

Iterative regularization algorithms for constrained image deblurring on graphics processors

Valeria Ruggiero · Thomas Serafini ·
Riccardo Zanella · Luca Zanni

Received: 1 December 2008

Abstract The ability of the modern graphics processors to operate on large matrices in parallel can be exploited for solving constrained image deblurring problems in a short time. In particular, in this paper we propose the parallel implementation of two iterative regularization methods: the well known expectation maximization algorithm and a recent scaled gradient projection method. The main differences between the considered approaches and their impact on the parallel implementations are discussed. The effectiveness of the parallel schemes and the speedups over standard CPU implementations are evaluated on test problems arising from astronomical images.

Keywords Image deblurring · Gradient projection methods · Graphics processing units

1 Introduction

Image deblurring has given rise to interesting optimization problems and stimulated fruitful advances in numerical optimization techniques. Nowadays several domains of applied science, such as medical imaging, microscopy and astronomy, involve large

This research is supported by the PRIN2006 project of the Italian Ministry of University and Research *Inverse Problems in Medicine and Astronomy*, grant 2006018748.

V. Ruggiero
Dipartimento di Matematica, Università di Ferrara, Polo Scientifico Tecnologico, Blocco B,
Via Saragat 1, I-44100 Ferrara, Italy
E-mail: rgv@unife.it

T. Serafini · R. Zanella · L. Zanni
Dipartimento di Matematica, Università di Modena e Reggio Emilia, Via Campi 213/B, I-41100
Modena, Italy
E-mail: thomas.serafini@unimore.it

R. Zanella
E-mail: riccardo.zanella@unimore.it

L. Zanni
E-mail: luca.zanni@unimore.it

scale deblurring problems whose variational formulations lead to optimization problems with millions of variables that should be solved in a very short time. To face these challenging problems a lot of effort has been put into designing effective algorithms that have largely improved the classical optimization strategies usually applied in image deblurring. Nevertheless, in many large scale applications also these improved algorithms do not provide the expected reconstruction in a suited time. In these cases, the modern multiprocessor architectures represent an important resource for reducing the reconstruction time. To fully exploit these architectures, parallel implementations of the optimization-based deblurring strategies must be carefully designed for the special multiprocessor system available (the interested reader can refer to [10,11,18,25,26] and references therein for examples of parallel optimization methods).

In this paper we discuss the benefits arising from facing the image deblurring problems on Graphics Processing Units (GPUs), that are a non-expensive parallel processing device available on many up-to-date personal computers. Originally developed for 3D graphics applications, GPUs have been employed in other scientific computing areas by showing very exciting speedups in comparison with standard CPU implementations (a complete list of general purpose GPU applications is available at <http://developer.nvidia.com/cuda>). In particular, GPU implementations of algorithms for signal and image reconstruction have recently been proposed by Lee and Wright in [16]. The model considered in [16] for the image deblurring problem leads to the minimization of an objective function given by the sum of a least squares fit-to-data term and a total variation regularization term [22]; the solution of this regularized problem is obtained by means of the primal-dual gradient descent approach described in [31]. The promising speedups obtained in [16] suggest that the design of GPU implementations for other popular image deblurring formulations could be an interesting and useful task.

This work deals with the maximum likelihood formulation of the image deblurring problem in the case of Poisson noise [2], a model of noise describing the effect of photon counting in applications such as emission tomography, microscopy and astronomy. Following this approach and denoting a two-dimensional image $\mathcal{X} \in \mathbb{R}^{N \times N}$ as a vector $\mathbf{x} = (x_1, \dots, x_n)^T \in \mathbb{R}^n$, $n = N^2$, an approximation $\mathbf{x}^* \in \mathbb{R}^n$ of the image to be reconstructed is obtained by solving the minimization problem

$$\begin{aligned} \min \quad & J(\mathbf{x}) = \sum_{i=1}^n \left(\sum_{j=1}^n A_{ij} x_j + bg - b_i - b_i \ln \frac{\sum_{j=1}^n A_{ij} x_j + bg}{b_i} \right) \\ \text{sub. to} \quad & \mathbf{x} \geq \mathbf{0}, \end{aligned} \quad (1)$$

in which the functional $J(\mathbf{x})$ represents the Kullback–Leibler divergence [4] of $(A\mathbf{x} + bg)$ from the observed noisy image \mathbf{b} , where $A \in \mathbb{R}^{n \times n}$ is the blurring operator and bg denotes a constant background term. Due to the ill-posedness of the image deblurring problem, the exact solution of (1) does not provide a sensible estimate of the unknown image and, consequently, iterative methods are usually applied to the problem (1) to obtain acceptable (regularized) solutions by early stopping. In this computational study we consider two iterative regularization methods: the Expectation Maximization (EM) [24] or Richardson–Lucy method [17,21], and the Scaled Gradient Projection (SGP) method recently introduced in [3]. The EM method, one of the most popular algorithms for astronomical and medical image deblurring, is attractive because of its simplicity, the low computational cost per iteration and the ability to preserve the non-negativity of the iterates; however, it usually exhibits very slow convergence rate that highly limits its practical performance. The SGP method is a more complicated scheme that

combines diagonally scaled gradient directions and special steplength selection rules to achieve a good convergence rate and provide the same reconstruction accuracy of EM in a much lower time. We also recall that SGP is suited to solve general differentiable minimization problems subject to simple constraints and it can be applied to other imaging problems.

Our main purpose is to develop GPU implementations of these algorithms that can enable to significantly reduce the time for processing images of several mega-pixels on low cost architectures. Furthermore, we aim to verify how much the strategies on which SGP is based are suited for parallel implementation on GPUs.

The paper is organized as follows: in section 2 we briefly recall the EM and SGP methods, in section 3 we describe the GPU architecture and the parallel implementations of the algorithms; numerical experiments on astronomical images are presented in section 4 to evaluate the CPU and GPU implementations of the two methods and, finally, the main conclusions are discussed in section 5.

2 Iterative regularization algorithms

Before to state the EM and SGP algorithms it is useful to recall some basic properties of the objective function $J(\mathbf{x})$ in (1). First of all we observe that the gradient and the hessian of $J(\mathbf{x})$ can be written as

$$\nabla J(\mathbf{x}) = A^T \mathbf{e} - A^T Y^{-1} \mathbf{b} \quad (2)$$

$$\nabla^2 J(\mathbf{x}) = A^T B Y^{-2} A, \quad (3)$$

where $\mathbf{e} \in \mathbb{R}^n$ is a vector with unitary entries, $Y = \text{diag}(A\mathbf{x} + b\mathbf{g})$ is a diagonal matrix with the entries of $(A\mathbf{x} + b\mathbf{g})$ on the main diagonal and $B = \text{diag}(\mathbf{b})$. The matrix A is generally dense, with nonnegative entries and such that $A^T \mathbf{e} = \mathbf{e}$ and $\sum_j A_{ij} > 0$, $\forall i$. Moreover, we must recall that A comes from the discretization of the Fredholm integral equation that models the image formation process; by imposing periodic boundary conditions for the discretization, the derived matrix is block-circulant with circulant blocks. This property implies that the matrix-vector products involving the matrix A can be performed with $\mathcal{O}(n \log n)$ complexity, by employing the Fast Fourier Transform (FFT) [7] (remember that $A\mathbf{x}$ is just the cyclic convolution of \mathbf{x} with an array usually called *point spread function* (PSF)). Concerning the components b_i of the observed image \mathbf{b} , we remark that they are nonnegative and, consequently, the problem (1) is a convex optimization problem since the hessian matrix (3) is positive semidefinite in any point of the nonnegative orthant.

The Karush–Kuhn–Tucker (KKT) first order optimality conditions for (1) can be stated as follows:

$$\begin{aligned} \mathbf{x}^T \nabla J(\mathbf{x}) &= 0 \\ \nabla J(\mathbf{x}) &\geq \mathbf{0} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned} \quad (4)$$

Since we assume $A^T \mathbf{e} = \mathbf{e}$, the complementarity condition (4) can be rewritten as

$$\mathbf{x} = \text{diag}(\mathbf{x}) A^T Y^{-1} \mathbf{b}.$$

Algorithm SGP (Scaled Gradient Projection Method for problem (1))

Choose the starting point $\mathbf{x}^{(0)} \geq \mathbf{0}$, set the parameters $\beta, \theta \in (0, 1)$, $0 < \alpha_{min} < \alpha_{max}$ and fix a positive integer M .

FOR $k = 0, 1, 2, \dots$ DO THE FOLLOWING STEPS:

STEP 1. Choose the parameter $\alpha_k \in [\alpha_{min}, \alpha_{max}]$ and the scaling matrix $S_k \in \mathcal{S}_L$;

STEP 2. Projection: $\mathbf{y}^{(k)} = P(\mathbf{x}^{(k)} - \alpha_k S_k \nabla J(\mathbf{x}^{(k)}))$;

 If $\mathbf{y}^{(k)} = \mathbf{x}^{(k)}$ then stop, declaring that $\mathbf{x}^{(k)}$ is a stationary point;

STEP 3. Descent direction: $\mathbf{d}^{(k)} = \mathbf{y}^{(k)} - \mathbf{x}^{(k)}$;

STEP 4. Set $\lambda_k = 1$ and $J_{max} = \max_{0 \leq j \leq \min(k, M-1)} J(\mathbf{x}^{(k-j)})$;

STEP 5. Backtracking loop:

 IF $J(\mathbf{x}^{(k)} + \lambda_k \mathbf{d}^{(k)}) \leq J_{max} + \beta \lambda_k \nabla J(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)}$ THEN
 go to Step 6;

 ELSE

 set $\lambda_k = \theta \lambda_k$ and go to Step 5;

 ENDIF

STEP 6. Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda_k \mathbf{d}^{(k)}$.

END

The EM method is based on the previous fixed point formulation, and its iteration is defined by

$$\mathbf{x}^{(k+1)} = X_k A^T Y_k^{-1} \mathbf{b} = \mathbf{x}^{(k)} - X_k \nabla J(\mathbf{x}^{(k)}), \quad (5)$$

where $X_k = \text{diag}(\mathbf{x}^{(k)})$ and $Y_k = \text{diag}(A\mathbf{x}^{(k)} + b\mathbf{g})$. We observe that EM can also be viewed as a scaled steepest descent method with diagonal scaling given by the matrix X_k . Starting from a positive initial point $\mathbf{x}^{(0)}$, the non-negativity of A , $b\mathbf{g}$ and \mathbf{b} guarantees that all the successive iterates remain feasible and, under the assumption $b\mathbf{g} = \mathbf{0}$, the convergence to a solution of (1) has been proved by several authors [13–15, 19, 27]. The EM method is attractive because of its low computational cost, consisting in $\mathcal{O}(n \log n)$ operations per iteration (needed to perform the two matrix-vector products involving the matrix A), but in many cases it converges very slowly and does not ensure satisfactory performance.

Among the different strategies suggested to improve the convergence rate of EM, the scaled gradient projection method, recently proposed in [3], seems very appealing for both its performance and its general form that well adapts to other interesting optimization problems in imaging. This method combines non-expensive diagonally scaled gradient directions with special steplength selection rules to obtain a good convergence rate; furthermore, it exploits line-search strategies along the feasible direction to ensure global convergence properties. In the following, we describe the version of the method that applies to the problem (1). First, we denote by $P(\cdot)$ the projection operator onto the feasible region of (1):

$$P(\mathbf{x}) \equiv \arg \min_{\mathbf{z} \geq \mathbf{0}} \|\mathbf{z} - \mathbf{x}\|_2, \quad (6)$$

where $\|\cdot\|_2$ denotes the usual 2-norm of vectors. Then, for a given positive scalar $L > 1$, we define the set \mathcal{S}_L of the diagonal matrix $S = \text{diag}(s_1, s_2, \dots, s_n)$ such that

$$\frac{1}{L} \leq s_i \leq L, \quad i = 1, 2, \dots, n. \quad (7)$$

The main steps of the method are summarized in Algorithm SGP.

This SGP version differs from the general scheme presented in [3] for the projection step, that is rewritten taking into account the special constraint of the problem (1),

Algorithm SS (SGP Steplength Selection)

```

IF  $k = 0$ 
  set  $\alpha_0 \in [\alpha_{min}, \alpha_{max}]$ ,  $\tau_1 \in (0, 1)$  and a nonnegative integer  $M_\alpha$ ;
ELSE
   $\mathbf{r}^{(k-1)} = \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}$ ;    $\mathbf{z}^{(k-1)} = \nabla J(\mathbf{x}^{(k)}) - \nabla J(\mathbf{x}^{(k-1)})$ ;
  IF  $\mathbf{r}^{(k-1)T} D_k^{-1} \mathbf{z}^{(k-1)} \leq 0$  THEN
     $\alpha_k^{(1)} = \alpha_{max}$ ;
  ELSE
     $\alpha_k^{(1)} = \max \left\{ \alpha_{min}, \min \left\{ \frac{\mathbf{r}^{(k-1)T} D_k^{-1} D_k^{-1} \mathbf{r}^{(k-1)}}{\mathbf{r}^{(k-1)T} D_k^{-1} \mathbf{z}^{(k-1)}}, \alpha_{max} \right\} \right\}$ ;
  ENDF
  IF  $\mathbf{r}^{(k-1)T} D_k \mathbf{z}^{(k-1)} \leq 0$  THEN
     $\alpha_k^{(2)} = \alpha_{max}$ ;
  ELSE
     $\alpha_k^{(2)} = \max \left\{ \alpha_{min}, \min \left\{ \frac{\mathbf{r}^{(k-1)T} D_k \mathbf{z}^{(k-1)}}{\mathbf{z}^{(k-1)T} D_k D_k \mathbf{z}^{(k-1)}}, \alpha_{max} \right\} \right\}$ ;
  ENDF
  IF  $\alpha_k^{(2)} / \alpha_k^{(1)} \leq \tau_k$  THEN
     $\alpha_k = \min \left\{ \alpha_j^{(2)}, j = \max \{1, k - M_\alpha\}, \dots, k \right\}$ ;    $\tau_{k+1} = \tau_k * 0.9$ ;
  ELSE
     $\alpha_k = \alpha_k^{(1)}$ ;    $\tau_{k+1} = \tau_k * 1.1$ ;
  ENDF
ENDIF

```

and for the diagonal assumption on the scaling matrix. However, the convergence result proved in [3] still holds and, due to the definition of the objective function $J(\mathbf{x})$, it ensures that every accumulation point of the sequence $\{\mathbf{x}^{(k)}\}$ generated by applying algorithm SGP to the problem (1) is a minimum point. The computational study reported in [3] highlights that the convergence rate of SGP is strictly related to the choices of the scaling matrix S_k and of the steplength parameter α_k . An appropriate choice for the scaling matrix is obtained by simulating the EM scaling:

$$S_k = \text{diag} \left(s_1^{(k)}, \dots, s_n^{(k)} \right), \quad s_i^{(k)} = \min \left\{ L, \max \left\{ \frac{1}{L}, x_i^{(k)} \right\} \right\}, \quad i = 1, \dots, n. \quad (8)$$

A suited steplength selection is derived by rewriting the Barzilai-Borwein steplength rules [1] in the case of scaled gradient directions and then by using an adaptive alternation of these rules. We report in Algorithm SS the alternation strategy suggested in [3] and refer to [8] for the derivation of this steplength selection (see also [5, 6, 9, 23, 28, 30] for further results on the BB alternation rules).

Given the above details, we may compare EM and SGP in terms of the computational cost per iteration. In both the approaches the main computational operations consist in two matrix-vector products, that is, two (two-dimensional) FFT/IFFT pairs with $\mathcal{O}(n \log n)$ complexity. Besides this common cost, SGP requires several additional vector-vector operations with $\mathcal{O}(n)$ complexity, mainly for managing the steplength and the backtracking loop. Due to these additional operations, the time per iteration in SGP is greater than in EM, sometimes more than twice, depending on the size of the image, the test platform/environment and the optimization level of the main computational tasks. Nevertheless, the remarkable reduction in the number of iterations allows SGP to reach the same reconstruction accuracy of EM by saving time up to more than

one order of magnitude (see the numerical results in [3] and in section 4). We will investigate in the following computational study how much the GPU implementation of SGP will be penalized by the additional operations of its iterations.

We end this section by suggesting a suited setting for the SGP parameters that will be exploited in the experiments on astronomical images discussed in section 4:

- line-search: $\beta = 10^{-4}$, $\theta = 0.4$, $M = 10$ (nonmonotone line-search);
- steplength: $\alpha_{min} = 10^{-10}$, $\alpha_{max} = 10^5$, $\alpha_0 = 1.3$, $\tau_1 = 0.5$ and $M_\alpha = 2$;
- scaling matrix: $L = 10^{10}$.

3 GPU implementations

We base our GPU computational study on the NVIDIA Graphics adapters. NVIDIA provides a complete framework, called CUDA (Compute Unified Device Architecture), for programming their GPUs (see <http://www.nvidia.com/cuda>). By means of CUDA it is possible to program a GPU using a C-like programming language. A CUDA program contains instructions both for the CPU and the GPU. The CPU controls the instruction flow, the communications with the peripherals and it starts the single computing tasks on the GPU. The GPU performs the raw computation tasks, using the problem data stored into the graphics memory. The GPU core is highly parallel: it is composed by a number of streaming multiprocessors, which depend on the graphics adapter model. Each streaming multiprocessor is composed by 8 cores, a high speed ram memory block, shared among the 8 cores and a cache. All the streaming multiprocessors can access to a global main memory where, typically, the problem data are stored. For our numerical experiments we used CUDA 2.0 [20] and a NVIDIA GTX 280 graphics card, which has 30 streaming multiprocessors (240 total cores) running at 1296 MHz. The total amount of global memory is 1GB and the connection bus with the cores has a bandwidth of 141.7 GB/sec. The peak computing performance is 933 GFLOPS/sec. The GPU is connected to the CPU with a PCI-Express bus, which grants a 8GB/sec transfer rate. It should be noted that this speed is much slower than the GPU-to-GPU transfer so, for exploiting the GPU performances, it is very important to reduce the CPU-GPU memory communications and keep all the problem data on the GPU memory. Besides, the full GPU-to-GPU bandwidth can be obtained only if a coalesced memory access scheme (see NVIDIA documentation [20]) is used; so, all our GPU computation kernels are implemented using that memory pattern. For the implementation of EM and SGP, two kernel libraries of CUDA are very important: CUFFT and CUBLAS. By the CUFFT library we can compute 1-D, 2-D and 3-D FFT: complex-to-complex, real-to-complex and complex-to-real versions are available. The CUBLAS library is a GPU implementation of the main BLAS subroutines for levels 1, 2 and 3. The use of these libraries is highly recommended for maximally exploiting the GPU performances.

As already observed, the main computational block in both the EM and SGP iteration consists in a pair of forward and backward FFTs for computing the image convolutions. We face these operations by means of the CUFFT subroutines: after computing a 2-D real-to-complex transform, the spectral multiplication between the transformed iterate and PSF is carried out and the 2-D inverse complex-to-real transform is computed. Furthermore, both the algorithms need a division for each pixel in the image, while the computation of the objective function in SGP requires also a logarithm for each pixel. These tasks are particularly suited for a GPU implementation: in fact there

is no dependency among the pixels and the computation can be distributed on all the scalar processors available on the GPU. Besides, divisions and logarithms require a higher number of clock cycles than a simple floating point operation, thus the GPU memory bandwidth is not a limitation for these operations. Finally, we must discuss a critical part involved by the SGP: the scalar products for updating the steplength. The “reduction” operation necessary to compute a scalar product implies a high number of communications among the processors and there are dependencies that prevent a straight parallelization. In [12], NVIDIA reports an analysis of 7 different strategies for computing a reduction and suggests which is the best one for a high volume of data. In our experiments, the CUBLAS function for scalar product (cublasSdot) generally achieves the best performance and then we exploit the kernel libraries provided by NVIDIA also for this task. Anyway, scalar product computation on GPU has a negative impact on performances compared to CPU; this means that we may expect a loss of speedup in the SGP method compared to the EM.

4 Numerical experiments

In order to evaluate the effectiveness of the proposed GPU implementations, we consider some deblurring problems on astronomical images corrupted by Poisson noise. Our test platform consists in a personal computer equipped with an AMD Athlon X2 Dual-Core at 3.11GHz, 3GB of RAM and the graphics processing unit NVIDIA GTX 280 described in the previous section. We consider CPU implementations of EM and SGP in Matlab 7.5.0 and in C (double precision) within the Microsoft Visual Studio 2005 environment; the GPU implementations are developed in mixed C and CUDA language (single precision) within Microsoft Visual Studio 2005.

The test problems are generated as described in [3]: we convolve original 256×256 images with an ideal PSF, then we add a constant background term and we perturb the resulting images with Poisson noise. Two different levels of noise are considered by changing the total flux, that is, the total number of counts, of the original images (the noise level is increasing when the total flux is decreasing). The original images, denoted by the letters A and B, are shown in the upper panels of Figure 1. In the other panels of Figure 1 we report, for each image, the blurred noisy images corresponding to a low level of noise (images A1 and B1 in the middle panels; the total flux is 4.43×10^9) and to a high level of noise (images A2 and B2 in the lower panels; the total flux is 4.43×10^7). We obtain test problems of larger size by expanding the original images and the PSF by means of a zero-padding technique on their FFTs. The expansion is made by preserving the medium value of the pixels and by using the same value of the background; as a consequence, the noise levels of the new larger images are comparable with those of the corresponding blurred noisy images sized 256×256 . In this way, from each of the four test problems A1, B1, A2 and B2, we derive three other test problems with sizes 512×512 , 1024×1024 and 2048×2048 , on which the scaling properties of the iterative regularization algorithms can be evaluated.

Concerning the number of iterations that must be required to obtain a suited regularized solution, we exploit values able to provide, for each test problem, a relative reconstruction error (defined as $\|\mathbf{x}^{(k)} - \mathbf{x}\|_2 / \|\mathbf{x}\|_2$, \mathbf{x} being the original image to be reconstructed) close to the best one. As an example, we show in Figure 2 the behaviour of the relative error as a function of the number of iterations in the case of the 256×256 images B1 (left panel) and B2 (right panel). In these experiments, for each class of test

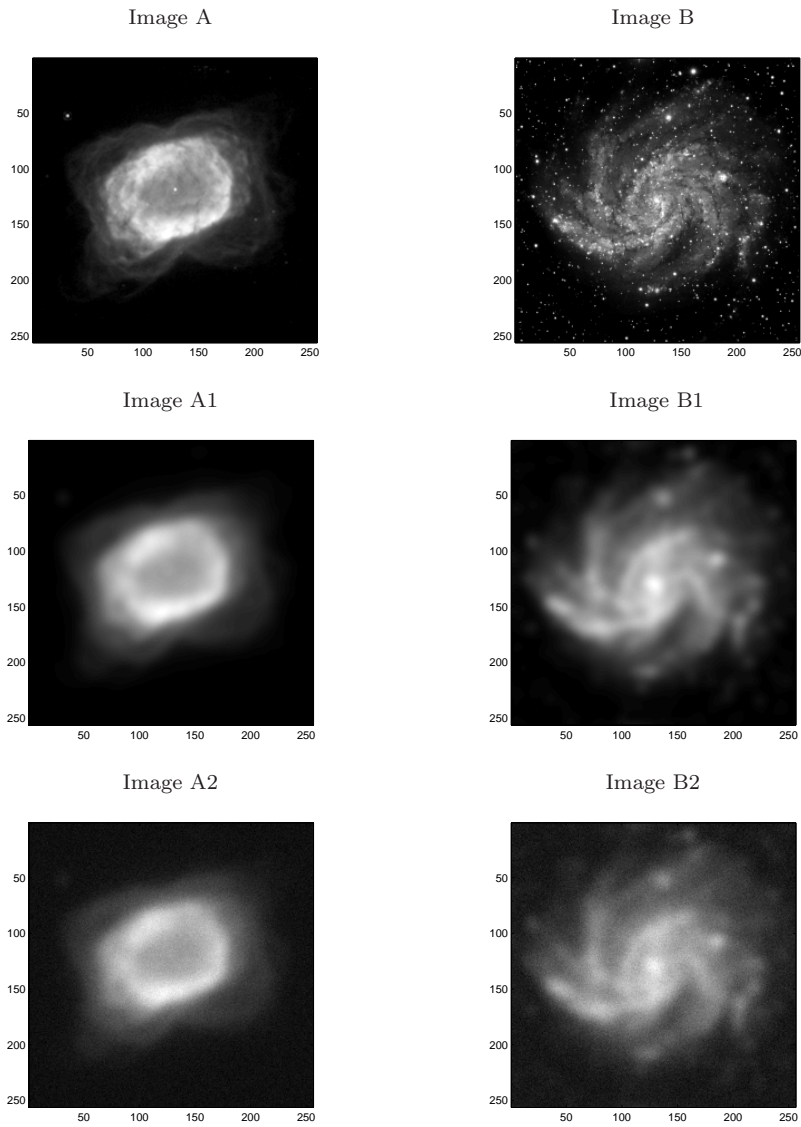


Fig. 1 Original images (upper panels) and blurred noisy images with low level on noise (middle panels) and high level of noise (lower panels).

problems, we determine a suited number of iterations for the images sized 256×256 and use the same number of iterations also on the corresponding images of larger sizes, since we observed that it provides satisfactory reconstructions. The reconstructed images corresponding to the blurred noisy images of size 256×256 are shown in Figure 3.

Before to evaluate the effectiveness of the proposed parallel implementations, we show the behaviour of the Matlab and C serial implementations of both EM and SGP. In Table 1, for the test problems A1 and A2, we report the relative reconstruction

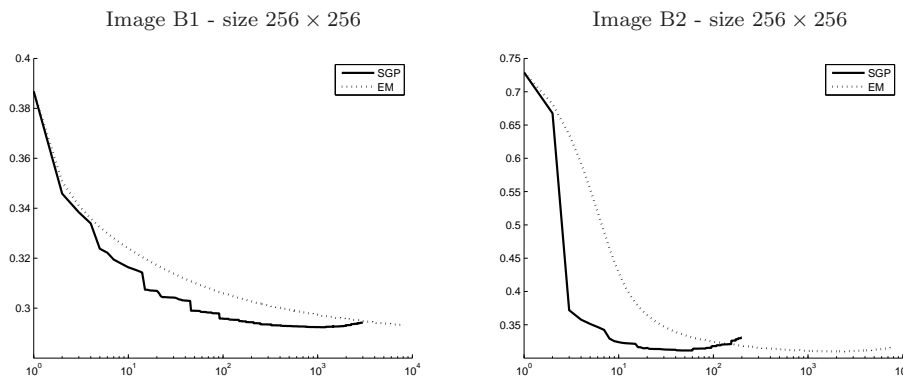


Fig. 2 EM and SGP relative reconstruction error.

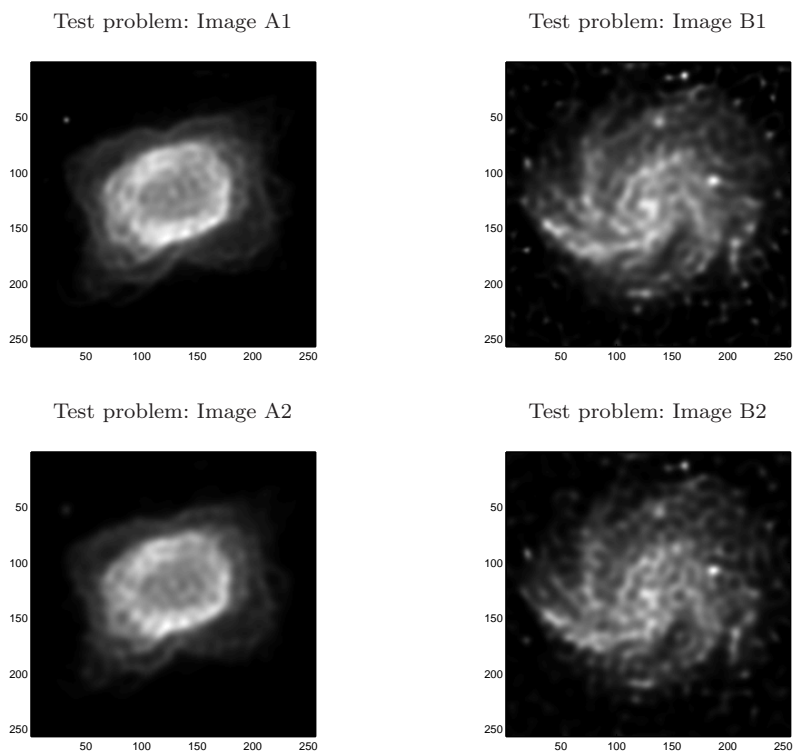


Fig. 3 Reconstructed images.

error (err.) and the computational time in seconds (time) corresponding to the two implementations. Even if the Matlab implementation is based on very well optimized functions for computing FFTs and other basic operations within the SGP iteration, the C version is able to achieve remarkable time reductions and appears definitely preferable. For these reason, the speedups corresponding to the parallel versions of the

algorithms will be computed with respect to the serial C version. The numerical results provided by the C_CUDA parallel implementations are summarized in Tables 2-5.

Table 1 C and Matlab implementations on CPU

Test problem	Algorithm	n	CPU (C-based)		CPU (Matlab-based)	
			err.	time	err.	time
A1	SGP it. = 260	256^2	0.052	4.78	0.051	21.09
		512^2	0.051	20.33	0.051	86.59
		1024^2	0.051	94.00	0.051	342.50
		2048^2	0.051	421.58	0.051	1418.27
A2	SGP it. = 29	256^2	0.070	0.72	0.071	2.47
		512^2	0.065	2.69	0.064	9.92
		1024^2	0.064	10.66	0.062	39.56
		2048^2	0.064	49.81	0.062	163.61

The main conclusion that can be drawn from these experiments is that the GPU implementations allow us to save time over the CPU implementation for more than one order of magnitude, without no significant differences in the reconstruction accuracy. The speedups are between 10.6 and 20.9 for the SGP and between 21.9 and 30.6 for the EM; this confirms that the additional operations required in each SGP iteration make this algorithm less suited than EM for a parallel implementation on GPU. However, it is important to observe that SGP, due to its better convergence rate, largely outperforms EM also in the GPU environment. In fact, the relative time saving of SGP over EM is between 79% and 95% for the tests on CPU and between 56% and 93% for the GPU implementations. Thus, taking also into account that the time reduction provided by SGP increases for increasing size of the problem, we may consider SGP an useful tool for solving large scale image deblurring problems on modern GPU parallel devices, more suited than the simple EM approach.

5 Conclusions

We have presented parallel versions of iterative regularization methods for solving constrained image deblurring problems on graphics processors. We have considered the expectation maximization method, a very simple but slowly convergent approach, and a recent scaled gradient projection method that exhibits superior convergence rate but a more expensive iteration. Parallel implementations of these algorithms for NVIDIA GPUs have been developed in a mixed C and CUDA language. A computational study on deblurring problems in astronomical imaging has shown that, for both the algorithms, the GPU implementation provides a time saving of at least one order of magnitude with respect to the CPU version. Concerning the relative performance of the two iterative approaches, the scaled gradient projection scheme largely outperforms the expectation maximization method on both the CPU and GPU environments. Thus, the GPU implementation of the gradient projection approach proposed in this paper

Table 2 Test problem: Image A1

Algorithm	n	CPU (C-based)		GPU (C_CUDA-based)		Speedup
		err.	time	err.	time	
SGP it. = 260	256 ²	0.052	4.78	0.052	0.45	10.6
	512 ²	0.051	20.33	0.052	1.64	12.4
	1024 ²	0.051	94.00	0.052	5.99	15.7
	2048 ²	0.051	421.58	0.052	27.72	15.2
EM it. = 8000	256 ²	0.051	66.91	0.051	3.06	21.9
	512 ²	0.051	311.31	0.051	14.09	22.1
	1024 ²	0.051	1533.28	0.051	57.72	26.6
	2048 ²	0.051	8360.94	0.051	368.27	22.7

Table 3 Test problem: Image B1

Algorithm	n	CPU (C-based)		GPU (C_CUDA-based)		Speedup
		err.	time	err.	time	
SGP it. = 750	256 ²	0.292	13.66	0.294	1.27	10.8
	512 ²	0.292	56.81	0.293	4.47	12.7
	1024 ²	0.291	268.86	0.293	16.20	16.6
	2048 ²	0.292	1206.92	0.293	76.91	15.7
EM it. = 8000	256 ²	0.293	67.02	0.293	3.05	22.0
	512 ²	0.293	314.02	0.293	14.02	22.4
	1024 ²	0.293	1551.00	0.293	57.43	27.0
	2048 ²	0.293	8148.89	0.293	366.09	22.3

represents an effective tool for large scale image deblurring and its generalization to other imaging problems will be a future work.

References

1. Barzilai, J., Borwein, J.M.: Two point step size gradient methods. *IMA J. Numer. Anal.* **8**, 141–148 (1988)
2. Bertero, M., Boccacci, P.: *Introduction to Inverse Problems in Imaging*. Institute of Physics Publishing, Bristol (1998)
3. Bonettini, S., Zanella, R., Zanni, L.: A scaled gradient projection method for constrained image deblurring. *Inverse Problems* **25**, 015002 (2009). Available: <http://cdm.unimo.it/home/matematica/zanni.luca/>
4. Csiszár, I.: Why least squares and maximum entropy? An axiomatic approach to inference for linear inverse problems. *Ann. Stat.* **19**, 2032–2066 (1991)
5. Dai, Y.H., Fletcher, R.: On the asymptotic behaviour of some new gradient methods. *Math. Programming* **103**(3), 541–559 (2005)
6. Dai, Y.H., Hager, W.W., Schittkowski, K., Zhang, H.: The cyclic barzilai-borwein method for unconstrained optimization. *IMA J. Numer. Anal.* **26**, 604–627 (2006)
7. Davis, P.J.: *Circulant Matrices*. John Wiley & Sons, (1979)
8. Frassoldati, G., Zanghirati, G., Zanni, L.: New adaptive stepsize selections in gradient methods. *J. Industrial and Management Optim.* **4**(2), 299–312 (2008)

Table 4 Test problem: Image A2

Algorithm	n	CPU (C-based)		GPU (C.CUDA-based)		Speedup
		err.	time	err.	time	
SGP it. = 29	256 ²	0.070	0.72	0.071	0.05	14.7
	512 ²	0.065	2.69	0.065	0.16	16.8
	1024 ²	0.064	10.66	0.064	0.58	18.4
	2048 ²	0.064	49.81	0.063	2.69	18.5
EM it. = 500	256 ²	0.070	4.41	0.071	0.19	23.2
	512 ²	0.064	19.91	0.064	0.89	22.4
	1024 ²	0.063	97.92	0.063	3.63	27.0
	2048 ²	0.063	523.03	0.063	23.05	22.7

Table 5 Test problem: Image B2

Algorithm	n	CPU (C-based)		GPU (C.CUDA-based)		Speedup
		err.	time	err.	time	
SGP it. = 45	256 ²	0.311	1.00	0.312	0.06	16.7
	512 ²	0.308	3.89	0.308	0.25	15.6
	1024 ²	0.307	18.39	0.307	0.88	20.9
	2048 ²	0.306	76.19	0.306	4.22	18.1
EM it. = 1475	256 ²	0.310	12.47	0.311	0.56	22.3
	512 ²	0.307	58.33	0.308	2.61	22.3
	1024 ²	0.306	326.64	0.307	10.66	30.6
	2048 ²	0.306	1542.97	0.306	67.94	22.7

9. Friedlander, A., Martínez, J.M., Molina, B., Raydan, M.: Gradient method with retards and generalizations. *SIAM J. Numer. Anal.* **36**, 275–289 (1999)
10. Galligani, E., Ruggiero, V., Zanni, L.: Parallel solution of large-scale quadratic programs. In: De Leone, R., Murli, A., Pardalos, P.M., Toraldo, G. (eds.) *High Performance Algorithms and Software in Nonlinear Optimization*, Applied Optimization 24, pp. 189-205. Kluwer Academic Publ., Dordrecht (1998)
11. Gergel, V.P., Sergeyev, Ya.D., Strongin, R.G.: A parallel global optimization method and its implementation on a transputer system. *Optimization* **26**, 261–275 (1992)
12. Harris, M.: Optimizing parallel reduction in CUDA. NVidia Tech. Rep. (2007). Available: <http://developer.download.nvidia.com/compute/cuda/1.1/Website/projects/reduction/doc/reduction.pdf>
13. Iusem, A.N.: Convergence analysis for a multiplicatively relaxed EM algorithm. *Math. Meth. Appl. Sci.* **14**, 573–593 (1991)
14. Iusem, A.N.: A short convergence proof of the EM algorithm for a specific Poisson model. *REBRAPE* **6**, 57–67 (1992)
15. Lange, K., Carson, R.: EM reconstruction algorithms for emission and transmission tomography. *J. Comp. Assisted Tomography* **8**, 306–316 (1984)
16. Lee, S., Wright, S.J.: Implementing algorithms for signal and image reconstruction on graphical processing units. Submitted (2008). Available: http://www.optimization-online.org/DB_HTML/2008/11/2131.html
17. Lucy, L.B.: An iterative technique for the rectification of observed distributions. *Astronom. J.* **79**, 745–754 (1974)
18. Migdalas, A., Toraldo, G., Kumar, V.: Parallel computing in numerical optimization. *Parallel Computing* **24**(4), 373-551 (2003)

-
19. Mülthei, H.N., Schorr, B.: On properties of the iterative maximum likelihood reconstruction method. *Math. Meth. Appl. Sci.* **11**, 331–342 (1989)
 20. NVIDIA: NVIDIA CUDA Compute Unified Device Architecture, Programming Guide. Version 2.0 (2008). Available at: http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide-2.0.pdf
 21. Richardson, W.H.: Bayesian-based iterative method of image restoration. *J. Opt. Soc. Amer. A* **62**, 55–59 (1972)
 22. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. *Physica D* **60**, 259–68 (1992)
 23. Serafini, T., Zanghirati, G., Zanni, L.: Gradient projection methods for quadratic programs and applications in training support vector machines. *Optim. Meth. Soft.* **20**(2-3), 343–378 (2005)
 24. Shepp, L.A., Vardi, Y.: Maximum likelihood reconstruction for emission tomography. *IEEE Trans. Med. Imaging* **1**, 113–122 (1982)
 25. Strongin, R.G., Sergeyev, Ya.D.: Global multidimensional optimization on parallel computer. *Parallel Computing* **18**, 1259–1273 (1992)
 26. Strongin, R.G., Sergeyev, Ya.D.: *Global optimization with non-convex constraints: sequential and parallel algorithms*. Kluwer Academic Publ., Dordrecht (2000)
 27. Vardi, Y., Shepp, L.A., Kaufman, L.: A statistical model for positron emission tomography. *J. Amer. Statist. Soc.* **80**(389), 8–37 (1985)
 28. Zanni, L.: An improved gradient projection-based decomposition technique for support vector machines. *Comput. Management Sci.* **3**, 131–145 (2006)
 29. Zanni, L., Serafini, T., Zanghirati, G.: Parallel software for training large scale support vector machines on multiprocessor systems. *Journal of Machine Learning Research* **7**, 1467–1492 (2006)
 30. Zhou, B., Gao, L., Dai, Y.H.: Gradient methods with adaptive step-sizes. *Comput. Optim. Appl.* **35**(1), 69–86 (2006)
 31. Zhu, M., Chan, T.F.: An efficient primal-dual hybrid gradient algorithm for total variation image restoration. CAM Report 08-34, Mathematics Department, UCLA (2008).