International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.3 (2016), pp.179-186
http://dx.doi.org/10.14257/ijmue.2016.11.3.17

A Fast Speckle Reduction Algorithm Based on GPU for Synthetic Aperture Sonar

Xu Kui[1], Zhong Heping[1] and Huang Pan [1]

[1] *Naval Institute of Underwater Acoustic Technology,*
Naval University of Engineering,
Wuhan 430033, China
E-mail: 57869588@qq.com;

Abstract

Synthetic aperture sonar (SAS) is a kind of high resolution imaging sonar, but speckle exists in SAS image for that SAS is a coherent imaging system, which makes it very difficult to visually and automatically interpret. In this paper, a fast speckle reduction algorithm for SAS is proposed in GPU environment, which helps to solve the speckle reduction problem of SAS image in real-time. Firstly, the original SAS image is partitioned into small rectangular blocks with partly overlapped and uploaded into the shared memory of GPU by block of threads. Secondly, the Lee filtering process is performed on every blocked SAS image by the multi-core of GPU simultaneously. Finally, the whole processed result is obtained by merging those local filtered images. The feasibility and high efficiency of the method are confirmed by the speckle reduction experiment on the real SAS image.

Keywords*: Synthetic aperture sonar; GPU; Speckle; Lee Filter*

1. Introduction

Synthetic aperture sonar (SAS) is a new high resolution underwater imaging technology. It uses a much smaller physical aperture which moves in a straight path at constant velocity to synthesize a large aperture to produce an image that has high azimuth resolution independent of range and wavelength. SAS is widely used in small underwater target search, channel surveying and mapping and so on. For that SAS imaging system is a coherent system, which makes speckle appear. Speckle seriously influences the quality of image, which makes the SAS image automatic interpretation become very difficult.

In order to improve the quality of SAS image, speckle reduction operation must be performed after SAS imaging. A good speckle reduction algorithm should well eliminate the coherent noise of SAS image and effectively keep the edges and detailed texture information. In addition, the efficiency of the speckle reduction algorithm should meet the requirements of SAS system in real time. The common used speckle reduction algorithms have local statistics adaptive method [4-7], the wavelet method and diffusion equation method [8]. During those algorithms, Lee filtering algorithm [4] is an effective method, which can effectively eliminate the spackle of SAS image. It disadvantage is its low efficiency, especially for processing large SAS image, which makes it difficult to use in practice. GPU technology provides a new method of real-time speckle reduction for SAS image using Lee filter [9].

This paper proposes a real-time speckle reduction method for SAS image in GPU environment. Firstly, the SAS image is loaded into the memory of GPU, then the whole image is divided into blocks with partly overlap according to the number of threads. Secondly, the GPU kernel function is called on the host, which uses multiple stream processors to process the blocked SAS image simultaneously, and then the final filtered SAS image is reconstructed for the blocked filtered SAS image and downloads to the

ISSN: 1975-0080 IJMUE

host. Finally, the efficiency of the proposed method is verified by speckle reduction experiment on real SAS image.

2. CUDA Programming Model

In CUDA programming model, the side of CPU is often called host, while the side of GPU is often called device. Computing tasks are completed by both CPU and GPU. CPU is good at logical and sequential computing work, while GPU mainly focuses on highly parallel data-processing work. CPU has its own memory and registers, and GPU also has its own memory and registers, but they are independent of each other. In CUDA program, the procedure called by the host executing on the GPU is called kernel function, which will be executed in parallel. If the parallel part of a program has been determined, the whole program can be divided into two parts. One part is several kernel functions which run on the device; the other is serial parts running on the host. Thus, a complete CUDA program is usually composed by the serial code executed on the CPU and a number of core functions executed on the GPU.

The parallel execution of kernel functions is realized by the CUDA threads, which is managed by the thread-block-grid three levels of structure. One, two or three thread block can be composed by lots of threads that perform the same task, and number of thread blocks can form grid with one or two dimension. There are two levels of parallel in a kernel function, which are the coarse parallel between the blocks in one grid and the fine parallel between the threads in one block. W distinguish from other thread hen the kernel function is executed, every thread executes the kernel function once, and which distinguishes from other thread by its index. One thread can communicate between threads with in the same block, while threads in different blocks cannot communicate, which can only communicate through the synchronization instructions. The thread hierarchy of CUDA programming model is shown in Figure.1.

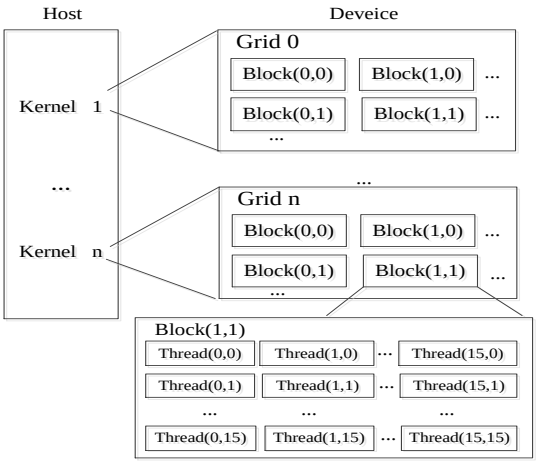


Figure 1. The Thread Hierarchy of CUDA Programming Model

3. Lee Filtering Algorithm

3.1 Basic Principle

Lee filtering is a typical speckle reduction algorithm for SAS image based on the local statistical characteristics, which is unnecessary to establish the precise statistical model. This algorithm is based on the multiplicative noise model of fully developed, and assuming that the speckle noise is white noise. During the process of processing, a local

area should be selected and assume that the prior mean and variance can be calculated the mean and variance of the local area.

Suppose the intensity image of SAS is denoted by Z. Z can also be expressed by $Z = XV$, in which X and Z stand for the actual scattering intensity and speckle intensity, and they are independent random variables. In local window, the mean of speckle intensity is equal 1, which means $\overline{V} = 1$. The mean of the intensity image $\overline{Z} = \overline{XV} = \overline{X} \cdot \overline{V} = \overline{X}$, which means the observed intensity is equal to the mean of speckle intensity. The variance of the observed intensity is derived as follows.

$$\begin{aligned} \mathrm{var}(Z) &= E[(Z - E(Z))^2] \\ &= E[(XV - E(XV))^2] \\ &= E[(XV)^2] - E^2(XV) \\ &= E(X^2)E(V^2) - E^2(X)E^2(V) \end{aligned}$$

Suppose the area of the local window is flat, then we can get $E(X^2) = \overline{X}^2$, and

$$\begin{aligned} \mathrm{var}(Z) &= E(X^2)E(V^2) - E^2(X)E^2(V) \\ &= \overline{X}^2(E(V^2) - E^2(V)) \\ &= \overline{X}^2\delta_V^2 \end{aligned}$$

Therefore, the variance of noise in the local window is

$$\delta_V^2 = \frac{\mathrm{var}(Z)}{\overline{X}^2} = \frac{\mathrm{var}(Z)}{\overline{Z}^2}$$

The actual scattering intensity is linear estimated using minimum mean-square error criterion under the multiplicative speckle model in space. Assuming the actual scattering intensity can be expressed by the observed intensity and the mean of which as follows,

$$\hat{\mathrm{X}} = a\overline{Z} + bZ$$

Where $\hat{\mathrm{X}}$ represents the estimation of actual scattering intensity. a、b are the coefficient to be determined, which value should make the minimum mean square error of $\hat{\mathrm{X}}$. By solving $J = E[(X - \hat{\mathrm{X}})^2]$, we can get

$$a = 1 - b = 1 - \frac{\mathrm{var}(X)}{\mathrm{var}(Z)}$$

$$b = \frac{E(X^2) - \overline{Z}^2}{E(X^2)E(V^2) - \overline{Z}^2} = \frac{\mathrm{var}(X)}{\mathrm{var}(Z)}$$

Then we can get the final expression of $\hat{\mathrm{X}}$ as follows,

$$\hat{\mathrm{X}} = \overline{Z} + \frac{\mathrm{var}(X)}{\mathrm{var}(Z)}(Z - \overline{Z})$$

3.2 Parallelization

Suppose the SAS image is denoted by $X(m,n)$, and $1 \le m \le M$, $1 \le n \le N$, which stand for the number of row and column of SAS image respectively. For any point (i, j) in the image, the estimated value $\hat{\mathrm{X}}(i, j)$ is based on the local image result with the centered (i, j) and the size of window $2W_x + 1$ and $2W_y + 1$ respectively. When the local window is 5×5 and the center point is $(7,7)$ in the SAS image, the sketch

map of Lee filtering is shown in Figure.2. Firstly, the local SAS image block is obtained according to the coordinate of the current point. Then the filtered value of the current point is estimated according to the formula of Lee filtering. Finally, the filtered result is written back to the result array. Lee filtering process is very simple, but its computation density is very high, especially when local window increases, the efficiency decreased significantly, which serious influence of SAS image automatic subsequent processing.

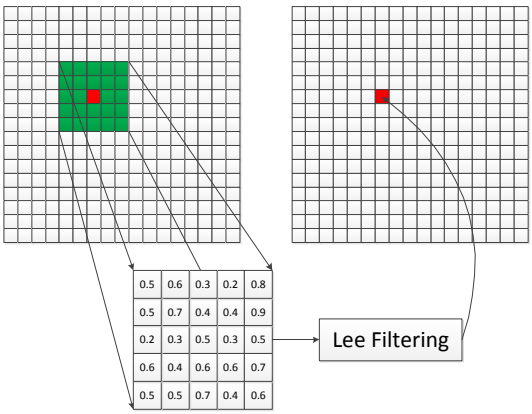


Figure 2. Sketch Map of the Lee Filtering By 5×5 Window Applied on Pixel of Coordinates(7,7)

GPU provides a new method for high efficient speckle reduction of SAS image based on Lee filtering algorithm. The flow chart of speckle reduction by Lee filtering in GPU environment is as follows:

(1) Allocate memory in GPU by calling cudaMalloc function according to the number of row and column of SAS image.

(2)Upload the SAS image from the memory of host to GPU by calling cudaMemcpy function.

(3)Start parallel computing function of Lee filter by calling the kernel function at the host. Firstly, every block thread begins to compute the local blocked image to be processed according to the index of the current thread block and the size of the local window. Then the local SAS image is uploaded to the shared memory by the threads of current block simultaneously and a synchronization operation is needed at last. Finally, every thread calculates a value of one point according to its local thread index, and then writes the result to the global memory.

(4)Download the result from the memory of GPU to host by calling cudaMemcpy function.

Suppose the size of local window is $(2W_x+1)\times(2W_y+1)$, and the size of thread block is set to 16×16 due to hardware limitations. Each thread computes one filter value. In order to accurately calculate the filtered value on the edge, we need to extend the boundary of the local blocked SAS image shown in Figure.3, which is determined by the size of local window. The width of extended boundary are W_x and W_y, which are the half size of local window, then the size of the local SAS image becomes $(16+2W_x)\times(16+2W_y)$. When the local SAS image is loaded into the shared memory, some threads need to load more than one data due to the expending. In order to keep the balance of loading data, the thread with the index $(i/16, j/16)$ is used to load the local data with the index (i, j), which can maximize the loading balance of different thread. In order to ensure the local SAS image is loaded before filtering, the sync threads function should be called.

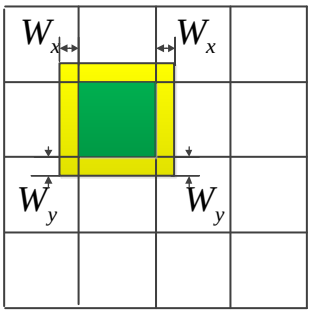


Figure 3. Sketch Map of Blocking

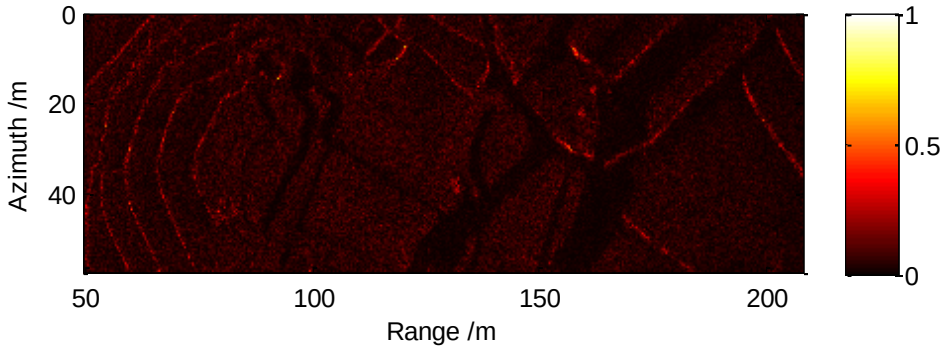
In addition, to ensure that estimation based on the calculated value, all local image data already contained in the shared memory required to load data after the sync threads function is called, contained in data synchronization.

4. Experiment

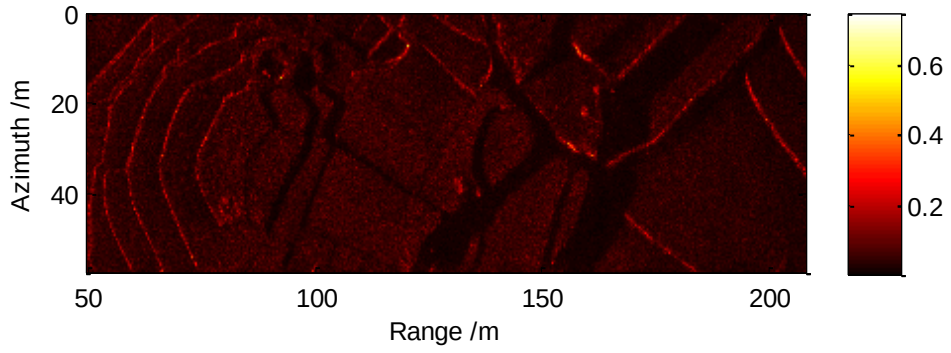
In this section, the performance of the fast speckle reduction algorithm in GPU environment is analyzed. The hardware configuration of the testing system is: CPU: Intel(R) Xeon(R) X5650 2.67G (2 GPU 12 cores), MEMORY: 48G, Display card: Tesla C2050, Operation System: Windows 7, Software Environment: Visual Studio 2008 and CUDA 5.0. The raw data sets have been acquired during the lake trial in July 2010 in QianDao Lake, Zhejiang Province of China by the Institute of Underwater Acoustics, Naval University of Engineering. The image result shown in Figure.4(a) is acquired by CS imaging algorithm, and the size of the image is 8800×2880. Before filtering, the image result has been normalized. It can be seen the terrain of this area is very rich, but the speckle is very seriously, which affects the subsequent image interpretation automatically. The parameter of the InSAS system is shown in Table. 1.

Table 1. System Parameter of the Trial

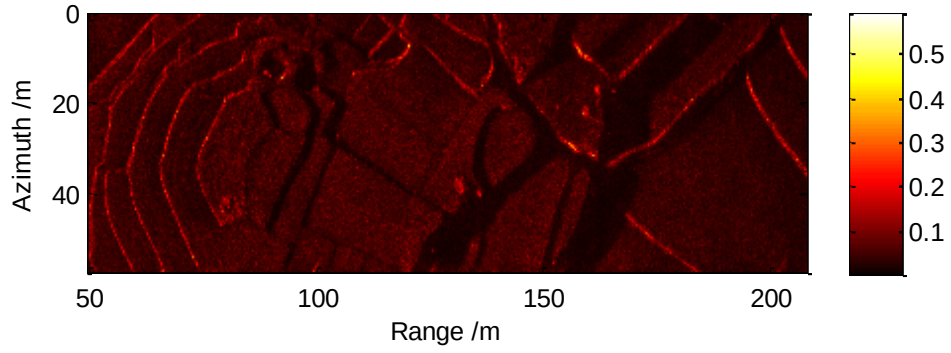
parameter	value	parameter	value
bandwidth (kHz)	20	pulse repetition period (ms)	320
carrier frequency (kHz)	150	length of sub-array (m)	0.04
pulse width (ms)	20	number of receiver	40
velocity (m/s)	1446	platform speed (m/s)	2.5
sampling rate (kHz)	40	range of sampling（m）	51-231



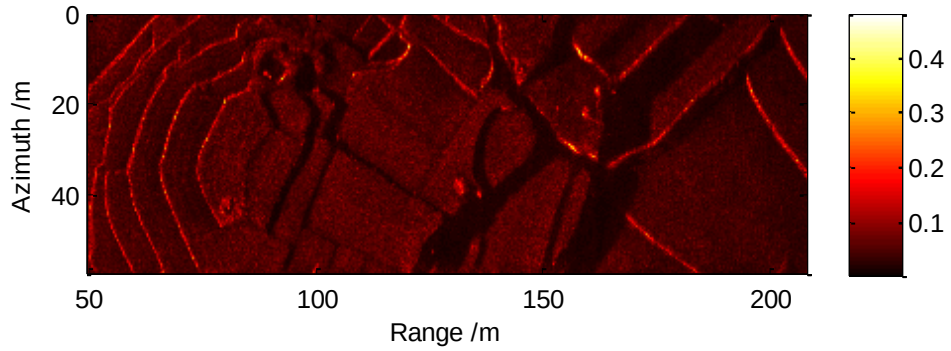
(a) Image Result of SAS



(b) Result by 3×3 Lee Filtering



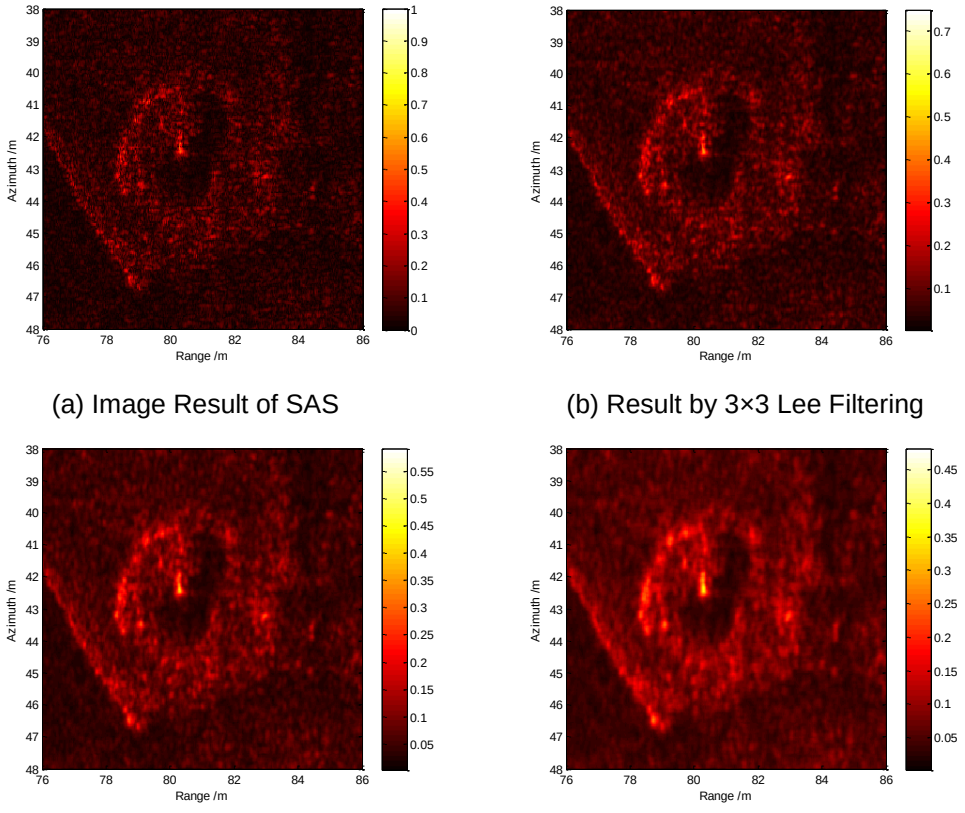
(c) Result by 5×5 Lee Filtering



(d) Result by 7×7 Lee Filtering

Figure 4. Experiment on the Real SAS Image Data

The filtered result by Lee filtering algorithm is shown in Figure.(b)-(d), which are processed with the size of window 3x3, 5x5 and 7x7 respectively. It can be seen the image radiometric resolution becomes higher with the local size of window increasing, and the speckle has been effectively suppressed. In order to compare the filtered results with different size of window, a local area image has been selected from the whole image, which starts from 76m to 86m in range and 38m to 48m in azimuth. The local image result and the filtered result corresponding to the Figure.4(a)-(d) are shown in Figure.5(a)-(d).



(a) Image Result of SAS

(b) Result by 3×3 Lee Filtering

(c) Result by 5×5 Lee Filtering

(d) Result by 7×7 Lee Filtering

Figure 5. Experiment on the Real Data (Local area)

The performance and efficiency comparison of the Lee filtering with different size of window on CPU and GPU is shown in Table.2. It can be seen the coefficient of noise and edge kept index are decreasing, while the radiometric resolution and the equivalent number of looks are increasing with the size of local window increasing. When the size of local window is 7×7, the filtered results shown in Figure.4(d) and Figure.5(d) are very good. The efficiency comparison is analyzed in the following under two different computing platforms. When GPU is used to reduce speckle, we should firstly allocate memory in GPU, then upload the data to GPU and download the filtered result to the memory of the host. When dealing with the test data, we needs 19.8 ms and 22.0 ms, respectively. When the size of local windows is 3×3, 5×5 and 7×7, the computing time processed by single CPU is 1654 ms, 2999 ms and 5134 ms, respectively. When processed by GPU, the kernel function only needs 16.5ms, 23.7ms and 34.9ms to run. The total filtering time by GPU is the sum of running time of kernel function, upload and download time, which are 58.3 ms, 65.5 ms and 76.7 ms, respectively. The speed up is increasing with the size of local window increasing which can up to 28, 45 and 66, respectively, which can meet the demand of real-time speckle reduction of SAS system.

Table 2. The Performance and Efficiency Comparison of the Lee Filtering with Different Size of Window

A	B	C	D	E	F	G	H
3×3	0.59	2.89	4.32	0.32	1654	16.5	28
5×5	0.54	3.46	4.56	0.18	2999	23.7	45
7×7	0.51	3.85	4.72	0.12	5134	34.9	66

A-size of window, B- coefficient of noise (standard deviation or mean value), C-

equivalent number of looks, D- radiometric resolution, E- edge kept index, F-computing time by CPU (ms), G- GPU kernel time (ms). H- speed up.

5. Conclusion

This paper presents a GPU-based speckle reduction method for SAS image, which can meet the demand of real-time processing. Lee filter is taken as a filtering algorithm to reduce the speckle of SAS image, which achieves a good result, but there exists the edge fuzzy problems inherent by Lee filter algorithm. When the Lee filter is optimized, this usually increases the calculation time and reduces the efficiency of the algorithm. Take the method of this paper, which only needs millisecond to reduce speckle of SAS image using GPU. This will break the bottle-neck of real-time speckle reduction for SAS image. Therefore, the next step is working on how to improve the precision of speckle reduction.

Acknowledgement

This work was supported by National Natural Science Foundation of China (Grant No.41304015).

References

[1] L. Jingnan, Y. Fanlin and Z, Jianhu, "Elementary Introduction to Synthetic Aperture Sonar and Interferometric Synthetic Aperture Sonar", Hydrographic Surveying and Charting.
[2] M. P. Hayes and P. T. Gough, "Synthetic aperture sonar: a review of current status", IEEE Journal of Oceanic Engineering, vol. 34, no. 3, **(2009)**, pp. 207-224.
[3] Z. X. Bo, T. J. Song and Z. H. Ping, "Chirp Scaling Imaging Algorithm for Synthetic Aperture Sonar Based on Data Fusion of Multi-receiver", Journal of Harbin Engineering University, vol. 34, no. 2, **(2013)**, pp. 1-6.
[4] J. S. Lee, "Speckle suppression and analysis for synthetic aperture radar images", Optical Engineering, vol. 25, no. 5, **(1986)**, pp. 636-642.
[5] D. T. Kuan, A. A. Sawchuk and T. C. Strand, "Adaptive noise smoothing filter for images with signal-dependent noise", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 7, no. 2, **(1985)**, pp. 793-802.
[6] V. S. Frost, J. A. Stiles and K. S. Shanmugan, "A model for radar images and its application to adaptive digital filtering of multiplicative noise", IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI, vol. 4, **(1982)**, pp. 157–166.
[7] M. Bhuiyanr, M. Ahmad and M. Swamy, "Spatially adaptive wavelet-based method using the cauchy prior for denoising the SAR images", IEEE Transactions on Geoscience and Remote Sensing, vol. 17, no. 4, **(2007)**, pp. 500-507.
[8] Y. Yu and S. Acton, "Speckle reducing anisotropic diffusion", IEEE Transactions on Image Processing, vol. 11, no. 11, **(2002)**, pp. 1260-1270.
[9] X. Han, Z. Qinglei and Z. Zuxun, "Parallel Algorithm of Harris Corner Detection Based on Multi-GPU", Geomatics and Information Science of Wuhan University, vol. 37, no. 7, **(2012)**, pp. 876-881.

Author

Kui Xu, Lecturer, received the B.Eng. degree from the Radar Engineering, Naval University of Engineering, Wuhan, China, in 2002 and the M. Eng. degree from the Signal and Information Processing, Nankai University, Tianjin, China in 2008. He has published more than 10 technical journal papers and international conference papers. His research interests include underwater acoustical signal processing.

