Fully Parallelized Lattice Boltzmann Scheme for Fast Extraction of Biomedical Geometry

Zhiqiang Wang[a,*], Ye Zhao[a,*], Huidan (Whitney) Yu[b], Chen Lin[c], Alan P. Sawchuck[c]

[a]*Department of Computer Science, Kent State University, Kent, Ohio, USA*
[b]*Mechanical Engineering Department, Indiana University-Purdue University, Indianapolis, Indiana, USA*
[c]*School of Medicine, Indiana University, Indiana, USA*

Abstract

We develop a fully parallel numerical method which quickly performs 2D and 3D segmentation on GPU to extract anatomical structures from medical images. The algorithm solves the level set equations completely within a Lattice Boltzmann model (LBM). Compared with existing LBM-based segmentation approaches, a parallel distance field regularization is added in the LBM computing scheme to keep computation stable with large time step iteration. This approach also avoids external regularization which has been a major impediment to direct parallelization of level set evolution with LBM. It allows the whole computing process to be efficiently executed on GPU. Moreover, the method can be incorporated with different image features to adopt in various image segmentation tasks. Therefore, our method enables fully GPU accelerated geometric extraction from medical images, leading to high computing performance which is demanded in many practical applications. This method is used to exact accurate 2D and 3D anatomical structures from many real world CT and MRI images. The achieved results can also directly feed required boundary information to LBM-based hemodynamic simulation.

Keywords: Parallel Image Segmentation, Biomedical Geometry Extraction, Lattice Boltzmann Method, GPU Computing

*Corresponding authors, Email: zwang22@kent.edu; zhao@cs.kent.edu

1. Introduction

Segmenting 3D geometry from large biomedical images is an important task for extracting anatomical structures, identifying their features, and facilitating biomedical engineering tasks. In clinical research and applications, efficiently extracting anatomical structures from medical images is a key step to offer patient-specific diagnosis in a timely manner and make large population studies possible. For example, Patient-Specific Computational Hemodynamics (PSCH) simulates blood flows in the arteries extracted from patients' angiography data [1]. Punctual and accurate segmentation is demanded in such applications to promote clinical analysis and assessment. In this paper, we propose a completely parallel computing scheme of active contour models for biomedical geometry extraction. The model is built up on solving level set equations (LSE) with a fully parallelized lattice Boltzmann model (LBM), enabling direct GPU acceleration to achieve very fast geometry extraction.

Level set methods have been successfully employed in image segmentation by tracking active contours to match geometric boundaries. They create accurate geometries from noisy raw data and easily handle complex topology. However, solving level set equations cannot achieve time efficiency easily. Many numerical algorithms based on explicit finite difference discretization have to use very small time steps for stable computation. Consequently, a large number of numerical iterations make the entire segmentation procedure rather time-consuming. Therefore, implicit numerical approaches are used to overcome the problem [2]. However, implicit methods are difficult to parallelize since they need to solve a global linear system. Other strategies including narrow band [3] and multigrid method [4] can improve the efficiency by limiting the computation in part of the whole domain. Some researchers have deliberately designed GPU implementations to accelerate these approaches [5]. They built heterogeneous data structures, such as virtual memory system [6] or dynamic list [7], to maintain irregular and dynamic computing domain. For biomedical geometry extraction tasks, the boundary structures of target objects (e.g., arteries) are often very complex, and sometimes they span over the whole domain. In such cases, the large un-coalesced GPU memory access would decrease data transfer bandwidth and limit the achieved performance.

Recently, LBM has been developed as an new numerical method for solving LSEs [8, 9, 10, 11, 12, 13, 14]. The explicit scheme of LBM is second order accurate and can utilize larger time step than direct LSE discretization. Moreover, its computing scheme is very simple to program and inherently parallel with local data access, making it greatly amenable for parallel acceleration. However, there still exist gaps between existing LBM works and a fully parallel instrument to solving LSE in 3D biomedical geometry extraction. In this paper, our LBM scheme provides a complete GPU-based solution of fast medical image segmentation. The main contributions of our method, compared with existing LBM approaches, are as follows:

First, we develop both 2D and 3D LBMs to perform fast segmentation over images and volumetric data. The proposed algorithms can iterate with large time steps. Meanwhile, they are suitable for parallel computing. In contrast, most of the existing LBM methods are aimed only at 2D image segmentation and do not need to consider parallel computing efficiency. Alternatively, extracted contours over 2D slices are connected to create a 3D shape [14], which however, is not accurate and smooth. A simple 3D LBM is implemented for segmentation [9], but the method is not effective for noisy images, since it only uses a linear diffusion and a simple pixel difference comparison for edge detection.

Second, our method is the first to integrate distance field regularization into the LBM computing scheme. It does not need to stop the LBM simulation and explicitly locate the zero level set by recomputing Euclidean distance as existing works do. Therefore, our approach only adds minimal computation load with no extra memory consumption. Moreover, the computation is embarrassingly parallel and robust in noisy image.

The distance field regularization is an essential part of tracking active contours, which is required to maintain the correct distance function when numerically solving the LSE. Many existing LBM segmentation methods do not perform this step (e.g., [10] and [12]). Consequently, the gradients of distance field are unbounded during iterations leading to inaccurate and rough results. Some approaches address this problem out of the LBM framework. In particular, Yang et al. [13] successfully segmented 2D auroral oval images by combining LBM with narrow band methods. They

updated zero level set by a sparse field method inside a narrow band. This approach is hard to parallelize since it needs to explicitly find the zero level set and compute the distances from those pixels to the contour. In 2D image segmentation, Sun et al. [11] repeatedly recomputed the Euclidean distance of each pixel to zero level set contours after several iterations. In these methods, the regularization of distances involves global contour information, which is implemented out of the LBM framework. As a result, these methods are not easily extended to 3D image segmentation accelerated on GPU. On the other hand, Chen et al. [14] added a penalty term to LSE to force the distance field smooth based on [15, 16]. This term may adversely move the zero level set, and eventually the active contour cannot converge to achieve correct results. Thus, this method has to add an extra edge detection step by Canny operator to handle noisy images. In these methods, the whole computing process is no longer fully parallel, so that the advantage of LBM-based segmentation is not completely exploited. Extra CPU computation and GPU-CPU data exchange is needed which can greatly impede acceleration performance. In contrast, our approach implements the distant field regularization inside LBM computing processes.

Third, our method offers a generalized LBM scheme to solve Geometric Active Contour (GAC) models. Different LSE approaches that use various image features (e.g., edge and region information) can be directly implemented within the scheme. The scheme can thus be adopted in different image segmentation tasks.

We apply our method to CT and MRT image datasets to show its quality and performance. The remainder of the paper is organized as follows. In Section 2, we introduce the level set segmentation method. Then our fully parallelized LBM is presented in Section 3. The acceleration on GPU is discussed in Section 4. Section 5 provides a set of examples using our method in 2D and 3D geometric extraction. Finally, we conclude the paper in Section 6.

2. Level Set Segmentation

2.1. Level Set Equation

Level set methods track an active contour (or a 3D evolving surface) which evolves to match structural boundaries in image or volume datasets, where a distance field

implicitly represents the contour or surface C. A signed distance field, $\phi : R^3 \rightarrow R$ for $p \in R^3$, is defined as the closest distance to C with the function:

$$\phi(p) = sign(p) \cdot min\{|p-q| : q \in C\} \tag{1}$$

where a positive distance refers to outside of C and a negative distance means inside of C. Then, C can be seen as the zero level set including all points with zero distance.

In image segmentation, the Geometric Active Contour (GAC) starts from an arbitrary starting shape and evolves itself by a particular LSE [17]:

$$\frac{\partial \phi}{\partial t} = div(\alpha \frac{\nabla \phi}{|\nabla \phi|})|\nabla \phi| + \beta |\nabla \phi|. \tag{2}$$

The first term in the right side is a smoothing term that represents curvature flow, where α determines the level of smoothness in the results. In the second term, β is a speed function that attracts the evolving level set to target regions as a driving force. Various image features can be integrated into the parameters α and β for different image segmentation tasks [18].

2.2. Distance Field Regularization

The level set function ϕ is initialized as a distance field (Equation 1). It satisfies $|\nabla \phi| = 1$ [19, 20]. Therefore, Equation 2 can be further simplified as:

$$\frac{\partial \phi}{\partial t} = div(\alpha \nabla \phi) + \beta. \tag{3}$$

This LSE describes the evolution of geometric active contour. However, solving this partial differential equation (PDE) on a discrete grid often introduces numeric errors and distorts the distance function around C. Therefore, ϕ needs to be updated (i.e., regularized) in order to keep $|\nabla \phi| = 1$, usually after a small number of time steps. Distance filed regularization usually needs to locate the zero level set explicitly and then recompute the distance field to it [21, 22, 23]. This process cannot be easily implemented on parallel platforms such as GPUs, since it often involves global data access in the whole domain. The reason is that these methods need to relocate zero level set when the distance regulation is applied. This relocation process involves an additional step and data structure to acquire global contour, which is not easy to parallelize. In

contrast, our approach avoids the relocation of zero level set for distance regulation, so that all the computation is local and becomes very suitable for GPU acceleration. Therefore, it can seamlessly intergrated with the LBM parallel computation framework for level set based image segmentation.

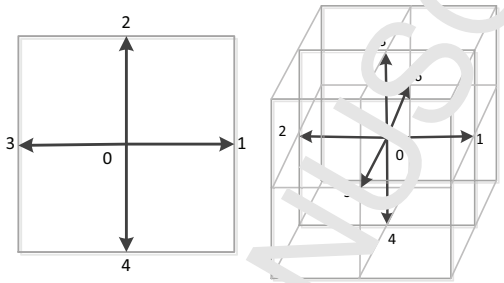


Figure 1: D2Q5 (left) and D3Q7 (right) lattice models.

3. Fully Parallelized LBM for Segmentation

3.1. LBM Solution to LSE

With its programming simplicity and embarrassingly parallel computation, LBM has been used to solve the LSE Equation 3 which is a nonlinear diffusion equation. Given a discrete computation grid over 2D/3D images, each grid cell located at $\vec{x}$ has a set of associated variables $f_i, i = 0 \ldots N$. N is the number of links starting from the cell to its immediate neighbors and itself, which is determined by different LBM lattice models. These variables f_i are summed up to define the distance function:

$$\phi(\vec{x},t) = \sum_i f_i(\vec{x},t), i = 0 \ldots N. \tag{4}$$

Each f_i interacts with one of its neighbours following a corresponding direction vector $\vec{e}_i$. Figure 2 shows a D2Q5 ($N = 5$) and a D3Q7 ($N = 7$) lattice model of a grid cell. D3Q7 model refers to 3D computation using seven $f_i, i = 0 \ldots 6$ (six to its axial neighbors and one to itself).

Moreover, a set of equilibrium variables corresponding to f_i are defined as

$$f_i^{eq} = A_i \phi, i = 0 \ldots N, \tag{5}$$

where A_i is a scalar coefficient determined by the lattice model. For D3Q7, $A_i = 1/7, i = 0 \ldots 6$ and for D2Q5, $A_i = 1/5, i = 0 \ldots 4$.

When $t = 0$, f_i is initialized as $f_i = f_i^{eq}$. At a simulation time step $t + \triangle t$, the variables of f_i are updated from the variables in the previous step t as:

$$f_i(\vec{x} + \vec{e}_i, t + \triangle t) - f_i(\vec{x}, t) = \frac{1}{\tau}(f_i(\vec{x}, t) - f_i^{eq}(\vec{x}, t)) + \triangle t \vec{F}_i, \tag{6}$$

where τ is a constant relaxation parameter and $\vec{F}_i$ is the external force driving the evolving level set. Once f_is are updated, the distance ϕ at $t + \triangle t$ is calculated by Equation 4, and f_i^{eq}s are updated by Equation 5 for next iteration. This iterative computation repeats in multiple simulation steps with the given time step $\triangle t$. It stops when the zero level set $\phi = 0$ converges to the aimed boundary C with respect to a giving stopping condition.

It can be proved (see details in [24]) that the LBM scheme (Equation 6) recovers the nonlinear diffusion equation through Chapman-Enskog expansion:

$$\frac{\partial \phi}{\partial t} = div(\frac{1}{3}(\tau - \frac{1}{2})\nabla \phi) + \frac{\vec{F}_i}{A_i}. \tag{7}$$

Comparing Equation 3 with Equation 7, LBM parameters τ and $\vec{F}_i$ are defined by the LSE-based image segmentation parameters as:

$$\tau = 3\alpha + \frac{1}{2}. \tag{8}$$

$$\vec{F}_i = A_i \beta. \tag{9}$$

From Equation 8, it can be seen that τ is larger than 0.5, whenever α is positive. Since α is the diffusion coefficient, it is always larger than zero. Therefore τ is always larger than 0.5 which has been known leading to stable LBM computation for any time step size $\triangle t$ [25].

In image segmentation, D2Q5 and D3Q7 lattice models provide minimal computing time and memory use while still maintain good segmentation results. There exists other lattice models, such as D3Q15, D3Q19 or D2Q9 [11, 12, 13], which will increase the computational load. Though these models are more popular in flow simulation, D2Q5 and D3Q7 are good enough for 2D and 3D segmentation tasks. Next, we show how to implement the distance field regularization inside the LBM scheme.

3.2. Distance Field Regularization by LBM

When the distance field ϕ is numerically computed, it needs to be regularized as ϕ^R to satisfy $|\nabla\phi^R| = 1$. However, some of the existing LBM-based segmentation methods do not perform regularization [9, 10, 12] and therefore cannot effectively segment noised images. A few methods apply external data structure and computation (e.g., fast marching [13], distance re-computation [11], and using an extra forcing term [14]) to regulate distance field, but they cannot be easily parallelized to get good computing performance.

The regularization can be achieved by solving a time dependent PDE which is introduced in multi-phase flow problem [19]

$$\frac{\partial \phi^R}{\partial t} + sign(\phi^R)(|\nabla \phi^R| - 1) = 0, \tag{10}$$
$$\phi^R(x,0) = \phi(\vec{x}).$$

The signum term in Equation 10 is of great importance to maintain the zero level set as a satisfied distance field. Meanwhile, the distance field is kept smooth during iterations, so that eventually the extracted contour or surface will have sub-voxel accuracy. Next, we design a new scheme to solve this equation in the LBM scheme so that the whole algorithm is very suitable for parallel computing.

3.2.1. Efficient Implementation in LBM

From Equation 5, the distance function can be represented by the equilibrium variables as $\phi = f^{eq}/A_i$. We realized that specifically in D2Q5 and D3Q7 models, A_i has the same value (i.e., 1/5 for D2Q5 and 1/7 for D3Q7, respectively) for all directions i. Therefore, only one variable f^{eq} is needed at each grid cell. Meanwhile, the distance function ϕ can be achieved by f^{eq}/A_i. Therefore, Equation 10 which solves the regulated distance function ϕ^R can be rewritten as

$$\frac{\partial f^{eq}}{\partial t} + sign(f^{eq})(|\nabla f^{eq}| - A_i) = 0. \tag{11}$$

In this way, the regularization process becomes part of the LBM computation with f^{eq}. Solving this equation (see below) can be triggered after a few normal LBM steps, after which we set $f_i = f^{eq}$ to continue the LBM iterations.

3.2.2. Parallel Regularization Solver

The solution of Equation 11 has been studied by using different macroscopic spatial and temporal discretizations [26]. To solve it in a parallel program, we use an explicit scheme with a first order ENO (Essentially Non-Oscillatory) finite difference in spatial discretization [19]. Please see the details in Section 4 for GPU implementation. This approach is very fast regarding convergence which can be achieved in only a few iterations. More importantly, the computation is performed in parallel for each grid cell, fitting seamlessly within the LBM parallel scheme.

3.3. Handling Edge Stopping Models

Our LBM-based level set solver can be utilized for active contour models using different image features to define the LSE parameters, α and β. We show two models below.

3.3.1. Edge Stopping Function with Gradient

Let I_0 be an image to be segmented, an edge stopping function g can be used to control the evolution of the contour in LSE [27]. A robust form of g is defined as [28]:

$$g = e^{-((|\nabla G_\sigma * I_0|)^2/k_1^2)}, \tag{12}$$

where $|\nabla G_\sigma * I_0|$ denotes the gradient of the Gaussian smoothed image. σ is a smoothing parameter and k_1 is an estimated threshold of the edge gradient of I_0. In LBM implementation, we simply set $\alpha = g$ and $\beta = \lambda_1 g$ in Equation 8 and Equation 9. Then the LBM Equation 6 solves the GAC evolution of

$$\frac{\partial \phi}{\partial t} = div(g\nabla\phi) + \lambda_1 g. \tag{13}$$

Here $\lambda_1 > 0$ is a constant to control the moving speed of the evolving contours. Using this approach, the LBM segmentation will stop when the zero level sets reach the large gradient edges defined by k_1 where $g \mapsto 0$.

3.3.2. Edge Stopping Function with Local Average

Using the gradient-based stopping function in LSE may not work well when the edges are not easily discovered by image gradients such as in a noisy image. In such

cases, regional information of I_0 can be applied to overcome the noise. In our implementation, we combine g with an estimated local average k_2. Then, we set $\alpha = g$ and $\beta = \lambda_2((G_\sigma * I_0) - k_2)$ in Equation 8 and Equation 9. Consequently, the LBM Equation 6 solves the evolution of GAC as

$$\frac{\partial \phi}{\partial t} = div(g\nabla\phi) + \lambda_2((G_\sigma * I_0) - k_2). \tag{14}$$

Similarly, $\lambda_2 > 0$ is a constant to control the contour surface moving speed. G_σ is the Gaussian convolution kernel with the variance σ. Then, the LBM segmentation will stop when the zero level sets reach the image edges whose local average is close to k_2.

The two models are well suited to parallel computation because the calculation of g and other values at each grid cell only involves local data access in its neighborhood. Therefore the LBM segmentation can be fully parallelized for fast performance. In level set image segmentation tools, the constants like λ and k are usually defined by users empirically.

3.4. Computational Procedure

In recapitulation, the LBM based image segmentation (using D2Q5 or D3Q7) is implemented in the following steps:

1. Define and compute the level set edge stopping variables (e.g., g) from an input image;
2. Generate an initial distance field ϕ from a starting shape (zero level set), such as a 2D circle/rectangle or a 3D sphere/cubic;
3. Initialize LBM: f_i, f_{eq} from ϕ by Equation 5;
4. At each grid cell, compute α, β for the corresponding GAC models, and then use them to define LBM computing parameters τ and F_i by Equations 8-9;
5. Perform LBM evolution following Equation 6;
6. Accumulate the f_i values at each grid cell by Equation 4, which generates an updated ϕ that is directly used to update f_{eq};
7. If the number of iterations is bigger than M, perform distance field regularization by solving Equation 11 and then reset $f_i = f_{eq}$.

8. If the zero level set converges with no significant changes, stop the LBM iterations and output the segmentation results.
9. Otherwise, go back to Step 4.

To fully leverage the parallel nature of the algorithm, the iterative computation steps (Step 3-9) are put into the GPU computation pipeline. We first implement the traditional LBM numerical iterator on GPU for Step 5. Then we focus on migrating the new regularization computation (Step 6-7) to GPU. In the next section, we discuss the GPU implementation in details.

4. GPU Acceleration

In this section, we describe the GPU implementation and optimization for the proposed LBM image segmentation. The whole algorithm is implemented using the CUDA toolkit v5.5 created by nVidia. CUDA provides developers the CUDA-accelerated libraries to access its runtime API on CUDA-enabled GPUs. In imple-menting our LBM algorithm, we divide the whole computation procedure into two CUDA kernel functions. Each kernel is executed in parallel by a given number of threads on GPU. The first one implements the traditional LBM iterations (Equation 6). The second one is newly developed for the distance field regularization (Equation 11). For each kernel, one computing thread is responsible for the operation of one LBM grid cell, which refers to one 2D pixel or one 3D voxel. Thus, multiple threads corresponding to all the pixels or voxels facilitate parallel execution of the local neigh-borhood operations. Inside each kernel function, the read/write conflicts are avoided as the source and destination memories are separated.

The thread synchronization inside each kernel is implicitly implemented by CUDA. In addition, the second kernel starts when all the threads of the first kernel are completed, so that the computations of the LBM evolution and distance field regulation are not overlapped. Therefore, explicit synchronization points are not needed inside one kernel. Such synchronization scheme leads to correct and fast iterative computation. Next we show the details of GPU implementation.

In the LBM iteration kernel of solving Equation 6, a temporary array f_i^{temp} is used

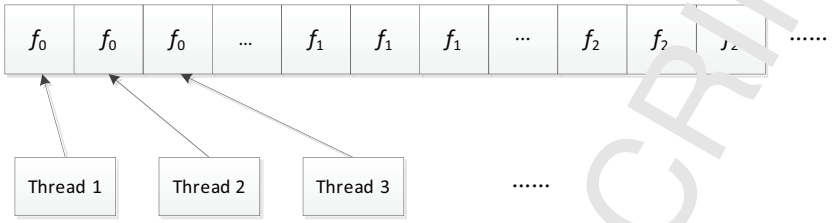


Figure 2: Memory access pattern. (Top) Array of Structures (Bottom) Structure of Arrays.

to swap the streaming results with f_i to avoid the memory race problem [8, 29]:

$$f_i^{temp}(\vec{x}+\vec{e}_i,t+\triangle t) = (1-\frac{1}{\tau})f_i(\vec{x},t) + \frac{1}{\tau}f_i^{eq}(\vec{x},t) + \triangle t\vec{F}_i(\vec{x},t) \tag{15}$$

Moreover, the storage of arrays f_i^{temp} and f_i is arranged to employ a Structures of Array (SoA) format as shown in Fig. 2. Compared to the classic format of Array of Structures (AoS), $f[(z*Ny*Nx+y*Nx+x)*7+i]$, the SoA arrangement stores f_i in the order of i and then by spatial coordinates. For example, assuming the computation domain of a D3Q7 is $Nx \times Ny \times Nz$, f_i in SoA is addressed as $f[i*Nx*Ny*Nz+z*Ny*Nx+y*Nx+x], i = 1 \cdots 7$. SoA makes the threads within one CUDA warp (e.g., 32 threads) to read consecutive memory. The coalesced memory access can largely improve the throughput of globe memory access on GPU [30].

There are some existing work using shared memory and intra-warp shuffle operation to improve the throughput of memory [31, 32]. However, it has been shown that this approach is ineffective in improving the performance on modern GPU architecture [30]. Therefore, our implementation utilizes direct access to globe memory, which leads to lower register usage and does not need any additional control flow.

The regularization kernel is implemented by solving Equation 11 through the first order ENO scheme. The 3D version of the GPU implementation is shown as follows (2D version has the similar implementation):

First, six monotone spatial differences are calculated for each grid cell (i,j,k) as

$$\begin{aligned} a &= D^-{}_x f^{eq} = f^{eq}_{i,j,k} - f^{eq}_{i-1,j,k}, \\ b &= D^+{}_x f^{eq} = f^{eq}_{i+1,j,k} - f^{eq}_{i,j,k}, \\ c &= D^-{}_y f^{eq}, d = D^+{}_y f^{eq}, \\ e &= D^-{}_z f^{eq}, f = D^+{}_z f^{eq} \end{aligned} \tag{16}$$

Second, we extract

$$\begin{aligned} a_+ = max(a,0), a_- = min(a,0), \\ \cdots, \\ f_+ = max(f,0), f_- = min(f,0). \end{aligned} \tag{17}$$

Third, the steeper gradient in each direction is computed as

$$\begin{aligned} a = max({a_+}^2, {b_-}^2), b = max({a_-}^2, {b_+}^2), \\ \cdots, \\ e = max({e_+}^2, {f_-}^2), f = max({e_-}^2, {f_+}^2). \end{aligned} \tag{18}$$

Fourth, we compute the gradient $|\nabla f^{eq}|$ based on the sign of f^{eq}

$$\begin{aligned} |\nabla f^{eq}| &= \sqrt{a+c+e}, (f^{eq} > 0), \\ |\nabla f^{eq}| &= \sqrt{b+d+f}, (f^{eq} < 0). \end{aligned} \tag{19}$$

Finally, with a smoothed sign function $sign(f^{eq}) = \frac{f^{eq}_t}{\sqrt{(f^{eq}_t)^2+1}}$, f^{eq} is updated as

$$f^{eq}_{t+1} = f^{eq}_t - \triangle t \frac{f^{eq}_t}{\sqrt{(f^{eq}_t)^2+1}} (|\nabla f^{eq}_t| - A_i). \tag{20}$$

These steps are executed in the distance regularization kernel repeatedly until convergence. In practice, this is usually achieved in a few iterations. In GPU implementation, the regularization step is also optimized using the similar strategies as in the first kernal to efficiently update f^{eq}.

In summary, due to the inherent parallel nature, the computing procedure of both kernels is explicit and only involves the nearest neighbor grid cells, which lead to high GPU computation performance reported in the next section.

5. Case Studies

In this section, we present several case studies of geometry extraction from 2D and 3D medical images.

5.1. Experiment Evaluation and Parameter Setting

Our method provides a fast and parallel numerical method for LSE-based segmentation. To evaluate its segmentation quality, we use the segmentation results of a standard level set solver, the upwind difference method [17] as the ground truth. The upwind difference method is the reliable and accurate numerical solver of LSE equations [17], which takes into account the gradient direction of the evolving interfaces. This method is widely used in solving level set segmentation problems [33, 18]. For quantitative measurement, we consider the segmentation result of the upwind difference method as the ground truth. The parameters of the level set stopping criteria play an important role in both segmentation accuracy and efficiency. The parameter setting rules have been discussed by many researchers [27, 28, 18]. While our approach focuses on improving the computational efficiency, different segmentation parameters can be used per the rules. In implementation, we choose the segmentation parameters including k_1, k_2, λ_1, λ_2, which can yield good segmentation results in the standard upwind difference method.

The upwind difference method uses small time steps and thus leads to long computing time. In contrast, LBM method can keep stable numerical iterations using larger time steps. It accelerates the computational speed and reduces the number of iterations to converge. In our experiments, we found that setting the time step size of LBM iterations between 1 and 2 is a good compromise between the efficiency and the accuracy.

We further compute the mismatched pixels/voxels in the segmented results between the ground truth and our LBM method. Then an error rate is computed by dividing the number of these mismatched ones by the total number of the correctly segmented pixels/voxels in the ground truth. For evaluating the computing performance, we show the computational time of our method in both CPU and GPU versions. We then compare them with the upwind difference method and several related existing methods. In our implementation, the CPU based serial algorithms are executed on a PC with an Intel

i7-3770 CPU at 3.4GHZ and 8G RAM. The GPU based parallel algorithms are run on a consumer GPU, NVIDIA Geforce GTX 780 at 900MHZ and 3GB memory.

5.2. 2D Medical Image Segmentation

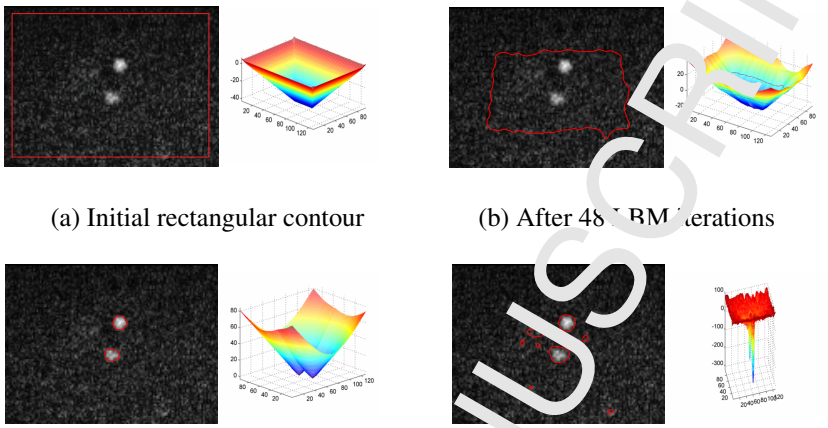
We first investigate 2D applications of our method to show its benefits. In the first case, we evaluate the robustness of our approach to noisy images and with different initial conditions. In the second and third cases, we compare our approach with other LBM based segmentation methods.

5.2.1. 2D Segmentation of Carotid Artery from MRI Images

The LBM scheme extracts carotid arteries from phase contrast MRI images. Figure 3 shows an image with the size of 128×196 including both left internal and external carotid arteries. The image is very noisy with blurred edges of the arteries. We apply the edge stopping function with local average as described in Section 3.3.2, where $k_1 = 2$, $k_2 = 75$, $\lambda_2 = 0.02$, and $\sigma = 1$.

In the first experiment, we set a large rectangle as the initial contour as Figure 3(a), and compute the distance field ϕ based on this initial contour. In LBM iterations, the time step is set to 1 and $M = 2$. That is, we apply regularization of the distance field every 2 LBM steps. Figure 3(b) shows the evolving contour after 48 iterations. Figure 3(c) shows the converged contour after 152 iterations, i.e., when the movement of the level set is less than one pixel in all positions. The result successfully finds the boundaries of the left and right carotid arteries. The distance field is very smooth during the evolution as visualized in the figures. In comparison, we implement the LBM algorithm of [2] to segment this image with the same parameters. Since there is no distance field regularization, the achieved contours fail to converge at the boundary of the carotid arteries (Figure 3(d)).

In the second experiment, we set the initial contour as a set of circles in the domain (Figure 4(a)). The other parameters are not changed. Moreover, instead of computing the distance at each grid cell to these circles to define initial contours, we simply apply a binary function to define if a grid cell is inside/outside a circle. This approach largely reduces the computing burden of the regularization. Even with this simplified initial condition, our LBM algorithm still keeps stable during the iterations as shown in



(a) Initial rectangular contour (b) After 48 LBM iterations

(c) Converged after 152 LBM iterations (d) LBM method of [9]

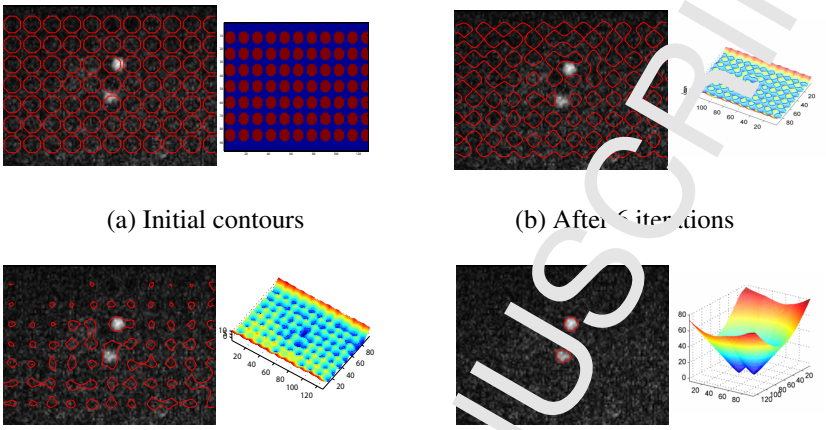
Figure 3: 2D carotid artery segmentation from a phase contrast MRI image.

Figures 4(b)(c). Furthermore, since the distance between the initial circles and the arteries is much closer than the first experiment, the active contours can quickly converge and stop at the boundaries of the carotid arteries with fewer iterations (48 iterations). Figure 4(d) shows the same segmentation result as Figure 3(c).

5.2.2. 2D Segmentation of Aortic Artery from CT Images

We extract a 2D aorta artery from a CT image with a size of 338×196. The edge of the aorta artery in the CT image is clearer than in the previous MRI image. We apply the edge stopping function with gradient (Section 3.3.1). The parameters are set as: $\sigma = 1$, $k_1 = 2$ to catch artery edges.

Figure 5 shows the ground truth result computed from the classic upwind difference method [17], compared with the results of our method and two existing LBM algorithms from [11] and [14], respectively. In order to keep the algorithm stable, we set the time step to 0.1 for the upwind differential method. For all LBM algorithms, the time step is set to 1. Here, we allow the distance field regularization to be executed every $\mu = 5$ step in the proposed algorithm. Our result achieves smoother contour than Sun's result. On the other hand, Chen's model fails to find the correct boundary. Since there is a very narrow gap between the aorta and heart, the algorithm moves the evolv-



(a) Initial contours (b) After 6 iterations

(c) After 12 iterations (d) Converged after 48 iterations

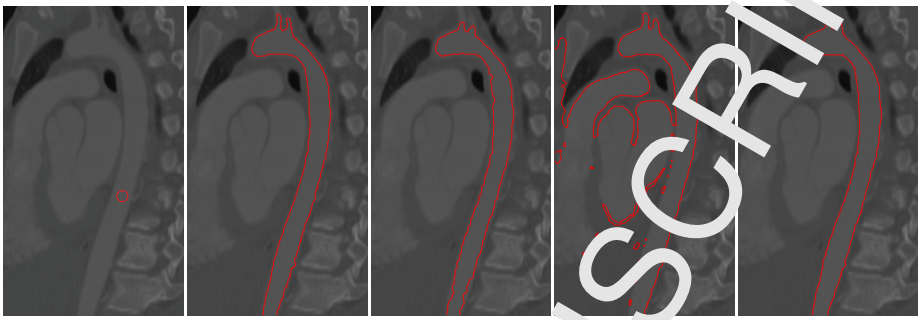
Figure 4: 2D carotid artery segmentation from a phase contrast MRI image with binary initial condition.

Table 1: Computing performance for 2D segmentation of an aorta image (338 × 196).

Methods	step size	number of iterations	error rate	time (sec)	speed up
Upwind (ground truth) [17]	0.1	8852	0	121.1	1
[11]	1	1255	3.7%	19.8	6.1
[14]	1	1200	fail	16.5	7.3
Our method (CPU)	1	1176	2.1%	12.5	9.7
Our method (GPU)	1	1176	2.1%	0.13	930

ing level set out of the aorta since it adds a penalty term to LSE to force the distance field smooth.

The computing performance and quality for all methods is demonstrated in Table 1. Our method has 2.1% error rate, while Sun's error rate is 3.7%. For computing efficiency, our CPU implementation uses 12.5 seconds which is about 40% faster than Sun's method at 19.8 seconds. Our method is almost ten times faster than the upwind scheme at 121.1 seconds. By GPU acceleration, our algorithm achieves near 100 times speedup at 0.13 seconds compared to its CPU version and Sun's method, which is more than 900 times faster than the classic upwind differential scheme.



(a) (b) (c) (d) (e)

Figure 5: 2D aorta segmentation result comparison. (a) Initial contour; (b) Ground truth; (c) Result of [11]; (d) Result of [14]; (e) Result of our algorithm.

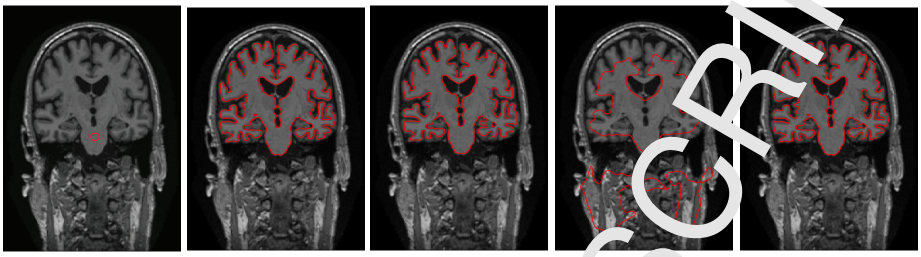
Table 2: Computing performance for 2D segmentation of a brain image (388×251).

Methods	step size	number of iterations	error rate	time (sec)	speed up
Upwind (ground truth) [17]	1	4235	0	88.7	1
[9]	1	500	fail	10.2	8.69
[11]	1	438	4.2%	14.5	6.1
Our method (CPU)	1	470	1.8%	9.6	9.2
Our method (GPU)	1	470	1.8%	0.09	940

5.2.3. 2D Segmentation of Brain from MRI images

We further perform a 2D segmentation of complex brain structure from an MRI image whose size is 388×251. The edge stopping function with local average is used, where $k_1 = 2$, $k_2 = 70$, $\lambda_2 = 0.02$, and $\sigma = 1$. In this case, we compare our segmentation result and efficiency with the ground truth and two other LBM methods of [11] and [9]. The time step is 0.1 for the upwind method and 1 for all LBM methods, and $M = 4$ for distance field regularization.

As shown in Figure 6, our result is smooth and close to the ground truth. Since there is no distance field regularization, the LBM approach of [9] fails to find the structure (Figure 6(d)). The performance and quality is reported in Table 2. In particular, our GPU implementation can achieve the result in 0.09 second which is around 940 speed-up to the upwind difference method. It also achieves 1.8% error rate compared to 4.2% of Sun's method.



(a) (b) (c) (d) (e)

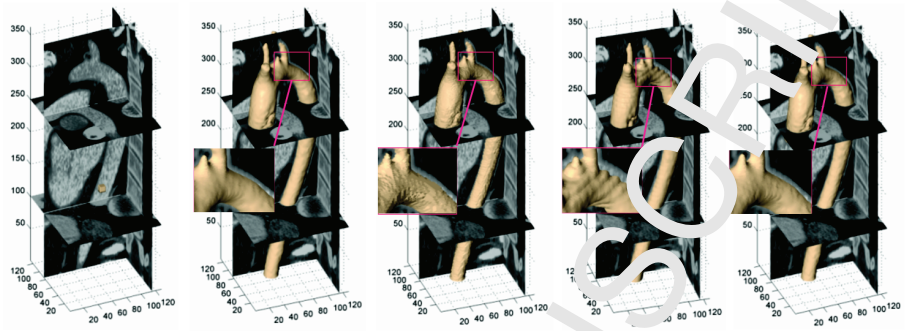
Figure 6: 2D segmentation result of brain from an MRI image. (a) Initial contour; (b) Ground truth; (c) Result of [11]; (d) Result of [9]; (e) Result of our algorithm.

5.3. 3D Geometry Extraction of Medical Images

In this section, we investigate 3D geometry extraction from CT and MRI medical images. For 3D cases, the computational load increases greatly by adding an extra dimension. Thus, our LBM based parallel approach of 3D extraction is time-efficient so it can be very helpful for real applications. There exist very little work using the parallel LBM scheme in 3D cases. [9] presented a 3D algorithm, which however, usually fails in our experiments of 3D segmentation, because it does not apply distance field re-initialization and only uses a simple pixel comparison method in stopping function. So we do not compare our method with it. In particular, we mainly compare our approach with three 3D geometric extraction methods of solving LSE functions including: (**Ground truth**) Upwind difference method [17]; (**M2**) The approach of [14] who used a simple method which connects segmented 2D slices together to form a 3D shape; (**M3**) An alternative approach which replaces our unified re-initialization approach in the LBM scheme with the distance field regulation method proposed by [11]. It should be noted that [11] did not implement 3D geometric extraction in their work. We extend their approach to 3D cases in order to compare its distance regulation method with ours. These methods are not suited to GPU acceleration, since they are not fully integrated into the LBM's parallel scheme, which is the unique feature of our approach.

5.3.1. 3D Segmentation of Aorta Artery from CT Images

First, we investigate 3D geometry extraction of aorta artery from CT images. The volume size of images is $128 \times 128 \times 372$. The edge stopping function with gradient



(a) (b) (c) (d) (e)

Figure 7: 3D aorta artery segmentation from CT images. (a) Initial contour; (b) Ground truth; (c) Result of M3; (d) Result of M2; (e) Result of our method.

information is used (Section 3.3.1), where $\kappa_2 = 2$, $\kappa_1 = 1$, and $\sigma = 1$.

Figure 7 shows the results of our algorithm and other approaches. We also show a zoomed-in region for details. It can be seen that our approach yields the results fairly similar to the ground truth from the upwind difference method. M2 method forms a 3D shape from 2D segmented slices leading to obvious artifacts. In practice, this method needs to perform initialization for every slice and cannot handle topological change automatically in the vertical direction. Compared with M3, our re-initialization method has smoother results than using Sun's method. Moreover, Sun's distance regulation cannot be easily parallelized as it uses globe information of the zero level set. This also makes it hard to implement M3 on GPUs.

Table 3 shows the report for this case study. The time step of the upwind difference algorithm is set to 0.1, while the time step of other methods are set to 1 for fast run. Our method has about 2.7% error rate while M3 has 4.8% error rate. With GPU acceleration, our fully parallel approach can complete in 22.2 seconds which is around 100 times faster than the CPU version, and much faster than the upwind approach and M3.

5.3.2. 3D Segmentation of Carotid Artery from CT Images

Clinical assessment of stroke risk has been heavily reliant on the degree of luminal stenosis of carotid artery. When a carotid stenosis narrows the artery by a diameter of more than 60%, a carotid endarterectomy or carotid artery stent is performed to decrease the risk of patients having a future stroke. The diameter of carotid artery is

Table 3: Computing performance for 3D segmentation of aorta artery from CT images ($128 \times 128 \times 372$).

Methods	step size	number of iterations	error rate	time (sec)	speed up
Upwind (ground truth)	0.1	7034	0	23058	1
M3	1	1955	4.8%	3568	6.4
Our method (CPU)	1	947	2.7%	2258	10.2
Our method (GPU)	1	947	2.7%	22.2	1038

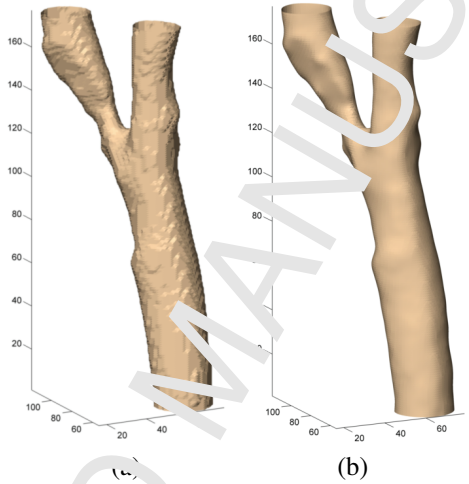


Figure 8: 3D segmentation of carotid artery from CT images. (a) Result of M3; (b) Result of our method.

only about 4-6 millimeters (mm) [34], and the common resolution of a CT image is 0.4 mm per pixel which stands for nearly 10% of the artery diameter. In such cases, getting accurate segmentation result is critical for a clinic application.

Figure 8 shows the geometry extraction results of a stenosed carotid artery from CT images. The volume size is $128 \times 128 \times 177$. Our method gets accurate and smooth geometry of the carotid artery compared with M3. Our regularization method is included in the LBM scheme which can reach sub-grid accuracy. In comparison, M3 performs distance field regularization at the accuracy of grid cells. Table 4 shows that our method achieves better quality with a smaller error rate than M3. Moreover, our fully parallel method reconstructs the complete 3D geometry totally on GPU in 4.6 seconds, faster than other approaches.

Table 4: Computing performance for 3D segmentation of carotid artery from CT images ($\cdot \times 128 \times 177$).

Methods	step size	number of iterations	error rate	time (sec)	speed up
Upwind (ground truth)	0.1	3135	0	4640	1
M3	1	332	4.9%	785	6.7
Our method (CPU)	1	448	2.8%	435	10.9
Our method (GPU)	1	448	2.8%	4.6	1010

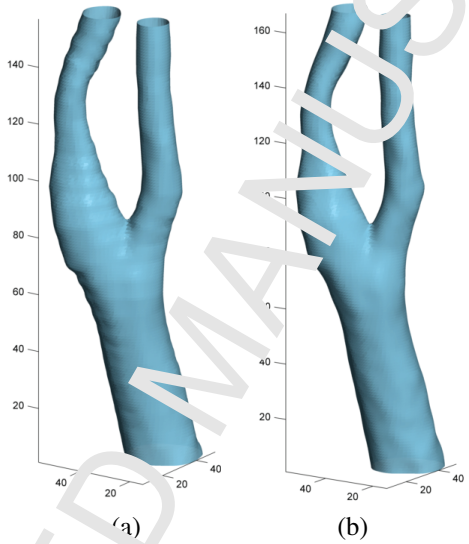


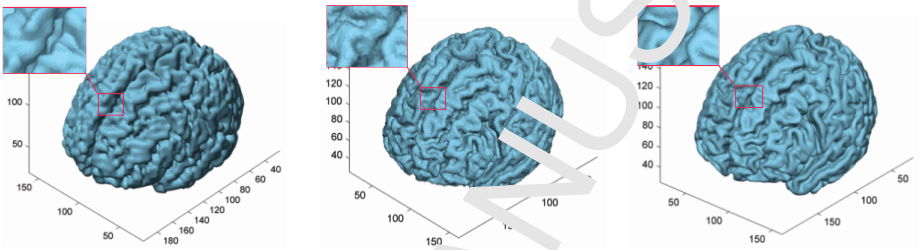
Figure 9: 3D segmentation of carotid artery from MRI images. (a) Result of M2; (b) Result of our method.

5.3.3. 3D Segmentation of Carotid Artery from MRI Images

We also work on MRI images for the carotid artery extraction. The image size is $64 \times 64 \times 57$. Since the MRI images we achieved from clinical practice are of relatively low-quality, we apply the edge stopping function with local average with $\sigma = 1$, $\lambda_2 = 0.07$, $k_2 = 85$ and $M = 2$. Figure 9(a) shows the result of M2 which creates 3D structure from 2D segmentations. It shows some salient horizontal ring effects. In contrast, Figure 9(b) shows that our method achieves smoother result with 2.4% error rate from the ground truth (the result image is omitted here). Table 5 also shows that our GPU approach completes the task in less than 0.5 seconds which can contribute to time critical applications such as clinical bio-flow simulation.

Table 5: Computing performance for 3D segmentation of carotid artery from MRI images ($64 \times 64 \times 167$).

Methods	step size	number of iterations	error rate	time (sec)	speed up
Upwind (ground truth)	0.1	626	0	455	1
M2	1	245	16%	151	3.1
Our method (CPU)	1	152	2.4%	45.7	9.9
Our method (GPU)	1	152	2.4%	0.48	947



(a) (b) (c)

Figure 10: 3D brain segmentation result. (a) Result of M2; (b) Result of M3; (c) Result of our method.

5.3.4. 3D Segmentation of Brain Structure from MRI Images

Finally, Figure 10 illustrates a 3D brain segmentation from MRI images. We apply the edge stopping function with local average with $\sigma = 1$, $\lambda_2 = 0.02$, $k_2 = 98$ and $M = 2$. The image size is $181 \times 217 \times 181$.

Figure 10(a) shows the result of M2 approach, which cannot construct the brain's shape correctly. Figure 10(b) is the result of M3, its surface is rather rough. In comparison, our method (Figure 10(c)) creates the smooth and accurate result with a 1.9% error rate. Furthermore, Table 6 shows the fast computing speed of our method which can finish the segmentation in 5 seconds on GPU.

5.3.5. Experiments on More Datasets

To further evaluate the proposed algorithm, we test our method on more 3D biomedical datasets. First, we extract the carotid artery of 20 different patients from their 3D MRI images. The image size is $64 \times 64 \times 196$ for each 3D data set. Comparing to the upwind difference method, our method generally achieves good performance. Table 7 reports the average values among these twenty datasets of the number of iterations, error rates, computing times, and speedups compared to the upwind method. In particular, the GPU acceleration has an average speedup of 947 with an average 2.5%

Table 6: Computing performance for 3D segmentation of brain structure from MRI image ($181 \times 217 \times 181$).

Methods	step size	number of iterations	error rate	time (sec)	speed up
Upwind (ground truth)	0.1	1026	0	5124	1
M3	1	143	4.1%	824	6.2
Our method (CPU)	1	152	1.9%	511	10.0
Our method (GPU)	1	152	1.9%	5.0	1024

Table 7: Average computing performance for the segmentation of carotid arteries of 20 patients from their 3D MRI images.

Methods	Step size	Number of iterations	Error rate	Total time (s)	Speed up
Upwind (ground truth)	0.1	732	0	644	1
Our method (CPU)	1	184	2.5%	68	9.5
Our method (GPU)	1	184	2.5%	0.7	947

error from the ground truth in the extracted results.

Moreover, we further test our method on several 3D image datasets from a public volume data library [1]. These datasets include different biomedical geometries with different volume sizes, including human head, foot, tooth, knee, and a frog. Table 8 is the experiment results of segmenting geometrical structures from these datasets. The table shows that the parallel implementation on GPU can achieve about 100 speedups comparing to its serial version on CPU, which is about 900 time faster than the upwind difference method with less than 3% difference in the segmentation results.

6. Conclusion and Future Work

In this paper, we propose a new parallel geometric extraction method from medical images. This model is fully parallel by incorporating the necessary distance field regularization of LSE into the LBM-based level set solver. Our method can completely run on GPUs which achieves great performance for biomedical geometry extraction from CT and MRI images. Recently, LBM computations have been implemented on multi-core CPU platforms [35], GPU clusters [36] and heterogeneous CPU/GPU clusters [37]. Our LBM algorithm has similar computational structure and procedure to

[1] http://lgdv.cs.fau.de/External/vollib/

Table 8: Computing performance for 5 biomedical datasets from a public library [1].

Datasets	Methods	Step size	Number of iterations	Error rate	Total time (s)	Speed up
Head $256 \times 256 \times 53$	Upwind (ground truth)	0.1	525	0	995	1
	Our method (CPU)	1	142	1.8%	21	9.2
	Our method (GPU)	1	142	1.8%	1.2	975
Foot $256 \times 256 \times 128$	Upwind (ground truth)	0.1	765	0	1448	1
	Our method (CPU)	1	201	1.9%	67	9.3
	Our method (GPU)	1	201	1.9%	0.7	925
Tooth $256 \times 256 \times 161$	Upwind (ground truth)	0.1	923	0	995	1
	Our method (CPU)	1	272	1.6%	106	9.4
	Our method (GPU)	1	272	1.6%	1.1	905
Frog $256 \times 256 \times 44$	Upwind (ground truth)	0.1	472	0	138	1
	Our method (CPU)	1	122	2.2%	14	9.9
	Our method (GPU)	1	122	2.2%	0.16	862
Knee $256 \times 256 \times 44$	Upwind (ground truth)	0.1	532	0	152	1
	Our method (CPU)	1	162	2.3%	16	9.5
	Our method (GPU)	1	162	2.3%	0.18	894

these methods, while the extra regularization step keeps the parallelism and locality. Therefore, we expect the proposed method to be applied in these platforms in addition to GPU, which will be our immediate future work.

In geometric active contour models, the stopping functions largely affect the segmentation results. We use image gradient and local average to define edge stopping functions, while other image features can be applied in a similar manner. If these features can be computed from locally neighboring cells (pixels/voxels), the LBM scheme can be directly used and easily parallelized. However, some models use global image attributes, such as the high order statistical descriptors [18] or clustering [12], where the computation needs special parallel algorithms, which can be combined with our parallel solution for fast performance.

In general, our method can quickly extract 3D geometry with accurate and smooth implicit representation from medical images. One direction of the future work is to combine this solver and the computation domain reduction methods (e.g., multi-grid or narrow band) together to further enhance the computational efficiency. Another direction is in bio-flow modeling as LBM is also a good parallel flow solver. The segmentation results can seamlessly feed to LBM based bio-flow simulation (e.g., [38, 3]) without explicitly generating the meshes. We will further combine this approach

with the LBM flow simulation towards a unified hemodynamics simulation system.

References

[1] J. Dong, K. Inthavong, J. Tu, Image-based computational hemodynamics evaluation of atherosclerotic carotid bifurcation models, Comput. Biol. Med. 43 (10) (2013) 1353–1362.

[2] J. Weickert, B. Romeny, M. Viergever, Efficient and reliable schemes for nonlinear diffusion filtering, IEEE Transactions on Image Processing 3 (1997) 398–410.

[3] D. Peng, B. Merrimann, A PDE-based fast local level set method, Journel of Computing Physics 155 (1999) 410–438.

[4] G. Papandreou, P. Maragos, Multigrid geometric active contour models, IEEE Transactions on Image Processing 16 (2007) 229–240.

[5] A. Eklund, P. Dufort, D. Forsberg, S. LaConte, Medical image processing on the GPU - past, present and future, Medical Image Analysis 17 (2013) 1073–1094.

[6] A. E. Lefohn, J. M. Kniss, C. D. Hansen, R. T. Whitaker, A streaming narrow-band algorithm: Interactive computation and visualization of level sets, IEEE Transcation on Visualization and Computer Graphics 10 (4) (2004) 422–433.

[7] M. Roberts, J. Packer, M. C. Sousa, J. R. Mitchell, A work-efficient GPU algorithm for level set segmentation, Proceedings of the conference on High Performance Graphics (HPG '10) (2010) 123–132.

[8] Y. Zhao, Lattice Boltzmann based PDE solver on the GPU, The Visual Computer 24 (5) (2008) 323–333.

[9] A. Hagan, Y. Zhao, Parallel 3D image segmentation of large data sets on a GPU cluster, Proceedings of the 5th International Symposium on Visual Computing (2009) 960–969.

[10] Z. Wang, Z. Yan, G. Chen, Lattice Boltzmann method of active contour for image segmentation, IEEE International Conference of Image and Graphics (2011) 338–343.

[11] X. Sun, Z. Wang, G. Chen, Parallel active contour with lattice Boltzmann scheme on modern GPU, IEEE International Conference of Image Processing (2012) 1709–1722.

[12] B. A. Souleymane, X. Gao, B. Wang, A fast and robust level set method for image segmentation using fuzzy clustering and lattice Boltzmann method, IEEE Transactions on systems, man, and cybernetics 43 (2013) 910–920.

[13] X. Yang, X. Gao, D. Tao, Improving level set method for fast auroral oval segmentation, IEEE Transactions on Image Processing 23 (7) (2014) 2854–2865.

[14] Y. Chen, L. Navarro, Y. Wang, G. Courbebaisse, Segmentation of the thrombus of giant intracranial aneurysms from CT angiography scans with lattice Boltzmann method, Medical Image Analysis 18 (2014) 1–8.

[15] C. Li, C. Xu, C. Gui, M. D. Fox, Level set evolution without re-initialization: A new variational formulation, IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) (2005) 430–436.

[16] C. Li, C. Xu, C. Gui, M. D. Fox, Distance regularized level set evolution and its application to image segmentation, IEEE Trans. Image Process. 19 (12) (2010) 3243–3254.

[17] J. A. Sethian, Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, Cambridge University Press.

[18] A. Mitiche, I. A. Ismail, Variational and level set methods in image segmentation, Springer.

[19] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1994) 146–159.

[20] D. Hartmann, M. Meinke, W. Schroder, The constrained reinitialization equation for level set methods, Journal of Computaional Physics 229 (2010) 1514–1535.

[21] J. A. Sethian, A fast marching level set method for monotonically advancing fronts, Proceedings of the National Academy of Science of USA 93 (1996) 1591–1595.

[22] R. T. Whitaker, A level-set approach to 3D reconstruction from range data, International Journal of Computer Vision 29 (1998) 203–231.

[23] K. Krissian, C.-F. Westin, Fast sub-voxel reinitialization of the distance map for level set methods, Pattern Recognition Letter 29 (2008) 1532–1542.

[24] B. Shi, Z. Guo, Lattice boltzmann model for nonlinear convection diffusion equation, Physical Review E 79 (2009) 016701.

[25] D. W. Gladrow, A lattice boltzmann equation for diffusion, Journal of Statistical Physics 79 (1995) 1023–1032.

[26] C. Min, On reinitializing level set functions., J. Comput. Physics 229 (8) (2010) 2764–2772.

[27] V. Caselles, R. Kimmel, G. Sapiro, Geodesic active contours, International Journal of Computer Vision 22 (1997) 61–79.

[28] M. J. Black, G. Sapiro, D. H. Marimont, Robust anisotropic diffusion, IEEE Transactions on Image Processing 7 (1998) 421–432.

[29] F. Kuznik, C. Obrecht, G. Rusaouen, J.-J. Roux, LBM based flow simulation using GPU computing processor, Computers & Mathematics with Applications 59 (7) (2010) 2380–2392.

[30] J. M. Mark, J. R. Alistair, Memory transfer optimization for a lattice Boltzmann solver on kepler architecture nvidia GPUs, Computer Physics Communications 185 (2014) 2566–2574.

[31] F. Rinaldi, E. Dari, M. Venere, A. Clausse, A lattice boltzmann solver for 3d fluid simulation on gpu, Simulation Modelling Practice and Theory 25 (2012) 163–171.

[32] M. Astorino, J. Sagredo, A. Quarteroni, A modular lattice boltzmann solver for gpu computing processors, SeMA 59 (2013) 53–57.

[33] G. Aubert, P. Kornprobst, Mathematical problems in image processing, Springer.

[34] J. Krejza, M. Arkuszewski, S. Kasner, J. Weigele, A. Usty-owicz, R. Hurst, B. Cucchiara, S. Messe, Carotid artery diameter in men and women and the relation to body and neck size, Stroke 37 (4) (2006) 1103–1105.

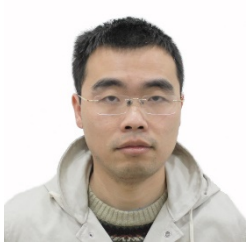
[35] S. Schmieschek, L. Shamardin, S. Frijters, T. Krueger, U. Schiller, J. Harting, C. PV., Lb3d: A parallel implementation of the lattice-boltzmann method for simulation of interacting amphiphilic fluids, Computer Physics Communications 217 (2017) 149–161.

[36] E. Calore, A. Gabbana, E. Pellegrini, S. Schifano, Massively parallel lattice boltzmann codes on large gpu clusters, Journal of Parallel Computing 58 (2016) 1–24.

[37] C. Riesinger, A. Bakhtiari, M. Schreiber, P. Neumann, H.-J. Bungartz, A holistic scalable implementation approach of the lattice boltzmann method for cpu/gpu heterogeneous clusters, Computation 5 (2017) 1–26.

[38] H. Yu, X. Chen, Z. Wang, D. Deep, E. Lima, Y. Zhao, D. S. Teague, Mass-conserved volumetric lattice Boltzmann method for complex flows with willfully moving boundaries, Physical Review E 89 (063304).

[39] Z. Wang, Y. Zhao, A. P. Sawchuck, M. C. Dalsing, H. W. Yu, GPU acceleration of volumetric lattice Boltzmann method for patient-specific computational hemodynamics, Computers and Fluids 115 (2015) 192–200.



Zhiqiang Wang is a PhD candidate in the Department of Computer Science, Kent State University. His research interest includes lattice Boltzmann model and its use in image processing and flow simulation.

Ye Zhao is a professor in the Department of Computer Science at the Kent State University, Ohio, USA. He has co-authored more than 50 refereed technical papers and served in many program committees including IEEE SciVis and VAST conferences. His current research projects include visual analytics of urban transportation data, multidimensional, text, and animated information visualization, medical image processing, and computational hemodynamics modeling

Huidan (Whitney) Yu is an associate professor of Mechanical Engineering at Indiana University Purdue University. Her research field is in computational fluid dynamics in general with expertise on kinetic theory based lattice Boltzmann method to model and simulate complex flows including turbulence.

Chen Lin is a clinical associate professor of Radiology at School of Medicine, Indiana University. His research focuses on clinical radiology and imaging sciences.

Alan P. Sawchuck is a professor of Surgery at School of Medicine, Indiana University. He is a vascular surgery doctor is affiliated with multiple hospitals in Indianapolis and conduct research in related topics.