Fast tomographic reconstruction on parallel hardware

Gábor Takács*, Zoltán Horváth* and Gábor Veres†

*Széchenyi István University, Győr, Hungary, {gtakacs, horvathz}@sze.hu
†KFKI Research Institute for Particle and Nuclear Physics, Budapest, Hungary, veres@rmki.kfki.hu

Abstract.
Tomographic reconstruction is the mathematical procedure of approximating a function f, based on the integrals of f along a set of line sections. The need for fast tomographic reconstruction arises for example in the challenging problem of real time control of some plasma parameters in a fusion reactor. In this paper, we present a fast algorithm for tomographic reconstruction. A good property of our approach is that it fits well to hardware with two levels of parallelism (e.g. a GPU cluster). We also propose an objective evaluation method for measuring the quality of reconstruction on real datasets where f is unknown. We will demonstrate that our algorithm is able to perform more than 50 000 reconstructions per second at reasonably good quality, running on a relatively cheap hardware.

Keywords: tokamak, tomography, parallelization
PACS: 02.60.Dc, 42.30.Wb, 52.55.Fa

INTRODUCTION

Tomographic reconstruction is the mathematical procedure of approximating a function, based on the integrals of the function along a set of line sections. An interesting area where fast and accurate tomographic reconstruction is needed is the challenging and yet not satisfactorily solved task of real time control of some plasma parameters in a tokamak fusion reactor (see e.g. [1] for further information).

A possible formalization of the tomographic reconstruction problem is the following. Assume that $f : \mathbb{R}^d \mapsto \mathbb{R}$ is an unknown function called *image*, and $\tau : \left(\mathbb{R}^d \mapsto \mathbb{R}\right) \mapsto \mathbb{R}^I$ is a known operation that calculates the scaled line integrals of f along a given set of lines. The formula of the i-th output of τ is $[\tau(f)]_i = e_i \int_{L_i} f(\mathbf{x})ds$, where L_i is a line section in $\mathbb{R}^d$, and $e_i \in \mathbb{R}$ is the scaling factor corresponding to the i-th integral ($i = 1\ldots,I$). The vector $\tau(f)$ is called the *tomogram* of f. Assume that $\varepsilon : \left(\mathbb{R}^d \mapsto \mathbb{R}\right)^2 \mapsto \mathbb{R}$ is a known function called *error measure*. The task is to approximate f by $\hat{f}$ based on $\tau(f)$, so that $\varepsilon(f,\hat{f})$ is small.

An important variant of the problem is *batch reconstruction*, where a sequence of functions $f_1,\ldots,f_K$ has to be approximated on the basis of $\tau(f_1),\ldots,\tau(f_K)$. The task is almost always batch reconstruction in real world cases, since it is impractical to build a tomographic device for the reconstruction of a single image.

In the field of plasma tomography, some known approaches for computing $\hat{f}$ are analytical methods (e.g. [2, 3, 4]) and pixel based methods (e.g. [5, 6]). A common error measure is the relative sum squared error that can be defined as $\varepsilon_{\mathrm{RSSE}}(\hat{f},f) = \sum_{j=1}^{J}\left(\hat{f}(\mathbf{x}_j) - f(\mathbf{x}_j)\right)^2 / \sum_{j=1}^{J}\left(f(\mathbf{x}_j)\right)^2$, where $\mathbf{x}_1,\ldots,\mathbf{x}_J$ are sampling points, usually located regularly in a region of interest.

In this paper, we present a basis function based approach for tomographic reconstruction. Our approach can be viewed as a generalization of pixel based methods. A good property of our approach is that it fits well to hardware with two levels of parallelism (e.g. a cluster of multi-core CPUs or a cluster of GPUs).

We also propose an error measure that assesses the quality of reconstruction without using the values of f. Error measures that require f are not applicable in real environments, since f is unknown there. Our error measure does not have this limitation. We will demonstrate that our approach is able to perform more than 50 000 reconstructions per second at reasonably good quality, running on a relatively cheap hardware.

THE PROPOSED APPROACH

A possible strategy for solving the tomographic reconstruction problem is the basis function approach. Assume that we have a set of functions $\{g_j : \mathbb{R}^d \mapsto \mathbb{R},\ j = 1,\ldots,J\}$ called *basis functions*. In the basis function approach the

reconstruction formula is

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{J} w_j g_j(\mathbf{x}), \tag{1}$$

where the $w_j \in \mathbb{R}$ values called *weights* are the parameters of the model. The goal is to set $\mathbf{w} = [w_1, \ldots, w_J]^T$ so that $\tau(\hat{f})$ is similar to $\tau(f)$ and $\hat{f}$ is "simple" in a sense. This requirement can be formalized as the minimization of the regularized loss function

$$\delta(\mathbf{w}) = \sum_{i=1}^{n} \left(F_i - \sum_{j=1}^{d} w_j G_{ij} \right)^2 + \rho(\mathbf{w}), \tag{2}$$

where $F_i = [\tau(f)]_i$, $G_{ij} = [\tau(g_j)]_i$, and $\rho : \mathbb{R}^J \mapsto \mathbb{R}$ is called the regularizer. In plasma tomography some common choices for ρ are Tikhonov and minimum Fisher regularizers [5].

In this work, we considered pixel and spherical Gaussian basis functions, arranged in a rectangular grid. The pixel basis function corresponding to grid cell C returns 1 for input $\mathbf{x}$, if $\mathbf{x} \in C$, and returns 0 otherwise. The formula of the Gaussian basis function corresponding to cell C is $g(\mathbf{x}) = \exp\left(-(\mathbf{x}-\mathbf{c})^T(\mathbf{x}-\mathbf{c})/(2r^2)\right)$, where $\mathbf{c}$ is the centroid of C and r is the half diagonal length of C. The basis function set also contained a constant 1 function in both cases.

We applied Tikhonov regularization $\rho(\mathbf{w}) = \lambda \mathbf{w}^T \mathbf{w}$, $\lambda > 0$, meaning that large weight absolute values are penalized. In our case, minimizing δ is equivalent with solving the system of linear equations $\left(\mathbf{G}^T\mathbf{G} + \lambda \mathbf{I}\right)\mathbf{w} = \mathbf{G}^T\mathbf{f}$, where $\mathbf{G}$ is the I-by-J matrix of G_{ij} values, $\mathbf{f}$ is the I-by-1 vector of F_i values, and $\mathbf{I}$ is a J-by-J identity matrix. Note that $\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I}$ is determined by the line integrals of the basis functions, and does not depend on the image. Therefore, in the case of batch reconstruction the task is to solve K systems with the same coefficient matrix.

Implementation

In this subsection, we present some details of the implementation, using low level building blocks. Besides standard matrix operations we need only three simple linear algebraic routines:

- chol($\mathbf{A}$): Compute the Cholesky decomposition of the symmetric matrix $\mathbf{A}$, and return the lower Cholesky-factor.
- $\mathbf{L}\backslash_F\mathbf{b}$: Solve the lower triangular system $\mathbf{L}\mathbf{x} = \mathbf{b}$ with forward substitution, and return the solution.
- $\mathbf{L}^T\backslash_B\mathbf{b}$: Solve the upper triangular system $\mathbf{L}^T\mathbf{x} = \mathbf{b}$ with backward substitution, and return the solution.

We will also use the notation $\mathbf{L}\backslash_F\mathbf{B}$ and $\mathbf{L}^T\backslash_B\mathbf{B}$, meaning that the system is solved for each column of $\mathbf{B}$ and the solution column vectors are arranged into a matrix. Note that if $\mathbf{A}$ and $\mathbf{L}$ are n-by-n, then the (sequential) computational complexity of the chol($\mathbf{A}$) is $O(n^3)$, while the complexity of $\mathbf{L}\backslash_F\mathbf{b}$ and $\mathbf{L}\backslash_B\mathbf{b}$ is $O(n^2)$.

The reconstruction of a singe image can be done as follows:

$$\mathbf{L} \leftarrow \text{chol}(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I}), \qquad \mathbf{w} \leftarrow \mathbf{L}^T\backslash_B\left(\mathbf{L}\backslash_F\left(\mathbf{G}^T\mathbf{f}\right)\right).$$

If the task is to reconstruct K images instead of a singe one, then it is worthwhile to compute the Cholesky decomposition before looping over the K images. In a parallel environment one can calculate the reconstruction of the K images independently (the pseudocode of this variant can be seen in Algorithm 1).

However, this solution does not fit well to a two level parallel environment, because the operations $\backslash_F$ and $\backslash_B$ are not easily parallelizable [1]. For such environments it is better to calculate the inverse of $\mathbf{A}$ before the loop (using chol, $\backslash_F$, and $\backslash_B$), and apply parallel matrix–vector multiplications in the loop (see Algorithm 2).

The sequential computational complexity is $O\left(J^2(I+J+K)\right)$ for both algorithms. In a sequential environment Algorithm 1 is a bit faster than Algorithm 2, since it does not require the calculation of $\mathbf{A}^{-1}$ (and the cost of the loop is nearly equal in the two cases).

In a parallel environment the two algorithms can have about the same speed, if certain conditions are met (K is large enough, there are enough parallel processing units, the implementation is done carefully). However, Algorithm 2 is easier to implement than Algorithm 1, because parallelizing the matrix–vector multiplication is easier than parallelizing the loop of Algorithm 1. Another advantage of Algorithm 2 is that it fits particularly well to two level parallel environments.

---

[1] And implementing the loop on the low level processing units can be difficult

$\mathbf{A} \leftarrow \mathbf{G}^T\mathbf{G} + \lambda\mathbf{I}$; // matrix-matrix multiplication
$\mathbf{L} \leftarrow \text{chol}(\mathbf{A})$; // Cholesky decomposition
for $k \leftarrow 1$ **to** K **do in parallel**
  $\mathbf{b} \leftarrow \mathbf{G}^T\mathbf{f}_k$; // matrix-vector multiplication
  $\mathbf{w}_k \leftarrow \mathbf{L}^T\backslash_B (\mathbf{L}\backslash_F \mathbf{b})$; // forward & backward substitution
end

Algorithm 1: Parallel batch reconstruction.

$\mathbf{A} \leftarrow \mathbf{G}^T\mathbf{G} + \lambda\mathbf{I}$; // matrix-matrix multiplication
$\mathbf{L} \leftarrow \text{chol}(\mathbf{A})$; // Cholesky decomposition
$\mathbf{A}^{-1} \leftarrow \mathbf{L}^T\backslash_B (\mathbf{L}\backslash_F \mathbf{I})$; // forward & backward substitution J times
for $k \leftarrow 1$ **to** K **do in parallel**
  $\mathbf{b} \leftarrow \mathbf{G}^T\mathbf{f}_k$; // parallel matrix-vector multiplication
  $\mathbf{w}_k \leftarrow \mathbf{A}^{-1}\mathbf{b}$; // parallel matrix-vector multiplication
end

Algorithm 2: Two level parallel batch reconstruction.

EXPERIMENTS

We tested the proposed approach on artificial images (called phantoms) and on a real tomogram sequence (called shot) originating from the TCV tokamak. In the TCV tokamak the dimension of the images is $d = 2$ and the number of line integrals (channels) is $I = 140$. The geometry of the tokamak and the line segments corresponding to the channels can be seen in Figure 1(a). For reconstruction we used $J = 432$ pixel or Gaussian basis functions, arranged in 36-by-12 rectangular grid over the bounding rectangle of the tokamak's geometry.

Evaluation

Measuring the quality of reconstruction in the case of real tomograms is not a trivial problem, since the image f is unknown. A possible solution that we propose here is to apply an L-fold cross validation procedure. This means that we partition the channels randomly into L nearly equally sized parts, and run L experiments. In the l-th experiment we do not use the channels of the l-th part for reconstruction, but we use them for validating the components of $\tau(\hat{f})$ corresponding to the l-th part.

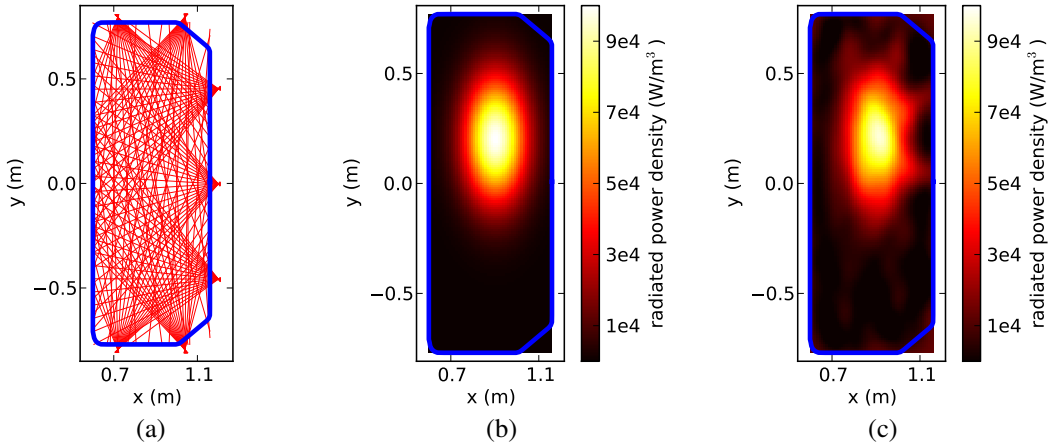

FIGURE 1. The geometry of the TCV tokamak (a), a phantom (b), and its reconstructed version (c).

TABLE 1. Reconstruction quality with different settings.

λ	$\varepsilon_{\mathrm{CV}}^{\mathrm{pixel}}$	$\varepsilon_{\mathrm{CV}}^{\mathrm{Gaussian}}$	$\varepsilon_{\mathrm{CV}}^{\mathrm{Gaussian}}/\varepsilon_{\mathrm{CV}}^{\mathrm{pixel}}$
10^{-18}	0.30 ($\pm$0.14)	0.24 ($\pm$0.10)	0.90 ($\pm$0.47)
10^{-17}	0.19 ($\pm$0.08)	0.13 ($\pm$0.07)	0.81 ($\pm$0.54)
10^{-16}	0.21 ($\pm$0.13)	0.12 ($\pm$0.09)	0.61 ($\pm$0.27)
10^{-15}	0.31 ($\pm$0.15)	0.19 ($\pm$0.14)	0.60 ($\pm$0.14)

Assuming batch reconstruction, the relative sum squared error of the l-th experiment is calculated as

$$\varepsilon_l = \frac{\sum_{k=1}^{K} \sum_{i \in I_l} \left([\tau(\hat{f}_k)]_i - [\tau(f_k)]_i \right)^2}{\sum_{k=1}^{K} \sum_{i \in I_l} \left([\tau(f_k)]_i \right)^2}, \tag{3}$$

where the set I_l contains the channel indices of the l-th part ($l = 1, \ldots, L$). The proposed error measure is the average of the numbers $\varepsilon_1, \ldots, \varepsilon_L$ (denoted by $\varepsilon_{\mathrm{CV}}$), supplemented with the empirical standard deviation of the numbers.

The above cross validation procedure does not compare $\hat{f}$ with f directly, but it compares $\tau(\hat{f})$ with $\tau(f)$ in a fair way. In a sense, a low $\varepsilon_{\mathrm{CV}}$ error means that the reconstruction method is able to predict the values of new channels.

Results

The result of our reconstruction method on a phantom (using Gaussian basis functions and $\lambda = 10^{-18}$) is shown in Figures 1(b)-(c). We also tested our method on a real shot containing $K = 200\,000$ tomograms. The 10-fold cross validation errors with different regularization coefficients and basis functions can be seen in Table 1.

$\varepsilon_{\mathrm{CV}}^{\mathrm{Gaussian}}$ was less than the corresponding $\varepsilon_{\mathrm{CV}}^{\mathrm{pixel}}$ in all cases. The parentheses contain the (unbiased) empirical standard deviation of the quantities. The standard deviation of $\varepsilon_{\mathrm{CV}}$ is often quite high, meaning that prediction was much more difficult for some channel subsets than for others. However, it can be seen in the last column that the ratio $\varepsilon_{\mathrm{CV}}^{\mathrm{Gaussian}}/\varepsilon_{\mathrm{CV}}^{\mathrm{pixel}}$ was always significantly less than 1, which means that if a subset of channels was more difficult for the Gaussian basis function method, then it was typically more difficult for the pixel basis function method too. Therefore, it is rightful to say that the Gaussian variant performed better than the pixel variant.

We also measured the reconstruction speed of our method (Algorithm 2). Here we assumed that matrix $\mathbf{G}$ and vectors $\mathbf{f}_1, \ldots, \mathbf{f}_K$ are in the main memory, and calculated K/T, where T is the wall clock time required to produce $\mathbf{w}_1, \ldots, \mathbf{w}_K$ in the main memory. In the first experiment used a single CPU (Intel Pentium E2200) core. The reconstruction speed was $\sim$800 images / second. In the second experiment we accelerated the matrix–vector multiplications of Algorithm 2 by a GPU (NVIDIA GeForce GTX 280). The reconstruction speed was $\sim$56 000 images / second, which means a speedup factor of 70.

ACKNOWLEDGMENTS

The authors' research was supported by the National Development Agency and the European Union within the frame of the project TAMOP 4.2.2-08/1-2008-0021 at the Széchenyi István University, entitled "Simulation and Optimization – basic research in numerical mathematics".

REFERENCES

1. J. P. Freidberg, *Plasma Physics and Fusion Energy*, Cambridge University Press, 2007.
2. A. M. Cormack, *Journal of Applied Physics* **35**, 2908–2913 (1964).
3. T. P. Ma, L. Q. Hu, and K. Y. Chen, *Physics Letters A* **372**, 6187–6192 (2008).
4. P. J. Carvalho, H. Thomsen, R. Coelho, P. Duarte, C. Silva, and H. Fernandes, *Fusion Engineering and Design* **85**, 266–271 (2010).
5. M. Anton, H. Weisen, M. J. Dutch, W. von der Linden, F. Buhlmann, R. Chavan, B. Marletaz, P. Marmillod, and P. Paris, *Plasma Physics and Controlled Fusion* **38**, 1849–1878 (1996).
6. A. K. Chattopadhyay, A. Anand, and C. V. S. Rao, *Review of Scientific Instruments* **76** (2005).