HIGH SPEED PROCESSING OF BIOMEDICAL IMAGES USING PROGRAMMABLE GPU

Jin Young Hong and May D. Wang[*]

Medical Informatics & Bioimaging Lab
The Wallace H. Coulter Department of Biomedical Engineering
Georgia Institute of Technology and Emory University
313 Ferst Drive, Atlanta, GA 30332, USA

ABSTRACT

In this paper, we report our research results on high speed processing of large size biomedical images. The biomedical images usually contain various shapes of bio-organism. To accurately quantify these objects, shape-independent image processing techniques are needed. One of such techniques is level set (LS) method. However, its application to large size images is constrained by the extremely high computational cost. This is because a set of numerical simulations has to be performed repeatedly on every pixel of an image, and the general-purpose central processing unit (CPU) has only one execution core and limited memory bandwidth. Thus, we researched for techniques that can perform a large number of iterative tasks effectively. As a result, we designed and developed a graphics processing unit (GPU) based level set (LS) algorithm, GPU-LS, to process large size of biomedical images. In this paper, we report how we perform LS processing utilizing features of advanced graphics hardware. Comparing CPU-LS and GPU-LS, we have achieved average 12-13 times increase in processing speed.

1. INTRODUCTION

Imaging technology plays an important role in modern biology and medicine. Once the images are acquired by different image modalities (CT, MRI, PET, optical imaging etc.), the important next step is to identify histological and anatomical features from these images. These images usually contain various shapes of bio-organisms that need robust segmentation techniques for processing. Level set (LS) technique has become one of the attractive candidates in processing biomedical images due to its topological independence. It is a numerical method that keeps tracking of the evolution of contours or surfaces in an image by solving a partial differential equation (PDE) [9]. The evolving curves can freely split and merge, and eventually converge to the boundaries of arbitrary-shaped objects [7, 9, 12].

However, most of the biomedical images have large size due to the need for accuracy. The level set scheme is pixel-based iteration-intensive process. The computational cost of LS becomes so high that hinders its wide application in medicine. Thus, there have been attempts to improve the processing speed, such as narrow band [1] and fast marching methods [7].

In our research, one major goal of analyzing the biomedical images is to reconstruct the bio-organism's behaviors in 3-D space to assist in biological function study or clinical diagnosis and treatment planning. As such, we have been working on 3-D graphical rendering besides image analysis. One of the recent trends in graphics research is to improve the computational performance of numerical schemes by using graphics hardware. For example, Rumpf et al. [11] proposed to use graphics card to solve partial differential equations numerically, and implemented LS type scheme using graphics hardware with a constant speed term [10] in year 2000 to 2001. However, at that time, the graphics card supported only low precision and had no GPU programmability. Thus exploring hardware solution to increase PDE solution was in a preliminary stage.

Recently, the graphics hardware such as Geforce FX and ATI Radeon 9700, 9800 series provide both vertex and pixel-level programmability. For instance, Bolz et al. [2] and Krüger et al. [5] showed that implementing numerical schemes of partial differential equation using programmable GPU outperforms CPU implementation. Lefon et al. [6] implemented 3D narrow band level set on GPU using their own communication protocol between CPU and GPU to show that 3D visualization using level set solver can be done at interactive rate. But for a simpler application, this approach is unnecessarily complicated.

In this paper, we show that except for initial setup stage, a simple level set (LS) time stepping can be achieved completely in GPU without any communication with CPU. To analyze large size biomedical images, we

[*] Author to which the correspondence should be addressed (maywang@bme.gatech.edu).

0-7803-8554-3/04/$20.00 ©2004 IEEE.

have designed the fast level set image segmentation algorithm by using two off-screen buffers (pbuffers) [8]. In the next few sections, first, the Osher-Sethian level set method [7, 9, 12] will be reviewed. Next, the GPU-based LS solver is proposed. Then the speed and image results of the GPU-LS solver are presented and discussed with conclusion.

2. OVERVIEW OF STANDARD LEVEL SET METHOD

In this section, the general level set formulation and numerical solution is reviewed [7, 9, 12]. The evolving contour in 2D can be represented as a partial differential equation with a given initial condition:

$$\frac{\partial \phi}{\partial t} + F\,|\nabla \phi| = 0, \quad \text{given} \quad \phi(x, y, t = 0), \quad (1)$$

where $\phi(x, y, t)$ is the level set function, and F is the speed of moving contour. The evolution speed F can be divided into two components. The first is the constant propagation speed (F_0), which helps the moving contour expand or contract into the desired location. The second is the curvature speed ($-\kappa = -\nabla \cdot (\nabla \phi / |\nabla \phi|)$, which is related to the smoothness of the contour. Therefore, the level set equation (1) can be rewritten as the following:

$$\frac{\partial \phi}{\partial t} = -F_0\,|\nabla \phi| + \kappa\,|\nabla \phi|, \quad (2)$$

This curve evolution equation can be applied for image segmentation by integrating the image gradient information ($g_I = 1/(1+|\nabla(G_\sigma * I(x,y)|^m)$) with the speed term as follows:

$$\frac{\partial \phi}{\partial t} = -g_I \alpha F_0\,|\nabla \phi| + g_I \varepsilon \kappa\,|\nabla \phi| + \beta \nabla P \cdot \nabla \phi, \quad (3)$$

Here, $\alpha, \beta, \varepsilon$ are the weights on each terms, and P is the gradient of potential field defined as $P = -|\nabla(G_\sigma * I(x,y)|$. This new term acts as an additional attractive force to facilitate the convergence of the evolving curve to the boundary [3, 7]. The numerical solution for equation (3) is illustrated in [12].

In our project, we propose programmable GPU based LS solver to increase processing speed for equation (3).

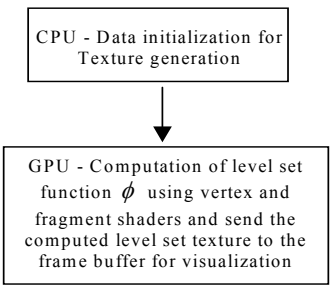

Figure 1. One pass communication from CPU to GPU

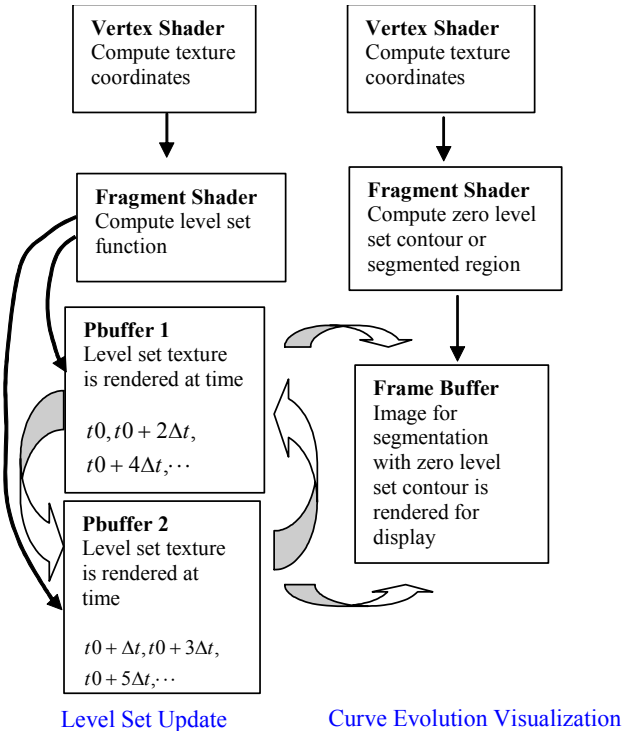

Figure 2. Flow of GPU-LS Solver

3. GPU IMPLEMENTATION

In this section, we describe the idea of using pbuffers in GPU to develop a GPU-LS solver for high speed processing of large size biomedical images. GPU has multiple processing pipelines and wider memory bandwidth than CPU, which enables graphics tasks to be achieved efficiently. To maximize this feature, we avoid CPU and GPU communication by allowing only one-pass communication from CPU to GPU in the initial step for solving level set equation (3). This design diagram is illustrated in **Figure 1.** The CPU reads the image for segmentation, computes its gradient image, and initializes the level set function $\phi(x, y, t = 0)$ as a signed distance field. To solve equation (3) numerically, $\phi(x,y,t)$, $g_I(x,y)$, and $\nabla P(x,y)$ are needed for whole image field. These data are transferred to the GPU as a 2D floating-point texture image represented as 4 channels RGBA (red, green, blue, and alpha index), and is rendered in pbuffer.

The level set equation (3) can be computed using finite difference method [12]. To compute level set function $\phi(x, y, t = t_0 + \Delta t)$, we need to know neighboring level set values of the previous time step, such as $\phi(x, y, t = t_0)$, $\phi(x + \Delta x, y, t = t_0)$, $\phi(x, y + \Delta y, t = t_0)$, etc. Because the level set function $\phi(x, y, t)$ is represented as a 2D texture image in the GPU, the pixel coordinates *(x, y)* correspond to the

texture coordinates. Because computing the neighboring texture coordinates in fragment shader would consume significant amount of instructions, we shift the coordinates by one pixel in neighboring directions in vertex shader to pass to the fragment shader. Using the texture coordinates of the neighboring pixels, the fragment shader can read its own and neighboring values at the same time to update the level set. This design is illustrated in **Figure 2.**

We used Cg (C for Graphics) language [4] to program the vertex and fragment shader. At time t_0, the initial level set is copied into Pbuffer1 by rendering a Pbuffer-size polygon using $\phi(x, y, t_0)$ as an input texture. At time $t_0 + \Delta t$, using Pbuffer1 as an input texture, the vertex and fragment shader computes the new level set values in Pbuffer2. At the next time step $t_0 + 2\Delta t$, using Pbuffer2 as an input texture, the new level set values can be computed in Pbuffer1. By using the other pbuffer as an input texture, we can avoid transferring data between system memory and video memory when updating the level set values to save time. In previous research, Lefon [6] reported penalty in switching pbuffers, citing the earlier paper by Bolz [2]. In our work, by replacing the fragment shader by a constant color one, and maintain all other settings the same as a full LS solver, we observe 1700 frames/second processing speed. This indicates that the pbuffer switching delay does not exist any more and the penalty disappeared.

To monitor the LS evolution, typically, the software needs to first compute contour or region and then draw them by lines or pixels on the image for visualization. This is a slow process if being done in CPU. In our work, we accelerate this process by using vertex and fragment shader for a frame buffer, and by using the Pbuffer and original image as input textures. The zero level set contour or segmented region are easily computed and rendered in the frame buffer. As a result, we achieve the real-time processing of relatively large size images.

4. RESULTS

4.1. Image Segmentation Results Using GPU-LS

To solve the long processing time problem, we have developed GPU-LS to process large size of images that contain different shapes of bio-organisms from micro (cell) to macro (organ anatomy) levels. For example, to study cell behavior for cancer diagnosis and drug response, in vitro imaging data are acquired using bio-nanoimaging acquisition technology. Specifically, in **Figure 3**, two types of cells' in vitro states were studied by (1) injecting blue-color and red-color tagged contrast agents into the cell environment; and by (2) acquiring images using optical imaging device. To obtain a precise description of the cell behavior change, expression value for each cell is obtained by finding the shape of the cell. As shown in **Figure 3**, we apply our GPU-LS solver to process these 1024x1024 cell images. We started by putting multiple initial seeds on the images**.** We applied max-min filtering first to enhance the intensity at region of interest (i.e. cell region) for better tracking of evolution curve. We show the intermediate tracking frames in the GPU-LS solver working progress, and the final converging results. In both cases, the segmentation results are good using floating-point precision GPU. This segmentation results gives accurate representation of each cell's behavior state.

As another example, we also apply our GPU-LS solver to visible male human anatomy image of 1024x1024 to assist clinical diagnosis and surgical operation study for another project. Even with sophisticated brain texture, the LS gives great segmentation results as shown in **Figure 4.**

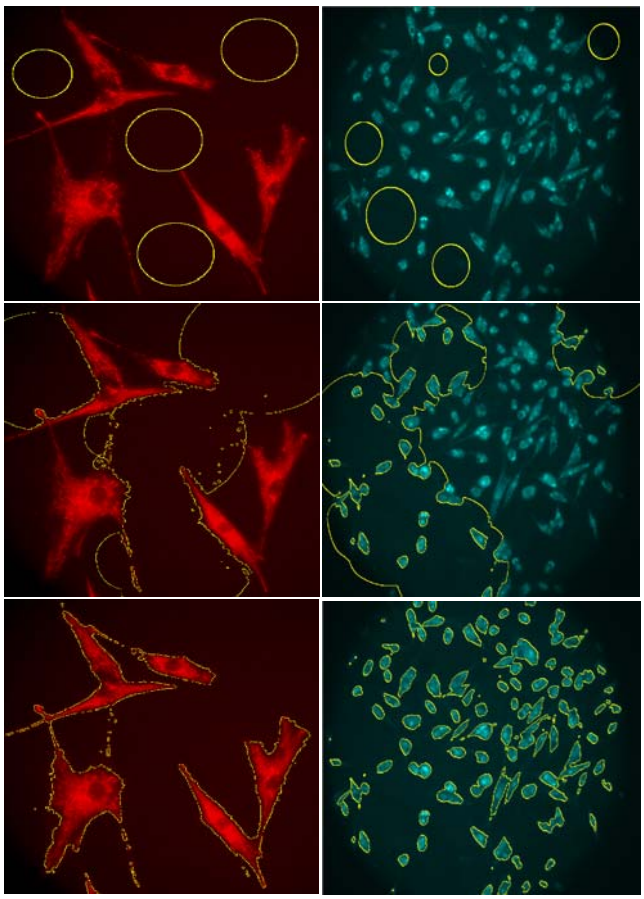

Figure 3. GPU-LS solver segmentation process in tracking cancer cell behaviors

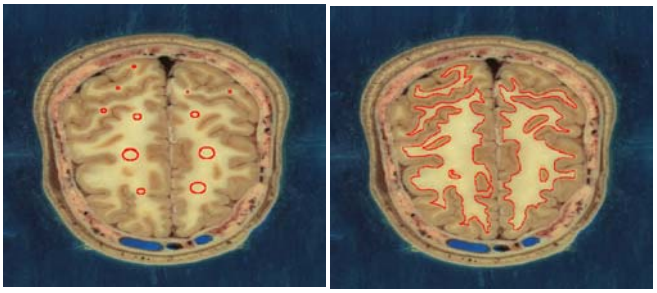

Figure 4. Human Anatomy Data (Head)

4.2. Computation Performance

Besides the fact that our GPS-LS solver can give highly accurate segmentation results for studying either cell behavior or human anatomy, the biggest advantage of our system is that it reduces computation time significantly. We performed a quantitative performance comparison between our GPU-LS solver and CPU-LS solver. The test image sizes are 256x256, 512x512, and 1024x1024. The test platform is Pentium IV 2.8GH with ATI Radeon 9700 graphics card. We record the processing frame rates in processing each image with 500 iterations, and report in Table 1.

	GPU (fps)	CPU (fps)
256x256	646.4	58.4
512x512	169.2	13.9
1024x1024	40.81	3.04

Table 1. Computation performance comparison of GPU-LS and CPU-LS Solvers for different size images.

As we can see the computation by GPU is at least 10 times faster than by CPU for all cases. We observe that as the image size increases, our GPU-LS solver computational performance increases more.

5. CONCLUSIONS AND FUTURE WORK

In this research, we illustrate that: first, the level set (LS) is a great scheme for processing diverse types of biomedical images from cell to organ level; and the second GPU-based LS solver can achieve at least ***10 times*** speed up in processing the images in comparing to standard CPU-LS solver. We also illustrate that a relatively simple and effective strategy can avoid CPU and GPU communication.

One of the biggest challenges in biomedical imaging research now is how to integrate multi-modality (e.g., MRI, optical images) and multi-organism level (e.g., molecular to cell to tissue to organ) imaging data for patient care. The LS solver is a perfect choice for our long-term project of integrating molecule level imaging data, with MRI and digital human anatomy data to visualize cancer call growth to metastasis processes inside human body for early diagnosis, and treatment planning.

The molecular imaging (10G+bytes), the anatomy imaging (55G+bytes), and the medical imaging (2G+bytes) data are so huge. In order to get 3-D integrated analysis in reasonable time, high speed processing has to be done at every possible way. Because our GPU-LS provides at least 10 folds faster performance, we conclude that GPU is very useful in helping us achieve the goal. As the graphics card gets improved, more complicated vertex and fragment shader programmability will be provided (i.e. longer codes and flow controls).

In the future, we plan to continually develop joint software-hardware optimized algorithms to perform high speed processing and visualization in our research of integrating high-resolution human anatomy, molecular imaging, and medical imaging data for personalized medicine.

REFERENCES

[1] D. Adalsteinsson and J. Sethian, "A Fast Level Set Method for Propagating Interfaces", *J. Computational Physics*, Vol 118, No. 2, pp. 269-277, 1995

[2] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder, "Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid", SIGGRAPH 2003

[3] V.Caselles, R. Kimmel, and G. Sapiro, "Geodesic Active Contours", *Proc. IEEE ICCV*, pp. 694-699, 1995

[4] R. Fernando and M. Kilgard, *The Cg Tutorial,* NVIDIA, Addison-Wesley, 2003

[5] J. Krüger and R. Westermann, "Linear Algebra Operators for GPU Implementation of Numerical Algorithms", SIGGRAPH, 2003

[6] A. Lefohn, J. Kniss, C. Hansen, and R. Whitaker, "Interactive Deformation and Visualization of Level Set Surfaces Using Graphics Hardware", *IEEE Visualization*, pp 75-82, 2003

[7] R. Malladi and J.A. Sethian "A real-time algorithm for medical shape recovery," *Int. Conference on Computer Vision*, pp. 304-310, 1998.

[8] C. Oat. Rendering to an off-screen buffer with WGL_ARB_pbuffer, http://www.ati.com/developer/techpapers.html

[9] S. Osher and J. Sethian, "Fronts propagation with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, pp.12–49, 1988.

[10] M. Rumpf and R. Strzodka, "Level set segmentation in graphics hardware", *Proc. ICIP*, pp.1103-1106, 2001

[11] M. Rumpf and R. Strzodka, "Nonlinear diffusion in graphics hardware", *Proc. of EG/IEEE Symposium on Visualization VisSym*, pp. 75-84, 2001

[12] J. A. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, 1996