RESEARCH ARTICLE

GPU-Accelerated Compartmental Modeling Analysis of DCE-MRI Data from Glioblastoma Patients Treated with Bevacizumab

Yu-Han H. Hsu[1], Ziyin Huang[1¤], Gregory Z. Ferl[2], Chee M. Ng[1,3]*

1 Division of Clinical Pharmacology and Therapeutics, The Children's Hospital of Philadelphia, Philadelphia, PA, United States of America, **2** Early Development Pharmacokinetics and Pharmacodynamics, Genentech, South San Francisco, CA, United States of America, **3** Department of Pediatrics, Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA, United States of America

¤ Current address: Department of Materials Science and Engineering, Drexel University, Philadelphia, PA, United States of America
* ngc1@email.chop.edu

Citation: Hsu Y-HH, Huang Z, Ferl GZ, Ng CM (2015) GPU-Accelerated Compartmental Modeling Analysis of DCE-MRI Data from Glioblastoma Patients Treated with Bevacizumab. PLoS ONE 10 (3): e0118421. doi:10.1371/journal.pone.0118421

Academic Editor: Mara Cercignani, Brighton and Sussex Medical School, UNITED KINGDOM

Received: August 29, 2014

Accepted: December 25, 2014

Published: March 18, 2015

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: The authors have no support or funding to report.

Competing Interests: The authors have declared that no competing interests exist.

Abstract

The compartment model analysis using medical imaging data is the well-established but extremely time consuming technique for quantifying the changes in microvascular physiology of targeted organs in clinical patients after antivascular therapies. In this paper, we present a first graphics processing unit-accelerated method for compartmental modeling of medical imaging data. Using this approach, we performed the analysis of dynamic contrast-enhanced magnetic resonance imaging data from bevacizumab-treated glioblastoma patients in less than one minute per slice without losing accuracy. This approach reduced the computation time by more than 120-fold comparing to a central processing unit-based method that performed the analogous analysis steps in serial and more than 17-fold comparing to the algorithm that optimized for central processing unit computation. The method developed in this study could be of significant utility in reducing the computational times required to assess tumor physiology from dynamic contrast-enhanced magnetic resonance imaging data in preclinical and clinical development of antivascular therapies and related fields.

Introduction

Dynamic contrast-enhanced magnetic resonance imaging (DCE-MRI) is a noninvasive quantitative tool that allows analysis of tumor vascular characteristics that might change in response to drug treatments without the use of ionizing radiation. With DCE-MRI, vascular properties of tumors such as vascular permeability, rate of perfusion, and vascular leakage space can be quantified using standard MRI imaging system in routine clinical practice [1–3]. Therefore, this imaging technique is increasingly used to evaluate antivascular therapies and can also be applied to select drug doses for clinical studies, identify subpopulations enriched for clinical response, and predict patients' benefits [4]. Numerous studies have utilized DCE-MRI as a quantitative imaging biomarker to guide preclinical and early clinical development of antiangiogenic agents, including anti-VEGF (vascular endothelial growth factor) antibodies such as bevacizumab (Avastin), receptor tyrosine kinase inhibitors, and vascular disrupting agents such as 5,6-dimethylxanthenone-4-acetic acid (DMXAA) [1, 4, 5].

Several analysis methods have been developed to quantify changes in microvascular physiology using DCE-MRI data [6–9]. Compartmental models such as the Tofts and the extended Tofts models are commonly used for kinetic analysis of imaging data and, when using an appropriate model structure, are the golden standards for quantification of DCE-MRI data [10, 11]. The analysis process involves using numerical minimization algorithms to fit the mathematical models to the observed contrast agent concentration-time profiles in blood and tissue and deriving model parameters that can estimate physiological properties. For instance, two tumor properties, fractional interstitial volume (v_e) and fractional transfer rate (K^{trans}), are commonly derived and used to assess drug effects on tumor. While this kinetic modeling approach to analyze DCE-MRI data is straightforward in concept, it is often very time consuming and computationally expensive, as model fitting has to be performed separately on each voxel in a DCE-MRI image, requiring tens of thousands of minimization function calls. When a dataset contains multiple DCE-MRI scans from many patients, with multiple slices in each scan, the analysis time cost may become a significant burden in a fast-paced clinical or research environment.

The graphics processing unit (GPU) is a dedicated numerical processor that has evolved from a highly specialized graphics processor to a versatile, highly programmable and energy efficient architecture for scientific computing [12–14]. Several of the world's fastest supercomputers, including Titan at Oak Ridge National Laboratory and Piz Daint at Swiss National Supercomputing Centre, are partly powered by GPU-computing technology [15]. Compared to standard central processing unit (CPU), current GPU has hundreds of numerical processor cores on a single chip and can be programmed to perform many numerical operations simultaneously to achieve extremely high arithmetic intensity for complex numerical analysis. Because of this unique architecture, running computations on GPU can be significantly faster than on CPU when the computations can be parallelized and distributed to GPU's numerous processor cores. GPU parallel computing has been used in various medical imaging applications for faster processing [16–18]. For DCE-MRI reconstruction and analysis, since each voxel in a DCE-MRI image is normally treated as an independent element during compartmental modeling, the process is perfectly suited for parallel implementations on the GPU. Recently, we successfully used GPU to accelerate the performance of relatively simple model-independent nonparametric method in the analysis of clinical DCE-MRI data [19]. However, to our best knowledge, there is no published study or report of using GPU to improve the performance of standard compartmental model method in DCE-MRI data analysis. Therefore, in this article, we present the first GPU-accelerated compartmental modeling method for DCE-MRI data analysis that can drastically reduce the amount of time required to evaluate changes in microvascular physiology in clinical patients before and after antivascular therapies.

Methods

Compartmental Model for Describing the Blood and Tumor Concentration-Time Profiles of Contrast Agent

The complete compartmental model used to describe the contrast agent kinetics in each DCE-MRI scan is shown in Fig. 1. A linear two-compartment model was used to represent systematic contrast agent kinetics (Fig. 1, compartments 1 and 2), thus the contrast agent concentration-time course in arterial blood (compartment 1) can be described by a bi-exponential decay

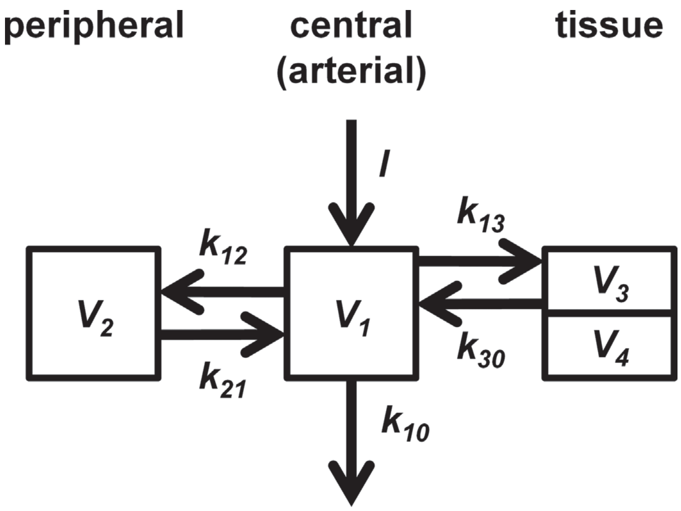

Fig 1. Compartment models for Gd-DTPA kinetics. The schematic shows the kinetics relationship between the tumor tissue (right box) and the remainder of the body (center and left box). Drug input (*I*) goes into arterial blood in compartment 1 (center box, volume: V_1), which exchanges with a general peripheral compartment 2 (left box, volume: V_2) via fractional clearances k_{12} and k_{21} and eliminates drug via k_{10}. The tumor (and other brain tissue) is represented by 2 compartments: compartment 3 (top of right box, volume: V_3) exchanges with compartment 1 via k_{13} and k_{30}; compartment 4 (bottom of right box, volume: V_4) exchanges with compartment 1 so rapidly that its drug concentration is practically the same as that of compartment 1.

doi:10.1371/journal.pone.0118421.g001

equation [20–22]:

$$C_1(t) = \frac{R_0}{V_1}\left[\frac{(1-e^{\lambda_1\tau})(k_{21}-\lambda_1)e^{-\lambda_1(t-t_0-t_{lag})}}{-\lambda_1(\lambda_2-\lambda_1)} + \frac{(1-e^{\lambda_2\tau})(k_{21}-\lambda_2)e^{-\lambda_2(t-t_0-t_{lag})}}{-\lambda_2(\lambda_1-\lambda_2)}\right] \quad (1)$$

Where R_0 is zero-order infusion rate; V_1 is the volume of compartment 1; t_0 is the starting time of infusion; t_{lag} is lag time before infusion starts to affect concentration observation; τ is the apparent elapsed time of infusion: $\tau = 0$ for $t \leq t_0 + t_{lag}$, $\tau = t - t_0 - t_{lag}$ for $t_0 + t_{lag} < t \leq t_1 + t_{lag}$ (t_1 is the time at the end of infusion), and $\tau = t_1 - t_0$ for $t > t_1 + t_{lag}$; and $\lambda_{1,2} = \frac{1}{2}(k_{10} + k_{12} + k_{21} \pm \sqrt{(k_{10}+k_{12}+k_{21})^2 - 4k_{10}k_{21}})$ [20]. As shown in Fig. 1, k_{10}, k_{12}, and k_{21} are fractional clearances between the indicated compartments.

The extended Tofts model was used to depict contrast agent kinetics in tumor and other brain tissue. Since tissue consists of both extravascular extracellular space (EES) and a blood plasma volume, its total contrast agent concentration at time *t* is [21]:

$$C_{tissue}(t) = v_e C_e(t) + v_p C_p(t) \quad (2)$$

Where $C_e(t)$ and $C_p(t)$ are the contrast agent concentrations in EES and in blood plasma, respectively, and v_e and v_p are their corresponding volumes per unit volume of tissue. The EES and blood plasma compartments correspond to compartments 3 and 4, respectively, in Fig. 1.

The EES (compartment 3) concentration-time course, $C_e(t)$, can be described by [20, 23, 24]:

$$C_3(t) = k_{30}\frac{R_0}{V_1}\left[\frac{(1-e^{\lambda_1\tau})(k_{21}-\lambda_1)e^{-\lambda_1(t-t_0-t_{lag})}}{-\lambda_1(\lambda_2-\lambda_1)(k_{30}-\lambda_1)} + \frac{(1-e^{\lambda_2\tau})(k_{21}-\lambda_2)e^{-\lambda_2(t-t_0-t_{lag})}}{-\lambda_2(\lambda_1-\lambda_2)(k_{30}-\lambda_2)} + \frac{(1-e^{k_{30}\tau})(k_{21}-k_{30})e^{-k_{30}(t-t_0-t_{lag})}}{-k_{30}(\lambda_1-k_{30})(\lambda_2-k_{30})}\right] \quad (3)$$

(the variables have same definitions as in Equation 1; k_{30} is used instead of k_{31} to indicate that contrast agent transfer from compartment 3 back into compartment 1 is ignored when defining $C_1(t)$). The blood plasma (compartment 4) concentration-time course, $C_p(t)$, can be estimated by the $C_1(t)$ model in Equation 1, since the relatively fast blood exchange between compartments 1 and 4 would make their concentrations practically indistinguishable. It is important to note that $k_{30} = K^{trans} / v_e$, where K^{trans} is the volume transfer coefficient between arterial blood and EES. Thus the extended Tofts model relates tissue concentration data to both v_e and K^{trans}, two important parameters that are commonly used to assess tumor physiology and vascular function in clinical studies. These parameters are obtained by using the nonlinear optimization algorithm such as Newton and Quasi-Newton methods to fit ESS compartmental model (Equation 3) to the observed contrast agent concentration-time course in individual voxel of human tissue image scans. Because each image scan of the human tissue contains many thousands voxels, this particular step of repeatedly fitting the ESS model to observed contrast agent concentration-time courses requires tens of thousands of minimization function calls and therefore, is extremely time consuming and computationally expensive to perform.

A GPU-based Implementation

Data parallelism is an essential requirement for an algorithm to benefit from GPU execution. The most computational step of the compartment analysis approach is the fitting of the ESS model to the observed contrast agent concentration-time course of many thousands of voxel in each human tissue image scan. However, this particular computation step may be formulated in a GPU-friendly data parallel manner as it mainly consist of voxel-wise computations.

Analysis Flow of the GPU-accelerated compartmental modeling approach. Fig. 2 summarizes the analysis flow of the GPU-accelerated compartmental modeling program which developed using MATLAB (The MathWorks, Inc., Natick, MA) and MATLAB-compatible GPU computing toolbox Jacket v2.0 (Accelereyes Inc., Atlanta, GA) which became part of MATLAB parallel computing toolbox, and each step is described in more detail below, and the MATLAB function code of the GPU-accelerated compartment modeling program was included as the supplementary materials (S1–S3 Programs).

1. Arterial Blood Data Fitting. After DCE-MRI data is loaded by the MATLAB script, the linear two-compartment model described by Equation 1 is fit to the arterial contrast agent concentration ($\boldsymbol{C_v}$) data by using the CPU-based Nelder-Mead algorithm to minimize the least squares objective function:

$$f(p) = \sum_{i=1}^{n_f} (c_i - \hat{c}_i(p))^2 \quad (4)$$

Where p is the set of model parameters to be estimated, including R_0/V_1 (treated as one parameter), k_{10}, k_{12}, k_{21}, and t_{lag}; c_i is the observed arterial concentration at each time point; and $\hat{c}_i(p)$ is the corresponding concentration value predicted by Equation 1 with the given p values. Inputs to the Nelder-Mead algorithm, such as initial estimates and boundaries of the model

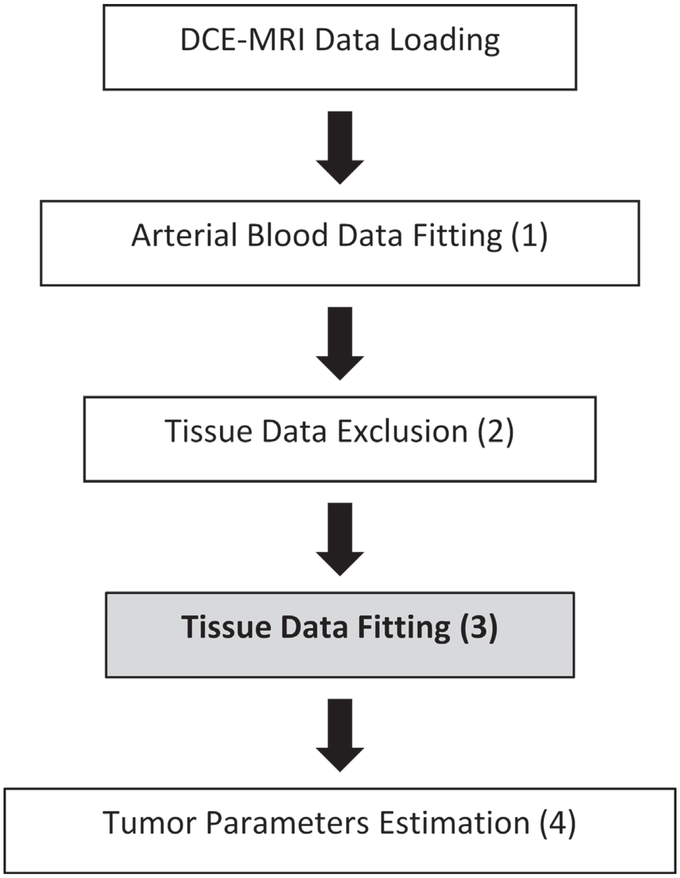
DCE-MRI Data Loading
Arterial Blood Data Fitting (1)
Tissue Data Exclusion (2)
Tissue Data Fitting (3)
Tumor Parameters Estimation (4)

Fig 2. Flow chart of the analysis steps executed by the MATLAB script. Step done on the GPU is in bold font.

doi:10.1371/journal.pone.0118421.g002

parameters, the number of iterations (300), and the convergence threshold (1e-5), are set based on past literature and preliminary analysis results. After $\boldsymbol{C_v}$ data fitting is completed, fitted parameter values for R_0/V_1, k_{10}, k_{12}, and k_{21} are stored and used for fitting tissue data in Step 3 (Fig. 2).

2. Tissue Data Exclusion. Before tissue data fitting is performed, the tissue contrast agent concentrations ($\boldsymbol{C_T}$) matrix is screened to exclude any voxels whose data contain: a) missing values, b) only negative values, or c) values that exceed a maximum threshold (defined based on fitted arterial data). This procedure removes extremely noisy data, often from voxels outside of the brain, and saves unnecessary computational time in the following analysis steps. For the DCE-MRI data analyzed in this study, more than 50% of the voxels in the field of view (FOV) can be excluded from further analysis in this step.

3. Tissue Data Fitting. After noisy voxels are removed, GPU-accelerated Nelder-Mead minimization of the objective function in Equation 4 is performed on each remaining voxel's $\boldsymbol{C_T}$ data. In this case, c_i is the observed tissue concentration in a single voxel at each time point and $\hat{c}_i(p)$ is calculated from the extended Tofts model (Equation 2). The set of model parameters to be estimated (p) are v_e, k_{30}, v_b, and t_{lag}. Other parameters, including R_0/V_1, k_{10}, k_{12}, and k_{21}, are set to the values derived from arterial data fitting in Step 1 (Fig. 2).

4. Tumor Parameters Estimation. After all tissue data fitting is completed, the resulting parameter estimates are generated by the MATLAB script. The tumor parameters v_e and K^{trans}

can be derived from results of the ***ROI*** voxels (using $k_{30} = K^{trans} / v_e$) and can be visualized in heat maps after adjusting for extreme outliers. Median ***ROI*** v_e and K^{trans} values can be calculated to compare tumor physiology and function in the scans acquired before and after bevacizumab treatment.

A GPU-accelerated Nelder-Mead simplex-based minimization algorithm. *1. Clinical Imaging Study and Image Processing*. The DCE-MRI data analyzed in this study were collected in a phase II clinical trial of bevacizumab in patients with grade III-IV glioma that was approved by the Duke Institutional Review Board and informed consent was obtained from every patients [9, 25]. DCE-MRI scans were obtained from the patients one day before and one day after bevacizumab treatment, using gadolinium-diethylene triamine pentaacetic acid (Gd-DTPA) as the contrast agent. Details of the DCE-MRI procedure and preliminary image processing were described in previous studies [7, 9, 19].

2. Model Fitting Minimization Algorithm. A Nelder-Mead simplex-based minimization algorithm was developed and used to fit the concentration-time models to the DCE-MRI blood and tissue data and to estimate the model parameters. The classic Nelder-Mead method [26] is often used to solve unconstrained minimization problem in which the goal is to minimize a function of *n* variables. In brief, the method constructs a simplex with *n+1* vertices in the search space (where each vertex represents a set of variable values), then performs a series of transformations on the simplex to decrease the function values of its vertices until convergence is reached (i.e. the vertices or their corresponding function values become close enough).

The minimization algorithm implemented in this study was modified to add an extra step to the classic Nelder-Mead method so that it can be used to solve constrained optimization problem [27]. More specifically, the algorithm takes in a set of initial estimates (x_0) for the function variables along with lower (*lb*) and upper (*ub*) bound restrictions on the variable values, then performs an arcsine transformation:

$$x_u = \arcsin\left(\frac{2(x_0 - lb)}{ub - lb} - 1\right) \tag{5}$$

to convert x_0 into its counterpart (x_u) in an unconstrained search space. After this transformation, the normal Nelder-Mead process is carried out to find a temporary solution in the unconstrained space, then the temporary solution is reversely transformed to produce a final solution that confines to the original boundary restrictions.

The code in this constrained Nelder-Mead algorithm was also modified extensively to remove and/or replace branching (e.g. if-else and switch case statements) and conditional termination, since these operations are computationally expensive and cannot run properly in parallel on the GPU without major modifications. Thus while a normal Nelder-Mead function would terminate whenever it reaches convergence or a pre-determined iteration number, the Nelder-Mead algorithm developed in this analysis runs a pre-determined number of iterations in two sequential steps. Initially, a first and short minimization process is performed on all voxels with a pre-defined number of iterations regardless of convergence. After the first minimization process is completed, voxels that have reached convergence are removed from subsequent analysis. Then a second minimization process is performed on the remaining unconverged voxels using the best parameter estimates obtained in the first step as the initial estimates. This multiple-step approach prevents the branching problem associated with early conditional termination of the minimization algorithm on GPU, which can produce unreliable results and decrease the efficiency of GPU computation. Testing on simulated data showed that these changes in code structure do not significantly alter minimization results.

To evaluate the performance of the GPU-accelerated kinetic analysis program developed in this study, both the algorithm that performs the identical analysis on the CPU (constrained Nelder-Mead algorithm with branching and conditional termination removed) and the algorithm optimized for CPU calculation (constrained Nelder-Mead algorithm without branching and conditional termination removed) were implemented to generate results for comparison purposes.

The following data is used to perform voxel-wise kinetic analysis of each DCE-MRI slice: a) ***Time***, a $1 \times n_f$ vector that indicates data acquisition time points, where n_f is the total number of image frames, b) $\boldsymbol{C_v}$, a $1 \times n_f$ vector that describes the vascular contrast agent concentrations in arterial blood, c) $\boldsymbol{C_T}$, an $n_x \times n_y \times n_f$ matrix that describes the contrast agent concentrations in tumor and other brain tissue, where $n_x \times n_y$ are the dimensions of the FOV, and d) ***ROI*** (region of interest), an $n_x \times n_y$ matrix that indicates which voxels in the FOV are in the tumor region ($n_x \times n_y$ for both $\boldsymbol{C_T}$ and ***ROI*** is 256 x 256, since FOV has a dimension of 256 x 256 mm) [19].

All analyses were executed on a 64-bit Windows 7 (Microsoft Corporation, Redmond, WA) desktop computer with Intel Xeon X5690 CPU (Intel Corporation, Santa Clara, CA) and a NVIDIA Tesla C2070 GPU card (NVIDIA, Santa Clara, CA) that contains 448 numerical processor cores and 6 GB onboard SDRAM memory.

Results

Estimations of v_e and K^{trans}

DCE-MRI scans taken one day before (pre-treatment) and one day after (post-treatment) administration of a single 10mg/kg bevacizumab dose in a glioblastoma patient were used to assess the performance of the GPU-accelerated kinetic analysis method implemented in this study. The v_e values derived from a single axial slice within each scan are displayed as heat maps in Fig. 3. The tumor tissue (*ROI*) is clearly visible in the lower right corner of the brain (left and middle panels), and there is an overall decrease in tumor v_e intensity after treatment. The K^{trans} heat maps also display similar patterns as the v_e results (S1 Fig.). Across all *ROI* voxels in this slice, median v_e values (unitless) are 0.242 in the pre-treatment scan and 0.103 in the post-treatment scan, while median K^{trans} values are 0.106 and 0.0943 min^{-1} before and after treatment, respectively (Table 1). These results indicate a 57.4% decrease in tumor v_e and an 11.4% decrease in tumor K^{trans} following treatment. These numbers are comparable with results from past studies [9, 10], and also agree with the expectation that tumor v_e and K^{trans} should decrease after antiangiogenic treatment. Furthermore, results generated from the GPU script were compared to results of a CPU script that performs the same analysis steps in serial to verify that parallelization on the GPU does not alter the outcome of the analysis method: most of the GPU and CPU results (99.9%) are very similar and show less than 5% difference.

Time Performance

The time performance of the GPU-accelerated compartmental modeling analysis method is significantly better than a CPU-based method that performs the same analysis steps in serial. The time results shown here came from timing the analyses of 16 different DCE-MRI slices. The statistical analyses of the performances are shown in Table 2. On average, using a single NVIDIA C2070 GPU card, the GPU-based method could analyze a DCE-MRI slice in ∼44 seconds, which is approximately 124 times faster than the CPU method. Taking the number of voxels included in each analysis into account, the GPU script could analyze ∼593 voxels per second, while the CPU script analyzed fewer than 5 voxels per second. Even when compared to an optimized CPU method that uses a more efficient Nelder-Mead algorithm (with conditional branching and termination check), the GPU method still perform more than 17 times faster.

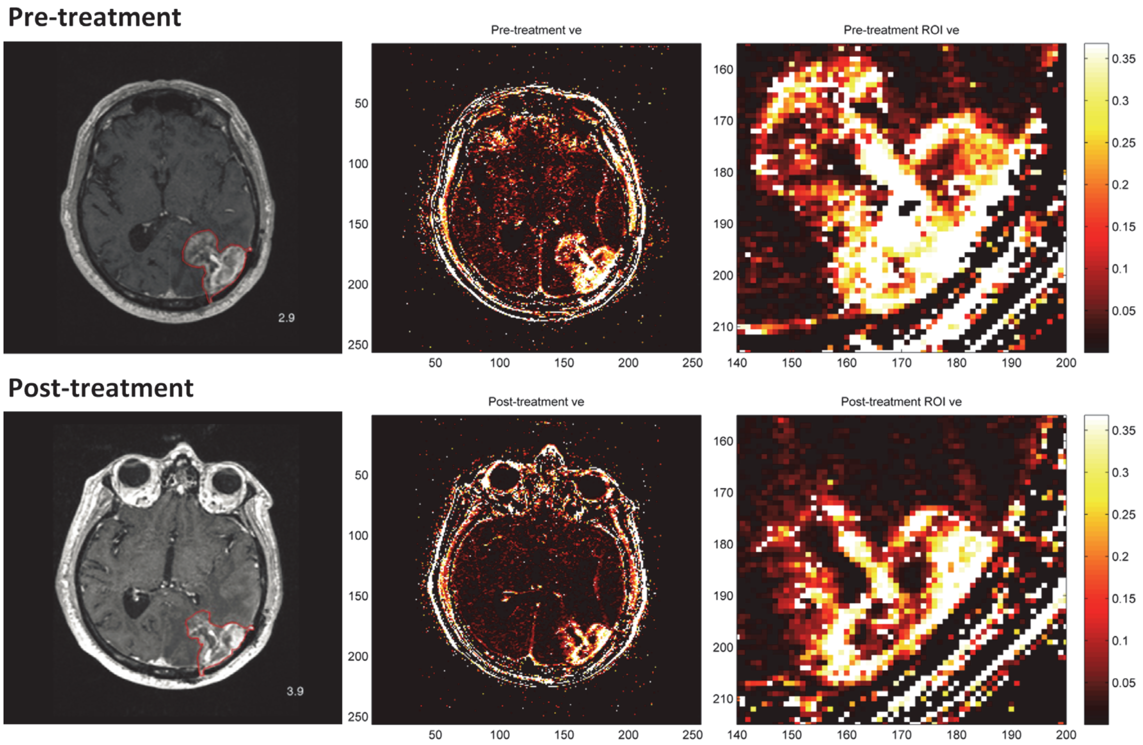

Fig 3. The v_e results of a single axial slice from DCE-MRI brain scans. The top row is from a pre-treatment scan and the bottom row is from a post-treatment scan. The left panel shows the original slice images with the tumor tissue (*ROI*) circled in red, the middle panel displays v_e values derived from the slices, and the right panel shows close-up v_e heat maps of the *ROI*.

doi:10.1371/journal.pone.0118421.g003

Discussion

In this study, we utilized the power of parallel computing and developed a GPU-accelerated compartmental modeling method for analyzing clinical DCE-MRI data. Executed with a single GPU card, our method offers a 124-fold speedup and yields similar parameter estimations compared to a CPU-based method that contains analogous serial code. This article is the first published report of a GPU-based compartmental modeling method of DCE-MRI data, and proposes an efficient way to assess important tumor physiological properties before and after antivascular therapies in clinical patients.

The method described in this study was specifically designed to utilize the power of GPU-computing technology in order to analyze DCE-MRI data efficiently, thus many choices we made during method development reflect this main purpose. In the early phase of development, the computational times of all analysis steps were assessed and evaluated for the feasibility of implementing the program code in a GPU-computing platform. The tissue data fitting process for each DCE-MRI voxel was identified as the most computationally intensive and

Table 1. Median v_e (unitless) and K^{trans} (min^{-1}) values in the *ROI* for data considered in Fig. 3.

Parameter	Pre-treatment	Post-treatment	% change[1]
v_e	0.242	0.103	–57.4
K^{trans}	0.106	0.0943	–11.4

[1] % change: the percentage change in pre- and post-treatment values.

doi:10.1371/journal.pone.0118421.t001

Table 2. Time performance of analyzing the GPU-accelerated method developed in this study versus a CPU-based methods with the same analysis steps.

Method	# of seconds per slice (avg. ± std. dev.[1])	# of voxels per second (avg. ± std. dev. [1])	Compared to GPU
GPU	44.20 ± 1.30	593.72 ± 13.01	1x
CPU (same algorithm structure with GPU)	5468.70 ± 218.82	4.80 ± 0.05	~124x slower
CPU (optimized algorithm structure for CPU)	761.84 ± 34.32	34.46 ± 0.49	~17x slower

[1] avg. ± std. dev.: average ± standard deviation

doi:10.1371/journal.pone.0118421.t002

rate-limiting step of our method because data from each voxel has to be analyzed separately and the Nelder-Mead minimization process has to be performed many times during this step. Consequently, we focused on parallelizing this step on the GPU in order to improve the computing efficiency of our method.

First, we chose to implement a Nelder-Mead algorithm that can simultaneously perform minimization on tissue data from many voxels on the GPU. This simplex-based direct search algorithm was selected over other minimization algorithms because its operations are deterministic and its implementation is relatively straightforward on the GPU. Other deterministic minimization algorithms such as the Gauss-Newton method were considered during early development, but they involve complicated numerical derivative and hessian matrix calculations that are difficult to implement properly or execute efficiently for DCE-MRI analysis on the GPU. A probabilistic search minimization algorithm, simulated annealing, was also tested, but this method produced inconsistent results due to its stochastic nature and would require a large number of iterations to yield more satisfactory results. In comparison, the Nelder-Mead algorithm is a relatively simple derivative-free minimization method that requires a relatively small number of iterations and computations, so it is suited for GPU parallel computing and its results are easily reproducible during testing. After selecting the Nelder-Mead algorithm as the minimization function in our DEC-MRI analysis method, we modified it to optimize its execution on the GPU. In particular, our GPU Nelder-Mead function was designed to run a pre-determined number of iterations on all voxels analyzed in parallel. This design is necessary because conditional termination could not be properly implemented in our Nelder-Mead function: when voxels are analyzed in a single kernel pass on the GPU, if a single voxel's minimization process terminates due to convergence, other voxels' processes would also terminate prematurely, thus producing unrealistic results.

During preliminary analysis, we noted that while the majority of the DCE-MRI voxels reach convergence relatively early in the minimization process, and only a small group of voxels requires many more iterations. For instance, for the data shown in Fig. 3, more than 50% of the voxels converge with fewer than 100 iterations during analysis. Therefore, it is not very time efficient to set a large number of iterations for all voxels while running the Nelder-Mead algorithm. We decided to break up the minimization process into two steps: in the first step, a small number of iterations are performed on all voxels; in the second step, more iterations are performed only on voxels that have not achieved convergence in the first step. Several iteration combinations for the first and second minimization steps were tested during preliminary analysis, and a combination (150 and 200) that had the best time performance for our dataset was used in the final minimization process. Using this design, we were able to further improve the best time performance of the entire GPU-accelerated kinetic analysis script from more than a minute to around 40 seconds.

The Compute Unified Device Architecture (CUDA) is a programming platform developed by NVIDIA for general-purpose computing on CUDA-compatible NVIDIA graphics processor cards. The development of CUDA allows scientific researchers to take advantage of GPU's parallel computing power for complex numerical computations without using any complicated graphics-specific application programming interface [28]. The GPU-computing feature described in this study was implemented in MATLAB using Jacket (now part of its parallel computing toolbox), a numerical computing platform which compiles MATLAB code into CUDA-compatible code that runs on CUDA-enabled GPUs. The parallel analysis steps utilizes the GFOR (GPU for) loop construct implemented in Jacket to simultaneously launch and perform the minimization process on many voxels' data on the GPU [29]. Developing the algorithm using MATLAB is convenient because it allows scientific researchers to rapidly develop prototype programs for GPU-accelerated data analysis. Furthermore, the developed GPU code can easily be converted to standalone, license-free and MATLAB-independent executables for deployment to larger user bases in the scientific community. While the MATLAB script of our GPU-based method performs well compared to CPU methods, potential time performance improvement may be achieved by using a lower-level GPU-computing platform (e.g. C for CUDA) to implement the method. Also, while the results shown in our study were obtained using only a single GPU card, it is expected that our program can be even more efficient in a multi-GPU computing environment. Further study is ongoing to assess the performance improvement our method can achieve using multiple GPU cards.

Conclusion

The findings of this study suggest that GPU-accelerated method in this study can be a practical and efficient alternative to previously reported DCE-MRI compartmental modeling analysis methods. The method significantly reduces the computational time required to obtain important tumor physiological properties, allowing researchers to quickly identify potential regions of interest when analyzing a scan and assess presence or absence of a treatment effect when comparing pre- and post- treatment scans. Furthermore, although this paper focuses on applying the GPU-accelerated method to brain tumor data and uses the extended Tofts model for analysis, the method can easily be extended to analyze various types of DCE-MRI data using various mathematical models. This flexibility would be especially advantageous when a single model is not optimal for all voxels in a given DCE-MRI dataset due to the presence of "kinetic heterogeneity" in malignant tumors [7, 20, 21]. If the fast GPU-based kinetic modeling approach proposed by this study can be perfected, one can construct an efficient DCE-MRI analysis method in which multiple model fittings are performed on each voxel's data in order to find the best-fitting model that accurately represents the underlying physiology of that voxel. Results from this more thorough modeling process would not only provide better parameter estimates of the data, but would also allow the researchers to gain a deeper understanding of the pathology and the drug treatment in question.

Supporting Information

S1 Program. Main MATLAB program code of GPU-accelerated compartment model analysis.
(M)

S2 Program. A GPU-accelerated Nelder-Mead simplex-based minimization algorithm.
(M)

S3 Program. Extended Tofts model that used to depict contrast agent kinetics in tumor and other brain tissue.
(M)

S1 Fig. The K^{trans} results of a single axial slice from DCE-MRI brain scans. The top row is from a pre-treatment scan and the bottom row is from a post-treatment scan. The left panel displays K^{trans} values derived from the slices, and the right panel shows close-up K^{trans} heat maps of the *ROI*.
(TIF)

Acknowledgments

We thank Daniel P. Barboriak (Duke University Medical Center) for providing the clinical DCE-MRI data analyzed in this study.

Author Contributions

Conceived and designed the experiments: YH GF CN. Performed the experiments: YH ZH. Analyzed the data: YH ZH CN. Contributed reagents/materials/analysis tools: YH ZH CN. Wrote the paper: YH ZH GF CN.

References

1. O'Connor JP, Jackson A, Parker GJ, Jayson GC. DCE-MRI biomarkers in the clinical evaluation of antiangiogenic and vascular disrupting agents. Br J Cancer. 2007; 96(2):189–195. PMID: 17211479
2. Bergamino M, Bonzano L, Levrero F, Mancardi GL, Roccatagliata L. A review of technical aspects of T1-weighted dynamic contrast-enhanced magnetic resonance imaging (DCE-MRI) in human brain tumors. Physica Medica. 2014; 30(6):635–643. doi: 10.1016/j.ejmp.2014.04.005 PMID: 24793824
3. Wang C-H, Yin F-F, Horton J, Chang Z. Review of treatment assessment using DCE-MRI in breast cancer radiation therapy. World Journal of Methodology. 2014; 4(2):46–58. doi: 10.5662/wjm.v4.i2.46 PMID: 25332905
4. Zweifel M, Padhani AR. Perfusion MRI in the early clinical development of antivascular drugs: decorations or decision making tools? Eur J Nucl Med Mol Imaging. 2010; 37 Suppl 1:S164–182. doi: 10.1007/s00259-010-1451-z PMID: 20461374
5. O'Connor JP, Jackson A, Parker GJ, Roberts C, Jayson GC. Dynamic contrast-enhanced MRI in clinical trials of antivascular therapies. Nat Rev Clin Oncol. 2012; 9(3):167–177. doi: 10.1038/nrclinonc.2012.2 PMID: 22330689
6. Ferl GZ, Port RE. Quantification of Antiangiogenic and Antivascular Drug Activity by Kinetic Analysis of DCE-MRI Data. Clin Pharmacol Ther. 2012; 92(1):118–124. doi: 10.1038/clpt.2012.63 PMID: 22588603
7. Port RE, Bernstein LJ, Barboriak DP, Xu L, Roberts TPL, van Bruggen N. Noncompartmental Kinetic Analysis of DCE-MRI Data From Malignant Tumors: Application to Glioblastoma Treated With Bevacizumab. Magnetic Resonance in Medicine. 2010; 64(2):408–417. doi: 10.1002/mrm.22399 PMID: 20665785
8. Yang X, Knopp MV. Quantifying tumor vascular heterogeneity with dynamic contrast-enhanced magnetic resonance imaging: a review. Journal of biomedicine & biotechnology. 2011; 2011:732848.
9. Ferl GZ, Xu L, Friesenhahn M, Bernstein LJ, Barboriak DP, Port RE. An automated method for nonparametric kinetic analysis of clinical DCE-MRI data: application to glioblastoma treated with bevacizumab. Magn Reson Med. 2010; 63(5):1366–1375. doi: 10.1002/mrm.22335 PMID: 20432307
10. Ferl GZ. DATforDCEMRI: An R Package for Deconvolution Analysis and Visualization of DCE-MRI Data. Journal of Statistical Software. 2011; 44(3):1–18.
11. Leach MO, Brindle KM, Evelhoch JL, Griffiths JR, Horsman MR, Jackson A, et al. The assessment of antiangiogenic and antivascular therapies in early-stage clinical trials using magnetic resonance imaging: issues and recommendations. Br J Cancer. 2005; 92(9):1599–1610. PMID: 15870830
12. Stone SS, Haldar JP, Tsao SC, Hwu WM, Sutton BP, Liang ZP. Accelerating Advanced MRI Reconstructions on GPUs. J Parallel Distrib Comput. 2008; 68(10):1307–1318. PMID: 21796230

13. Huang S, Xiao S, Feng W, editors. On the energy efficiency of graphics processing units for scientific computing. IPDPS &rsquo;09 Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing; 2009: IEEE Computer Society, Washington, DC, USA.

14. Shams R, Sadeghi P, Kennedy RA, Hartley RI. A Survey of Medical Image Registration on Multicore and the GPU. Ieee Signal Processing Magazine. 2010; 27(2):50–60.

15. List. Top 500 Supercomputer Sites; 2014; Available from: http://www.top500.org/lists/2014/06/.

16. Eklund A, Dufort P, Forsberg D, LaConte SM. Medical image processing on the GPU—Past, present and future. Medical Image Analysis. 2013; 17(8):1073–1094. doi: 10.1016/j.media.2013.05.008 PMID: 23906631

17. Pratx G, Xing L. GPU computing in medical physics: A review. Medical Physics. 2011; 38(5):2685–2697. PMID: 21776805

18. Shi L, Liu W, Zhang H, Xie Y, Wang D. A survey of GPU-based medical image computing techniques. Quantitative Imaging in Medicine and Surgery. 2012; 2(3):188–206. doi: 10.3978/j.issn.2223-4292.2012.08.02 PMID: 23256080

19. Hsu YH, Ferl GZ, Ng CM. GPU-accelerated nonparametric kinetic analysis of DCE-MRI data from glioblastoma patients treated with bevacizumab. Magnetic resonance imaging. 2013; 31(4):618–623. doi: 10.1016/j.mri.2012.09.007 PMID: 23200680

20. Port RE, Knopp MV, Hoffmann U, Milker-Zabel S, Brix G. Multicompartment analysis of gadolinium chelate kinetics: Blood-tissue exchange in mammary tumors as monitored by dynamic MR imaging. J Magn Reson Imag. 1999; 10(3):233–241. PMID: 10508282

21. Tofts PS. Modeling tracer kinetics in dynamic Gd-DTPA MR imaging. J Magn Reson Imaging. 1997; 7 (1):91–101. PMID: 9039598

22. Rowland M, Tozer T. Clinical pharmacokinetics: concepts and applications. 3rd ed. Baltimore: Williams &Wilkins; 1995.

23. Benet LZ. General treatment of linear mammillary models with elimination from any compartment as used in pharmacokinetics. Journal of Pharmaceutical Sciences. 1972; 61(4):536–541. PMID: 5014309

24. Nakashima E, Benet L. An integrated approach to pharmacokinetic analysis for linear mammillary systems in which input and exit may occur in/from any compartment. Journal of Pharmacokinetics and Biopharmaceutics. 1989; 17(6):673–686. PMID: 2635739

25. Vredenburgh JJ, Desjardins A, Herndon JE 2nd, Dowell JM, Reardon DA, Quinn JA, et al. Phase II trial of bevacizumab and irinotecan in recurrent malignant glioma. Clin Cancer Res. 2007; 13(4):1253–1259. PMID: 17317837

26. Nelder JA, Mead R. A Simplex-Method for Function Minimization. Computer Journal. 1965; 7(4):308–313.

27. Oldenhuis R. Minimize (File ID: #24298). MATLAB Central File Exchange; 2009 [Jan 1, 2012]; Available from: http://www.mathworks.com/matlabcentral/fileexchange/24298.

28. Sanders J, Kandrot E. CUDA by example: an introduction to general-purpose GPU programming. Upper Saddle River, NJ: Addison-Wesley; 2011. xix, 290 p. p.

29. McClanahan C. Jacket for Multidimensional Scaling in Genomics. GPU Technology Conference. 2012.