_t^* = \sum_{s=1}^{S} k_{ts} u_s = E(g|u) \tag{26}$$

is the average of the conditional statistic of g_t. The probability $p\,(g_t|u)$ of g_t given u is:

$$p\,(g_t|u) = \text{Poisson}\left(\sum_{s=1}^{S} k_{ts} u_s\right) = e^{-\left(\sum_{s=1}^{S} k_{ts} u_s\right)^{g_t}} \frac{\left(\sum_{s=1}^{S} k_{ts} u_s\right)^{g_t}}{g_t!} \quad . \tag{27}$$

Replacing (26) in the (27) gives:

$$p\,(g_t|u) = e^{-(g_t^*)^{g_t}} \frac{(g_t^*)^{g_t}}{g_t!} \quad . \tag{28}$$

then:

$$u^* = \arg\max_u \left\{ \sum_{t=1}^{T} \left(g_t \log \frac{g_t^*}{g_t!} - g_t^* \right) + \log p\,(u) \right\} \quad , \tag{29}$$

u^* is solution of (**??**) if [35]:

$$u_s \frac{\partial l(u)}{\partial u_s}\bigg|_{u_s=u_s^*} = 0, \quad u_s > 0 \tag{30}$$

$$\frac{\partial l(u)}{\partial u_s}\bigg|_{u_s=u_s^*} \le 0 \quad u_s = 0 \tag{31}$$

where

$$l(u) = \sum_{t=1}^{T} \left(g_t \log \frac{g_t^*}{g_t!} - g_t^* \right) \quad .$$

then:

$$0 = u_{s'} \frac{\partial l(u)}{\partial u_{s'}} = u_{s'} \left(-\sum_{t=1}^{T} k_{ts'} + \sum_{t=1}^{T} \frac{k_{ts'} g_t}{\sum_{s=1}^{S} k_{ts} u_s} \right), \tag{32}$$

or:

$$u_{s'} \sum_{t=1}^{T} k_{ts'} = u_{s'} \quad . \sum_{t=1}^{T} \frac{k_{ts'} g_t}{\sum_{s=1}^{S} k_{ts} u_s}. \tag{33}$$

We now introduce the matrix:

$$[K]_3 := (k_{ts})_{\substack{t=1,\ldots,T \\ s=1,\ldots,S}}$$

then (34) gives:

$$u\left([K]_3^T [1]\right) = u\left([K]_3^T \frac{g}{([K]_3 u)}\right) \tag{34}$$

$$0 = -u\left([K]_3^T [1]\right) + u\left([K]^T \frac{g}{([K]_3 u)}\right) \tag{35}$$

$$0 = -u\left\{\left([K]_3^T [1]\right) + \left([K]_3^T \frac{g}{([K]_3 u)}\right)\right\} \tag{36}$$

where $[1] \in \Re^t$ is a vector with all elements equal to 1, and where operations between vectors are intended component-wise.

3 The algorithms: Fixed Point and EM

We use two standard algorithms, the Fixed Point (FP) and the Expectation Minimization (EM), suitably modified for solving (23) and (36).

If $\omega = \frac{\nabla[u]}{|\nabla[u]|}$, 23) can be written as:

$$\begin{cases} \mathcal{F}(u,\omega) = [K^*]_3([K]_3[u] - [g]) + \alpha \nabla \cdot \omega = 0 \\ \omega = \frac{\nabla[u]_0}{\sqrt{(\nabla[u]_0)^2 + \beta^2}} \end{cases} \tag{37}$$

```
procedure PCG(K,K*,M,f,g);
* Computation of the residual at the zero step.
$r^{(0)} = g$ ;
* Preconditioning at the zero step.
$z^{(0)} = M^{-1}\, r^{(0)}$;
* Search direction at the zero step.
$p^{(0)} = z^{(0)}$;
$\gamma_0 = 0$;
for k=1, maxit do
    * I. Computation of $\gamma_k$
    $\gamma_k = \|r^{(k-1)}\|_2^2$;
    * II. Computation of $\beta_k$
    $\beta_k = \gamma_{k-1} \,/\, \gamma_k$;
    * III. Update of the vector $p^{(k)}$
    $p^{(k)} = r^{(k)} + \beta_k p^{(k-1)}$
    * IV. Update of the vector $q^{(k)}$
    $q^{(k)} = K^* M^{-1} K\, p^{(k)}$;
    $\alpha_k = \gamma_k / \|q^{(k)}\|_2^2$;
    * V. Update of the solution $f^{(k)}$
    $f^{(k)} = f^{(k-1)} + \alpha_k p^{(k)}$;
    * VI. Update of the residual $r^{(k)}$
    $r^{(k)} = r^{(k-1)} - \alpha_k q^{(k)}$;
endfor
end procedure PCG.
```

Figure 4: *PCG algorithm.*

Elimination of ω from the first equation leads to:

$$[K^*]_3([K]_3[u] + [g]) + \alpha\nabla \cdot \left(\frac{\nabla[u]}{\sqrt{(\nabla[u])^2 + \beta^2}} \right) = 0 \quad . \tag{38}$$

If $[u]_0 = [g]$ as in [13], we have:

$$[K^*]_3[K]_3[u]^{m+1} - [u]_0 + \alpha\nabla \cdot \left(\frac{\nabla[u]^{m+1}}{\sqrt{(\nabla[u]^m)^2 + \beta^2}} \right) = 0 \quad . \tag{39}$$

The iterative scheme (39) has a linear convergence [22].
The FP, shown in fig. 5, is made of two nested loops where the main operations are:

1. *I loop:* construction of the matrices in (39)
   - 1.a computation of $[K]_3$;
   - 1.b computation of $[K^*]_3$;
   - 1.c computation of $L^m([u]^m)$;
2. *II loop*: solution of a linear system, by using the PCG :

   The diagonal preconditioner is described in [29]. PCG is shown in fig. 4.

From (36) it follows the iterative scheme for computing the approximation u^* of (11) is:

$$u^{m+1} = \frac{[1]}{[K]_3^T\,[1]} u^m [K]_3^T \frac{g}{[K]_3 u^m} \qquad m = 1, 2, \ldots . \tag{40}$$

which is known as the EM algorithm.

Theorem 3.1 *The iterative scheme (40) converges to the solution u^* of the problem (**??**) [35].*

The explicit form of the EM algorithm is:

$$u_{s'}^{m+1} = u_{s'}^m \frac{1}{\sum_{t=1}^T k_{ts'}} \sum_{t=1}^T \frac{g_t k_{ts'}}{\sum_{s=1}^S k_{ts} u_s^m} \tag{41}$$

The EM (40) has a slow convergence rate [30], then the accelerated version, called EMOSn (*Expectation Maximization Ordered Subset*)[30] has been used. Let $\Xi = \{\Omega_i : \ i = 1, \ldots, n\}$ be a partition of the set $\Omega = \{1, \ldots, T\}$, that is:

$$\begin{aligned} \Xi &= \{\Omega_i\}_{i=1,\ldots,n}, \\ \Omega_i &\subseteq \Omega, \\ \bigcup_{i=1}^n \Omega_i &= \Omega, \\ \Omega_i \bigcap \Omega_j &= \emptyset, \quad \forall i \neq j \end{aligned} \tag{42}$$

```
begin procedure FP
{
  Construction of the matrix $F$;
   $F = [K^*]_3[g]$;

  * Fixed Point Iteration

  for $i = 1, \cdots, maxit$ do
     Computation of the matrix $L^m([u]^m)$;
     Solution of the system $([K^*]_3[K]_3 + \alpha L^m([u]^m))[u] = F$
     by the $PCG$ method (nested cycle);
  endfor
}end procedure FP
```

Figure 5: *FP algorithm.*

The partition Ξ leads to a row-block formulation of $[K]_3$:

$$[K]_3 = \begin{bmatrix} [K_1]_3 \\ \vdots \\ [K_n]_3 \end{bmatrix}, \quad [K_i]_3 \in \Re^{\frac{T}{n} \times S} \tag{43}$$

$[K_i]_3$ is a restriction of the operator $[K]_3$ to Ω_i. In the *2D+1* model, each block $[K_i]_3$ can be factorized as (16):

$$[K_i]_3[u] = ([A]_1 \otimes [E]_2)_i[u] \quad i = 1, \ldots, n \quad . \tag{44}$$

In fig. 6 we show the EMOSn algorithm.

3.1 Introduction of Total Variation inside EM.

Introducing TV in (36) leads to the following scheme [34]:

$$u_s^{m+1} = u_s^m \frac{1}{\sum_{t=1}^{T} k_{ts} + \alpha \left(L\left(u^m\right) u^m\right)_s} \sum_{t=1}^{T} \frac{g_t k_{ts}}{\sum_{s=1}^{S} k_{ts} u_s^m} \quad m = 1, 2, \ldots \tag{45}$$

TV-based EMOSn, is shown in figure 7. EMOSn-TV is made of two loops which perform the following operations:

1. *I loop (EM):* definition of matrices in (**??**):

```
**procedure** EMOS$n$
  1. $m = 0$
  2. Let $\hat{u}^m$ be a constant and positive value
  3. repeat until to the $\hat{u}^m$ convergences
     (a) $u^1 = \hat{u}^m$
     (b) $m = m + 1$
     (c) for each OS level $i = 1, \ldots, n$
         i. compute the product $\mu^i = [K_i]_3 u^i$
              $$\mu_t^i = \sum_{s=1,\ldots,S} k_{ts} u_s^i,\ t \in T_i$$
         ii. compute $u^{i+1} = \frac{[1]}{[K]_3^T [1]} u^i [K_i]_3^T \frac{g}{\mu^i}$
              $$u_s^{i+1} = u_s^i \frac{1}{\sum_{t=1}^{T} k_{ts}} \sum_{t \in T_i} \frac{g_t k_{ts}}{\mu_t^i},\ s = 1, \ldots, S$$
     (d) $\hat{u}^m = u^{n+1}$
  4. $u = \hat{u}^m$
**end procedure** EMOS$n$
```

Figure 6: *The EMOS*n algorithm

```
**procedure** EMOS$n$-TV(input: $K, g$ ; output: $u$)
    1. $m = 0$
    2. Let $\hat{u}^m$ be a constant and positive value
    3. repeat until to the $\hat{u}^m$ convergence
        (a) $u^1 = \hat{u}^m$
        (b) $m = m + 1$
        (c) computation of $L^m\left(u^1\right)$
        (d) For each OS level $i = 1, \ldots, n$
            i. compute the product $\mu^i = [K_i]_3 u^i$
                $$\mu_t^i = \sum_{s=1,\ldots,S} k_{ts} u_s^i,\ t \in T_i$$
            ii. computation of $u^{i+1} = \frac{[1]}{[K]_3^T[1] + \alpha L^m(u^1)u^i} u^i [K_i]_3^T \frac{g}{\mu^i}$
                $$u_s^{i+1} = u_s^i \frac{1}{\sum_{t=1}^T k_{ts} + \left(L^m\left(u^1\right)u^i\right)_s} \sum_{t \in T} \frac{g_t k_{ts}}{\mu_t^i},\ s = 1, \ldots, S$$
        (e) $\hat{u}^m = u^{n+1}$
    4. $u = \hat{u}^m$
**end procedure** EMOS$n$-TV
```

Figure 7: *TV-based EMOS*n *algorithm.*

1.a construction of $[K]_3$;

1.b construction of $[K^*]_3$;

1.c construction of $L^m([u]^m)$;

2. *II loop (OS)*:

2.a computation of $\mu^i = [K_i]_3 u^i$;

2.b computation of $L^m([u]^m)u^i$.

4 Performance analysis

Main difficulty of the development of efficient algorithms is the capability to analyze and predict their performance. MEDITOMO library has been designed with the aim of providing the most efficient algorithms. To this aim, first step is the performance analysis of the basic algorithms. We now estimate the flops of CG and EMOS$_n$ ([4]) considering the number of floating-point operations N_{cgXX} of one CG's step, and the number of floating-point operations N_{em2d+1} ([5]) of one EM's step. We will use the following notations:

`nsez`	L	number of sections
`kdim`	J	size of each projection
`nang`	M	number of angular views
`nsez`	L'	number of projections
`ndim`	N	size of each section
`nos`	n	number of *subset*
`kos`	$\frac{M}{n}$	number of angular views contained in each subset
`nmat`	-	size of working area

`nmat` is the number of points inside the square inscribed circle with diameter equals to `ndim`, then

$$\texttt{nmat} \approx \pi \left(\frac{\texttt{ndim}}{2} \right)^2 \quad .$$

One step of CG requires:

1. 2 scalar products between vectors;
2. 3 *axpy* operations([6]);
3. two matrix-vector operations([7]).

In case of the fully 3D model, steps 1. and 2., are performed using vectors of length `nmat` $\times$ `nsez`, than they require:

$$N_1 = 12 \times \texttt{nmat} \times \texttt{nsez} \quad \textit{flops}. \tag{46}$$

Step 3. requires [8]:

$$N_2 = \texttt{nang} \times \texttt{nsez} \times (24 + 12 \times \texttt{ndim} + 505 \times \texttt{nmat} + 45 \times \texttt{kdim}) \quad . \tag{49}$$

[4]Looking at the algorithms in fig. 5, it follows that the flops of one FP's step should be obtained summing the operations of one CG's iteration, the construction of $L^m([u]^m)$ at each step m, the product $L^m([u]^m)[u]^{m+1}$ at each step m. Numerical experiments show that the computational complexity of last two operations (matrix product and construction) is negligible. By the same way, the computational cost of one TV-based EMOSn's step (see figure 7) is equivalent to one EMOS $_n$'s step.

[5]Estimate is given only for the *2D+1* model.

[6]We denote by *axpy*, the operation

$$y \leftarrow y + \alpha x \quad .$$

[7]The cost of such operations is of $\mathcal{O}((J \times M) \times L \times (N \times N) \times L')$, but the sparsity reduces it to of about 95%.

[8]If we don't consider the sparsity of $[K]_3$, then the (49) should be:

$$\tilde{N_2} = 2 \times \texttt{nsez}^2 \times \texttt{nang} \times \texttt{kdim} \times \texttt{ndim}^2 \quad . \tag{47}$$

and:

$$\begin{aligned} N_{cg3d} &= 12 \times \mathtt{nmat} \times \mathtt{nsez} + \\ &+ \mathtt{nang} \times \mathtt{nsez} \times (24 + 12 \times \mathtt{ndim} + 505 \times \mathtt{nmat} + 45 \times \mathtt{kdim}) \quad . \end{aligned} \tag{50}$$

In case of the 2D+1 model, steps at 1 and 2, are performed using vectors of length $\mathtt{ndim}^2 \times \mathtt{nsez}$, and they require:

$$N_1 = 12 \times \mathtt{ndim}^2 \times \mathtt{nsez} + 1 \qquad flops. \tag{51}$$

Step 3. requires [8][9]:

$$N_2 = \mathtt{nsez} \times (\mathtt{nang} \times (106 \times \mathtt{ndim}^2 + 61 \times \mathtt{kdim})) \quad . \tag{53}$$

and:

$$\begin{aligned} N_{cg2d+1} &= 12 \times \mathtt{ndim}^2 \times \mathtt{nsez} + 1 + \\ &+ \mathtt{nsez} \times (\mathtt{nang} \times (106 \times \mathtt{ndim}^2 + 61 \times \mathtt{kdim})) \quad . \end{aligned} \tag{54}$$

4.1 Floating Point operations in EMOS$_n$.

At each iteration of EMOS$_n$, the following operations are performed:

1. 2 scalar products between vectors;
2. 2 matrix-vector operations ([9]).

Using the scheme shown in figure 6, step 1. is performed `nos` times, with vectors of length $\mathtt{ndim}^2 \times \mathtt{nsez}$, then it requires:

$$N_1 = 2 \times \mathtt{ndim}^2 \times \mathtt{nsez} \qquad flops. \tag{55}$$

If we assume that $\mathtt{nsez} = 25$, $\mathtt{nang} = 120$, $\mathtt{kdim} = \mathtt{ndim} = 128$ we have:

$$N_2 \sim 7 \times 10^9 \quad , \quad \widetilde{N_2} = 335 \times 10^9 \Longleftrightarrow \frac{\widetilde{N_2}}{N_2} = 2\% \tag{48}$$

(48) confirms that the cost of both matrix-vector operations, at each iteration, reduces of about 98%[7].

[9]If we don't consider the sparsity of the matrix $[K]_3$, then the (53) should have the value of (47). If we suppose that $\mathtt{nsez} = 25$, $\mathtt{nang} = 120$, $\mathtt{kdim} = \mathtt{ndim} = 128$, we have:

$$N_2 \sim 4 \times 10^9 \quad , \quad \widetilde{N_2} = 335 \times 10^9 \Longleftrightarrow \frac{N_2}{\widetilde{N_2}} = 1,5\% \qquad flops. \tag{52}$$

52) confirms that cost of both matrix-vector operations, at one CG step, reduces of about 98.5%.

The matrix-vector product at 2. requires [10] the following computational cost [8]:

$$N_2 = 2 \times \texttt{nsez} \times (\texttt{nang} \times (48 \times \texttt{ndim}^2 + 31 \times \texttt{kdim})) \quad . \tag{57}$$

and:

$$\begin{aligned} N_{em2d+1} &= 2 \times \texttt{nos} \times \texttt{nsez} \times \texttt{ndim}^2 + \\ &+ 2 \times \texttt{nsez} \times (\texttt{nang} \times (48 \times \texttt{ndim}^2 + 31 \times \texttt{kdim})) \quad . \end{aligned} \tag{58}$$

If we compare (54) and (58), and suppose $\texttt{nos} \approx 6$, then $N_{cg2d+1} \approx N_{em2d+1}$.

5 Parallelism into 3D SPECT algorithms

Starting the the performance analysis, parallelism has been introduced to reduce the exexution time of the most expensive operation:

$$z = [K^*]_3[K]_3 u \quad . \tag{59}$$

where (59) is the *projection/backprojection* operation.

5.1 Parallelization of the *fully 3D* model

The most suitable communication topology is a *1D periodic grid* of size $1 \times N_{proc}$, where N_{proc} is the number of processors. We use a *cyclic column block* distribution for $[K]_3$ and a *cyclic row block* distribution for $[K^*]_3$([11]) Each processor loads $r = J \times M/Nproc$ rows and $c = N \times N$ columns of $[E_i]_2$ and loads $r = N \times N$ rows and $c = J \times M/Nproc$ columns of $[E_i]_2^*$. The operations in (59) are performed as follows:

1. *projection* : $\mathbf{f}_p^i = \sum_{j=0}^{L'-1} [E_{(j+i-1,\text{mod}L')}]_2 \cdot \mathbf{u}^j$, $i = 0, \dots, L-1$;
2. *backprojection* : $\mathbf{z}_p^i = \sum_{j=0}^{L'-1} [E^*_{(j+i-1,\text{mod}L')})]_2 \cdot \mathbf{f}_p^j$, $i = 0, \dots, L-1$;
3. *collective sum* : $\mathbf{z}^i = \sum_{p=0}^{Nproc-1} \mathbf{z}_p^i$, $i = 0, \dots, L-1$;

In figure 8 we show the parallel algorithm.

[10] If we don't consider the sparsity of the matrix $[K]_3$, than the (57) should have the value of (47). If we suppose that $\texttt{nsez} = 25$, $\texttt{nang} = 120$, $\texttt{kdim} = \texttt{ndim} = 128$, we have:

$$N_2 \sim 2 \times 10^9 \quad , \quad \widetilde{N_2} = 335 \times 10^9 \Longleftrightarrow \frac{N_2}{\widetilde{N_2}} = 0,5\% \quad \textit{flops}. \tag{56}$$

(56) confirms that computational cost of both matrix-vector operations, at one EMOS$_n$ iteration, reduces, of about 99.5%.

[11] For the matrix $[K^*]_3$ can be used a block form as in (**??**).

```
procedure $[K^*]_3[K]_3 u$

Cycle on the number of processors.
  for $p = 0, Nproc - 1$

    Compute the partial product:  $f_p^i \in \Re^{J\cdot M/Nproc}$
    step 1. for $i = 0, \ldots, L$
      $\mathbf{f}_p^i = E_i(p \cdot (J \cdot M/nproc) : (p+1) \cdot (J \cdot M/Nproc) - 1; :) \cdot \mathrm{u}^i;$

    Compute the partial product: $z_p^i \in \Re^{N \cdot N}$
   step 2. for $i = 0, \ldots, L$
      $\mathbf{z}_p^i = E_i^*(:, p \cdot (J \cdot M/nproc) : (p+1) \cdot (J \cdot M/Nproc) - 1) \cdot \mathrm{f}_p^i;$

  endfor

 Collective sum operation.
 $\mathrm{z}^i = \sum_{p=0}^{Nproc-1} \mathbf{z}_p$

end procedure
```

Figure 8: *Parallel algorithm for $z = [K^*]_3[K]_3 u$ in the fully 3D problem.*

5.2 Parallelization of the *2D+1* model

The operation (59) has a block formulation also in the *2D+1* model. Indeed, from the (15) each component g^l of $[g]$, of size $J \times M$, can be computed performing L' products using $[E]_2$ and summing the resulting vectors multiplied by the $\alpha_{l'-l}$, that is:

$$\begin{pmatrix} \mathbf{g}^0 \\ \mathbf{g}^1 \\ \vdots \\ \mathbf{g}^{\mathbf{L-1}} \end{pmatrix} = \begin{pmatrix} \alpha_0[E]_2 & \alpha_1[E]_2 & \ldots & \alpha_{L'-1}[E]_2 \\ \alpha_{-1}[E]_2 & \alpha_0[E]_2 & \ldots & \alpha_{L'-1}[E]_2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{1-L}\,[E]_2 & \alpha_{2-L}\,[E]_2 & \ldots & \alpha_0[E]_2 \end{pmatrix} \begin{pmatrix} \mathbf{u}^0 \\ \mathbf{u}^1 \\ \vdots \\ \mathbf{u}^{\mathbf{L'-1}} \end{pmatrix} \quad . \tag{60}$$

If the partition (42) holds then the (60) is valid for each $[K_i]_3$, that is:

$$\begin{pmatrix} \left(\mathbf{g}^0\right)_i \\ \left(\mathbf{g}^1\right)_i \\ \vdots \\ \left(\mathbf{g}^{L'-1}\right)_i \end{pmatrix} = \begin{pmatrix} \alpha_0\left([E]_2\right)_i & \alpha_1\left([E]_2\right)_i & \ldots & \alpha_{L'-1}\left([E]_2\right)_i \\ \alpha_{-1}\left([E]_2\right)_i & \alpha_0\left([E]_2\right)_i & \ldots & \alpha_{L'-2}\left([E]_2\right)_i \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{1-L'}\left([E]_2\right)_i & \alpha_{2-L'}\left([E]_2\right)_i & \ldots & \alpha_0\left([E]_2\right)_i \end{pmatrix} \begin{pmatrix} \mathbf{u}^0 \\ \mathbf{u}^1 \\ \vdots \\ \mathbf{u}^{L'-1} \end{pmatrix} \quad i = 1, \ldots, n \quad . \tag{61}$$

We introduce two parallelization approaches for computing the product (61).

```
procedure $[K_i]_3 u$
    step 1. for $j = p * L_{loc} + 1, \dots, (p+1) * L_{loc}$
        $p = ([E]_2)_k \, \mathbf{f}_j$
        for $i = 0, 1, 2, \dots, L-1$
            $\left(\mathbf{g}^i\right)_k = \left(\mathbf{g}^i\right)_k + \alpha_{j-i} p$
        endfor
    endfor
    step 2. for $i = 0, 1, 2, \dots, L-1$
    Collective sum operation
        $\mathbf{g}^i = \sum_{p=0}^{N_{proc}} (\mathbf{g}^i)_p$
end procedure
```

Figure 9: *Parallel algorithm for* $\mathbf{g} = [K_i]_3 \, \mathbf{u}$ *in the 2D+1 model (First Strategy).*

5.2.1 First strategy

Each processor p loads $L_{loc} = L/N_{proc}$ blocks of columns of the matrix $[K_i]_3$, then operation (61) requires:

1. *projection*: $\left((\mathbf{g}^j)_i\right)_p = \sum_{k=p*L'_{loc}+1,\dots,(p+1)*L'_{loc}} \alpha_{j-k} \left([E]_2\right)_i \mathbf{u}^k \quad j = 0, \dots, L-1;$

2. *sum*: $\mathbf{g}^j = \sum_{p=0}^{N_{proc}} ((\mathbf{g}^j)_i)_p \quad , \quad j = 0, \dots, L-1.$

In figure 9 we show the algorithm. The parallel algorithm for computing $\mathbf{u} = [K_i^*]_3 \, \mathbf{g}$ is designed analogously.

5.2.2 Second strategy

Processor p loads $L'_{loc} = L'/N_{proc}$ vectors $\mathbf{u}^j$, $L_{loc} = L/N_{proc}$ vectors $\left(\mathbf{g}^i\right)_k$ and L'_{loc} blocks of columns of $[K_i]_3$. For each pair (p, q) of processors we introduce the set:

$$E_p^q \quad = \quad \left\{ i \in I^q : A_i^j \neq 0 \quad , \forall j \in J^p \right\}, \tag{62}$$

where I^p and J^p denote the set of the indices of the vectors $\left(\mathbf{g}^i\right)_k$ and $\mathbf{u}^j$ respectively assigned to processor p. Then E_p^q contains the indices of $\left(\mathbf{g}^i\right)_k$, which is assigned to processor q. Vectors $\mathbf{u}^j$, given to processor p contribute to the the computation of $\left(\mathbf{g}^i\right)_k$.

Computation of (61), can be performed by the following two steps:

1. *projection:*

   1.a *Computation of* $\left(\mathbf{g}^i\right)_k$ *of* I^p*:*

$$\left(\mathbf{g}^i\right)_k = \sum_{j \in J^p} \alpha_{i-j} \left([E]_2\right)_k \mathbf{u}^j, \quad \forall i \in I^p$$

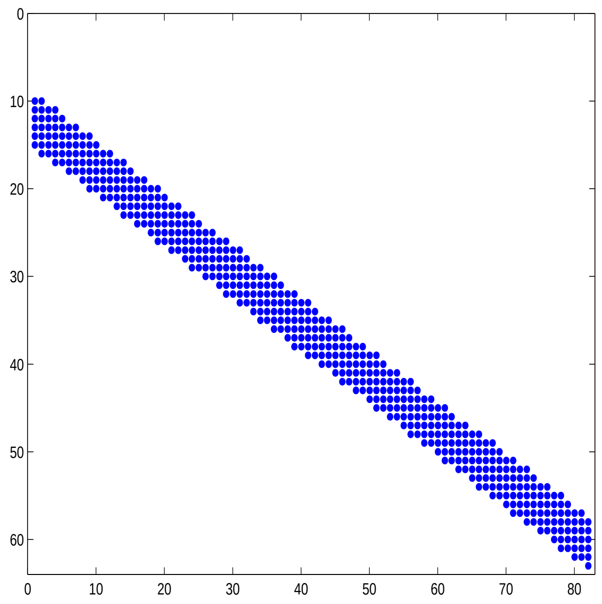


Figure 10: *Representation of the sparsity pattern of the matrix* $[A]_1$*, with* $L = 68$ *and* $L' = 82$ *.*

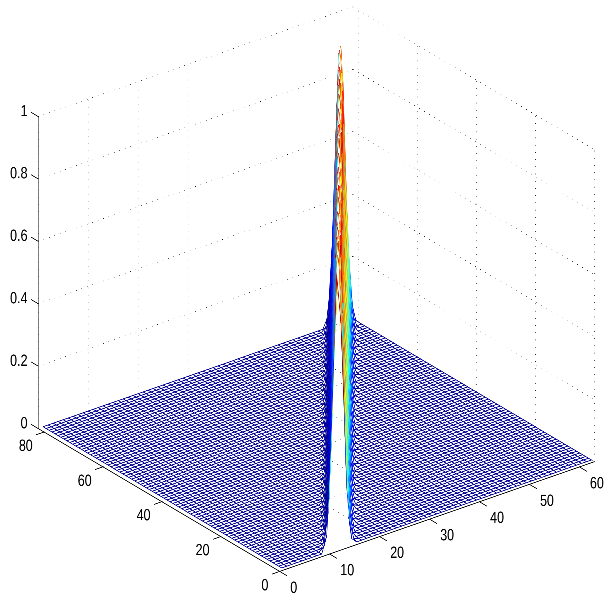


Figure 11: *Elements of* $[A]_1$*, with* $L = 68$ *and* $L' = 82$*.*

1.b *Contribution to* $\left(\mathbf{g}^i\right)_k$ *by* E_p^q*:*

$$\left(\left(\mathbf{g}^i\right)_k\right)_p = \sum_{j\in J^p} \alpha_{i-j}\left([E]_2\right)_k \mathbf{u}^j, \quad \forall q : E_p^q \neq \emptyset$$

2. *sum between the neighborhood processors:*

2.a *send* $\left(\left(\mathbf{g}^i\right)_k\right)_p$ to processor q, $\forall i \in E_p^q$.

2.b *receive* $\left(\left(\mathbf{g}^i\right)_k\right)_q$ from processor q, $\forall q : E_q^p \neq \emptyset$

2.c *local sum*:

$$\left(\mathbf{g}^i\right)_k = \left(\mathbf{g}^i\right)_k + \sum_{q:E_q^p\neq\emptyset} \left(\left(\mathbf{g}^i\right)_k\right)_q, \quad \forall i \in E_q^p$$

In figure 12 we describe the parallel algorithm.
Parallel algorithm for computing $\mathbf{u} = [K_i^*]_3\,\mathbf{g}$ has been designed by the same way.

5.2.3 Performance comparison

We compare the strategies described above in terms of N_{calc}, N_{com} and speed-up. Processors are logically organized in a 2D mesh of size $1 \times N_{proc}$. In step 1 of the first strategy, each processor p contributes to the computation of the L components of $\mathbf{g}$ (or equivalently of $(\mathbf{g}^i)_k$, $i = 1, \ldots L$). This computation requires a block matrix-vector which requires $\mathcal{O}((L \times L_{loc}) \times (J \times M) \times (N \times N))$ flops ([12]), then the computational cost is:

$$N_{calc}^{(1.First)} = \mathcal{O}(N_{proc} \times (L \times L_{loc}) \times (J \times M) \times (N \times N)) \quad . \tag{63}$$

Step 2 requires a *collective sum* which requires ([13]

$$N_{calc}^{(2.First)} = \mathcal{O}(N_{proc} \times log_2(N_{proc}) \times (J \times M) \times L) \quad flops, \tag{64}$$

and uses N_{com}^{First} communications([14]) where:

$$N_{com}^{First} = \mathcal{O}(N_{proc} \times log_2(N_{proc}) \times (J \times M) \times L) \quad , \tag{65}$$

because the N_{proc} processors, in couples, send one to each other $((\mathbf{g})_k)_p$ vectors of length $L \times (J \times M)$.

[12]Matrix-vector product uses local matrix and vector of size, respectively, $L \times (J \times M) \times (N \times N) \times L_{loc}$ and $(N \times N) \times L_{loc}$.

[13]We assume that collective sum between vectors $(\mathbf{g})_p$ (of length $(J \times M) \times L$) is performed by N_{proc} processors using a tree communication topology where with depth $log_2 N_{proc}$. At each level processors, organized in couples, communicate each other and update, using a sum operation, own vector $(\mathbf{g})_p$. Therefore, at each level, N_{proc} sum between vectors of length $(J \times M) \times L$ are performed (MPI function `MPI_ALLREDUCE` used to perform the collective sum uses such topology)

[14]At each level N_{proc} communications are performed, therefore the total number of communications performed during the collective sum operation is $N_{proc} \times log_2 N_{proc}$.

```
procedure $([K]_3)_k f\ (\ldots)$
    for each $j \in J^p$
    begin for each
    Computation of the 2D projections
        $\mathbf{tu} = ([E]_2)_k\, \mathbf{u}_j$
        Computation of contribution of the weights
        $\alpha_{i-j}$ to projections
        for each $i \in I^p$
        begin for each
            $(\mathbf{g}_i)_k = (\mathbf{g}_i)_k + \alpha_{i-j}\mathbf{tu}$
        end for each
    end for each
    send operation involving $((\mathbf{g}_i)_k)_p$
    for each $q \in \{1, \ldots, N_{proc}\}$ such that $E_p^q \neq \emptyset$
    begin for each
        for each $j \in J^p$
        begin for each
            $\mathbf{tu} = ([E]_2)_k\, \mathbf{f}_j$
            for each $i \in E_p^q$
            begin for each
                $((\mathbf{g}_i)_k)_p = ((\mathbf{g}_i)_k)_p + \alpha_{j-i}\mathbf{tu}$
                call Send$\Big(((\mathbf{g}_i)_k)_p\,, q\Big)$
            end for each
        end for each
    end for each
    Receive operations involving $((\mathbf{g}_i)_k)_q$
    for each $q \in \{1, \ldots, N_{proc}\}$ such that $E_q^p \neq \emptyset$
    begin for each
        for each $i \in E_q^p$
        begin for each
            call Receive$\Big(((\mathbf{g}_i)_k)_q\,, q\Big)$
            Local sum operation
            $(\mathbf{g}_i)_k = (\mathbf{g}_i)_k + ((\mathbf{g}_i)_k)_q$
        end for each
    end for each
end procedure
```

Figure 12: *Parallel algorithm for computing* $\mathbf{g} = [K_i]_3\, \mathbf{u}$ *of 2D+1 problem (Second Strategy).*

At first step of the second strategy, each processor p calculates $|J^p|$ ([15]) components of $\mathbf{g}$, i.e. calculates $(\mathbf{g}^i)_k$ where $i \in J^p$, and contributes to the computation of the other $|E_p^q|$ components of $\mathbf{g}$, i.e. to the other projections $(\mathbf{g}^i)_k$ where $i \in E_p^q$, by using a block matrix-vector operations, with a cost of $\mathcal{O}((|J^p| + |E_p^q|) \times (J \times M) \times (N \times N))$ flops. $|J^p| = L_{loc}$, then the cost is:

$$N_{calc}^{(1.Second)} = \mathcal{O}(N_{proc} \times ((L_{loc} + |E_p^q|) \times L'_{loc}) \times (J \times M) \times (N \times N)) \quad flops. \tag{66}$$

Second step of the second strategy requires a sum between at most $|E_p^q|$ neighborhood processors with a cost of:

$$N_{calc}^{(2.Second)} = \mathcal{O}(N_{proc} \times |E_p^q| \times (J \times M) \times |E_p^q|) \quad flops, \tag{67}$$

and a number of communications:

$$N_{com}^{(Second)} = \mathcal{O}(N_{proc} \times |E_p^q| \times (J \times M) \times |E_p^q|) \quad , \tag{68}$$

because, at each communication, processors p and q where $|E_p^q| \neq 0$, exchange sub-vectors of length $|E_p^q| \times (J \times M)$ of $(g)_k$.

If $L = L'$, that is $L_{loc} = L'_{loc}$:

L_{loc} is at most $|E_p^q| = |J^q|$ and:

$$L \times L_{loc} = \frac{L^2}{N_{proc}} > 2\frac{L^2}{N_{proc}^2} = (L_{loc} + |E_p^q|) \times L_{loc}) \quad ,$$

then:

$$\boxed{N_{calc}^{(1.First)} > N_{calc}^{(1.Second)}} \tag{69}$$

If N_{proc} increases also $log_2 N_{proc}$ increases, and L_{loc} decreases, then:

$$\boxed{N_{calc}^{(2.First)} > N_{calc}^{(2.Second)}} \tag{70}$$

From $N_{calc}^{(2.First)} = N_{com}^{(First)}$ and $N_{calc}^{(2.Second)} = N_{com}^{(Second)}$, and from (70) it follows:

$$\boxed{N_{com}^{(First)} > N_{com}^{(Second)}} \tag{71}$$

Let us now consider the speed-up $S_{N_{proc}}$:

$$S_{N_{proc}} = \frac{T_1}{T_{N_{proc}}} = \frac{N \times t_{calc}}{N_{calc} \times t_{calc} + T_{com} \times t_{com}} \quad , \tag{72}$$

[15] $|\mathcal{J}|$ denotes the cardinality of the set $\mathcal{J}$.

where T_1 and $T_{N_{proc}}$ denote the execution time respectively on one and N_{proc} processors, N and N_{calc} denotes the number of floating point operations performed respectively by one and by N_{proc} processors, t_{calc} is the execution time of one floating point operation, and t_{com} is the execution time of one communication between two processors.
The Product (61) requires:

$$N_{calc} = \mathcal{O}((L \times L) \times (J \times M) \times (N \times N)) \quad flops, \tag{73}$$

therefore:

$$T_1 = [(L \times L) \times (J \times M) \times (N \times N)] \times t_{calc} \quad . \tag{74}$$

If, for the sake of simplicity, we assume $J = M = N$ then (74) can simplified as follow:

$$T_1 = [L^2 \times N^4] \times t_{calc} \quad . \tag{75}$$

and:

$$T_{N_{proc}}^{(*)} = [N_{calc}^{(1.*)} + N_{calc}^{(2.*)}] \times t_{calc} + [N_{com}^{(*)}] \times t_{com} \quad , \tag{76}$$

where the symbol "*" means "*First*" or "*Second*".

Substituting (63), (64) and (65) in (76) gives:

$$\begin{aligned} T_{N_{proc}}^{(First)} &= [N_{proc} \times (L \times L_{loc}) \times (J \times M) \times (N \times N) + \\ &+ N_{proc} \times log_2(N_{proc}) \times (J \times M) \times L] \times t_{calc} + \\ &+ [N_{proc} \times log_2(N_{proc}) \times (J \times M) \times L] \times t_{com} \quad . \end{aligned} \tag{77}$$

If $t_{com} = 2t_{calc}$ and $J = M = N$, then (77) can be simplified as follows:

$$T_{N_{proc}}^{(First)} = [(N^2 \times L) + (N^2 \times L + 3N_{proc} \times log_2 N_{proc})] \times t_{calc} \quad , \tag{78}$$

then:

$$\begin{aligned} S_{N_{proc}}^{First} &= \frac{L^2 \times N^4}{L^2 \times N^4 + L \times N^2 \times 3N_{proc} \times log_2 N_{proc}} = \\ &= \frac{L \times N^2}{L \times N^2 + 3N_{proc} \times log_2 N_{proc}} \quad . \end{aligned} \tag{79}$$

From (79) it follows that, if N and L are fixed and N_{proc} increases, speed-up degrades.

Substituting (66), (67) and (68) in (76) gives:

$$\begin{aligned} T_{N_{proc}}^{(Second)} &= [N_{proc} \times ((L_{loc} + |E_p^q|) \times L'_{loc}) \times (J \times M) \times (N \times N) + \quad (80) \\ &+ N_{proc} \times |E_p^q|) \times (J \times M) \times |E_p^q|] \times t_{calc} + \\ &+ [N_{proc} \times |E_p^q| \times (J \times M) \times |E_p^q|] \times t_{com} \quad . \end{aligned}$$

If $t_{com} = 2t_{calc}$, and $J = M = N$, and $L = L'$, since L_{loc} is at the most $|E_p^q| = |J^q|$, (80) can simplified as follows:

$$T_{N_{proc}}^{(Second)} = [2 \times \frac{L^2}{N_{proc}} \times N^4 + 3\frac{L^2}{N_{proc}} \times N^2] \times t_{calc} \quad , \tag{81}$$

then:

$$\begin{aligned} S_{N_{proc}}^{Second} &= \frac{L^2 \times N^4}{2 \times \frac{L^2}{N_{proc}} \times N^4 + 3\frac{L^2}{N_{proc}} \times N^2} = \\ &= \frac{N^2 \times N_{proc}}{2(N^2+3)} \quad . \end{aligned} \tag{82}$$

Observe that if N and L are fixed and N_{proc} increases, also the speed-up increases, and the parallel algorithm scales with the processors number.

Second Strategy shows a better speed up. Other results are in [16].

6 The MEDITOMO library

In table 1 we list the software modules of the library ([16]):

- *fully 3D*
  - *fan* collimator
    - *Conjugate Gradient* (`3dtomo_fan_cg`)
    - *Fixed Point* (`3dtomo_fan_fp+tv`)
    - *EMOSn* (`3dtomo_fan_em`)
    - *TV regularizated EMOSn* (`3dtomo_fan_em+tv`)
  - *parallel* collimator

[16]`XXtomo_YYY_ZZ` is the name of each software:

- `XX` denotes the model (`3d` or `2d+1`);
- `YYY` denotes the geometry of the acquisition system (parallel `par` o fan `fan`);
- `ZZ` denotes the algorithm (conjugate gradient `cg`, expectation maximization `em` or fixed point `fp+tv`);

MEDITOMO Library	
`Acquisition Models`	
Fully 3D	*2D+1*
`Reconstruction Algorithms`	`Reconstruction Algorithms`
⋆ FP (TV) ⋆ CG ⋆ EMOS ⋆ EMOS-TV	⋆ FP (TV) ⋆ CG ⋆ EMOS

Table 1: MEDITOMO library

    * *Conjugate Gradient* (`3dtomo_par_cg`)

- *2D+1*
  - *fan* geometry
    * *Conjugate Gradient* (`2d+1tomo_fan_cg`)
    * *Fixed Point* (`2d+1tomo_fan_fp+tv`)
    * *TV-based EMOSn* (`2d+1tomo_fan_em+tv`)

Parallel software use the **M**essage **P**assing **I**nterface (MPI) communication library for the message passing operations. We compare results obtained using:

1. `3dtomo_fan_cg` and `3dtomo_fan_fp+tv`;
2. `3dtomo_fan_em` and `3dtomo_fan_em+tv`;
3. `2d+1tomo_fan_em` and `2d+1tomo_fan_cg`.

Tests are performed on synthetic images and on real data:

- synthetic data obtained *"blurring"* the Hoffman phantom; in fig. 14, a projection for a fixed angle (a) and a sinogram (b) are shown. Projections data set is obtained simulating the acquisition system, where the matrix $[K]_3$ is computed by using a Gauss function [9]. The values of (2) considered are $\sigma_{min} = 3$, $\sigma_{max} = 8$, $r = 192$. We generate 120 sinograms of size 128×128, where 120 is the number of angular views in $[0, \pi)$.
- in-vivo data([17]).

In figure 16 we show the *Restoration Error*:

$$RE = \frac{\| [u]_{true} - [u^*] \|_2}{\| [u]_{true} \|_2} \quad . \tag{83}$$

In figure 20 we show results for different iterations. Results refer to TV-based algortithms in figure 7 (a) and to the standard EMOSn (b).

[17]The data are courteously granted by the Careggi Hospital in Florence (Italy).

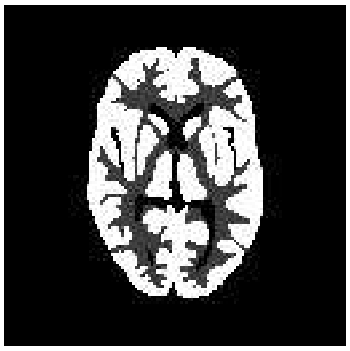
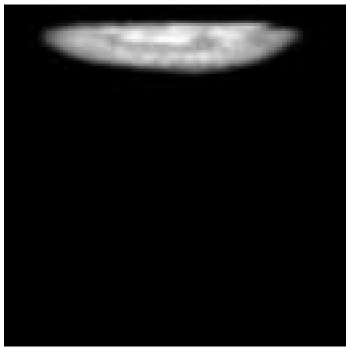
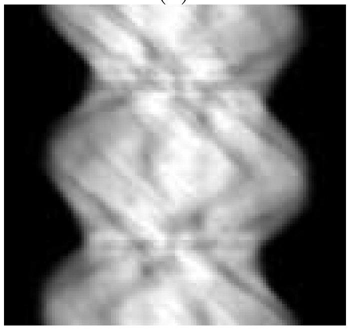


Figure 13: *An Hoffman phantom section.*



(a)



(b)

Figure 14: A projection section (a) and a sinogram (b) examples

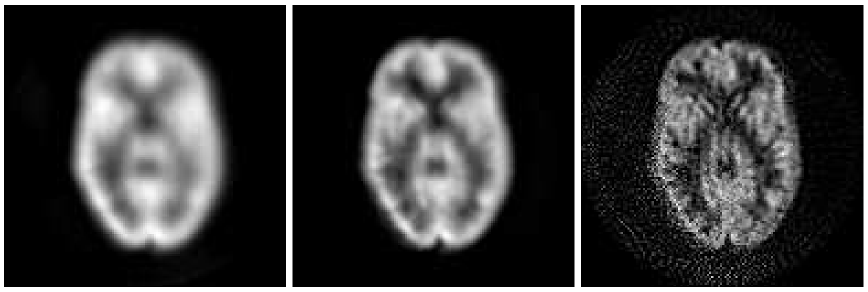


Figure 15: *Results using* `3dtomo_fan_cg` *of the Hoffmann phantom. Number of steps are* 7, 21, 63.

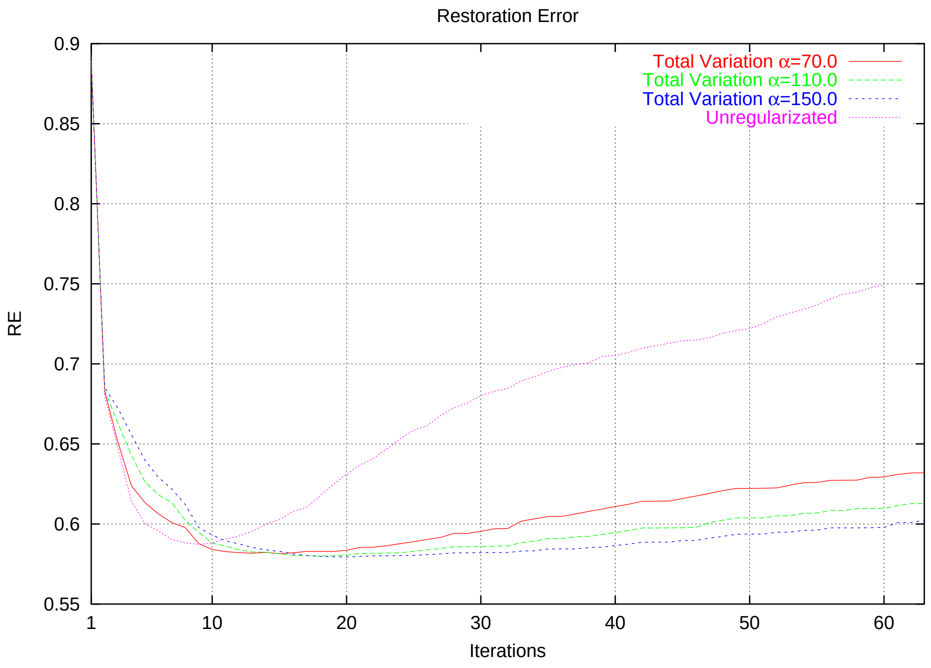


Figure 16: *RE for* `3dtomo_fan_cg` *and* `3dtomo_fan_fp+tv`. *We plot RE at different values of the regularization parameter* α.

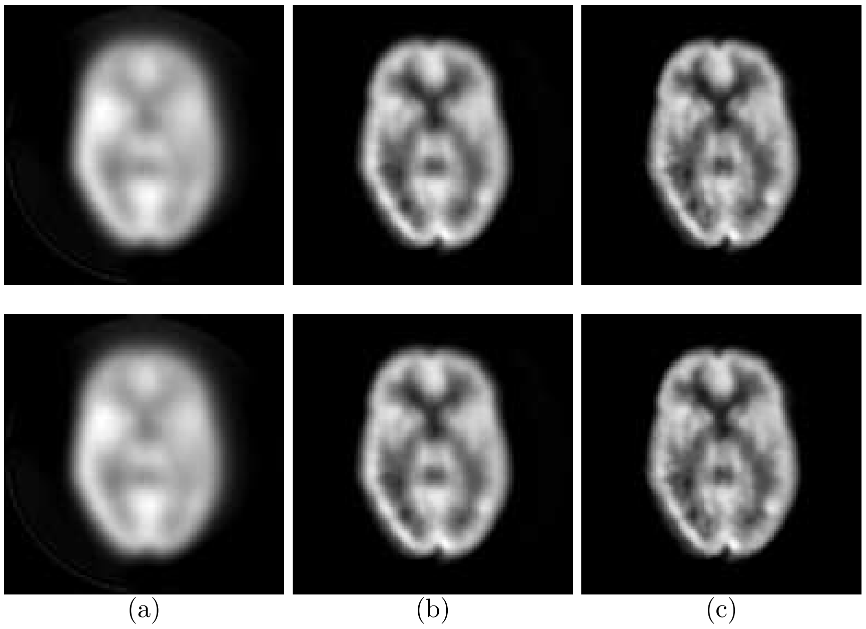


Figure 17: *Results of* `3dtomo_fan_fp+tv` *on the Hofmann phantom, at step* $m = 1$ *(a),* $m = 3$ *(b) and* $m = 9$ *(c) of FP. CG steps is* 7 *(the total number of iterations considered is than* 7, $21 and 63$ *respectively).* $\alpha = 70$ *at the first row and* $\alpha = 150$ *at the second row.*

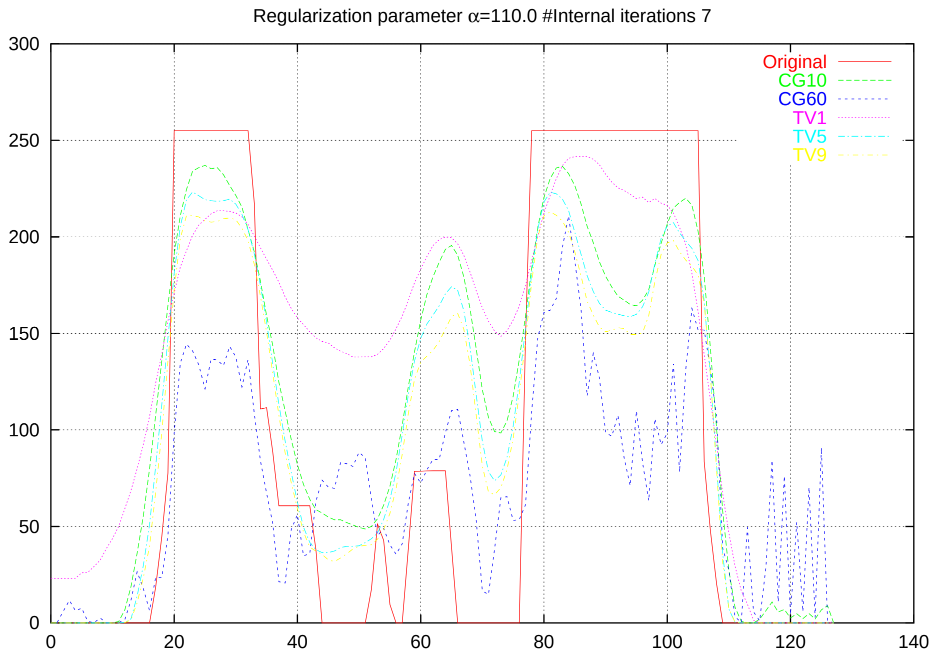


Figure 18: *Profiles obtained by* `3dtomo_fan_fp+tv` *and* `3dtomo_fan_cg`.

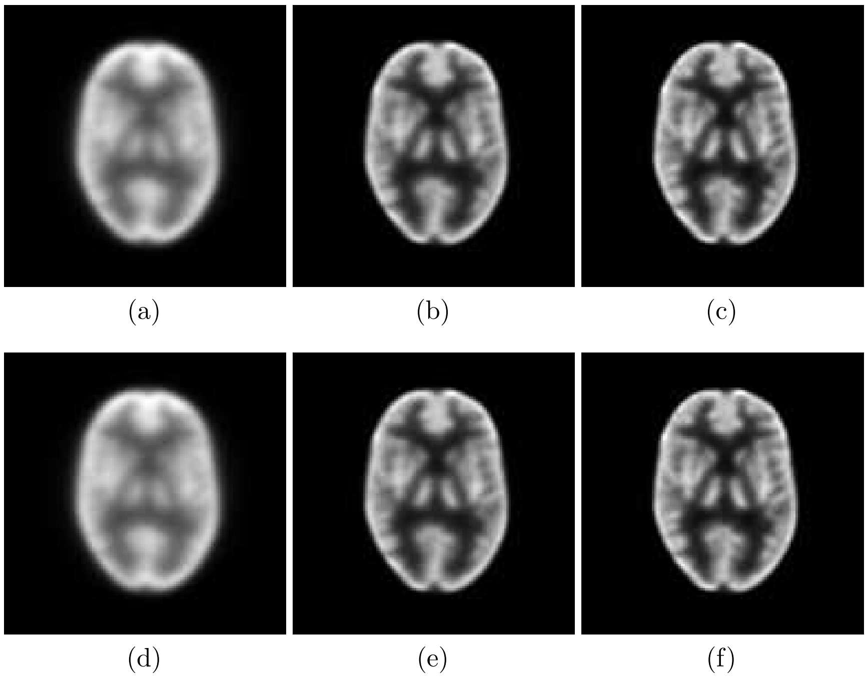
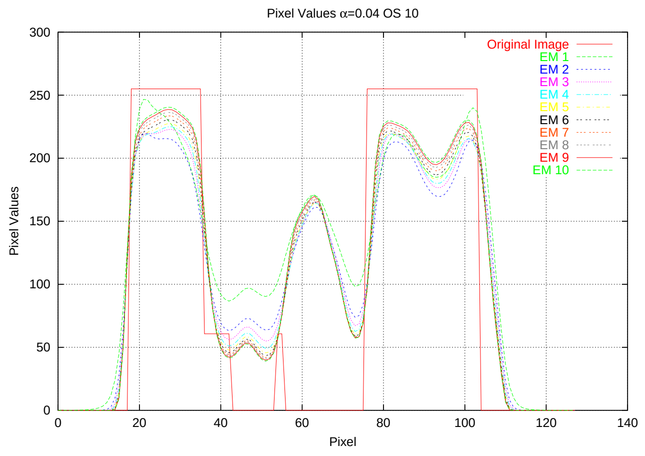
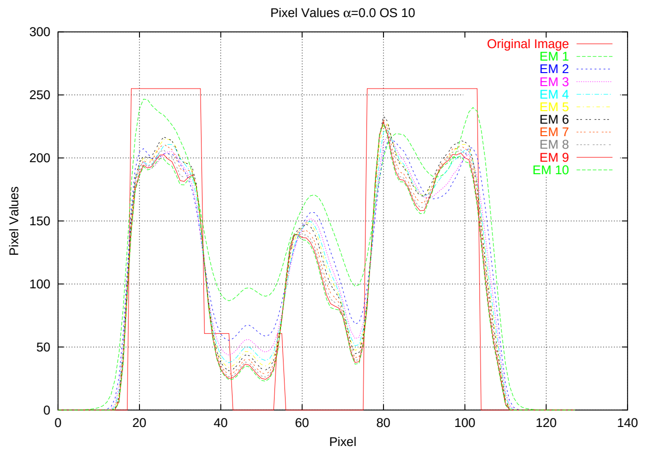


Figure 19: *Reconstruction of the Hoffman phantom, by* `3dtomo_fan_em` *module, at I step (a), VI step (b) and X step (c) of EM. Reconstruction of Hoffman phantom, by* `3dtomo_fan_em+tv`*, with* $\alpha = 0.04$*, at I step (d), VI step (e) and X step (f) of EM.*



(a)



(b)

Figure 20: *Cross section of images obtained by* `3dtomo_fan_em+tv` *(a) and* `3dtomo_fan_em` *(b) respectively, for different steps of EM.*

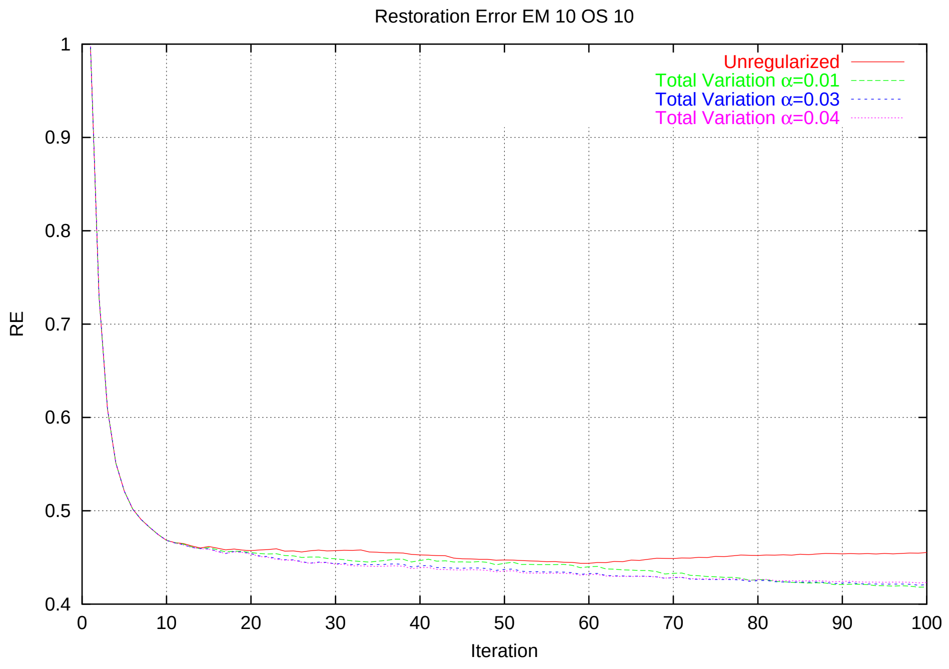


Figure 21: *RE for* `2d+1tomo_fan_em` *and* `2d+1tomo_fan_em+tv`.

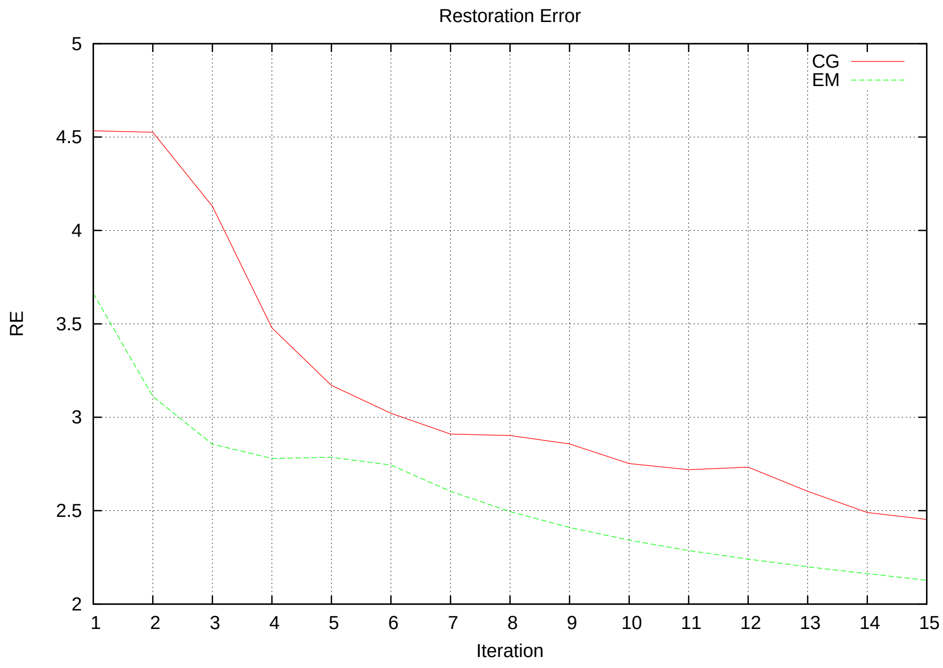


Figure 22: *RE relatively to* `2d+1tomo_fan_cg` *and* `2d+1tomo_fan_em` *modules. The values of RE is plotted for different iterations of OS.*

6.1 Conclusions

1. CG and EM appear to be *semiconvergent*: the RE, after reaching a minimum then grows while the TV-based algorihms show a more stable behaviour even with different values of α.

2. it_{opt} is 10 and 6 for CG and EM respectively.

3. TV-regularization provides results of better quality.

4. Choice of α and β:

   4.a Suitable values of α are $\alpha = 70$ for FP and $\alpha = 0.04$ for EM;

   4.b parameter β in (**??**) has been fixed to 0.01 and 0.001 for the CG and EM. The value of β in EM is smaller because EM works with normalized values of the brightness.

5. EM is more accurate than CG. Accuracy increases as the number n of the *subset* grows. Optimal value of n is about 15-20 for EM and 10 for CG.

6. MEDITOMO has been tested on MIMD distributed memory parallel systems. In figures 24, 25 and 26 we show perfomance, in terms of time, speed-up and efficiency, of the `3dtomo_fan_fp+tv` running the software on a *Beowulf* system made of 18 nodes, each whit 1.5 Ghz *Pentium IV* processor with a *Fast Ethernet 10/100 MBit* network card. Execution time of one iteration decreases from 10 to 1 minute. Efficiency of the most expensive code is at least of 60%.

7 Acknowledgements

We thank prof. Mario Bertero for his suggestions. We thank Andreas Robert Formiconi end Piero Calvini which gave us the prototype software and collaborate at the development of MEDITOMO.

We thank Careggi Hospital in Florence which gave data used in numerical experience. We also thank Careggi Hospital for the help of its doctors in nuclear medicine.

The authors

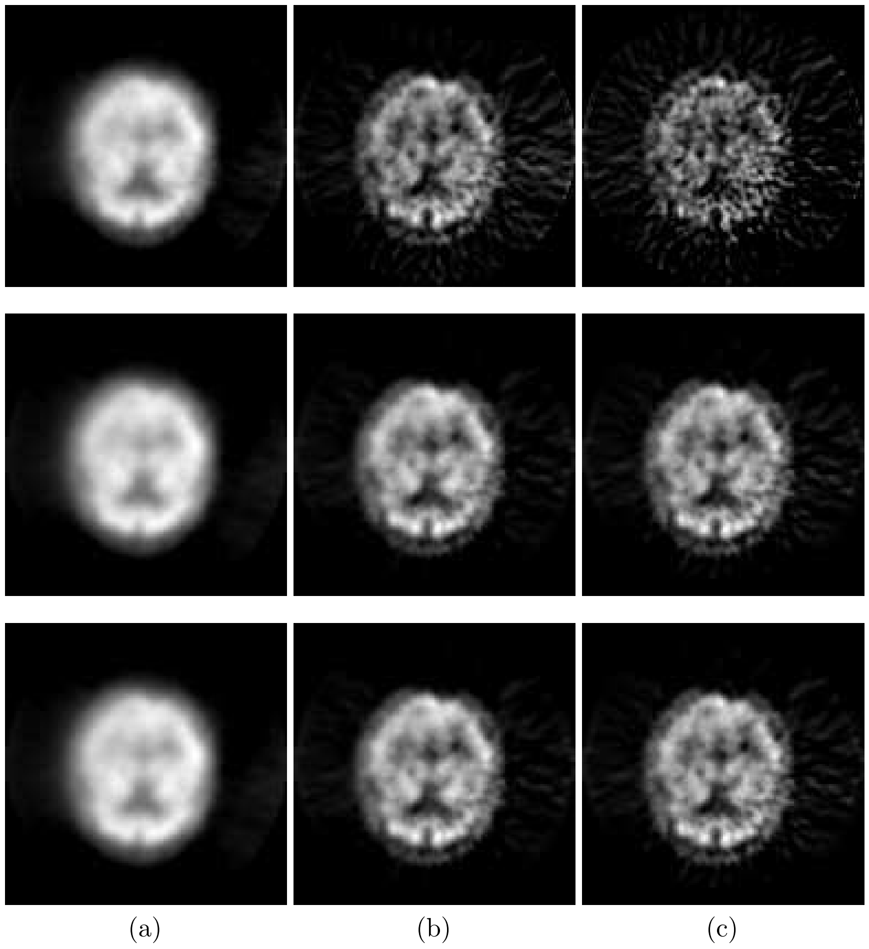


Figure 23: *First row: sections of human brain obtained by* `3dtomo_fan_cg`*. iteration number:* 5, 15 45 *; second and third rows: sections of human brain obtained by* `3dtomo_fan_fp+tv` *with* $\alpha = 50, 70$ *at step* $m = 1$ *(a),* $m = 3$ *(b) and* $m = 5$ *(c) of FP, and with CG steps* 5 *(Total number of iterations equals to* 5, 15 45 *respectively).*

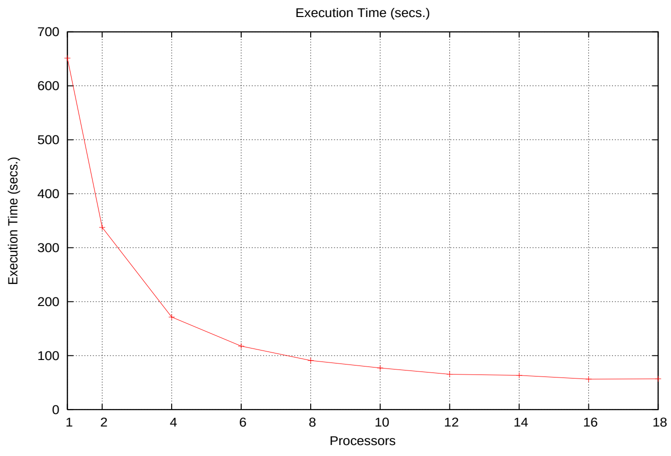


Figure 24: *Execution time of* `3dtomo_fan_fp+tv` *versus the number of processors.*

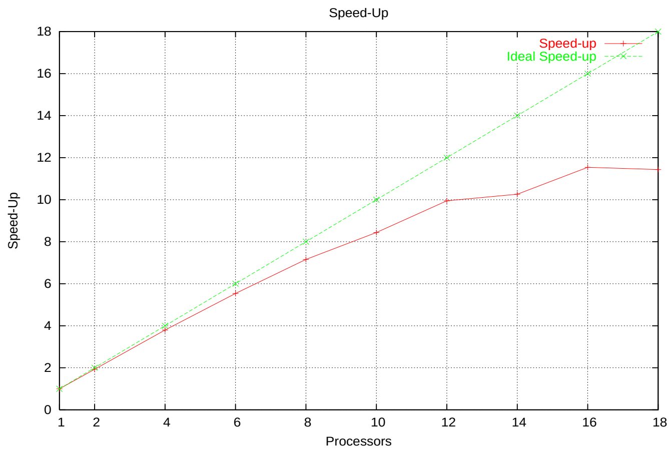


Figure 25: *Speedup of* `3dtomo_fan_fp+tv` *versus the number of processors.*

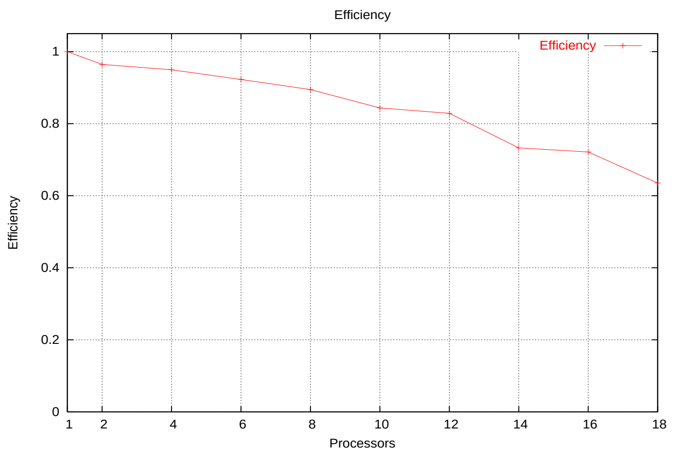


Figure 26: *Efficiency of* `3dtomo_fan_fp+tv` *versus the number of processors.*

References

[1] `L. Alvarez, F. Guichard, P.L. Lions, J.M. Morel` ,*Axioms and fundamental equations of image processing*, Arch. Rational Mechanics, 123, (1993).

[2] `L. Antonelli`, *Un software parallelo basato sul metodo dei minimi quadrati con vincoli*, Degree Thesis, (1999).

[3] `L. Antonelli, L. Carracciuolo, M. Ceccarelli, L. D'Amore, A. Murli`, *High Performance Edge Preserving Restoration in 3D SPECT Imaging*, Parallel Computing Special Issue "Parallel and Distribuited Image and Video Processing" (to appear).

[4] `L. Antonelli, L. Carracciuolo, M. Ceccarelli, L. D'Amore, A. Murli`, *Total Variation Regularization for Edge Preserving 3D SPECT Imaging in High Performance Computing Environments* Lecture Notes in Computer Science, (2002).

[5] `L. Antonelli, L. Carracciuolo, L. D'Amore, A. Murli`, Il funzionale di Totale Variazione nella ricostruzione di dati 3D SPECT mediante l'algoritmo di EMOS$_n$, ICAR-CNR technical report, TR-02-01, (2002)

[6] `L. Antonelli`, *Sulla risoluzione numerica di un problema inverso mal posto in ambiente di calcolo ad alte prestazioni*, PhD Thesis, (2003).

[7] `L. Antonelli, L. Carracciuolo, L. D'Amore, A. Murli`, *Stima teorica e validazione sperimentale del nucleo computazionale per la ricostruzione di dati SPECT 3D basato sull'algoritmo del Gradiente Coniugato*, ICAR-CNR technical report, TR-04-15, (2004).

[8] L. Antonelli, *Stima teorica e sua validazione sperimentale delle prestazioni degli algoritmi CG ed EM alla base della ricostruzione di dati SPECT*, ICAR-CNR technical report, TR-05-4, (2005).

[9] D. Baldini, P. Calvini, A.R. Formiconi, *Image reconstruction with conjugate gradient algorithm and compensation of the variable system response for an annular SPECT system*, Physic Medica, vol 14, (1998).

[10] M. Bertero, P. Boccacci, *Introduction to Inverse Problems in Imaging*, Institute of Physics Publishing, Bristol (1998).

[11] M. Bertero, P. Bonetto, L. Carracciuolo, L. D'Amore, A.R. Formiconi, M.R. Guarracino, G. Laccetti, A. Murli, G. Oliva, *MedIGrid: a medical image application for computational Grid*, 17th International Parallel and Distributed Processing Symposium, IPDPS-2003 (2003).

[12] P. Boccacci, P. Bonetto, P. Calvini, A. Formiconi, *A simple model for the efficient correction of collimator blur in 3D SPECT imaging*, Inverse Problems 15 (1999).

[13] P. Blomgren, T. Chan, P. Mulet, C. Wong, *Total Variation image restoration: numerical methods and extensions*, IEEE International Conference on Image Processing (1997).

[14] C. Byrne, *Accelerating the EMML algorithm and related iterative algorithms by rescaled block-iterative methods*, IEEE Transactions on Image Processing 7, (1998).

[15] C. Byrne, *Notes on block-iterative or ordered subset methods for image reconstruction*, University of Massachusettes technical report, MA01854, (2000).

[16] L. Carracciuolo, L. D'Amore, A. Murli, *Sviluppo di software parallelo per la ricostruzione di dati 3D SPECT basato sull'algoritmo EMOS$_n$*, ICAR-CNR technical report, TR-02-02, (2002).

[17] D. Casaburi, *Un software parallelo basato sul metodo di Newton*, Degree Thesis, (2002).

[18] Y. Censor, *Row-action methods for huge and sparse systems and their applications*, SIAm Review 23 (1981).

[19] T. Chan, P. Mulet, *Iterative methods for Total Variation image restoration*, SIAM Journal on Applied Mathematics, (1995).

[20] T.F. Chan, G.H. Golub, P. Mulet, *A non linear Primal-Dual Method for Total Variation-Based Image Restoration*, Lecture Notes in Control and Information Sciencies, (1996)

[21] L. D'Amore, V. De Simone, A. Murli, *The Wiener Filter and Regularization Methods for Image Restoration Problems*, Proceedings of the International Conference on Image Analysis and Processing, IEEE Computer Society Publisher, (1999).

[22] D. Dobson, C. Vogel *Convergence of an iterative method for Total Variation*, SIAM Journal on Numerical Analysis, (1997)

[23] R.E. Ewing, J. Shen, *A multigrid algorithm for the cell-center finite difference scheme*, 6th Copper Mountain Conference Multigrid Methods, Imperial College Press, (1993)

[24] A. Formiconi, *Geometrical response of multihole collimators*, Physics in Medicine and Biology 43 (1998).

[25] E. Giusti, *Minimal Surface and Function of Bounded Variation*, Birkhauser, (1984)

[26] P.J. Green, *Bayesian reconstruction from emission tomography data using a modified EM algorithm*, IEEE Transactions on Medical Imaging 9, (1990).

[27] P.J. Green, *On use of the EM algorithm for penalized likelihood estimation*, Journal of the Royal Statistical Society B 52, (1990).

[28] C.W. Groetsch, *Inverse problems in the mathematical sciences*, Vieweg Verlag, Wiesbaden, (1993).

[29] R.H. Huesman, G.T. Gullberg, W.L. Greenberg, T.F. Budinger, *User's Manual; Donner Algorithms for Reconstruction Tomography*, Lawrence Berkeley Laboratory, University of California (1977).

[30] H. Hudson, R. Larkin, *Accelerated image reconstruction using ordered subsets of projection data*, IEEE Transactions on Medical Imaging 4, (1994).

[31] M. Oman, C. Vogel, *Iterative method for Total Variation denoising*, SIAM Journal on Scientific and Statistical Computing (1996).

[32] M. Oman, C. Vogel, *Fast robust Total Variation-based reconstruction of noisy, blurred images*, IEEE Transactions on Image Processing 7 (1998).

[33] M. Oman, C. Vogel,*Continuation method for Total Variation denoising problem* SIAM Journal on Applied Mathematics (1995).

[34] V. Y. Panin, G.L. Zeng, G.T. Gullberg ,*Total Variation Regulated EM Algorithm*, IEEE Transactions on Nuclear Science 6, (1999).

[35] L. Shepp, Y. Vardi, *Maximum likelihood reconstruction for emission tomography*, IEEE Transactions on Medical Imaging 4, (1982).