GPU Accelerated Simulations of Magnetic Resonance Imaging of Vascular Structures

Krzysztof Jurczuk[1(✉)], Dariusz Murawski[1], Marek Kretowski[1], and Johanne Bezy-Wendling[2,3]

[1] Faculty of Computer Science, Bialystok University of Technology, Wiejska 45a, 15-351 Bialystok, Poland
{k.jurczuk,m.kretowski}@pb.edu.pl
[2] INSERM, U1099, 35000 Rennes, France
[3] University of Rennes 1, LTSI, 35000 Rennes, France

Abstract. Computer simulations of magnetic resonance imaging (MRI) are important tools in improving this imaging modality and developing new imaging techniques. However, MRI models often have to be simplified to enable simulations to be carried out in a reasonable time. The computational complexity associated with the tracking of magnetic fields' perturbations with high spatial and temporal resolutions is very high and, thus, it calls for using parallel computing environments. In this paper, we present a GPU-based parallel approach to simulate MRI of vascular structures. The magnetization calculation in different spatial coordinates is spread over GPU cores. We apply CUDA framework and take advantage of GPU memory hierarchy to efficiently exploit GPU computational power. Experimental results with different GPUs and various images show that the proposed algorithm substantially speedups the simulation. The proposed GPU-based approach may be easily adopted in modeling of other flow related phenomena like perfusion or diffusion.

Keywords: Computer simulation · Graphics processing unit (GPU) · Magnetic resonance imaging (MRI) · Parallel computing · Vascular structures

1 Introduction

Computational modeling and computer simulations play a very important role in contemporary science and industry. In medical field, they provide an additional insight to increase the comprehension of complex life processes that can be obscured during both *in vitro* and *in vivo* experiments [1]. Computer simulations are also usually cheaper and less time consuming than experimental investigations. Moreover, they can be performed with no need for patients to participate and thus are not limited by the examination duration.

In our research, we focus on modeling of magnetic resonance imaging (MRI) that is one of the most important diagnostic tools in modern medicine [2]. Although MRI is known as a highly detailed 3D imaging modality, there is still

R. Wyrzykowski et al. (Eds.): PPAM 2015, Part I, LNCS 9573, pp. 389–398, 2016.
DOI: 10.1007/978-3-319-32149-3_37

a lot of difficulties in magnetic resonance (MR) image formation and interpretation, especially in the area of blood flow. Imaging of such areas is crucial since vascular diseases are the cause of large mortality rate in the populations of developed countries [3]. Thus, we concentrate mainly on MRI of vascular structures.

The modeling of MR flow imaging is not a trivial task since it requires the integration of many processes/phenomenon (like physiology, hemodynamics, anatomy, imaging technology) in one computational model [4]. The challenge is also the decision about level of model detail (closeness to reality) as well as mathematical modeling itself. Another issue is high computational complexity of the model and, thus, the need for high performance computing to face up to such simulations. Many computations are required to simulate each physical phenomenon itself as well as interactions between them. The computational needs grow fast with the size of vascular structures. Large vascular simulations were shown to be vital to correctly investigate internal processes in human bodies [5].

There have been proposed many approaches in modeling of MR flow imaging [6–8] to name a few. The long simulation time was always one of the main factors limiting the extension of these models to a 3D version or to study complex vascular networks. Thus, it seems that further progress in computational modeling of MRI does not only depend on sophisticated equations, but also on the development of parallel architectures and algorithms. In our recent study [4], we successfully applied cluster computing in modeling of MR flow imaging. It allowed complex vascular structures to be investigated.

In this paper, we propose a GPU-based approach in modeling of MR flow imaging. Our motivation is to bring the possibility to perform MRI simulations fast on a single workstation. This way, complex vascular structure simulations can become independent of computer clusters that might be expensive, maintenance demanding and are not always accessible. In addition, modern GPUs are more often able to provide higher price/performance factor than computer clusters. What is also important that the proposed parallel approach may be easily applied in modeling of other flow related imaging like perfusion or diffusion MRI.

As far as we know, the proposed GPU-based algorithm is the first approach in modeling of MRI of vascular structures. Although, a GPU-based MRI simulator [9] was published recently, it allows only the stationary magnetization (without blood flow) to be investigated. The same research group also extended their simulator to model various motions [10]. However, their extended solution does not still enable vascular structures to be taken into account since the magnetization transport model (based blood flow) is significantly simplified.

In the next section, the model of MR flow imaging is described. In Sect. 3 the GPU-based approach is proposed. Section 4 presents the evaluation of the GPU-based algorithm. Section 5 provides conclusion and plans for future research.

2 Computational Model Description

In our previous research, we developed a three-component model of MR flow imaging [4] (Fig. 1). The first component is used for vascular structures generation based on physiological and hemodynamic parameters [11]. The second one

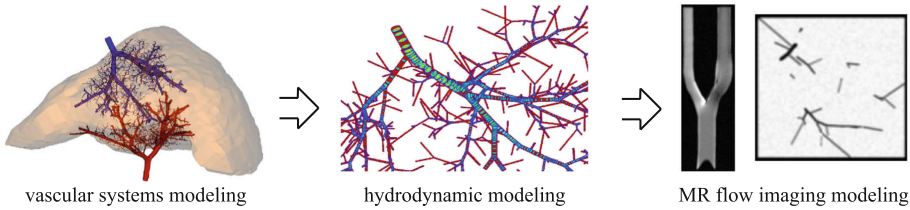

Fig. 1. Three-component MRI model overview. The model allows us to generate vascular structures, then simulate flow behaviour and finally reproduce MRI processes.

allows flow simulations to be performed in the generated vascular structures [8]. The last component makes use of the generated vascular structures and flow characteristics to simulate MRI of vascular structures. Since, in this paper, the algorithm of imaging is parallelized, only the last model component is described.

The 3D imaged area (object) is divided into cubic elements [8]. To each cubic element basic MR parameters (proton density, relaxation times), determined by the represented part of a tissue, are assigned. In addition, each cubic element contains hydrodynamic parameters, i.e. mean flow velocity and direction of the fluid filling it (generated by the flow model). For stationary tissue structures (e.g. bones, vessel walls), the flow velocity equals zero.

Imaging simulation is divided into small time portions called time steps Δt. After each time step, local magnetizations of all cubic elements are modified taking into account both the flow influence ($\Delta \mathbf{M}_F$) and MRI processes ($\mathbf{A}_{MRI}$):

$$\mathbf{M}(\mathbf{r}, t + \Delta t) = \mathbf{A}_{MRI}(\mathbf{r}, \Delta t)[\mathbf{M}(\mathbf{r}, t) + \Delta \mathbf{M}_F(\mathbf{r}, \Delta t)] \,, \tag{1}$$

where $\mathbf{M}$ is the magnetization of the cubic element at spatial position $\mathbf{r}$.

First, the flow influence is computed (see left part of Fig. 2). In each cubic element the magnetization fractions are propagated to the neighboring cubic elements (see black rectangles labeled by "a", "b" and "c") based on the flow velocity and direction ($\mathbf{u} = u_x\hat{\mathbf{i}} + u_y\hat{\mathbf{j}} + u_z\hat{\mathbf{k}}$, where $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, $\hat{\mathbf{k}}$ are unit vectors in x, y, z directions). This way, parts of magnetization can leave some cubic elements. At the same time, the magnetization fractions that leave some cubic elements enter to neighboring cubic elements. The mean magnetization changes during a time step Δt for a cubic element at $\mathbf{r}$ position are modeled as follows $\Delta \mathbf{M}_F(\mathbf{r}, \Delta t) = \Delta \mathbf{M}_{IN}(\mathbf{r}, \Delta t) - \Delta \mathbf{M}_{OUT}(\mathbf{r}, \Delta t)$ where:

$$\begin{aligned} \Delta \mathbf{M}_{IN}(\mathbf{r}, \Delta t) = {} & \mathbf{M}\left(\mathbf{r} - \Delta d\frac{u_x(\mathbf{r})}{|u_x(\mathbf{r})|}\hat{\mathbf{i}} - \Delta d\frac{u_y(\mathbf{r})}{|u_y(\mathbf{r})|}\hat{\mathbf{j}}, t\right)|u_x(\mathbf{r})||u_y(\mathbf{r})| + \\ & \mathbf{M}\left(\mathbf{r} - \Delta d\frac{u_x(\mathbf{r})}{|u_x(\mathbf{r})|}\hat{\mathbf{i}}, t\right)|u_x(\mathbf{r})|(1 - |u_y(\mathbf{r})|) + \\ & \mathbf{M}\left(\mathbf{r} - \Delta d\frac{u_y(\mathbf{r})}{|u_y(\mathbf{r})|}\hat{\mathbf{j}}, t\right)(1 - |u_x(\mathbf{r})|)|u_y(\mathbf{r})|, \end{aligned} \tag{2}$$

$$\begin{aligned} \Delta \mathbf{M}_{OUT}(\mathbf{r}, \Delta t) = {} & \mathbf{M}(\mathbf{r}, t)[|u_x(\mathbf{r})||u_y(\mathbf{r})| + \\ & |u_x(\mathbf{r})|(1 - |u_y(\mathbf{r})|) + (1 - |u_x(\mathbf{r})|)|u_y(\mathbf{r})|]. \end{aligned} \tag{3}$$

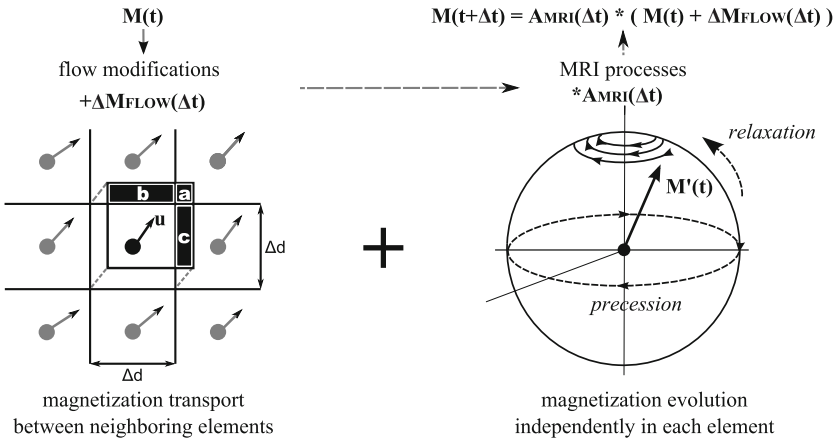

Fig. 2. The connection of MRI and magnetization transport algorithms.

In 3D modeling, each sum component in Eqs. 2 and 3 is made up of two more cases, accordingly, with $|u_z(\mathbf{r})|$ or $(1 - |u_z(\mathbf{r})|)$ term.

Later, the MRI influence is computed (right part of Fig. 2). It is modeled by the Bloch equation. We apply its discrete solution in the form of rotation matrices and exponential scaling [12]. Such an approach allows us to track the magnetization changes ($\mathbf{A}_{MRI}$) induced by MRI events without any integration:

$$\mathbf{A}_{MRI}(\mathbf{r}, \Delta t) = \mathbf{E}_{RELAX}(\mathbf{r}, \Delta t)\mathbf{R}_z(\Theta)\mathbf{R}_{RF}(\mathbf{r}, \Delta t) \ , \tag{4}$$

where $\mathbf{E}_{RELAX}$ is responsible for the relaxation phenomena, $\mathbf{R}_z$ is the rotation matrix about the z-axis through angle Θ used to model the influence of spatial encoding gradients and magnetic field inhomogeneities, while $\mathbf{R}_{RF}$ reproduces the excitation process [8].

Based on Faraday's law of an electromagnetic induction, the MR signal coming from the imaged object at a time t is expressed as a sum of the transverse magnetizations of all cubic elements. In MRI, such a signal is acquired many times during successive sequence repetitions, with different phase encoding gradients. These different gradients are used to encode spatial positions of particular signals that compose the total received signal. Within a single repetition period, the acquired signal fills one k-space (readout) matrix row. The MR image is created by applying the fast Fourier transform (FFT) to the fully filled matrix.

This coupled MR flow imaging model closely follows the physical process of MRI. In addition, it does not need additional mechanisms to consider the blood flow influence during most of MRI events, in contrast to prior works [6,7].

3 GPU-based Parallelization of MR Flow Imaging

The most time consuming part of the algorithm is the calculation of new magnetization values ($\mathbf{M}(\mathbf{r}, t + \Delta t)$). It has to be done many times after each time

step (Δt) that ought to be small enough (usually of the order of tens to hundreds of microseconds). Second, in each time step, both the blood flow and MRI influences have to be taken into account in each cubic element (which size is usually of the order of tens to hundreds of micrometers). Thus, data decomposition strategy is naturally applied (see Fig. 3).

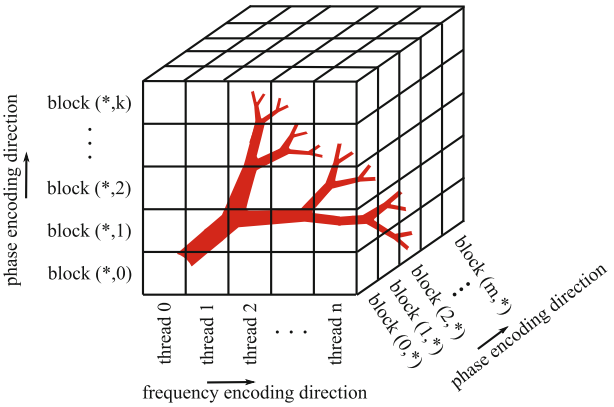

Fig. 3. Data decomposition strategy: 2D grid of blocks along the phase encoding directions and 1D block of threads along the frequency encoding direction.

We decide to use 2D grid of blocks along the directions of phase encoding gradients. The cubic elements along the direction of frequency encoding gradient are, in turn, spread over threads. By default, the number of blocks and threads are set to the number of cubic elements (object size) in the considered directions. If the object size excesses the maximum number of threads/blocks (which is hardware dependent), then the maximum value is used and a single thread/block processes more than one cubic element.

Figure 4 presents the idea behind the GPU-based algorithm of MR flow imaging. At the beginning, at host (CPU), the 3D object and the experiment parameters are initialized. Then, they are sent to the device (GPU) and saved in the allocated space in global memory. This CPU to GPU data transfer is performed only once before MRI procedure and this data is kept there during the whole MRI experiment. This way, each thread has an access to this data. Both the object (flow, MR parameters) and tracked magnetization values are organized as 3D matrices of floating point values arranged in one-dimensional arrays. The results (k-space matrix that is also arranged in a one-dimensional array) are transferred to CPU when all MRI repetitions are finished.

After the initialization, the MR image formation is started and here the GPU-based parallelization is applied. In each excitation/repetition, various MRI events are simulated and finally a single line of the MR signal is acquired and saved in the k-space matrix. Each MRI event (excitation, spatial encoding, relaxation, ...)

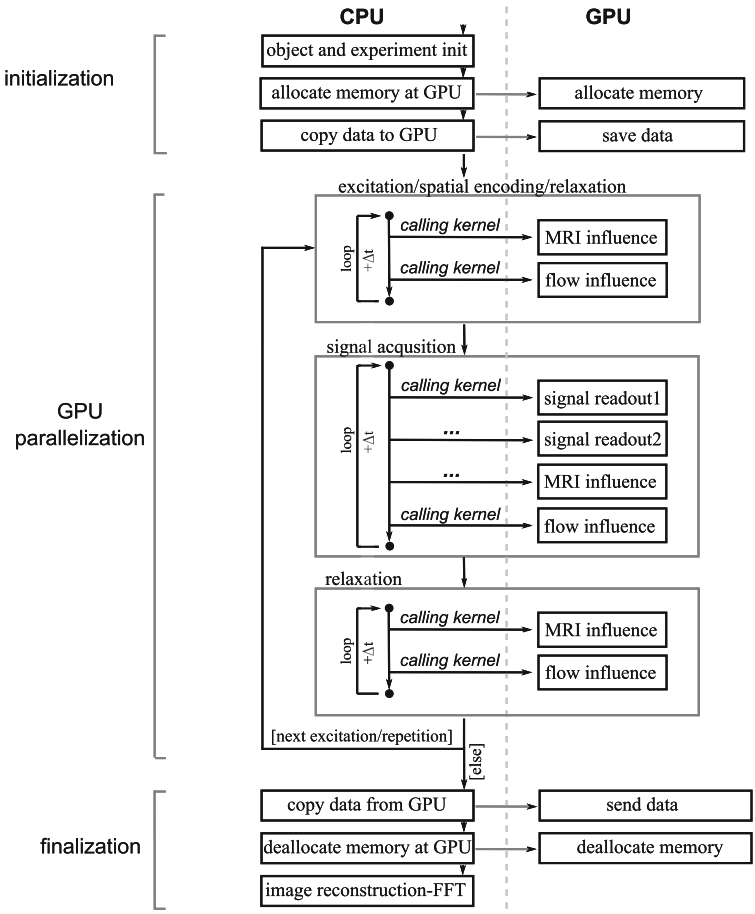

Fig. 4. The flowchart of the GPU-based algorithm of MR flow imaging

is performed in consecutive time steps (Δt). The magnetization of a cubic element in the next time step depends on its current value (time dependency), the MRI influence and magnetization of neighboring cubic elements (spatial dependency), accordingly with Eq. 1. Thus, in each time step, two kernel functions are called. The first one is responsible for the MRI influence, while the second one for the magnetization transport induced by blood flow. Such an approach provides the synchronization for all threads (both inside and between blocks) after each time step. Thereby, the time and spatial dependency between magnetizations of cubic elements is provided. During the MRI influence calculations, each thread accesses only the local magnetization values, while the magnetization transport phase needs to reach into the neighboring cubic elements.

The signal acquisition phase differs a little from other MRI events. Here, not only the MRI and flow influences are simulated, but also the signal from all cubic

elements has to be read and saved in the k-space matrix. This operation is carried out by two additional kernel functions (*signal readout1* and *signal readout2*, see Fig. 4). These two functions, in a single time step, provide the sum of all magnetizations (from all cubic elements). The first kernel performs the reduction of magnetization values for cubic elements inside blocks. It uses the mechanism of shared memory inside thread blocks. The second kernel finishes the reduction with the use of one block and a table in global memory.

After the signal acquisition, the relaxation process is simulated and if the next repetition is needed, the algorithm starts again from the excitation. Otherwise, the k-space matrix is transferred from the device to the host. Finally, the MR image is created by the application of FFT to the received matrix at CPU.

In order to efficiently use hardware resources, we take advantage of memory hierarchy of modern GPUs. The following algorithm improvements, among others, are applied: *(i)* Data stored in global memory and frequently used (e.g. cubic element magnetization) is transferred to local variables at the beginning of kernel functions. At the end of the kernel, the data is transferred back to global memory. *(ii)* In the serial algorithm, the results of some repeated calculations (e.g. partial computation of the magnetization increase during relaxation after Δt time step) are saved in auxiliary tables before the MRI procedure and used when needed. In the GPU-based algorithm, we do not use such global auxiliary tables. Results of frequently repeated calculations are saved locally in a kernel function when they are done for the first time. Such a mechanism increases the number of arithmetic operations but, at the same time, reduces redundant loads from GPU global memory. *(iii) float4* data type is used to store magnetization description in GPU global memory, despite the fact that the magnetization is a 3D vector and we only need three float values. This way, coalesced memory access may be provided, resulting in efficient memory requests and transfers [13]. As regards local thread variables, temporary magnetizations are stored in *float3* variables since in threads register space is an important issue.

4 Experimental Results

All experiments were performed on a workstation equipped with a quad-core processor (Intel Core i7-870, 8M Cache, 2.93 GHz), 32 GB RAM and a single graphics card. We tested three different Kepler-based NVIDIA graphics cards: *(i)* GeForce GTX 780 (3 GB memory, 2304(12×192) CUDA cores), *(ii)* Quadro K5000 (4 GB memory, 1536(8×192) CUDA cores) and *(iii)* GeForce GTX Titan Black (6 GB memory, 2880(15×192) CUDA cores).

We used 64-bit Ubuntu Linux 14.04.02 LTS as an operating system. The sequential algorithm was implemented in C++, compiled with the use of gcc version 4.8.2 and optimized many times since of long simulation time. The GPU-based parallelization was implemented in CUDA-C and compiled by nvcc CUDA 7.0 [14]. Single precision calculations were analyzed.

We present results for four various vascular structures investigated in our previous papers [4,8], where detailed flow and MRI settings were reported.

Figure 5 shows examples of simulated MR flow images representing more and more complicated vascular structure. The gradient echo (GE) imaging sequence was used to obtained the images. Other basic imaging parameters are reported next to the MR images.

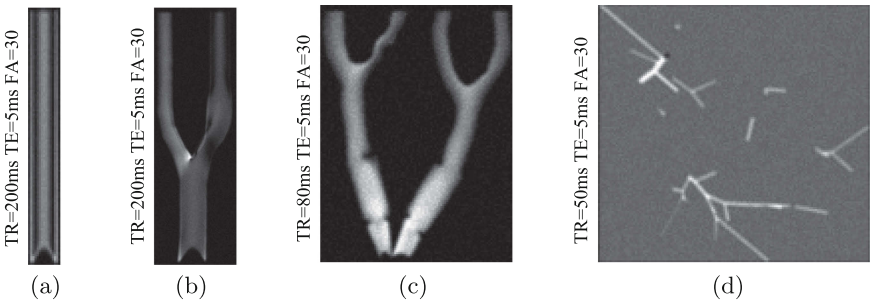

Fig. 5. Simulated MR images: (a) a single vessel, (b) a single bifurcation, (c) multiple bifurcations, (d) slice through the liver volume with many vascular structures.

Figure 6 shows the obtained mean speedup (over different imaging parameters) for four tested vascular structures, relative to the sequential implementation run on a single core. The speedup obtained with the use of four CPU cores by an OpenMP parallelization is also presented. It is clearly visible that the GPU-based parallelization provides a significant decrease in computation time. All GPUs are able to obtain a speedup at least one order higher than the OpenMP parallelization. Currently, the time needed to simulate e.g. MR image in Fig. 5(d) equals about 1.5 h with the use of Titan Black GPU, instead of about 10 days by a single core CPU or 5 days using OpenMP parallelization and four CPU cores. Moreover, the achieved speedup is comparable (images (a–c)) or even higher (image (d)) than the one obtained by a computer cluster of 16 nodes each equipped with 2 quad-core CPUs (Xeon 2.66 GHz) and 16 GB RAM [4].

In Fig. 6 we can also see that the tested GPUs provide different speedups. Concerning GTX Titan Black and GTX 780 GPUs, the difference is quite small and it could be explained by a slightly faster memory (7.0 vs 6.0 Gbps) and a stronger computational unit (2880 cores of 980 MHz vs. 2304 cores of 900 MHz) of the former one. As regards Quadro K5000 GPU, it achieves much smaller acceleration than other tested GPUs. Quadro family GPUs are known to be designed to especially accelerate applications to design, rendering and 3D visual modeling (like CAD software) at the expense of lower computational performance in games and GPGPU. They are also usually more expensive than GTX series GPUs and offer lower power consumption. Anyway, the obtained results show that such a engineering workstation equipped with a Quadro family GPU can also be successfully used in fast MR flow imaging simulations. Considering the price/performance factor, the GTX 780 GPU wins since, currently, it is about 3 times cheaper than Titan Black and 5 times than Quadro K5000.

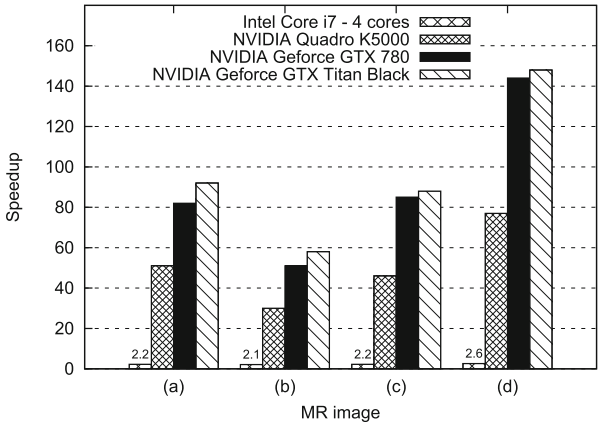

Fig. 6. Speedup of the GPU (as well as multi-core) accelerated algorithms relative to the serial implementation run on a single core of Intel Core I7 CPU. Images shown in Fig. 5 and various GPUs are tested.

There is also a difference in speedup between tested objects. The first reason could be the fraction of cubic elements that represent the object/image background where no MRI is simulated (black areas in images). The second one is the size of the object/image that does not always allow computational resources to be exploited efficiently. The object size and fraction of background elements were as follows: (a) 106×739, 0.37, (b) 167×543, 0.6, (c) 572×740, 0.7, (d) 1000×1000×70, 0. The simulation box can differ from the object size due to additional background cubic elements added to objects to apply FFT easily [8]. The worst results were obtained for the single bifurcation object where there was a lot of background elements and the horizontal object (167) as well as simulation box size (320) was too small to provide load balancing. The best speedup, in turn, was achieved for image (d) since there was no background elements and object size was big enough. We verified execution times with respect to the object size for the straight vessel geometry (Fig. 5(a)) (72×705, 144×1410 and 288×2820 original/flow grid size). The obtained speedup grew with the increase of object size, from about 80 to 120, respectively. The thorough evaluation of the scaling properties of the algorithm was left for future investigation.

5 Conclusion

In this paper we propose a GPU-based parallel approach to simulate MRI of vascular structures. The approach was tested at various GPUs and vascular structures. The results clearly show that it provides a significant speedup relative to the CPU implementation. Thus, it opens a perspective to simulate more and more complex (that is more realistic) vascular structures. Moreover, the proposed GPU-based approach may be easily adapted in other algorithms concerning related phenomena like diffusion or perfusion.

In the future, we want to deal with multi-GPU approaches, both at a single workstation as well as on computing clusters. We also plan to use the presented parallelization in the modeling of contrast agent propagation and dynamic contrast-enhanced MRI [15].

Acknowledgments. This work was supported by the grants W/WI/2/2014 and S/WI/2/2013 from Bialystok University of Technology.

References

1. Hoppensteadt, F.C., Peskin, C.S.: Modeling and Simulation in Medicine and the Life Sciences. Springer-Verlang, New York (2004)
2. Westbrook, C., Roth, C.K., Talbot, J.T.: MRI in Practice. Wiley-Blackwell, Malden (2011)
3. Aiyagari, V., Gorelick, P.B.: Hypertension and Stroke: Pathophysiology and Management. Humana Press, Totowa (2011)
4. Jurczuk, K., Kretowski, M., Eliat, P.-A., Saint-Jalmes, H., Bezy-Wendling, J.: In silico modeling of magnetic resonance flow imaging in complex vascular networks. IEEE Trans. Med. Imaging **33**(11), 2191–2209 (2014)
5. Grinberg, L., Cheever, E., Anor, T., Madsen, J.R., Karniadakis, G.E.: Modeling blood flow circulation in intracranial arterial networks: A comparative 3-D/1-D simulation study. Ann. Biomed. Eng. **39**(1), 297–309 (2011)
6. Lorthois, S., Stroud-Rossman, J., Berger, S., Jou, L.D., Saloner, D.: Numerical simulation of magnetic resonance angiographies of an anatomically realistic stenotic carotid bifurcation. Ann. Biomed. Eng. **33**(3), 270–283 (2005)
7. Marshall, I.: Computational simulations and experimental studies of 3D phase-contrast imaging of fluid flow in carotid bifurcation geometries. J. Magn. Reson. Imaging **31**(4), 928–934 (2010)
8. Jurczuk, K., Kretowski, M., Bellanger, J.-J., Eliat, P.-A., Saint-Jalmes, H., Bezy-Wendling, J.: Computational modeling of MR flow imaging by the lattice Boltzmann method and Bloch equation. Magn. Reson. Imaging **31**(7), 1163–1173 (2013)
9. Xanthis, C.G., Venetis, I.E., Chalkias, A.V., Aletras, A.H.: MRISIMUL: A GPU-based parallel approach to MRI simulations. IEEE Trans. Med. Imaging **33**(3), 607–617 (2014)
10. Xanthis, C.G., Venetis, I.E., Aletras, A.H.: High performance MRI simulations of motion on multi-GPU systems. J. Cardiovasc. Mag. Reson. **16**, 48 (2014)
11. Kretowski, M., Rolland, Y., Bezy-Wendling, J., Coatrieux, J.-L.: Physiologically based modeling for medical image analysis: application to 3D vascular networks and CT scan angiography. IEEE Trans. Med. Imaging **22**(2), 248–257 (2003)
12. Bittoun, J., Taquin, J., Sauzade, M.: A computer algorithm for the simulation of any nuclear magnetic resonance (NMR) imaging method. Magn. Reson. Imaging **2**(2), 113–120 (1984)
13. NVIDIA, CUDA C Best Practices Guide in CUDA Toolkit Documentation v7.0. http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/
14. Wilt, N.: Cuda Handbook: A Comprehensive Guide to GPU Programming. Addison-Wesley, Boston (2013)
15. Mescam, M., Kretowski, M., Bezy-Wendling, J.: Multiscale model of liver DCE-MRI towards a better understanding of tumor complexity. IEEE Trans. Med. Imaging **29**(3), 699–707 (2010)