Optimized GPU Framework for Semi-implicit AOS Scheme Based Speckle Reducing Nonlinear Diffusion

Tian Cao*[a], Bo Wang[a], Dong C. Liu[a]
[a]Computer Science College, Sichuan University, Chengdu, China

ABSTRACT

Ultrasound image quality is degraded because of the presence of speckle, which causes loss of image contrast resolution and makes the detection of small features difficult. The traditional nonlinear diffusion filtering of speckle reduction with explicit schemes can achieve desirable results, but they are only stable for very small time steps. Semi-implicit additive operator splitting (AOS) schemes for nonlinear diffusion are stable for all time size and more efficient than the traditional explicit schemes. However, the AOS schemes are still inefficient for real time speckle reducing of ultrasound images. Current graphics processing units (GPUs) offers an opportunity to boost the computation speed of AOS schemes through high computational power at low cost. In this paper, an optimized GPU framework for AOS schemes is presented. By using the well-established method of cyclic reduction of tridiagonal systems in our framework, we are able to implement the AOS schemes on GPU. Experiments from CPU implemented AOS schemes and our GPU based framework show that our method is about 10 times faster than the CPU implementation. Our presented framework deals with the local coherence anisotropic diffusion, but it can be generalized to the class of nonlinear diffusion methods which can be discretized by AOS schemes.

Keywords: Speckle Reduction, Anisotropic diffusion, GPU, Ultrasound imaging, Local coherence

1. INTRODUCTION

Ultrasound imaging is an essential tool for medical diagnosis. Due to the nature of ultrasound imaging, the speckle noise decreases the image contrast resolution, which is a major limitation for doctors' interpretation of the ultrasound images. Finding appropriate ways to reduce speckle noise to help the doctor's diagnosis is a hot area for researchers in medical image processing.

1.1 Current Methods

Recently, a class of new methods called nonlinear (anisotropic) diffusion has been proposed in the field of ultrasound image speckle reduction. Many literatures discuss and study this class of methods such as speckle reducing anisotropic diffusion (SRAD) [1], detail preserved anisotropic diffusion (DPAD) [2], speckle constrained anisotropic diffusion (SCAD) [3], and oriented speckle reducing anisotropic diffusion (OSRAD) [4]. This class of methods is favorable because of its capability in preserving edge and reducing speckle. However, most of these methods based on traditional explicit schemes are very slow in CPU implementation and only stable for small time step. In SRAD and DPAD, the authors employed the statistics characteristics of speckle to control the diffusion coefficients at each point. They are very good in reducing speckle; however, their speeds are very slow. The real-time speckle reduction (RTAD) [10] method is a fast anisotropic diffusion method based on semi-implicit and explicit discretizations. In [13] and [14], we also proposed two other semi-implicit discretization based fast anisotropic diffusion methods.

By using the semi-implicit additive operator splitting (AOS) schemes, the diffusion method is unconditionally stable and can be much faster than traditional methods. However, the AOS schemes are still inefficient to reach the real time requirement of ultrasound imaging on CPU. Modern computer graphics hardware provides tremendous computational power. The performance of these GPUs is also growing at an extraordinary rate. Recently, much effort has been focused on using the computational power of graphics processing units (GPUs) for general purpose computations [11]. J. Kruger and R. Westermann introduce a framework for the implementation of linear algebra operators on GPU in [13]. M. Kass,

*Tian Cao Email Address:ct.radiate@gmail.com, Tel: +86-13032890220

A. Lefohn and J. Owens applied GPU on depth-of-field computation [12]. Both of them are utilize the GPU computation power to solve the linear systems and can achieve real time frame rates in their applications. M. Rumpf and R. Strzodka used Graphics hardware in nonlinear diffusion of image processing [14]. But their method is still only stable for small time step.

To address the problem of reaching real-time AOS-based speckle reduction, an optimized GPU framework is presented in this paper. Our optimized GPU framework contains three parts, a low pass filtering, generating the tridiagonal matrix, and solving the tridiagonal system. Each part of the framework is implemented on GPU and is flexible to modify for different applications. The proposed framework use the well-established method of cyclic reduction to solve the tridiagonal systems in the AOS scheme based speckle reduction, because the traditional LU decomposition method which can be implemented on CPU is not suitable for GPU implementation. While cyclic reduction requires more arithmetic than an LU solver, it is still amenable to a parallel GPU implementation.

1.2 Overview

This paper is organized as follows. In Section 2, we present the anisotropic diffusion model and the fast discretization method, and then the optimized GPU framework is introduced. Experiments and Testing results of our method from in vivo images are shown in Section 3. Conclusions and future directions are given in Section 4.

2. METHODOLOGY

2.1 Semi-implicit AOS scheme based nonlinear diffusion

Weickert proposed semi-implicit AOS schemes to do fast anisotropic diffusion [5, 6]. The AOS scheme based anisotropic diffusion is based on CLMC filter [7], and the 1-D CLMC filter is as

$$\frac{I_i^{t+1}-I_i^t}{\tau}=\sum_{j\subset \mathrm{N}(i)}\frac{g_j^k+g_i^k}{2h^2}(I_j^t-I_i^t) \tag{1}$$

where I_i^t and I_i^{t+1} are the image pixel values of current and next steps respectively, τ is the time step size (TSS), h is the grid size, and N(i) is the set of two neighbours of the pixel *i* (boundary pixels have only one neighbour). The diffusivity function $g(\cdot)$ in (1) is as,

$$g_i^k := g[\frac{1}{2}\sum_{p,q\subset \mathrm{N}(i)}(\frac{I_p^k-I_q^k}{2h})^2] \tag{2}$$

$$g(s)=\begin{cases}1,(s\le 0)\\ 1-\exp[\frac{-3.315}{s/\lambda}],(s>0)\end{cases} \tag{3}$$

where λ is a threshold value. Weickert integrated (1) to the following matrix-vector notation as

$$\frac{I^{t+1}-I^t}{\tau}=A(I^t)I^t \tag{4}$$

where $A(I^t)$ is a tridiagonal matrix with $A(I^t)=[a_{ij}(I^k)]$ and

$$a_{ij}(I^t)=\begin{cases}(g_i^k+g_j^k)/2h^2 & [j\in \mathrm{N}(i)],\\ -\sum_{n\subset \mathrm{N}(i)}(g_j^k+g_n^k)/2h^2 & (j=i),\\ 0 & (\text{else}).\end{cases} \tag{5}$$

In order to reduce the time complexity, Weickert proposed the semi-implicit AOS scheme, for a K by L image, the 2-D formulation is as

$$I^{t+1}=\frac{1}{2}\sum_{l=1}^{2}[U+2\tau A_l(I^t)]^{-1}I^t \tag{6}$$

where U is a KL by KL identity matrix. Weickert also proved that AOS scheme is unconditionally stable.

2.2 An optimized GPU framework for Semi-implicit AOS scheme based nonlinear diffusion

In our previous work, Wang [13] proposed local coherence based fast speckle reducing anisotropic diffusion. We proposed our framework based on this method. There are three parts in our framework: a Low-pass filtering, generating the tridiagonal matrix by looking up the table of diffusivity and solving the tridiagonal system. Each part is independent and can be modified for different applications. Moreover, our framework is suitable for semi-implicit scheme based nonlinear diffusion method in ultrasound speckle reduction (SIND) [14]. The reason is because [13] and [14] are all based on semi-implicit AOS scheme, in the process of solving the tridiagonal linear equation, we can apply our GPU framework. More importantly, because we only use GPU instead of CPU for parallel computing, our method can get the same denoising results as each original AD method, at the same time, we could reduce much processing time.

The diffusion equation of [13] is as

$$\frac{\partial I}{\partial t}=div[l\cdot\nabla I] \tag{7}$$

where $l(\cdot)$ is the local coherence function which is the squared difference of eigenvalues of multiscale structure tensor, it is as

$$l(|\mu_1-\mu_2|)=1/1+(|\mu_1-\mu_2|/K)^2 \tag{8}$$

where μ_1 and μ_2 are the eigenvalues of structure matrix at each point, structure matrix is as $J(\nabla I_\sigma)=(\nabla I_\sigma\cdot\nabla I_\sigma^T)$. Here, ∇I_σ is the gradient of a smoothed version of I which is obtained by convolving I with a Gaussian of standard deviation σ. Converting (7) to a matrix-vector notation and adopting AOS scheme, (8) comes down to the following iteration scheme

$$I^{t+1}=\frac{1}{2}\sum_{l=1}^{2}[U+2\tau T_l(I^t)]^{-1}I^t \tag{9}$$

where U is a KL by KL identity matrix for a K by L image, τ is the time step size (TSS), $T_l(I^t)$ is a tridiagonal matrix, with $T_l(I^t)=[t_{ij}(I^t)]$ and

$$t_{ij}(I^t)=\begin{cases}l_j^t/h^2 & [j\in \mathrm{N}(i)],\\ -\sum_{n\subset \mathrm{N}(i)}l_n^t/h^2 & (j=i),\\ 0 & (\text{else}).\end{cases} \tag{10}$$

where l is the diffusion coefficient. $\mathrm{N}(i)$ is the set of the two neighbors of pixel i. From above equations we can see that the (9) is similar with (6) except the $T_l(I^t)$. The diffusion equation of [14] is as,

$$I^{t+1} = \frac{1}{2}\sum_{l=1}^{2}[U + 2\tau D_l(I^t)]^{-1} I^t \tag{11}$$

where U is a identity matrix as (9), τ is the time step size (TSS), $D_l = (d_{ij}l)_{ij}$ is the same as.

$$d_{ij}(I^t) = \begin{cases} (c_i^k + c_j^k)/2h^2 & [j \in \mathrm{N}(i)], \\ -\sum_{n \subset \mathrm{N}(i)} (c_j^k + c_n^k)/2h^2 & (j = i), \\ 0 & (\text{else}). \end{cases} \tag{12}$$

where *c(q)* is the diffusivity function which is a function of local statistics in the image

$$c(q) = \frac{1}{1 + [q^2 - q_0^2]/[q_0^2(1 + q_0^2)]} \tag{13}$$

(11) is also similar with (6) and (9) except the function of $D_l(I^t)$, so we can use a same GPU framework to deal with this kind of diffusion methods.

2.2.1 Low-pass Filter

The low-pass filtering is usually performed with a Gaussian filter kernel. In this paper, in order to reduce the influence of speckle noise, the image was regularized by a 5×5 Gaussian kernel.

The 2D Gaussian convolution can also be performed by convolving with two 1D Gaussian kernels along x and y axis directions which means that the 2D Gaussian filtering can be implemented in two GPU passes. The Gaussian smoothing can be replaced by a simpler low-pass filter, such as box filter.

2.2.2 Generating table of diffusivity (TOD) and the tridiagonal matrix

We can generate a TOD based on different diffusivity functions. The diffusivity function we mentioned before is defined in the CLMC filter. In our previous work, Wang [13] has proposed local coherence based fast speckle reduction anisotropic diffusion (LCSRAD) method. In LCSRAD, the diffusion function is based on the local coherence. In SIND [14], the diffusivity function is based on the signal-to-noise ratio, which involves the statistics of ultrasound speckle. Then our framework can be applied to LCSRAD, SIND and some other nonlinear diffusion methods which can be discretized by AOS scheme just by generating different TODs.

In LCSRAD the independent variables of the diffusivity function are the difference of two eigenvalues of structure matrix ($\nabla I \bullet \nabla I^T$) where these eigenvalues can be computed from *Ix*, *Iy* at a pixel. To fully utilize the power of GPU on memory access in parallel, the diffusion coefficient at each point can be determined from a table that includes the values of diffusivity function of all possible cases of *Ix*, *Iy*, i.e., a range of Ix and *Iy* from -255 to 55. We, therefore, generate a TOD with 511×511 size of coefficients, and at each point, we only need to use Ix, *Iy* as index to retrieve the diffusion coefficient from the TOD. Furthermore, we can generate the tridiagonal matrix using TOD. For each row in the tridiagonal matrix, we only need two lookup operations of the TOD to calculate three values, i.e., the diagonal and its two neighbors. On a GPU with *N* shader processors, our implementation can gain a speedup of N times faster due to its parallel architecture than that from the traditional CPU implementation.

2.2.3 Solving a tridiagonal linear system

Equation (6) describes a tridiagonal linear system of the form:

$$\begin{pmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & c_3 & \\ & & \ddots & \ddots & \ddots \\ 0 & & & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_n \end{pmatrix} \tag{14}$$

where a_i, b_i, c_i and F_i, i = 1,…, n, are given. The key to fast implementation of our AOS scheme is the efficient solution of the tridiagonal system of (14). Many researchers have developed solvers for a variety of linear system on graphics hardware, but most of them focused on general linear system and not appropriate for this particular case. Recently, Michael and Aaron used cyclic reduction method to solve the tridiagonal linear system in their computation of depth-of-field effects on GPU. In our paper we used this well-established cyclic reduction method to solve the tridiagonal linear system of Eqn. (14).

In AOS scheme on a 2D image, our implementation involves two stages: a sequence of tridiagonal solvers for each row (called row processing) followed by column processing along each column. For an m×n image, we have n (m×m) tridiagonal matrices. Each row of a tridiagonal matrix contains 3 nonzero elements (an, bn, cn), and those elements are stored in a single texel as (R, G, B) components. This means that we can save a tridiagonal matrix as m texels and n(m×m) tridiagonal matrices are now saved as a "texture" with a size of m×n. The row processing now deals with n-times tridiagonal solvers where one solver outputs m values along one row and finally generates an m×n image. Similarly, the column processing computes another m×n image. The final result will be the sum of these two images. The above strategy is specially designed for GPU platform where we can utilize its parallelism on memory storage/retrieve and suitable for parallel computation of our cyclic reduction based tridiagonal solver.

Kass and Lefohn's tridiagonal solver [12] has a time complexity of 3-times higher than the traditional LU solver in serial processing. In GPU platform, however, their cyclic reduction can achieve almost N-times faster where N is the number of shader processors. Therefore, our implemented tridiagonal solver has a speedup of N/3 compared with the traditional LU solver (e.g., a 10-times faster for NVIDIA 8600GT with 32 shader processors).

Fig. 2.1 is the Flow Chart of our optimized GPU framework.

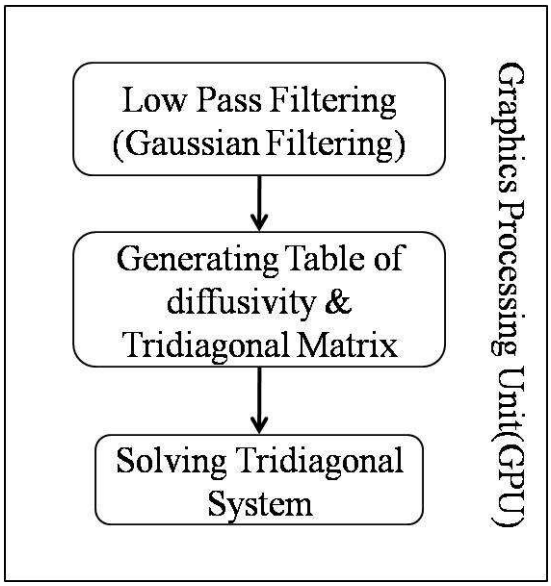


Fig. 2.1 Flow Chart of the optimized GPU Framework

3. RESULTS

3.1 Speckle reduction of ultrasound image

Results of our AOS based LCSRAD method are illustrated in Fig. 3.1 to 3.4 where Fig. 3.x (b) and Fig. 3.x (c) show images with different parameters. It's clear that, more iteration gives better speckle reduction but takes more processing time. GPUs are deeply pipelined architectures. In our work, the data is transferred to GPU at the very beginning. Then, all the procedures are processed in GPU without any feedback to CPU, and the result is directly sent to frame buffer which could be displayed on the screen.

Using our proposed GPU framework to do speedup will have the same speckle reduction result as LCSRAD. Because our framework is the same as original LCSRAD except we use different method to solve the tridiagonal linear equation of LCSRAD. The following experiments show the speckle reduction results of our proposed GPU framework.

3.2 Runtime Comparison between CPU and GPU

We have implemented our optimized GPU framework on a 2.20 GHz Athlon 64 4200+ system running Windows XP with an NVIDIA GeForce 8600GT GPU with 32 shader processors, see comparisons in Table 3.1 to 3.3 for each key part and Table 3.3 and Table 3.4 for image results from different iterations. It's clear that our GPU platform can have 10-times speedup compared with that from CPU and this advantage will make real-time speckle reduction imaging possible.

Image Size(pixels)	CPU time (ms)	GPU time (ms)
256×256	15	1
512×512	62	3

Table 3.1 Gaussian Filtering

Image Size(pixels)	CPU time (ms)	GPU time (ms)
256×256	32	2
512×512	131	5

Table 3.2 Generating tridiagonal-matrix

Image Size(pixels)	CPU time (ms)	GPU time (ms)
256×256	64	8
512×512	248	25

Table 3.3 Solving tridiagonal system

Image Size (pixels)	CPU Frame rate(frames/s)	GPU Frame rate(frames/s)
256×256	8-12	95-100
512×512	2-3	38-42

Table.3.3 LCSRAD, 1 iteration, TSS = 1.5.

Image Size (pixels)	CPU Frame rate(frames/s)	GPU Frame rate (frames/s)
256×256	4-7	35-40
512×512	1-2	12-15

Table.3.4 LCSRAD, 4 iterations, TSS = 1.5

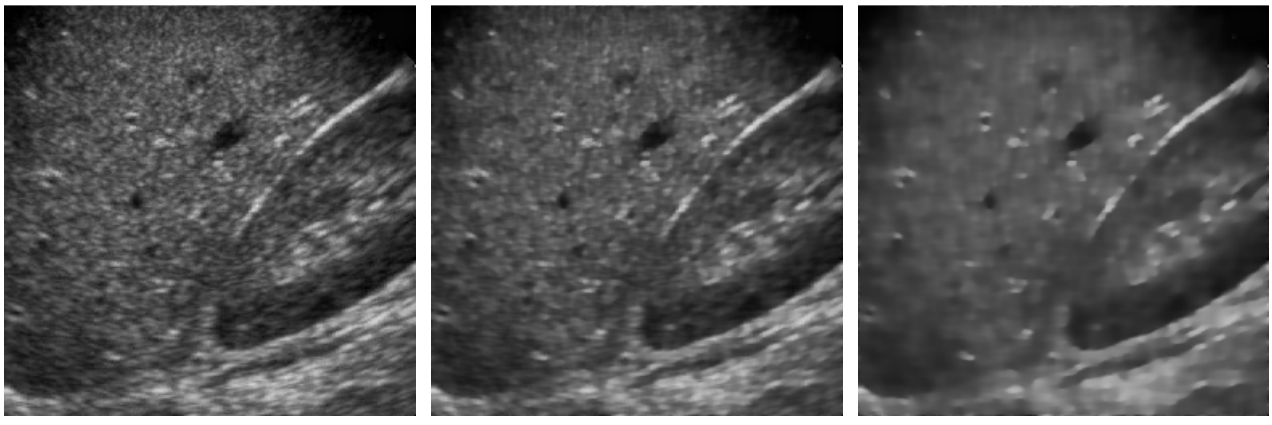
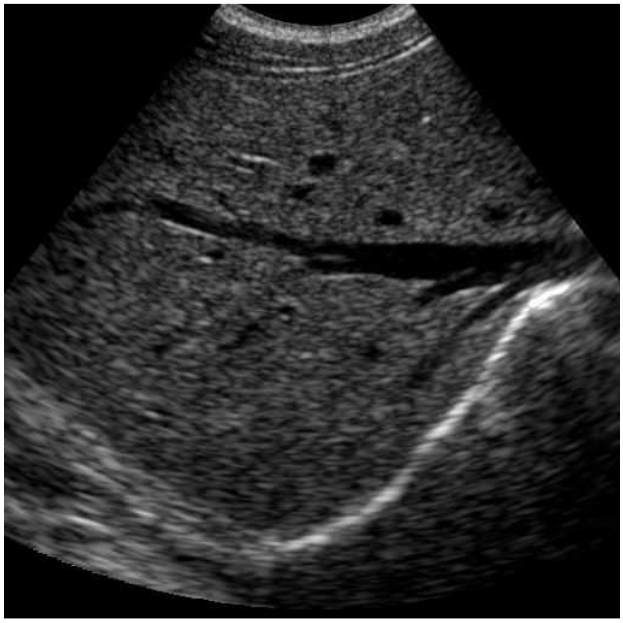


Fig. 3.1(a) Original ultrasound image (b) LCSRAD, 1 iteration, TTS=1.5 (c) LCSRAD, 4 iterations, TTS=1.5

Fig. 3.2(a) Original ultrasound image (b) LCSRAD, 4 iteration, TTS=1.5



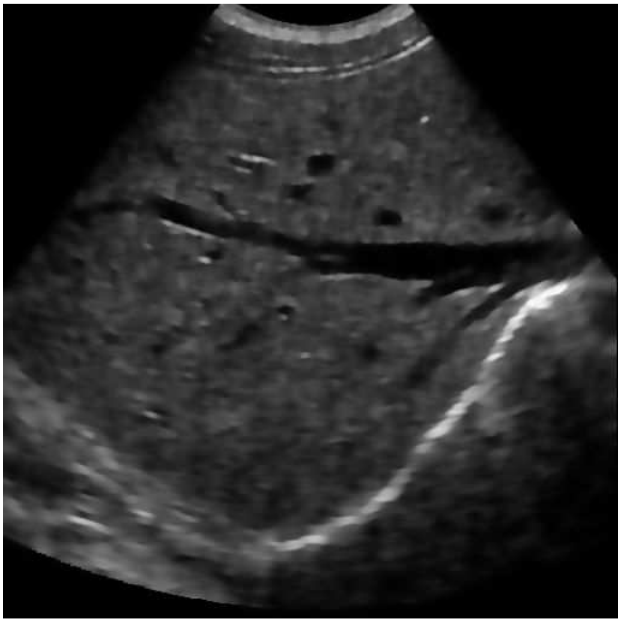


Fig. 3.3(a) Original ultrasound image (b) LCSRAD, 4 iteration, TTS=1.5

Fig. 3.4(a) Original ultrasound image (b) LCSRAD, 1 iteration, TTS=1.5 (c) LCSRAD, 4 iterations, TTS=1.5

4. CONCLUSIONS

Anisotropic diffusion is an edge-preserving smoothing algorithm that was successfully adapted to ultrasonic speckle reduction by using local statistics to determine the extent of blurring. However, numerical solution of the underlying PDE was very slow because of explicit discretization. Although there are several semi-implicit scheme based AD methods for ultrasound image speckle reduction, they are still not fast enough if we want to do real-time implementation or use them as fast preprocessing methods in other applications.

Moreover, in our presented GPU framework, the semi-implicit AOS scheme of nonlinear diffusion method can be used for real time ultrasound speckle reducing imaging. We use cyclic reduction method to solve tridiagonal linear system with specified memory storage/retrieve techniques which can be implemented on GPU. Our experiments show that the quality of speckle reduction by using advanced algorithms is the same as that implemented on CPU; while the GPU implementation offers increased frame rate by 10 times. Our framework is also flexible to be generalized to other similar nonlinear diffusion processes such as the NLAD [14].

ACKNOWLEDGMENT

We would like to thank Xiaohui Zuo of Saset Healthcare Inc. for obtaining the ultrasound images.

REFERENCES

[1] Yu, Y. and Acton, S. T., "Speckle reducing anisotropic diffusion," IEEE Trans. Image Processing, 11(11), 1260-1270(2002).

[2] Aja-Fernandez, S and Alberola-Lopez, C, "On the estimation of the coefficient of variation for anisotropic diffusion speckle filtering," IEEE Trans. Image Processing, 15(9), 2694-2701(2006).

[3] Krissian, K., Kikinis, R., Westin, C. F. and Vosburgh, K., "Speckle-constrained filtering of ultrasound images, " CVPR 2005, 2, 547-552(2005).

[4] Krissian, K., Westin, C. F., Kikinis, R. and Vosburgh, K. G., "Oriented speckle reducing anisotropic diffusion, " IEEE Trans. Image Processing, 16(5), 1412-1424(2007).

[5] Weichert, J., [Anisotropic Diffusion in Image Processing], BG Teubner, 278-290(1998).

[6] Weichert, J., Romeny, B. and Viergever, M. A. , "Efficient and reliable schemes for nonlinear diffusion filtering, " IEEE Trans. Image Processing, 7(3), 398-410(1998).

[7] Catte, F., Lions, P. L., Morel, J. -M., and Coll, T.,"Image selective smoothing and edge detection by nonlinear diffusion," SIAM Journal on Numerical Analysis, 29, 182-193(1992).

[8] Abd-Elmoniem, K. Z., Youssef, A. B. M. and Kadah, Y. M., "Real-time speckle reduction and coherence enhancement in ultrasound imaging via nonlinear anisotropic diffusion," IEEE Trans. Biomedical Engineering, 49(9), 997-1014(2002).

[9] Wang, Bo, Tan, Chaowei and Liu, Dong C., "Local Coherence based Fast Speckle Reducing Anisotropic Diffusion," Proceedings of 2008 International Pre-Olympic Congress on Computer Science, 304-309(2008).

[10] Wang, Bo and Liu, Dong C., "Semi-Implicit Scheme based Nonlinear Diffusion Method in Ultrasound Speckle Reduction," IEEE International Ultrasonics Symposium, Beijing, China, in press (2008).

[11] Owens, J. D., Luebke, D., Govindaraju, N., and etc., "A Survey of General-Purpose Computation on Graphics Hardware," Eurographics 2005, 21-51(2005).

[12] Kass, M., Lefohn, A. and Owens, J., "Interactive Depth of Field Using Simulated Diffusion on a GPU", Technical report, Pixar Animation Studios, 1-8(2006).

[13] Krüger, J., and Westermann R., "Linear Algebra Operators for GPU Implementation of Numerical Algorithms," ACM Transactions on Graphics. 22(3), 908-916(2003).

[14] Rumpf, M. and Strzodka, R., "Nonlinear Diffusion in Graphics Hardware," Proceedings of EG/IEEE TCVG Symposium on Visualization, 75-84(2001).