10.1515/msr-2015-0029

MEASUREMENT SCIENCE REVIEW, Volume 15, No. 4, 2015

GPU-Accelerated Reconstruction of T2 Maps in Magnetic Resonance Imaging

Jan Mikulka

Department of Theoretical and Experimental Electrical Engineering, Faculty of Electrical Engineering and Communication, Brno University of Technology, Technická 12, 616 00, Brno, Czech Republic, mikulka@feec.vutbr.cz

The main tissue parameters targeted by MR tomography include, among others, relaxation times T_1 and T_2. This paper focuses on the computation of the relaxation time T_2 measured with the Spin Echo method, where the sensing coil of the tomograph provides a multi-echo signal. The maxima of these echoes must be interleaved with an exponential function, and the T_2 relaxation can be determined directly from the exponential waveform. As this procedure needs to be repeated for each pixel of the scanned tissue, the processing of large images then becomes very intensive. For example, given the common resolution of 256x256 with 20 slices and five echoes at different times T_E, it is necessary to reconstruct $1.3 \cdot 10^6$ exponential functions. At present, such computation performed on a regular PC may last even several minutes. This paper introduces the results provided by accelerated computation based on parallelization and carried out with a graphics card. By using the simple method of linear regression, we obtain a processing time of less than 36 ms. Another effective option consists in the Levenberg-Marquardt algorithm, which enables us to reconstruct the same image in 96 ms. This period is at least 900 times shorter than that achievable with professional software. In this context, the paper also comprises an analysis of the results provided by the above-discussed techniques.

Keywords: CUDA; Spin Echo; T_2 relaxation; magnetic resonance imaging.

1. INTRODUCTION

MAGNETIC RESONANCE IMAGING (MRI) is currently one of the most advanced diagnostic techniques. In this method, the resulting image is generated via excitation of hydrogen nuclei in the examined tissue and, subsequently, sensitive response detection. A large number of imaging sequences are available to display different tissue properties. The most common tomographically acquired parameters include relaxation times T_1 (spin-lattice) and T_2 (spin-spin). Although both these times are identical in pure water, they can be advantageously applied for the recognition of biological tissues, whose T_2 is less than T_1. The more tissue parameters are known, the higher the possibility of their recognition or the correct diagnosis of a pathology. In this context, it is interesting to note that the multiparametric analysis [1] of diseased tissues became a fundamental factor to influence the concept of fast computation of T_2-based tissue maps [2], [3].

The relaxation time T_2, whose reconstruction is discussed in this paper, can be acquired by means of various sequences, including the Spin Echo (SE) and the Gradient Echo (GE) methods. The frequently applied SE approach [4] utilizes two radio frequency pulses with phases of 90° and 180°. After the former pulse, the protons' magnetization vector is flipped into the x-y plane, and T_2 relaxation begins to show; some protons thus precess at slightly higher frequencies, others at lower ones, and dephasing occurs. The latter (or the refocusing) pulse, however, flips the individual spins in the x-y plane to rephase them, and the receiving coil will detect a signal having an amplitude that depends on the T_2 relaxation of the tissue. This process repeats with the growing time T_E, where $T_E/2$ is the period between the pulses with the phases of 90° and 180°. The time sequence of the Spin Echo method is presented in Fig.1.

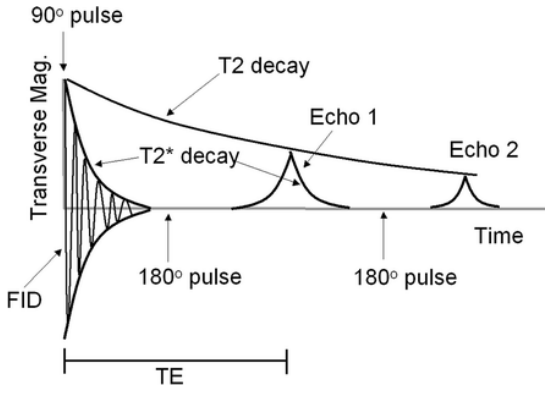

Fig.1. The time progression of the Spin Echo sequence and the T_2 exponential decay.

The relaxation time T_2 is determined from the waveform of the exponential function which passes through maximum values of the individual echoes. This function can be expressed as

$$E = e^{-\frac{T_E}{T_2}}, \tag{1}$$

where E is the envelope of the spin echoes, T_E is the time of the echo arrival from the first excitation pulse given by double of the time between the arrivals of the first and second excitation pulses.

The relaxation time T_2 is therefore determined via the following steps:

- acquisition of several images via the SE sequence with different times T_E;
- exponential regression at points located in the maxima of the individual echo signals;
- calculation of the T_2 relaxation.

This procedure is repeated for all pixels in the image. For the applied image with the size of 256x256 pixels and depth of 20 slices, we then have $1.3 \cdot 10^6$ exponential regressions, and there are 5 points from which the exponential function is reconstructed.

It follows from the outlined facts that the computing task is rather complex with respect to the data size. Generally, two solution options are available. The first (and the most common) approach consists of solving the task sequentially, which requires us to perform in every step exponential regression from the five measurements for each pixel. The second method is based on the parallelization of the task. The computation of the value of the relaxation time T_2 in each pixel depends only on the values obtained from the five measurements with the SE method, and therefore regression can be performed for all the pixels simultaneously. The parallel image processing technique then substantially reduces the time necessary to yield the actual result [5].

The reason for the proposed design of a method for fast computation of the T_2 relaxation time is the lack of a corresponding tool. At present, any extensive experimental research using the analysis of images obtained via MRI requires data correction and preprocessing. The computation of the relaxation times from the data acquired through imaging sequences constitutes the initial stage within both the image processing chain and the description of the examined biological tissues by means of real physical parameters [6]. In view of the presented problem, the aim of this paper is to introduce the procedure of T_2 map reconstruction utilizing fast graphics processors and various mathematical models. Moreover, the suggested approaches and solutions might also serve as inspiration and guidance for researchers intending to assemble or correct similar data processing methods.

The first section of the article comprises a survey of the following problems or aspects: state of the art in fast T_2 map reconstruction; database of MR test images; and the software/hardware used for the implementation of the fast reconstruction methods. In this connection, a detailed characterization of both the sequential and the parallel approaches to T_2 map computation is provided. The following chapter then compares the results obtained with the analyzed techniques and presents the related computation times, and the final part of the paper contains an overall evaluation of the problem.

The research in fast image processing methods is motivated by a major research track presently pursued at the DTEEE and partner institutes, where a large number of science projects involve the analysis of biological tissues. Recently, the procedure referred to as multiparametric analysis has been often applied; this technique is designed to view a tissue as a set of several parameter distributions, and therefore the tissue is not interpreted from the perspective of merely a single measured parameter. In current diagnostics, the most frequently examined images are those weighted by relaxation times T_1 and T_2 or by diffusion or perfusion coefficients; significant attention is also paid to derived images generated via defined calculation. The growing number of parameters increases the processing time and intensifies the computation, making necessary the search of suitable tools to either simplify or accelerate the actual processing. One of the projects in which the author of this paper currently participates in is based on investigating schizophrenia in mice. Within this project, sixty laboratory mice have been examined thus far, and images weighted by the T_1/T_2 relaxation times and by the diffusion coefficient were acquired from all the samples; the images weighted by the relaxation time T_2 were obtained via the SE imaging sequence. Given the size of the image database, it became obvious that a new tool for batch-based fast processing has to be created.

In recent years, graphics cards have become a popular instrument for parallel image processing [7], [8]. Methods enabling us to implement parallelization in image filtering [9], segmentation [10] and the otherwise very time-intensive registration [11], [12] are sufficiently described by several papers within the field of medical image processing. In MRI data processing, graphics cards are employed for a wide variety of computations. Sources [13] and [14], for example, then interestingly characterize the recent development and prospects of parallelization within medical data processing. Further, the problem of simple computation of the T_2 relaxation time is also analyzed in a number of articles and other sources, such as [15], [16], and [17]. Although the procedures presented in the indicated studies often utilize a simple linear model of the exponential function after logarithmic transformation, the processing time of the proposed techniques is not mentioned by the authors. However, time-intensive computation is expectable, given that the actual implementation was performed in the Matlab environment and assuming sequential execution of the program. The described regression function is also applied in our case, but the method principally prefers lower echo values. This drawback is discussed in references [18], [19]. Generally, we can point out that not many authors analyze the problem of employing parallelization to accelerate the computation of T_2; rather, most papers concentrate on the use of exponential fitting in concrete applications.

Several tools facilitating the computation of T_2 maps are also available, for example *MRI Processor* [20]. This software computes both T_1 and T_2 maps and is equipped with two approximation iterative algorithms, namely the Levenberg-Marquart and Simplex methods. The tool is conceived as a plug-in for the well-known environment *ImageJ*, and it supports the computation of data up to four dimensions. Another suitable software package to process tomographic data is *Marevisi* [21], which is designed specifically for MR data processing. The program facilitates the computation of relaxation times via the regression of a series of available exponential waveforms. The software is applicable only commercially.

2. SUBJECT & METHODS

The image resolution provided by experimental MR tomographs commonly ranges between 64x64 pixels and 512x512 pixels in a slice; three-dimensional imaging, however, enables us to obtain even 512^3 pixels. In multi-echo acquisition, the number of pixels must be also multiplied by the number of echoes, and as the reconstruction of T_2 images typically requires the acquisition

of 5 images, it is necessary to process $5 \cdot 512^3 = 0.67 \cdot 10^9$ values. The increasing image quality, which constitutes a decisive factor in tissue recognition, then obviously raises the computational difficulty of the entire processing chain. Thus, the actual conversion of such large data volumes has to be viewed in a completely different manner.

If the reconstruction of a T_2 map is carried out using a regularly available software package, the computation time required to process a single, average data volume will most probably remain within tens to hundreds of milliseconds. In higher resolution images, such computation may last even several minutes on a common PC; the processing of a whole set of experimental samples would therefore require minutes to hours of computation time.

The computation period can nevertheless be significantly reduced by parallel programming and suitable application of standard hardware, usually a graphics card, which is characterized by a large number of implemented processor cores. The computation can thus be distributed among the processor threads executed in parallel, and the processing will be accelerated without its quality being compromised.

A. Image description

The designed parallel processing method was tested using a three-dimensional image acquired with a Bruker Biospec 94/30 tomograph (9.4 T) at the Institute of Scientific Instruments of the Academy of Sciences of the Czech Republic. The image captures a brown rat. The resolution of the images is 256x256 pixels, and the depth corresponds to twenty slices. Each pixel is measured five times with the Spin Echo method; the selected echo times T_E are 15, 45, 75, 105, and 135 ms.

B. Hardware and software specifications

To compare the sequential and parallel execution of the T_2 map computation process, we used a PC equipped with an Intel Core2 Quad 2.66 GHz, 4 GB RAM (2 × DDR2 @ 400 MHz). The set of related software included Windows 7 64-bit and Matlab R2013a.

The parallel execution was implemented via a CUDA platform v. 7.0 with an nVidia graphics card, namely a GeForce GTX 770 (1024 threads per block; max thread block size 1024x1024x64; 4 GB of total memory at the frequency of 7 GHz; 1 GHz GPU).

C. Methods used

The aim of the processing was to acquire a map of the T_2 relaxations of tomographically represented tissues. The paper presents two methods suitable for the purpose: a) sequential computation, and b) parallel computation. For both these approaches, the same image was used to facilitate comparison; this image consists of n pixels, and n is expressed as

$$n = w \cdot h \cdot n_s = 256 \cdot 256 \cdot 20 = 1\,310\,720\,, \tag{2}$$

where w is the image width in pixels, h is the image height in pixels, and n_s is the number of slices.

To compare the different processing times, we tested three algorithms implemented in the Matlab environment via sequential and parallel programming. The first two algorithms utilize methods based on converting exponential regression to linear regression via logarithmization of the equation [19]. While the first of these two procedures exploits the simple application of the least squares method to linear regression, the second one embodies an improved approach to the problem: although built upon the first technique, it allows simple correction to eliminate the drawback of greater weighting of smaller values of the acquired echoes [19]. We assume measured values that approximately correspond to the functional dependence expressed as

$$y = a \cdot e^{bx}\,, \tag{3}$$

After logarithmization, this formula yields the equation of a line

$$\ln y = \ln a + \ln e^{bx}\,. \tag{4}$$

Using simple modification, we can explicitly express the value of the exponent b as

$$b = \frac{\ln y - \ln a}{x}\,. \tag{5}$$

By applying the least squares method, we obtain – for the calculation of the coefficient b – the formula for the first exponential regression method (ER1):

$$b = \frac{n \cdot \sum_{i=1}^{n} x_i \cdot \ln y_i - \sum_{i=1}^{n} x_i \cdot \sum_{i=1}^{n} \ln y_i}{n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i \right)^2}\,. \tag{6}$$

where the vector x represents the individual times of the echoes T_E, the vector y denotes the measured echo values, and n is the number of measurements. By comparing the formulas 1 and 3 and substituting the coefficient b from the equation 6, we obtain the relaxation time T_2 in the form

$$T_2 = -\frac{1}{b}\,. \tag{7}$$

Each thread of the process therefore computes a voxel of the T_2 map according to the following formula (ER1):

$$T_2 = -\frac{n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i \right)^2}{n \cdot \sum_{i=1}^{n} x_i \cdot \ln y_i - \sum_{i=1}^{n} x_i \cdot \sum_{i=1}^{n} \ln y_i}\,. \tag{8}$$

The second procedure assigns the same weight to all measured echoes, thus ensuring a more accurate result. The regression model of this method consists of a simple correction, after which the following function is minimized:

$$\sum_{i=1}^{n} y_i \left(\ln y_i - a - bx_i\right)^2 . \tag{9}$$

If we now apply the least squares method, we obtain (via a process analogical to that used in deriving the ER1) the formula enabling the calculation of the relaxation time T_2 through ER2:

$$T_2 = -\frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} \left(x_i^2 y_i\right) - \left(\sum_{i=1}^{n} x_i y_i\right)^2}{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} \left(x_i y_i \ln y_i\right) - \sum_{i=1}^{n} \left(x_i y_i\right) \sum_{i=1}^{n} \left(y_i \ln y_i\right)} . \tag{10}$$

The third algorithm is also based on the minimization of the function given by the least squares method; in this case, however, the minimization is performed with the Levenberg-Marquardt (LM) optimizing approach [22], which is given by numerical solution of the formula

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda \mathrm{diag}\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}\right)\right)\delta = \mathbf{J}^{\mathrm{T}}\left[\mathbf{y} - f\left(\boldsymbol{\beta}\right)\right]. \tag{11}$$

where $\boldsymbol{\beta}$ is the vector of searched parameters of the interleaved function f, and the size of the parameter λ determines the characteristics of the algorithm. For low values of λ, the Levenberg-Marquardt model approaches the Gauss-Newton optimization method; for high values of λ, however, the model is close to the gradient descent technique. For the numerical solution, it is necessary to specify both the initial estimation of the searched parameters and the convergence criteria. In our case, the aspects were set as follows:

- Estimation of searched parameters: $a_0 = 1000$; $b_0 = -1$.
- Convergence criteria: the maximum of 100 iterations
- $\lambda_0 = 0.01$.

D. CPU implementation

The three above-described techniques were all implemented in Matlab via sequential and parallel programming.

At the initial stage, the test images were processed sequentially via two different techniques. The first of these methods sequentially reads individual image entities by means of a `for` cycle directly in Matlab, and one pixel of the resulting image is calculated during each step. As Matlab belongs to the family of interpreted programming languages, this processing method appears to be the least suitable in terms of the time required; for this reason, the specified processing time is usually only approximate and cannot be straightly compared with parallelized computations. A Matlab code to calculate the T_2 map via a `for` cycle is shown as follows:

```
for j = 1:length(t2echo)
    b = (5*sum(te.*log(t2echo(j,:)))-
    sum(te)*sum(log(t2echo(j,:))))/(5*sum(te.^2)-
    sum(te)*sum(te));

    t2mapa(j) = -1/b;
end
```

The other CPU-based sequential processing technique utilizes the optimized MEX function (written in the C language) in Matlab. This approach markedly differs from the previously described one in that the time-intensive `for` cycle is transferred from the interpreted Matlab environment to the very fast processor code of the MEX function. Such a code representing computation of the T_2 map via the ER1 method is presented below.

```
void compT2(double *te, double *t2echo, double
            *t2, int n)
{
    int idx;
    double xx[5], yy[5], xxyy[5], logyy[5],
           xx2[5], xx2yy[5], xxlogyy[5],
           yylogyy[5], xxyylogyy[5];

    for (int i = 0; i < 256*256; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            idx = i*5+j;
            xx[j] = te[j];
            yy[j] = t2echo[idx];
            xxyy[j] = xx[j]*yy[j];
            logyy[j] = log(yy[j]);
            xx2[j] = xx[j]*xx[j];
            xx2yy[j] = xx2[j]*yy[j];
            xxlogyy[j] = xx[j]*logyy[j];
            yylogyy[j] = yy[j]*logyy[j];
            xxyylogyy[j] = xx[j]*yy[j]*logyy[j];
        }

        t2[i] = (n*sum(xxlogyy)-
                 sum(xx)*sum(logyy))/(n*sum(xx2)-
                 sum(xx)*sum(xx));
        t2[i] = -1.0/t2[i];

    }
}
```

The parallel, CPU-based processing was performed by means of three procedures. The first of these employs direct parallelization and enables optimization for concrete hardware, both thanks to the Matlab built-in function `arrayfun`; this function executes the particular processing type via the user-selected method in all elements of the vector/matrix. The efficiency of this solution is nevertheless hampered by two major problems, namely the already described slow execution of the interpreted code and the large amount of time required for calling the external m-function that facilitates the actual computation over each element of the processed vector. At this point, it is also necessary to mention the unavailability of detailed documentation for the software-based execution of the `arrayfun` function. The second parallelization technique uses sequential processing, only the Matlab cycle `for` is

substituted by the parallelized cycle `parfor` with the maximum number of threads (four in this case) allowed by the hardware.

The third approach to parallelized computation via CPUs consists in calling the MEX function, which computes the T_2 map very quickly, using a compiled code written in C++. Here, the actual parallelization is performed through Microsoft tools of the Parallel Patterns Library (PPL), namely using the parallelized cycle `parallel_for`. This cycle operates similarly to its common counterparts, but unlike them it utilizes all processor cores available. A MEX function to reconstruct the T_2 map via the ER1 method is shown in the following section:

```
void compT2(double *te, double *t2echo, double
            *t2, int n)
{
    parallel_for(0, 256*256,
                 [&t2, &te, &t2echo, &n](int i)
    {
        int idx;
        double xx[5], yy[5], xxyy[5], logyy[5],
               xx2[5], xx2yy[5], xxlogyy[5],
               yylogyy[5], xxyylogyy[5];

        for (int j = 0; j < 5; j++)
        {
            idx = i*5+j;
            xx[j] = te[j];
            yy[j] = t2echo[idx];
            xxyy[j] = xx[j]*yy[j];
            logyy[j] = log(yy[j]);
            xx2[j] = xx[j]*xx[j];
            xx2yy[j] = xx2[j]*yy[j];
            xxlogyy[j] = xx[j]*logyy[j];
            yylogyy[j] = yy[j]*logyy[j];
            xxyylogyy[j] = xx[j]*yy[j]*logyy[j];
        }

        t2[i] = (n*sum(xxlogyy)-
                 sum(xx)*sum(logyy))/(n*sum(xx2)-
                 sum(xx)*sum(xx));
        t2[i] = -1.0/t2[i];
    });
}
```

To simplify the implementation of the algorithms in the MEX function, the CPU program was (unlike GPU distributed programs) complemented with the `double` data type in all real variables.

E. GPU implementation

The parallel computation is invariably performed over only one slice of the three-dimensional image, which exhibits the resolution of 256x256 pixels; this condition corresponds to 65536 threads of the graphics processor. The computation of the three-dimensional T_2 map is then carried out in the serial-parallel manner.

The individual methods were implemented using CUDA cores. An example of the ER1 method implemented to calculate the exponential regression coefficients is shown here:

```
#define tel 5 // 5 echoes
// auxiliary function
__device__ float sum(const float vector[tel])
{
    double sum=0;
    sum+=vector[0];
    sum+=vector[1];
    sum+=vector[2];
    sum+=vector[3];
    sum+=vector[4];
    return sum;
}

// main CUDA kernel
__global__ void compT2(float * t2map,
                       const float * te,
                       const float * t2echo,
                       const int t2len)
{
    int idx = threadIdx.x +
              (((gridDim.x * blockIdx.y) +
              blockIdx.x)*blockDim.x);

    if (idx < t2len) {
       float b, xx[tel], yy[tel], xx2[tel],
             logyy[tel], xxlogyy[tel];
       float n = (float)tel;

       for (int i = 0; i < tel; i++) {
           xx[i] = te[i];
           yy[i] = t2echo[idx+i*65536];
           logyy[i] = log(yy[i]);
           xx2[i] = xx[i]*xx[i];
           xxlogyy[i] = xx[i]*logyy[i];
       }

       // compute the exponent T2=-1/b
       b = (n*sum(xxlogyy)-sum(xx)*sum(logyy))/
            n*sum(xx2)-sum(xx)*sum(xx));
       t2map[idx] = -1.0/b;
    }
}
```

In order to ensure high efficiency of the parallelized algorithm, we selected the data type `float` (32 bits) for all significant parameters, mainly because its accuracy is fully sufficient for the given task. From the `sum` function for the summation of vector elements, the original cycle `for` was removed over all members and substituted with five-step sum. At such a low number of vector elements, the actual amount of time required for the cycle initialization could manifest itself markedly.

3. RESULTS

The T_2 map computation as performed by the individual algorithms is analyzed considering the factors of processing time and accuracy of the obtained values. Due to marked differences between the sequential and parallel execution of the program, the comparison of the processing times is performed on a three-dimensional image, namely with the resolution of 256x256 and 20 slices. The comparison focused on the accuracy of the obtained values is carried out via the computation and analysis of the differential image and by comparing the computed relaxation times of concrete tissues. The related processing time values are presented in Tables 1.-3. Table 1. compares the CPU-based sequential and the GPU-based parallel processing types. The time values

related to parallel processing via a graphics card are invariably specified without the time necessary for the image data transfer to the graphics memory; this time was defined separately and corresponds to the value of 1.3 ms per transfer of one MR slice. It appears that, in a simple exponential regression (the ER1 and ER2 methods), the acceleration of algorithm distributions to the GPU is not markedly discernible; the acceleration value is approximately 4.5×, which virtually corresponds to the published achievable values [23]. In the more complex LM method, we can already speak of a three hundred-fold acceleration. But this value can be, on the one hand, misleading due to differences between the applied data types (*float* in the GPU, *double* in the CPU approaches) or because the code was not optimized for the concrete hardware; this latter aspect may include, for example, the SSE vectorization, which is certainly capable of providing a major increase in efficiency. Within the above- described solution, we accentuated fast, clear, and transferable implementation. On the other hand, however, the time required for sequential processing by means of the MEX function roughly corresponds to the processing time achievable with the Marevisi and MRI Processor programs. If we compare (Table 2.) the times necessary for the CPU and GPU-based parallel processing, it becomes obvious that in the simple methods (the ER1 and ER2) we obtain an almost double acceleration with the graphics card distribution. In the LM method, the acceleration is seventyfold in the given case. Here, too, the use of double time consuming data types may be of major importance. Table 3. compares the times required for the CPU-based processing of one and the same image via various sequential and parallel techniques. For example, it is apparent from this comparison that 4-thread parallelization in the MEX function becomes reasonably applicable only with the more complex algorithms (the LM), where the computation is positively accelerated almost four-fold. The simple procedures (the ER1 and ER2), however, are considerably burdened with the time-consuming parallelization control, and thus the parallelization process can be accelerated only twice.

Table 1. The processing times of the sequential (MEX function) and parallel (GPU) computation methods in Matlab.

Comp.	Comp. time 1 slice sequential (parallel)	Comp. time 20 slices sequential (parallel)
ER1	8.40 ms (1.80 ms *)	168 ms (36 ms *)
ER2	8.90 ms (1.90 ms *)	178 ms (38 ms *)
LM	1.45 s (4.80 ms *)	29.0 s (96 ms *)
Marevisi	4.3 s (n/a)	≈86 s (n/a)
MRI Proc.	12.0 s (n/a)	240 s (n/a)

*) The processing times without data transmission to the GPU RAM; the transmission time is approximately 1.3 ms/slice.

Table 2. The processing times of the CPU (MEX function) vs. the GPU parallel computation methods in Matlab.

Comp.	Comp. time 1 slice CPU (GPU)	Comp. time 20 slices CPU (GPU)
ER1	3.65 ms (1.80 ms *)	73 ms (36 ms *)
ER2	4.20 ms (1.90 ms *)	84 ms (38 ms *)
LM	370.8 ms (4.80 ms *)	7,416 s (96 ms *)

*) The processing times without data transmission to the GPU RAM; the transmission time is approximately 1.3 ms/slice.

Table 3. A comparison of the times required to process one slice via various CPU-based techniques.

Computation	ER1	ER2	LM
MEX function, seq. processing	8.40 ms	8.90 ms	1.45 s
MEX function, par. processing	3.65 ms	4.20 ms	370.8 ms
Matlab `for` cycle	710 ms	1.10 s	505 s
Matlab `parfor` cycle (4 workers)	225 ms	1.00 s	185 s
Matlab `arrayfun` function	1.54 s	1.91 s	475 s

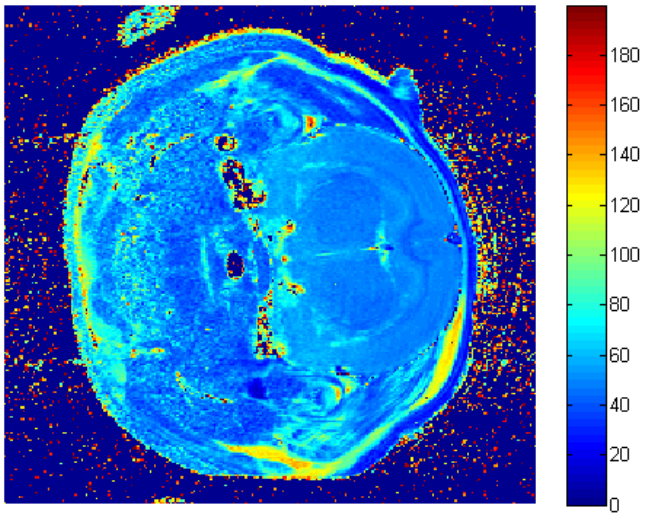

Fig.2. The resulting T_2 map obtained via the Levenberg-Marquardt algorithm (LM).

Fig.2. presents a selected T_2 map in the 9th slice (middle position) of the volume from the parallel computation using non-linear regression performed with the Levenberg-Marquardt algorithm. An image obtained using this method is comparable to images acquired by means of professional MR image processing programs. In the following sections of the comparison, the intensities of this image will therefore constitute reference values of the T_2 relaxations.

Fig.3. and Fig.4. then show the result of the parallel computation carried out with the linear regression technique according to formulas 8 (ER1) and 10 (ER2). The differential image to facilitate comparison of the results is indicated in Fig.5.

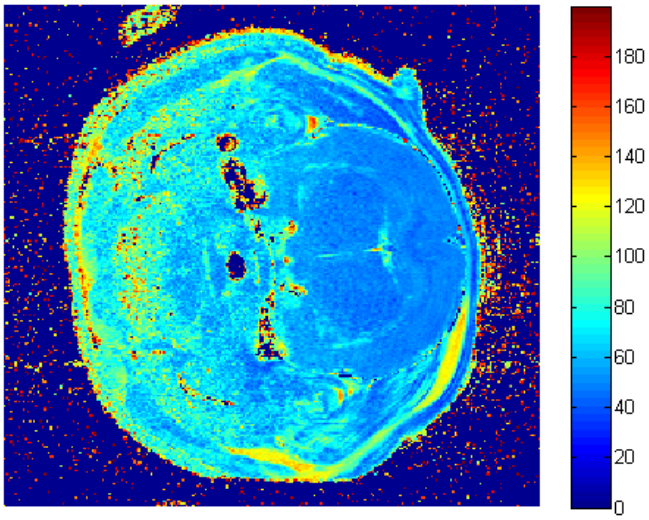

Fig.3. The resulting T_2 map acquired with the linear regression method (ER1).

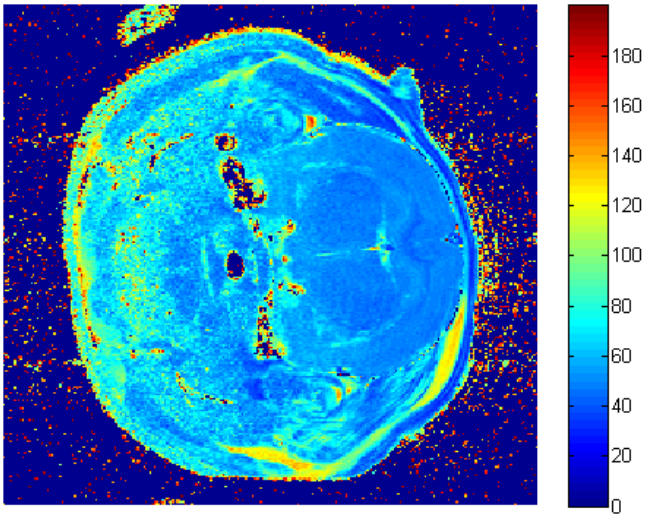

Fig.4. The resulting T_2 map obtained using the modified linear conversion method (ER2).

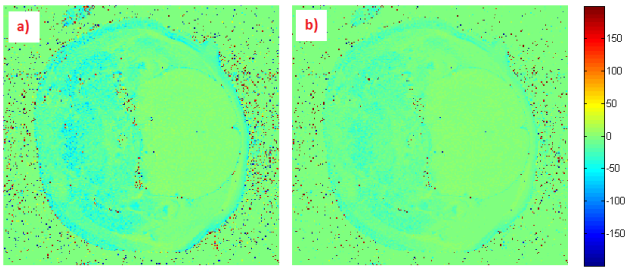

Fig.5. The differential images: a) LM-ER1; b) LM-ER2.

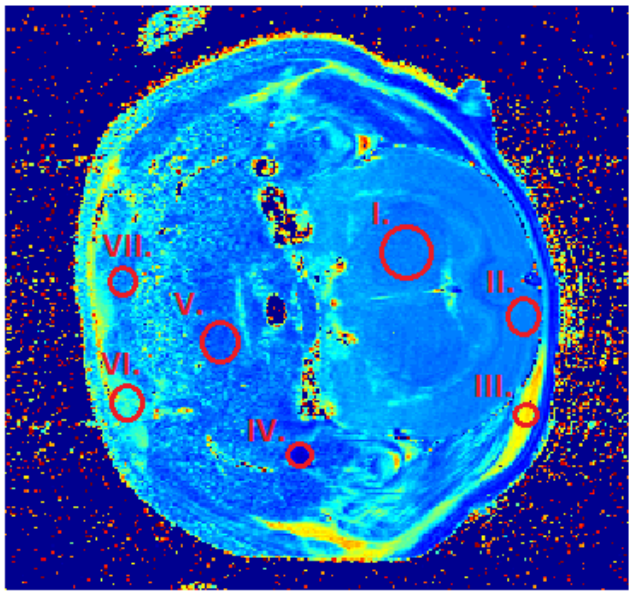

Fig.6. The ROIs for evaluating the relaxation times of concrete tissues.

Based on subjective evaluation of the images shown in Fig.3. and Fig.4., it is possible to confirm the accuracy of both methods used in the T_2 map computation. An analysis of the map acquired via the ER1 and ER2 methods will indicate a higher sensitivity to noise, which degrades the SE images already at the acquisition stage. This problem, which is visible mainly in the left portion of the image, is caused by weak response of the examined tissues, reflecting the fact that the image was made with a surface coil located on the right-hand side. The correctness of the result can be further demonstrated on the differential image presented in Fig.5. Here, again, a higher error rate can be identified in regions characterized by a lower level of the SE signal. After comparing the differential images presented in Figs.5.a) and b), it is possible to state that the ER1 is a method less accurate and more sensitive to noise than ER2. By comparing the sums of the differential image intensities, namely $\left|\sum \text{LM-ER1}\right| = 4.3 \cdot 10^5$ and $\left|\sum \text{LM-ER2}\right| = 2.4 \cdot 10^5$, we can conclude that the error exhibited by the method ER2 is smaller than that of the technique ER1. The results can be also compared by the calculated relaxation time values in individual tissues. Fig.6. shows the regions of interest in which we evaluated the T_2 relaxation values in images obtained via LM, ER1, and ER2.

Table 4. comprises values from the evaluation of the T_2 relaxation times related to concrete tissues, namely the ROIs shown in Fig.6. The regions are numbered from right to left; the right portion of the image corresponds to the real region where the tomograph sensing coil was located. As indicated in Table 4., the error rate of both ER1 and ER2 increases with the distance between the coil and the evaluation spot. The regions I, II, and III exhibit results comparable to those acquired with all three methods; in the regions IV to VII, major disagreement can be observed between the values of relaxation times computed with the ER1 and ER2 methods and the values obtained with the reference LM.

Table 4. Evaluation of the T_2 relaxation times in concrete tissues.

ROI	LM	ER1	ER2
I.	47.8±1.9 ms	49.2±3.2 ms	48.2±2.0 ms
II.	49.4±1.3 ms	49.4±1.7 ms	49.0±1.5 ms
III.	122.2±5.7 ms	122.7±2,6 ms	120.2±4.9 ms
IV.	18.0±5.0 ms	53.1±6.0 ms	37.1±3.2 ms
V.	40.4±4.1 ms	58.2±7.2 ms	51.1±5.5 ms
VI.	77.9±9.7 ms	85.2±11.7 ms	78.9±9.5 ms
VII.	50.4±7.1 ms	67.6±10.1 ms	60.4±8.3 ms

4. CONCLUSIONS

The above-presented results point to the benefit of algorithm parallelization to both the reconstruction of T_2 maps and the implementation of such algorithms in a graphics card processor. In this manner, the computation of the T_2 relaxation time maps was significantly accelerated without the processing quality being compromised. We described three reconstruction methods, namely two approaches based on linear regression and one utilizing nonlinear regression. The best results were achieved with the nonlinear solution, which consists of optimizing the exponential function parameters by means of the Levenberg-Marquardt algorithm. Interestingly, the results obtained via this method were identical with those provided by regularly accessible software tools for the processing of data from MR tomographs (*MRI Processor*, *Marevisi*). These programs also utilize the regression approach employing the Levenberg-Marquardt optimization, and thus agreement between the results was the precondition for correct implementation. A comparison of the processing times required to compute the applied image (256x256x20 pixels) will indicate a major decrease in the length of the execution period, and we then have the following values: *Marevisi* - approximately 1.5 min.; *MRI Processor* - 4 min.; GPU acceleration – 96 ms. It is therefore possible to conclude that a graphics processor will proceed 900 to 2500 times faster than regularly available software packages; if we compared the processing times characterizing the linear regression methods, the eventual processing pace would be even higher. However, such comparison is rather misleading because the linear regression methods may exhibit a significant failure rate in the reconstruction of T_2 maps. This fact was verified by comparing the processing results acquired with LM, ER1, and ER2. The ER1 method, which principally does not assign the same weight to all regression points, exhibited an error rate higher than that of the ER2 approach, which solves the given problem. In both these cases, however, a comparison of the reconstruction results or an analysis of the differential images has shown that the linear regression methods, in which the exponential regression problem is converted to the optimization of the linear model, are not suitable for the reconstruction of relaxation times. This is further proved by the results obtained from an analysis of the differential images, where the sum of intensities of the individual pixels is $4.3 \cdot 10^5$ in ER1 and $2.4 \cdot 10^5$ in ER2. A higher value points to a more prominent difference between the result of the given method and the results provided by the reference technique LM. The reconstruction error can be demonstrated in the relaxation times of individual tissues, especially in spots more distant from the measuring surface coil. In regions marked as IV, V, and VII, the error of T_2 reconstruction is expressed in tens to hundreds of percent, and we can therefore conclude that the simple linear regression methods ER1 and ER2 are not applicable for the reconstruction of maps from weak, noise-containing signals.

To provide a well-balanced comparison of the processing times, we included several types of implementation related to both the sequential and the parallelized techniques. Within this context, the easiest procedure appears to consist of comparing the processing speed of MEX functions programmed in the C language with the speed of graphics card-distributed computations. If we compare the purely sequential (pixel-by-pixel), CPU-based T_2 map computation using the MEX function with the GPU-based processing, we obtain the speed acceleration values of 4.6×, 4.7×, or 300× in the ER1, ER2 methods (the LM). However, it appears more accurate and balanced to observe the CPU-based parallelized computation vs. the GPU-based process using the MEX function, where the achieved acceleration equals 2×, 2.2×, and 77×. As pointed out above, the large increase in the LM method reflects the fact that the possibilities of the code optimization for the concrete CPU were exploited only partially. Assuming the values in Table 1., it is nevertheless possible to claim that although the reconstruction of MRI T_2 maps can be considered a time-consuming task within image processing, currently available tools (Marevisi, MRI Processor) do not employ any large-scale optimization of the CPU code and are markedly slower than the analyzed implementation.

ACKNOWLEDGMENT

This work was supported in part by the Grant GAČR 102/12/1104 and by the project CZ.1.07/2.3.00/30.0039 of Brno University of Technology.

REFERENCES

[1] Dvořák, P., Bartušek, K., Smékal, Z. (2014). Unsupervised pathological area extraction using 3D T2 and FLAIR MR images. *Measurement Science Review*, 14 (2), 357-364.

[2] Mikulka, J., Burget, R., Říha, K., Gescheidtová, E. (2013). Segmentation of brain tumor parts in magnetic resonance images. In *36th International Conference on Telecommunications and Signal Processing*, 2-4 July 2013. IEEE, 565-568.

[3] Hnilicová, P., Bittšanský, M., Dobrota, D. (2014). Optimization of brain T2 mapping using standard CPMG sequence in a clinical scanner. *Measurement Science Review*, 14 (2), 117-125.

[4] Hahn, E. (1950) Spin echoes. *Physical Review*, 80 (4), 580-594.

[5] Jiménez, J., Ruiz de Miras, J. (2012). Fast box-counting algorithm on GPU. *Computer Methods and Programs in Biomedicine*, 108 (3), 1229-1242.

[6] Gogola, D., Krafčík, A., Štrbák, O., Frollo, I. (2013). Magnetic resonance imaging of surgical implants made from weak magnetic materials. *Measurement Science Review*, 13 (4), 165-168.

[7] Eklund, A., Andersson, M., Knutsson, H. (2012). fMRI analysis on the GPU. *Computer Methods and Programs in Biomedicine*, 105 (2), 145-161.

[8] Zhu, F., Rodriguez-Gonzalez, D., Carpenter, T., Atkinson, M., Wardlaw, J. (2012). Parallel perfusion imaging processing using GPGPU. *Computer Methods and Programs in Biomedicine*, 108 (3), 1012-1021.

[9] Broxvall, M., Emilsson, K., Thunberg, P. (2012). Fast GPU based adaptive filtering of 4D echocardiography. *IEEE Transactions on Medical Imaging*, 31 (6), 1165–1172.

[10] Prassni, J.S., Ropinski, T., Hinrichs, K. (2010). Uncertainty-aware guided volume segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 16 (6), 1358-1365.

[11] Shams, R., Sadeghi, P., Kennedy, R.A., Hartley, R.I. (2010). A survey of medical image registration on multicore and the GPU. *IEEE Signal Processing Magazine*, 27 (2), 50-60.

[12] Fluck, O., Vetter, C., Wein, W., Kamen, A., Preim, B., Westermann, R. (2011). A survey of medical image registration on graphics hardware. *Computer Methods and Programs in Biomedicine*, 104 (3), e45-e57.

[13] Pratx, G., Xing, L. (2011) GPU computing in medical physics: A review. *Medical Physics*, 38 (5), 2011, 2685-2697.

[14] Eklund, A. et al. (2013) Medical image processing on the GPU – Past, present and future. *Medical Image Analysis,* 17 (8), 1073-1094.

[15] Giri, S., Chung, Y.C., Merchant, A., Mihai, G., Rajagopalan, S., Raman, S.V., Simonetti, O.P. (2009). T2 quantification for improved detection of myocardial edema. *Journal of Cardiovascular Magnetic Resonance*, 11 (1), 56.

[16] Jones, C.K. (2003). *T2 decay curve acquisition and analysis in MRI noise considerations, short T2, and B1 field encoding*. PhD thesis, University of British Columbia.

[17] Yoo, Y. (2003). *Robust and fast T2 decay analysis for measuring myelin water in MRI*. PhD thesis, University of British Columbia.

[18] Fraile, R., García-Ortega, E. (2005). Fitting an exponential distribution. *Journal of Applied Meteorology and Climatology*, 44 (10), 1620-1625.

[19] Rektorys, K. (1995). *Přehled užité matematiky II*. Praha: Prometheus.

[20] Prodanov, D. (2009). *MRI Processor, v. 1.1.6 22*. http://imagejdocu.tudor.lu/doku.php?id=plugin:filter:mri_processor:start.

[21] Starcuk, J., Starcuk, Z., Kozlowski, P. (2008). *Marevisi, v. 8.2*. More information at: http://www.nrc-cnrc.gc.ca/eng/solutions/licensing/multidimensional_mri_data_software.html.

[22] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P. (2007). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press.

[23] Lee, V.W., et al. (2010). Debunking the 100X GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU. In *ISCA'10*, 19-23 June 2010, Saint-Malo, France.

Received January 22, 2015.
Accepted August 12, 2015.