GPU Implementation of Levenberg-Marquardt Optimization for T_1 Mapping

Shufang Liu[1,2,3], Aurélien Bustin[1,2,3], Darius Burshka[1], Anne Menini[2], Freddy Odille[3,4]

[1] Technische Universität München, München, Germany
[2] GE Global Research Center, München, Germany
[3]IADI, INSERM U947 and Université de Lorraine, Nancy, France
[4]CIC-IT 1433, INSERM, CHRU Nancy and Université de Lorraine, Nancy, France

Abstract

T_1 mapping is an emerging MRI research tool to characterize diseased myocardial tissue. The T_1 map is generated by fitting an exponential relaxation curve to the acquired image data. Levenberg-Marquardt algorithm is a standard way to solve this nonlinear curve fitting problem. However, the execution on the standard CPU can be time-consuming and incompatible with clinical routine. In this paper, a GPU implementation is performed to reduce the computation time of the standard T_1 mapping. In addition, a new vectorized approach is proposed to include spatial regularization in the curve fitting process to improve the robustness. The GPU implementation is validated on NVIDIA K42000 GPU using cardiac T_1 data from 16 volunteers. The computation time shows significant decrease in both pixel-wise and vectorized curve fitting. The pixel-wise curve fitting is accelerated by a factor of 30+ compared to the standard sequential C code and the vectorized curve fitting is improved by a factor of 47 and 38 for 3-parameter and 2-parameter curve fitting compared to the Matlab code.

1. Introduction

T_1 mapping is an emerging MRI technique to distinguish the diseased myocardial tissue from the normal tissue. Compared to conventional protocols such as late-enhancement imaging, T_1 mapping is a quantitative measurement that does not depend on the MR system. This tool has gained more and more attention in recent years[1–3].

A typical T_1 map is generated by fitting an exponential model to a sequence of images for each pixel. The image signal is first flipped by an external radio frequency pulse, and then recovered to its equilibrium state. The relaxation process can be modeled by an exponential curve and T_1 is the relaxation constant of this exponential curve. There are different MR sequences to generate the images, named as inversion recovery (IR)[4] and saturation recovery (SR)[5]. The difference between these two sequences is that the signal is flipped by 180 degree for IR and 90 degree for SR. Usually 8 images are acquired for SR and 11 images are acquired for IR. IR techniques such as MOLLI are more widely used due to their larger dynamic range and better precision (i.e. reproducibility), whereas SR techniques such as SASHA or SMART1Map provide more accurate T_1 values [6]. Different models have been proposed, a 2-parameter model showing more stable curve fitting but lower accuracy, whereas a 3-parameter model is used to correct for the inhomogeneity of the initial flip angle [7].

Due to the different imaging protocols, different formulas have been used to describe the relaxation curve. Table 1 gives a summary of the models. More detailed descriptions can be found in [7].

Table 1. Summary of T_1 relaxation model

	N_p	Model
Saturation-recovery	3	$y_t = A * \left(1 - e^{-\frac{t}{T}}\right) + C$
	2	$y_t = A * \left(1 - e^{-\frac{t}{T}}\right)$
Inversion-recovery	3	$y_t = \left\lvert A * \left(1 - e^{-\frac{t}{T}}\right) + C \right\rvert$
	2	$y_t = \left\lvert A * \left(0.5 - e^{-\frac{t}{T}}\right) \right\rvert$

The current implementation of the curve fitting can take a long time. For a typical T_1 series of 8 images, the pixel-wise curve fitting algorithm implemented on Matlab usually take several minutes. A notable delay is observed. Thus either a mask or a ROI is used to select the pixels that are clinically relevant, which results in a reduction of the computation time. A vectorized version of the algorithm is also proposed, which is aimed to include a spatial regularization term in the Levenberg-Marquardt optimization to improve the quality of resulting T_1 maps.

The GPU architecture is characterized by the single instruction multiple data (SIMD) feature, which allows operations on multiple data simultaneously. Nowadays, more and more applications achieve better performance by taking the advantage of the GPU architecture. In the pixel-wise curve fitting, each pixel can be fitted individually, which is ideal for GPU. For the vectorized curve fitting with spatial regularization, the fitting of each pixel is no longer independent so the algorithm needs to be adapted. The essential part is to solve a large matrix inversion problem, which involves matrix multiplications and vector

 ISSN: 2325-887X DOI:10.22489/CinC.2017.003-070

operations, which can also fit GPU.

In this paper, we present the GPU implementation of the pixel-wise and vectorized Levenberg-Marquardt algorithms using the NVIDIA CUDA framework. Section 2 introduces the mathematical background of the optimizer. Section 3 describes the detail of the GPU implementation. Section 4 shows the performance of the GPU implementation.

2. Mathematical Background

2.1. Levenberg-Marquardt Algorithm

LM Algorithm [8–10] is a popular choice to solve non-linear curve fitting problems. In our case, we consider N_m measurements for each pixel, $N_m = 8$ for SR and $N_m = 11$ for IR. The unknown model can be denoted as $f(t;p)$, where p is the parameter set with N_p unknowns and t is the inversion time. In our case N_p= 2 or 3. Then the problem can be formulated as

$$min_p \|f(\cdot;p) - y\|^2 \quad (1)$$

To solve this problem, an iterative method is employed. Starting from an initial guess of the parameters p_o, the method consists of searching for an optimal refinement of the parameters δp, by linearizing the cost function around the current estimate:

$$\min_{\delta p} \|f(\cdot;p+\delta p) - y\|^2 = \min_{\delta p} \left\|J(p)\delta p - \left(y - f(\cdot;p)\right)\right\|^2 \quad (2)$$

where $J(p)$ is the Jacobian matrix of f with respect to the parameters, evaluated at the current guess p. $J(p)$ is a $N_m \times N_p$ matrix. Therefore LM involves solving a sequence of linear least squares problems using a regularized inversion of the Jacobian matrix in order to calculate the following update of the solution at a given iteration k:

$$\delta p = p_{k+1} - p_k = \left(J(p_k)^T J(p_k) + \lambda_k Id\right)^{-1} J(p_k)^T \left(y - f(p_k)\right) \quad (3)$$

Where Id is the identity matrix and λ_k is the LM regularization coefficient. This regularization coefficient is adapted throughout iterations. The rationale of LM is to start with a large value of λ_k, so the method behaves like a steepest gradient descent in the first iterations, then to decrease $\lambda_k$as p approaches the solution, so the method behaves like a quasi-Newton method in the last iterations. Such schemes are thought to yield optimal convergence speed in the nonlinear optimization literature. Several variations of the LM technique have been proposed depending on the choice of λ_0, update rule for λ_k and stopping condition. Here we choose the following update rule [10]:

$$\begin{cases} if\ \rho(p) > \varepsilon, set\ p_{k+1} = p_k + \delta p, \lambda_{k+1} = \min(\lambda_k \times 2, 10^7) \\ if\ \rho(p) \le \varepsilon, do\ not\ update\ p_k, \lambda_{k+1} = \max(\lambda_k/2, 10^{-7}) \end{cases}$$

$$with\ \rho(p) = \frac{\|f(p)\|^2 - \|f(p+\delta p)\|^2}{\|f(p)\|^2 - \|f(p) + J(p)\delta p\|^2}$$ and iterations are stopped when $\|p_{k+1} - p_k\| / \|p_{k+1}\| < \tau$, with τ a given tolerance, or when a maximal number of iterations was reached.

2.2. Vectorized Levenberg-Marquardt Algorithm

In this section, we vectorize the LM algorithm by grouping the same operations that are performed multiple times on different data into a single operation performed once on a large array of data. It is therefore possible to incorporate constraints such as spatial smoothness to improve the processing itself.

In order to formulate the vectorized version of LM for an image of N_{pix} pixels, we use the same algorithm framework as described in the previous section. However we redefine y to be the whole acquired T1-weighted dataset, a vector of $N_{pix}N_m$ elements, $y = \{y_{1,1}, y_{1,2}, \dots, y_{i,k}\}$, in which $y_{i,k}$ is the k-th acquisition of pixel i. p the concatenated parameter maps, a vector of $N_{pix}N_p$ elements, $p = \{p_{1,1}, p_{2,1}, \cdots, p_{i,m}\}$ where $p_{i,m}$ is the m-th parameter of pixel i and f is the fitting model function operating on images. Adding a spatial smoothness constraint leads to a vectorized version of Eq. (1):

$$\min_p \|f(p) - y\|^2 + \mu\|Gp\|^2 \quad (4)$$

where μ is a scalar controlling the spatial regularization weight and G is an operator returning a concatenation of the spatial gradients of each parameter map, computed by forward differences. G is a sparse matrix of size $N_{dim}N_{pix}N_p \times N_{pix}N_p$, with N_{dim} the number of dimensions in the image (here N_{dim}=2). This penalization term can reduce the local variations, thus making the method more robust to noise. The vectorized LM update formula becomes:

$$\delta p = \left(J(p_k)^T J(p_k) + \Lambda_k + \mu G^T G\right)^{-1} J(p_k)^T (y - f(p_k) - \mu G^T G p_k) \quad (5)$$

Note that the term $\lambda_k Id$ in Eq. (3) has been substituted by a diagonal matrix Λ_k, the diagonal elements of which contain the map of LM regularization coefficients at iteration k. $J(p_k)$ is now a large sparse matrix of size $N_{pix}N_m \times N_{pix}N_p$. The element in row $(i + k * N_{pix})$ corresponds to $y_{i,k}$, the intensity of pixel i at inversion time k. Since $y_{i,k}$ depends only on its own parameter $p_{i,m}(m \in [1, N_m])$, the $J(p_k)$ is a large sparse matrix, concatenated of several diagonal matrices. $J(p_k)^T J(p_k)$ is then a $N_{pix}N_p \times N_{pix}N_p$ maxtrix. For images of size 256x256, $J(p_k)^T J(p_k)$ is a 196608x196608 matrix for 3-parameter fitting and 131072x131072 matrix for 2-parameter fitting.

In contrast to the previous section, here the large sparse matrix inversion involved in Eq. (5) can be solved efficiently using iterative methods (as shown in section 3.2). It should be noted that the computational burden of such methods is mainly constrained by the application of the matrix operator to be inverted. Update rules for the LM regularization coefficient maps and stopping conditions are the same as in the pixel-wise case (note that the stopping condition now applies to the whole parameter map vector).

3. GPU implementation

3.1. Pixel-wise Levenberg-Marquardt

Here we use 16x16 block size and 16x16 grid size to cover the 256x256 image size. Each image pixel is processed independently by a thread.

3.2. Vectorized Levenberg-Marquardt

In the vectorized version, the same steps for LM algorithm are adopted, as shown in Figure 1. However, most of the work is done on the GPU, Figure 1. In this case, a large matrix J^TJ needs to be calculated. Since J^TJ is a block-wise diagonal matrix. The image intensity at pixel i only affect the value of J^TJ at $(i + m \cdot N_{pix}, i + n \cdot N_{pix})$ where $m, n \in \{0,1,..N_p - 1\}$.

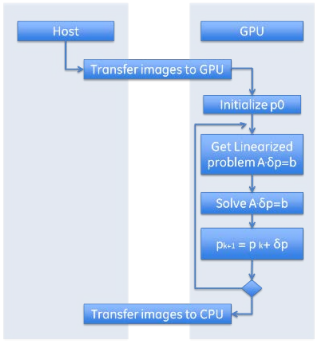

Figure 1. Workflow of vectorized curve fitting

The workload of the vectorized LM algorithm is mainly on the solving of the linear equation. The QR factorization method from the standard CUDA library 'cuSolver' was first tested. Then a conjugate gradient solver [11] was implemented to accelerate the computation. The conjugate gradient algorithm is an iterative way to solve the linear equation and is more suitable for large scale problems presented in this study. Our implementation relied on the CUDA cuBLAS library.

4. Experimental results

4.1 Data description

16 healthy volunteers were included in the study. The study protocol was approved by the ethics committee and written informed consent was obtained from all volunteers. For each volunteer, T1-weighted images were acquired using an SR acquisition technique (SMART1Map sequence). For 12 of them, data were also acquired using an IR acquisition technique (MOLLI sequence). All data were acquired with a General Electric 3T system (Signa HDxt, GE Healthcare, Milwaukee, USA). One short-axis slice was positioned in the mid-cavity and all acquisitions were synchronized to the ECG. Subjects were instructed to hold their breath during the acquisition, resulting in 8 (SR) to 11 (IR) T1-weighted images (256x256) being acquired.

4.2 Computing platform

All experiments were performed on a computer with Intel® Xeon CPU at 3.30GHz, 64G physical memory. The graphic card was a NVIDIA Quadro K4200, with 1344 cores, operating at 784MHz, a 256-bit memory interface, and a 4GB GDDR5 GPU memory with memory bandwidth of 173 GB/s.

4.3. Experiments

For pixel-wise curve fitting, the standard sequential c code and the CUDA code were both compiled as a "mex" file to be interfaced with Matlab (Mathworks, Natick, USA) and the C code was used as a reference for the GPU implementation.

For the vectorized curve fitting, initial experiments were done in Matlab, and the Matlab implementation is used as reference because the linear solver in Matlab is highly optimized. Then GPU was enabled by replacing the array variables by gpuArray ones in Matlab. Then the full CUDA implementation was used to further optimize the speed.

4.4 Parallel performance

Figure 2 shows an example of the T_1 maps of short-axis cardiac image, obtained using the vectorized curve fitting algorithm with different spatial regularization parameter. The T_1 maps get smoother as regularization increases.

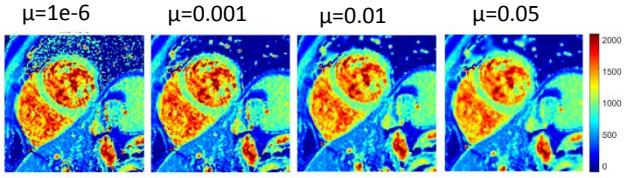

Figure 2. T_1 mapping with different regularization parameter from the vectorized curve fitting

For the pixel-wise curve fitting, we compared the C code (mex file) and the GPU implementation. Matlab is well known for its inefficiency at for-loop, thus it is unfair to compare to the Matlab implementation. Therefore, the Matlab mex file is used as a reference. Figure 3 (a) shows the computation time of each implementation. For one slice, the standard C implementation estimates the T_1 map in about 0.73±0.16 sec for 2 parameter fitting and 1.21±0.36 sec for 3 parameter curve fitting. The CUDA implementation cost 0.022±0.005 sec and 0.038±0.009 sec for 2- and 3-parameter respectively. The acceleration factor is 32 and 31.

For the vectorized curve fitting, the original Matlab implementation costs around 54.39±2.74 sec (3-parameter) and 31.74±4.4 sec (2-parameter) to generate one T_1 map. After replacing the array variables with gpuArray ones, which enables Matlab to calculate with GPU, the calculation time was reduced to 7.3±0.1 sec (3-parameter) and 4.2±0.7

sec (2-parameter). CUDA implementation further reduces the calculation time to 1.14±0.01 sec (3-parameter) and 0.84±0.02 sec (2-parameter). The acceleration ratio compared to the original Matlab implementation is 47 for 3 parameter case and 38 for 2 parameter case.

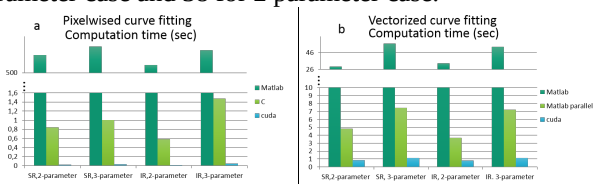

Figure 3. Computation time for T_1 mapping of 256x256 images, (a) pixel-wise fit; (b) vectorized fit

5. Discussion

In this paper, we implemented the Levenberg-Marquardt algorithm on GPU using CUDA framework. The GPU implementation significantly reduces the calculation time for T_1 mapping in both traditional pixel-wise solver and the new vectorized solver.

In the pixel-wise solver, each kernel solves the curve fitting problem for the corresponding pixel. The calculation time is the longest time among all the pixels. The data transfer only happens at the beginning, where images are copied to GPU, and at the end, where the estimated T_1 values are copied back to CPU. So the process is optimized.

The vectorized Levenberg-Marquadt cost longer time on GPU. However, the benefit of vectorized Levenberg-Marquardt algorithm is to include regularization in the curve fitting process, so that the stability could be improved by using the pixels from the neighborhood. In this vectorized Levenberg-Marquardt algorithm, simply calling the Matlab build-in GPU function could reduce the calculation time by a factor around 7, thus is a good choice for fast prototyping of new regularization schemes. More advanced edge-preserving regularization schemes (such as e.g. total variation[12] or Beltrami regularization[13]) may be of interest and could be investigated using the proposed framework. For larger datasets, e.g. larger image size, stacks of contiguous 2D slices or even 3D images, the calculation time will increase significantly, thus the full GPU implementation is feasible and more efficient.

6. Conclusion

With GPU implementation, the computation time for T_1 mapping can be significantly reduced. Further work includes investigation of advanced regularization schemes to improve the T_1 estimation and larger scale clinical evaluation to show the benefit of T_1 mapping in myocardial disease diagnosis.

Acknowledgements

This publication was supported by the European Community's Marie Curie Research Training Network Programme (grant number 605162). The content is solely the responsibility of the authors and does not necessarily represent the official views of the EU.

References

[1] L. Iles *et al.*, "Evaluation of Diffuse Myocardial Fibrosis in Heart Failure With Cardiac Magnetic Resonance Contrast-Enhanced T1 Mapping," *J. Am. Coll. Cardiol.*, vol. 52, no. 19, pp. 1574–1580, Nov. 2008.

[2] T. D. Karamitsos *et al.*, "Noncontrast T1 Mapping for the Diagnosis of Cardiac Amyloidosis," *JACC Cardiovasc. Imaging*, vol. 6, no. 4, pp. 488–497, Apr. 2013.

[3] V. M. Ferreira *et al.*, "Non-contrast T1-mapping detects acute myocardial edema with high diagnostic accuracy: a comparison to T2-weighted cardiovascular magnetic resonance," *J. Cardiovasc. Magn. Reson.*, vol. 14, no. 1, p. 42, 2012.

[4] D. R. Messroghli, A. Radjenovic, S. Kozerke, D. M. Higgins, M. U. Sivananthan, and J. P. Ridgway, "Modified Look-Locker inversion recovery (MOLLI) for high-resolutionT1 mapping of the heart," *Magn. Reson. Med.*, vol. 52, no. 1, pp. 141–146, Jul. 2004.

[5] C. K. al et, "Saturation recovery single-shot acquisition (SASHA) for myocardial T(1) mapping. - PubMed - NCBI." [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/23881866. [Accessed: 29-Aug-2016].

[6] P. Kellman and M. S. Hansen, "T1-mapping in the heart: accuracy and precision," *J. Cardiovasc. Magn. Reson.*, vol. 16, no. 1, pp. 1–20, 2014.

[7] P. Kellman, H. Xue, K. Chow, B. S. Spottiswoode, A. E. Arai, and R. B. Thompson, "Optimized saturation recovery protocols for T1-mapping in the heart: influence of sampling strategies on precision," *J. Cardiovasc. Magn. Reson.*, vol. 16, no. 1, Dec. 2014.

[8] K. LEVENBERG, "A METHOD FOR THE SOLUTION OF CERTAIN NON-LINEAR PROBLEMS IN LEAST SQUARES," *Q. Appl. Math.*, vol. 2, no. 2, pp. 164–168, 1944.

[9] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, Jun. 1963.

[10] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Numerical Analysis*, vol. 630, G. A. Watson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 105–116.

[11] A. V. Knyazev and I. Lashuk, "Steepest Descent and Conjugate Gradient Methods with Variable Preconditioning," *SIAM J. Matrix Anal. Appl.*, vol. 29, no. 4, pp. 1267–1280, Jan. 2008.

[12] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. Nonlinear Phenom.*, vol. 60, no. 1, pp. 259–268, Nov. 1992.

[13] D. Zosso and A. Bustin, "A primal-dual projected gradient algorithm for efficient Beltrami regularization," Tech. Report UCLA CAM Report 14-52, 2014.

Address for correspondence:
Freddy Odille
IADI (Inserm U947) - CIC-IT 1433
Bâtiment Recherche, CHRU de Nancy Brabois
54511 Vandoeuvre-lès-Nancy, FRANCE
E-mail: freddy.odille@inserm.fr