Graphics Hardware-Based Level-Set Method for Interactive Segmentation and Visualization

Helen Hong¹ and Seongjin Park²

¹ Division of Multimedia Engineering, College of Information and Media,
Seoul Women's University, 126 Gongreung-dong, Nowon-gu, Seoul 139-774 Korea

² School of Computer Science and Engineering, Seoul National University,
San 56-1 Shinlim-dong, Kwanak-gu, Seoul 151-741 Korea
hlhong@swu.ac.kr, sjpark@cglab.snu.ac.kr

Abstract. This paper presents an efficient graphics hardware-based method to segment and visualize level-set surfaces as interactive rates. Our method is composed of memory manager, level-set solver, and volume renderer. The memory manager which performs in CPU generates page table, inverse page table and available page stack as well as process the activation and inactivation of pages. The level-set solver computes only voxels near the iso-surface. To run efficiently on GPUs, volume is decomposed into a set of small pages. Only those pages with non-zero derivatives are stored on GPU. These active pages are packed into a large 2D texture. The level-set partial differential equation (PDE) is computed directly on this packed format. The memory manager is used to help managing the packing of the active data. The volume renderer performs volume rendering of the original data simultaneously with the evolving level set in GPU. Experimental results using two chest CT datasets show that our graphics hardware-based level-set method is much faster than software-based one.

Keywords: Segmentation, Level-Set, Volume rendering, Graphics hardware, CT, Lung.

1 Introduction

The level-set method is a numerical technique for tracking interfaces and shapes[1]. The advantage of the level-set method is that one can perform numerical computations involving curves and surfaces on a fixed Cartesian grid without having to parameterize these objects. In addition, the level-set method makes it easy to follow shapes which change topology. All these make the level-set method a great tool for modeling time-varying objects. Thus, deformable iso-surfaces modeled by level-set method have demonstrated a great potential in visualization for applications such as segmentation, surface processing, and surface reconstruction. However, the use of level sets in visualization has a limitation in their high computational cost and reliance on significant parameter tuning.

Several methods have been suggested for accelerate the computation time. Adalsteinson and Sethian [2] have proposed the narrow band method, which only computes the points near the front at each time step. Thus it is more efficient than the standard level-set approach. However, the computational time is still large, especially when the image size is large. Paragios and Deriche introduced the Hermes algorithm which propagates in a small window each time to achieve a much faster computation. Sethian [3] presented a monotonically advancing scheme. It is restricted to a one directional speed term and the front's geometric properties are omitted. Unfortunately, the stop criteria have to be decided carefully so that the front will not exceed the boundary. Whitaker[4] proposed the sparse-field method, which introduces a scheme in which updates are calculated only on the wavefront, and several layers around that wavefront are updated via a distance transform at each iteration.

To overcome those limitations of software-based level-set methods, we propose an efficient graphics hardware-based method to segment and visualize level set surfaces as interactive rates.

2 Level-Set Method on Graphics Hardware

Our method is composed of memory manager, level-set solver and volume renderer as shown in Figure 1. First, in order to help managing the packing of the active data, the memory manager generates page table, inverse page table and available page stack as well as process the activation and inactivation of pages. Second, level-set solver computes only voxels near the iso-surface like the sparse field level-set method. To run efficiently on GPUs, volume is decomposed into a set of small pages. Third, volume renderer performs volume rendering of the original data simultaneously with the evolving level set.

2.1 Memory Manager

Generally, the size of texture memory in graphics hardware is rather small. Thus, there has a limitation to load a large volume medical dataset which has over 1000 slices with 512 x 512 image size to texture memory. In order to overcome this limitation, it is required to load level sets only near the iso-surface, which called active pages. In this section, we propose an efficient method to manage these active pages.

Firstly, main memory in CPU and texture memory in GPU is divided into pages. Then data structure as shown in Fig. 2 is generated. In order to exchange the corresponding page number between main memory and texture memory, the page table which converts the page number of main memory to corresponding page number of texture memory and the inverse page table which converts the page number of texture memory to corresponding page number of main memory is generated respectively. In addition, the available page stack is generated to manage empty pages in texture memory.

Fig. 1. The flow chart of our method on graphics hardware

Fig. 2. Data structure for memory management

In level-set method, the page including front is changed as front is increased or decreased. To manage these pages, activation and inactivation is performed as shown in Fig. 3. The activation process is occurred when evolving front use the inactive page in texture memory. At this process, main memory asks new page of texture memory to available page stack. Then top page of available page stack is popped as shown in Fig. 3(a). The inactivation process is occurred when evolving front is departed from active pages of texture memory. As shown in Fig. 3(b), main memory asks the removal of active page to texture memory, and the removed active page is pushed to available page stack.

Fig. 3. The process of page activation and inactivation (a) page activation process (b) page inactivation process

During level-set computation in GPU, partial differential equation is computed using information of current and neighbor pixels. In case of referring inside pixel of page in texture memory, PDE can be calculated without preprocessing. In case of referring boundary pixel of page, neighbor page should be referred to know information of neighbor pixel. However, it is difficult to know such information during PDE calculation in GPU. In the case, vertex buffer is created in CPU to save the location of current and neighbor pixels. For this, we define nine different cases as shown in Fig. 4. In 1st, 3rd, 5th, 7th vertices, two pages neighbor to them are referred to know and save the location of neighbor pixel to vertex buffer with the location of current pixel. In 2nd, 4th, 6th, 8th vertices, one page neighbor to them are referred. In 9th vertex, the

location of current is saved to vertex buffer without referring of neighbor page. The location of neighbor pixel is calculated using page table and inverse page table like Eq. (1).

Fig. 4. Nine different cases for referring neighbor page

$$T_{num} = \frac{T_{addr}}{PageSize} \quad (1)$$

$$M_{num} = InversePageTable(T_{num})$$

$$neighbor(M_{num}) = M_{num} + neighborOffset$$

$$neighbor(T_{num}) = PageTable(neighbor(M_{num}))$$

where T_{num} is page number in texture memory, $T_{address}$ is page address in texture memory, M_{num} is page number in main memory, $PageSize$ is defined in 16×16 .

2.2 Level-Set Solver

The efficient solution of the level set PDEs relies on updating only those voxels that are on or near the iso-surface. The narrow band and sparse field methods achieve this by operating on sequences of heterogeneous operations. For instance, the sparse-field method keeps a linked list of active voxels on which the computation is performed. However, the maximum efficiency for computing on GPU is achieved when homogeneous operations are applied to each pixel. To apply different operations to each pixel in page has a burden in CPU-to-GPU message passing. To run efficiently on GPUs, our level-set solver applies heterogeneous operations to nine different cases divided in creation of vertex buffer.

Fig. 5 shows that vertex buffer is transferred to GPU during vertex shading, which is divided into apex (1^{st} , 3^{rd} , 5^{th} , 7^{th}), edge (2^{nd} , 4^{th} , 6^{th} , 8^{th}) and the inner parts (9^{th}). Sixteen vertex buffers which include the location of four different apex points for the apex case, eight ends points for the edge case, and four apex points for the inner case are transferred. Then level-set computations are achieved by using Eq. (2) and (3).

$$D(I) = \varepsilon - |I - T| \quad (2)$$

$$\phi = \phi - |\nabla \phi| \cdot D(I) \quad (3)$$

where I is the intensity value of the image, $D(I)$ is speed function, ϕ is level-set value, T and ε is the average intensity value and standard deviation of segmenting region, respectively.

Fig. 5. The process of efficient level-set operation in GPU

2.3 Volume Renderer

The conventional software-based volume rendering techniques such as ray-casting and shear-warp factorization have a limitation to visualize level-set surfaces as interactive rate. Our volume renderer performs a texture-based volume rendering on graphics hardware of the original data simultaneously with the evolving level set.

Firstly, updated level-set values in texture memory are saved to main memory through inverse page table. Then texture-based volume rendering is applied for visualizing original volume with level-set surfaces. For efficient memory use, we use only two channels for intensity value and level set value instead of using RGBA four channels. Then proxy geometry is generated using parallel projection. Finally, we map three-dimensional texture memory in GPU to the proxy geometry. The mapped slices onto the proxy geometry render using compositing modes that include maximum intensity projection.

3 Experimental Result

All our implementation and tests were performed using a general computer equipped with an Intel Pentium 4, 2.4 GHz CPU and 1GB memory. The graphics hardware was ATI Radeon 9600 GPU with 256 MB of memory. The programs are written in the

DirectX shader program language. Our method was applied to each unilateral lung of two chest CT datasets to evaluate its accuracy and processing time. The volume resolution of each unilateral lung is $512 \times 256 \times 128$. For packing active pages, the size of 2D texture memory is 2048×2048 . Fig. 6 and 7 show how our method segment accurately in two- and three-dimensionally. The segmented lung boundary is presented in red. In Fig. 7, original volume with level set surfaces is visualized by using maximum intensity projection.

Fig. 6. The results of segmentation using our graphics hardware-based level-set method

Fig. 7. The results of visualizing original volume with level-set surfaces

We have compared our technique with software-based level-set method under the same condition. Table 1 shows a comparison of the total processing time using the 2 different techniques. The total processing time includes times for performing page management and level-set computations. As shown in Table 1, our method is over 3.4 times faster than software-based level set method. In particular, our method for computing level set PDE is over 14 times faster than that of software-based level set method.

Table 1. The comparison results of total processing time using two different techniques.

(sec)					
Method	Dataset	Page manager	Level set solver	Total processing time	Average
Proposed graphics hardware-based method	AL	0.38	0.068	0.45	0.44
	AR	0.37	0.066	0.44	
	BL	0.38	0.073	0.45	
	BR	0.36	0.067	0.43	
Software-based method	AL	0.54	0.93	1.47	1.48
	AR	0.55	0.94	1.49	
	BL	0.55	0.94	1.49	
	BR	0.54	0.93	1.47	

L: left lung, R: right lung

4 Conclusion

We have developed a new tool for interactive segmentation and visualization of level set surfaces on graphics hardware. Our memory manager helps managing the packing

of the active data. A dynamic, packed texture format allows the efficient processing of time-dependent, sparse GPU computations. While the GPU updates the level set, it renders the surface model directly from this packed texture format. Our method was over 3.4 times faster than software-based level set method. In particular, our method for computing level set PDE was over 14 times faster than that of software-based level set method. The average of total processing time of our method was 0.6 seconds. The computation time for memory management took almost times in total processing time. Experimental results show that our solution is much faster than previous optimized solutions based on software technique.

Acknowledgement

This study is supported in part by Special Basic Research Program grant R01-2006-000-11244-0 under the Korea Science and Engineering Foundation and in part by Seoul R&D Program.

References

1. Osher S, Sethian J.A., Front propagating with curvature dependant speed: algorithms based on Hamilton-Jacobi formulation, *Journal of Computational Physics*, Vol. 79 (1988) 12-49.
2. Adalsteinson D, Sethian J.A., A fast level set method for propagating interfaces, *Journal of Computational Physics* (1995) 269-277.
3. Sethian J.A., A fast matching level set method for monotonically advancing fronts, *Proc. Natl. Acad. Sci. USA* Vol. 93 (1996) 1591-1595.
4. Whitaker R., A level-set approach to 3D reconstruction from range data, *International Journal of Computer Vision* (1998) 203-231.