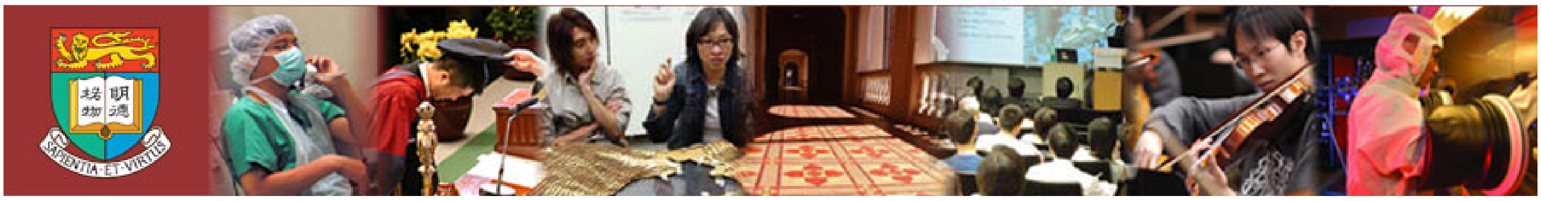
View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by HKU Scholars Hub

The HKU Scholars Hub The University of Hong Kong 香港大學學術庫



Title	Real-time GPU-based adaptive beamformer for high quality ultrasound imaging
Author(s)	Chen, J; Yiu, BYS; So, HKH; Yu, ACH
Citation	The 2011 IEEE International Ultrasonics Symposium (IUS), Orlando, FL., 18-21 October 2011. In IEEE International Ultrasonics Symposium Proceedings, 2011, p. 474-477
Issued Date	2011
URL	http://hdl.handle.net/10722/140228
Rights	IEEE International Ultrasonics Symposium Proceedings. Copyright © IEEE.

10.1109/ULTSYM.2011.0114

Real-Time GPU-Based Adaptive Beamformer for High Quality Ultrasound Imaging

Junying Chen*, Billy Y. S. Yiu†, Hayden K.-H. So* and Alfred C. H. Yu†

*Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam, Hong Kong SAR
†Medical Engineering Program, The University of Hong Kong, Pokfulam, Hong Kong SAR

Correspondance Emails: alfred.yu@hku.hk, hso@eee.hku.hk

***Abstract*—A real-time adaptive minimum variance (MV) beamformer realized using graphics processing units (GPUs) is presented. MV adaptive beamforming technique is attractive as it is capable of producing high quality images with narrow mainlobe width and low sidelobe level. However, because of its substantially higher computational requirements, realizing MV in real-time has been prohibitively difficult. Recent advancements in commodity GPUs have made very high performance computing possible at very affordable price. Using a commercial off-the-shelf GPU, an MV beamformer achieving real-time performance has been realized. Tradeoffs between computational throughput and image quality have been studied. Careful selection of algorithm parameters, including receive aperture and sub-aperture size, was demonstrated to be imperative for achieving real-time performance without sacrificing image qualities.**

***Index Terms*—adaptive beamforming, graphics processing units, parallel processing, real-time realization.**

I. Introduction

Beamforming is a crucial stage in medical ultrasound imaging which determines the output image quality. Different beamforming methods have been investigated in ultrasound research community for many years, among which, adaptive beamforming has gained concerns by ultrasound researchers in recent years [1]–[5] because of the high image quality provided.

Most adaptive beamforming techniques are based on the minimum variance (MV) method developed by Capon [6]. Comparing with conventional delay-and-sum (DAS) beamforming, which uses a set of predetermined weights, MV beamforming uses signal-dependent weights. MV beamforming could establish high resolution image quality, but is inherently sensitive to estimation errors of wavefield parameters [1]. Therefore, increasing the robustness of MV beamforming is necessary to maintain the high image quality [2], [3].

The amplitude estimate is calculated with apodization weighting computed based on the signal statistics of received echo signals. The calculation, which involves covariance matrix computation and inverse operation, is tremendous computational demanding. Therefore, real-time MV beamformer is very difficult to realize.

As a hardware accelerator, graphics processing units (GPUs) have gained great successes in different ultrasound imaging applications [7], [8]. The successes mainly contributed by its parallel architecture, which allows tasks to be partitioned and executed in parallel. In this study, GPUs are adopted to realize real-time MV beamforming. It is our intent to foster MV beamforming into a practically feasible algorithm and in turn enhance the image quality achievable in ultrasound imaging. In the following sections, we will discuss various computational strategies that can accelerate the overall process of MV beamforming and point out how GPUs are well-suited for this purpose. We will also present a performance-quality tradeoff analysis to provide insights on the practical efficacy of MV beamforming from both computational and image quality standpoints.

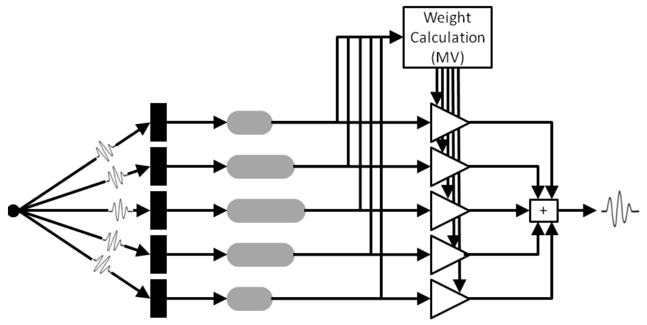


Fig. 1. Beamforming process. The reflected echo signals are received by transducer elements. They are delayed and then apodized by applied weights, and finally summed up to form an amplitude estimate.

II. Background: MV Beamforming

Fig. 1 illustrates the MV beamforming process. In general, for a MV beamformer with M receive channels, the adaptive apodization weights w(p) of each pixel located at p can be computed through the following equation:

$$\mathbf{w}(p) = \frac{\mathbf{R}^{-1}(p)\mathbf{a}}{\mathbf{a}^H\mathbf{R}^{-1}(p)\mathbf{a}}, \tag{1}$$

where $\mathbf{R}^{-1}(p)$ is the inverse of the covariance matrix for focus-delayed samples and $\mathbf{a}$ is simply a vector of ones because the channel data being processed are already delayed. The covariance matrix $\mathbf{R}(p)$ is usually estimated as:

$$\mathbf{R}(p) = \frac{1}{M-L+1} \sum_{k=0}^{M-L} \mathbf{x}_k(p)\mathbf{x}_k^H(p), \tag{2}$$

where $\mathbf{x}_k(p)$ is a $L \times 1$ vector of delayed echo samples in the k^{th} sub-aperture, which starts from k^{th} channel to $(k+L-1)^{th}$ channel. Furthermore, to ensure that the covariance matrix is invertible, the sub-aperture size is limited to $L \leq \frac{M}{2}$. Subsequently, the amplitude estimate $z(p)$ of the pixel p is obtained by averaging the weighted sum of focus-delayed echo signals from all sub-aperture channels.

$$z(p) = \frac{1}{M-L+1} \sum_{k=0}^{M-L} \mathbf{w}^H(p)\mathbf{x}_k(p). \tag{3}$$

MV beamformer achieves high resolution through only allowing reflections from receive focal point to pass through the beamformer with unity gain while others are suppressed. Yet, it is sensitive to wavefield parameters. For example, inaccurate delay calculation due to phase aberration could lead to underestimation of the amplitude estimates. By increasing robustness of the beamformer, the suppression of slight out of focus reflection can be tuned.

To improve robustness, one could reduce the sub-aperture length L or adding a constant to the diagonal elements of R(p). It should be noted that increasing the robustness of the beamformer would degrade the resolution of the output image. Therefore, a tradeoff has to be made.

III. Real-Time MV Beamformer Development

We developed a real-time MV adaptive beamformer utilizing GPU parallel compute architecture. In addition to parallelizing the computing steps of MV beamforming algorithm into GPU, we investigated the characteristics of MV beamforming algorithm and sought out three approaches to diminish the number of mathematical operations to carry out the MV beamforming.

A. Reduction of Mathematical Operations

We analyzed MV beamforming algorithm to avoid unnecessary operations and cut down redundant operations.

1) Robustness Improvement Approach Selection: As discussed in Section II, two methods are available to improve MV beamforming robustness. If the first method, spatial smoothing, is applied, the size of the sub-aperture L is reduced, given a specific M. The computational complexity of covariance matrix $\mathbf{R}$ calculation is $O(L^3)$, because $(M-L+1)$ times of $L \times L$ matrix element calculations are executed to find the value of $\mathbf{R}$. Consequently, the computation operations are reduced. If the other method, diagonal loading, is used, the beamformer needs to execute additional steps to obtain a suitable constant value and add this calculated value to the diagonal of $\mathbf{R}$, while the value of L is not reduced. Obviously, spatial smoothing has the computation advantage over diagonal loading.

2) Symmetry of Covariance Matrix: Rewrite (2) as:

$$\mathbf{R} = \frac{1}{M-L+1}\mathbf{X}\mathbf{X}^H, \tag{4}$$

where $\mathbf{X} = [\mathbf{x}_0\ \mathbf{x}_1\ \mathbf{x}_2\ ...\ \mathbf{x}_{M-L}]$, which is the assemble of all focus-delayed sample vectors.

As the beamforming data we are handling are real numbers, $\mathbf{X}^H = \mathbf{X}^T$. Hence, $\mathbf{R}$ exhibits a valuable feature that accelerates the beamforming process, symmetry. The explanation is as follows:

$$(\mathbf{X}\mathbf{X}^H)^T = (\mathbf{X}\mathbf{X}^T)^T = (\mathbf{X}^T)^T\mathbf{X}^T = \mathbf{X}\mathbf{X}^T, \tag{5}$$

so that

$$\mathbf{R}^T = \mathbf{R}. \tag{6}$$

Because of the symmetry, the MV beamformer can save computation time by calculating upper or lower triangular matrix instead of calculating the whole matrix.

3) Computing $\boldsymbol{R}^{-1}\boldsymbol{a}$, not $\boldsymbol{R}^{-1}$: Referring to (1), the MV beamformer may calculate $\mathbf{R}^{-1}$ first, and then $\mathbf{R}^{-1}\mathbf{a}$ and $\mathbf{a}^H\mathbf{R}^{-1}\mathbf{a}$. However, matrix inverse calculation is a burden to computing. Instead, we calculate $\mathbf{R}^{-1}\mathbf{a}$ to find the weight vector. The calculation of $\mathbf{R}^{-1}\mathbf{a}$ is to solve the following equation to find the value of **b**:

$$\mathbf{R}\mathbf{b} = \mathbf{a}. \tag{7}$$

The solution is obtained by performing row and column operations according to Gaussian-Jordan elimination on the augmented matrix $[\mathbf{R}|\mathbf{a}]$. The result of Gaussian-Jordan elimination is $[\mathbf{I}|\mathbf{b}]$, where **I** is the identity matrix and **b** is the exact solution. Subsequently , $\mathbf{a}^H\mathbf{R}^{-1}\mathbf{a}$ is found easily by summing all the elements in vector **b**.

The calculation of $\mathbf{R}^{-1}$ is to solve another equation as follows:

$$\mathbf{R}\mathbf{Y} = \mathbf{I}. \tag{8}$$

The equation is solved also by Gaussian-Jordan elimination on $[\mathbf{R}|\mathbf{I}]$. The result is $[\mathbf{I}|\mathbf{Y}]$, where $\mathbf{Y} = \mathbf{R}^{-1}$.

The advantages of calculating $\mathbf{R}^{-1}\mathbf{a}$ instead of $\mathbf{R}^{-1}$ is obvious. It saves row and column operations, because $[\mathbf{R}|\mathbf{a}]$ has $(L-1)$ columns less than $[\mathbf{R}|\mathbf{I}]$.

B. Computation Parallelization in GPU

1) Platform and Scenario Setup: The MV adaptive beamformer was implemented using a high-performance and relatively low cost GPU card, NVIDIA GeForce GTX 480. The host computing desktop platform consisted of a 2.4 GHz Intel Core 2 Quad Q6600 CPU and 4 GB DDR2 RAM, with Ubuntu 10.04 environment. The GTX 480 was the first commercially available GPU that realized the Fermi architecture [9]. It has a total of 480 execution cores available during run time. NVIDIA CUDA toolkit version 3.2 was used for program compilation.

We simulated a phantom of three 4-mm-diameter cysts aligned to the vertical center line of the phantom using Field

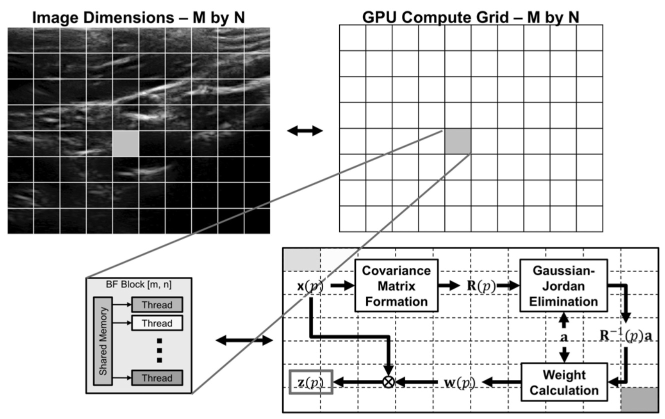


Image Dimensions – M by N
GPU Compute Grid – M by N
BF Block [m, n]
Shared Memory
Thread
Thread
Thread
$\mathbf{x}(p)$
Covariance Matrix Formation
$\mathbf{R}(p)$
Gaussian-Jordan Elimination
$\mathbf{a}$
$\mathbf{R}^{-1}(p)\mathbf{a}$
Weight Calculation
$\mathbf{w}(p)$
$z(p)$

Fig. 2. Operation parallelization inside GPU. The operations calculating one pixel amplitude estimate were conducted by one compute block. Within the compute block, the beamforming steps were further partitioned into parallel processes computed by parallel compute threads.

II simulator [10]. A 128-element linear array transducer with 5 MHz center frequency was used to transmit and receive ultrasound signals, focusing at 30-mm depth in transmissions. The returned echoes were sampled by an analog-to-digital converter (ADC) with 50 MHz sampling rate. The simulated raw data were fed into GPU processor from the host computer for beamforming. The post-beamform data were transmitted back to the host computer for final image display.

2) Execution Parallelization: The computation steps of the algorithm were first parallelized into the outer structure of the GPU parallel compute architecture, which was a grid of concurrent compute blocks. Within the compute blocks, the computation steps were further partitioned into the inner structure of GPU architecture, which was a set of independent compute threads running simultaneously. The overview of the operation parallelization in GPU is shown in Figure 2.

The beamforming process for each pixel is identical to any other pixels, hence the operations calculating one pixel amplitude estimate were conducted by one compute block. The total number of compute blocks activated was equivalent to the total number of pixels to be processed. All the compute blocks collectively form the GPU compute grid.

Within the compute block, the beamforming steps were further partitioned into parallel processes. The calculation of **R** is a good example to explain the parallelization in GPU. The calculation of one element of R_{ij} is derived from (2) as:

$$R_{ij} = \frac{1}{M-L+1} \sum_{k=0}^{M-L} x_{(i+n)} x_{(j+n)}, \tag{9}$$

where i and j are row index and column index of matrix **R**.

As indicated in (9), the calculation process of R_{ij} has no dependency in other elements' calculations. The process is self-contained so that it is well-suited to be completed by one compute thread. Because **R** contains $L \times L$ elements, $L \times L$ compute threads were used in our MV beamformer design for **R** calculation process.

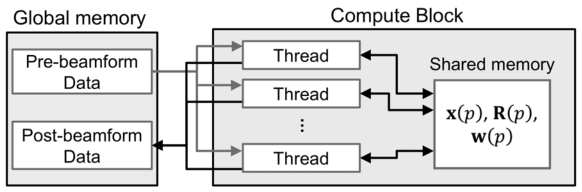


Fig. 3. GPU memory assignment.The utilization of the slowest global memory was restricted to store input and output data of beamforming process. The faster shared memory was used as storage of intermediate results during the beamforming process. The fastest register files were assigned to hold the temporary results within beamforming steps.

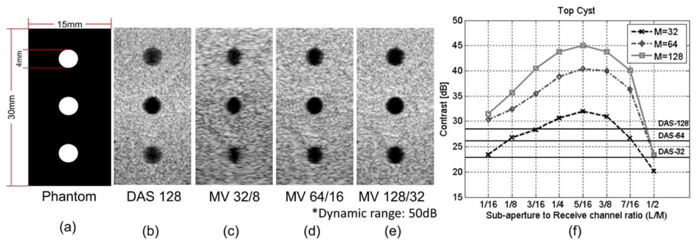


Fig. 4. Image contrast changing with M and L. (a) is the simulated phantom which can be set as the ideal image. From (b) to (d), when M increases, better contrast is shown.

3) GPU Memory Assignment: Figure 3 demonstrates the memory assignment inside the GPU. The memory access rate decreases from on-chip register files to on-chip shard memory, and then to the off-chip global memory.

IV. Tradeoff between Throughput and Quality

Not only the throughput of the MV beamformer was concerned, but also its image quality. We built a MV beamformer with real-time video frame rate but not sacrificing the image quality.

The throughput is measured by processing frame rate. Human visual system is capable to process 10 to 12 separate images in one second. During the maintenance period of one image in human visual cortex, if another image is perceived by human vision, an illusion of image continuity is incurred, making an impression of smooth motion of a sequence of still images. That is to say, when the beamformer processing frame rate is higher than 12 frames per second, a smooth image video is formed, meaning that the beamformer achieves real-time video frame rate.

Image quality is analyzed from two aspects of contrast and lateral resolution. Contrast is measured as the ratio of the mean amplitude estimate from the speckle region and that inside the cyst region at the same imaging depth. Lateral resolution is measured as the full width at the half maximum of the mainlobe in the point spread function at the transmit focal depth.

A. Relationship between M, L, and Image Quality

Figure 4 demonstrates the contrast comparison of output images with different M and L. As seen from the contrast value curves in Figure 4 (f), larger M exhibits better contrast,

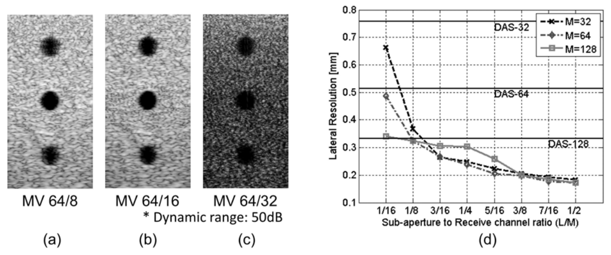


Fig. 5. Image lateral resolution varying with M and L. While M is fixed, larger L leads to higher lateral resolution of the output images.

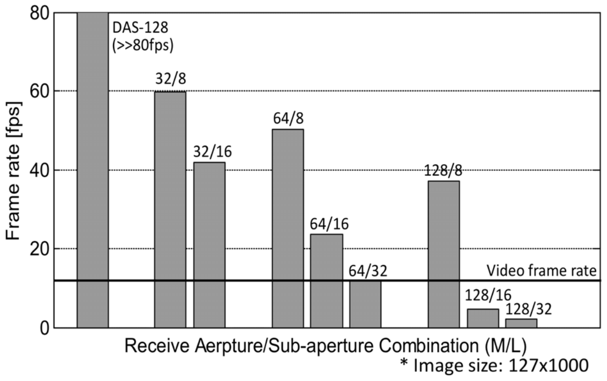


Fig. 6. Beamforming throughput.High imaging throughput is achieved if smaller M and L is chosen. This is because the computation operations are reduced subsequently.

which is confirmed in the output images shown in Figure 4 (b) to (d).

Figure 5 illustrates the lateral resolution of MV beamforming method for various M and L. Additional information observed from Figure 5 (a) to (c), although bigger $\frac{L}{M}$ provides higher lateral resolution, but the contrast is worse, which is also indicated in Figure 4 (f). Furthermore, when robustness of the beamformer is considered, smaller $\frac{L}{M}$ is expected.

B. Tradeoff between Beamforming Throughput and Image Quality

Figure 6 presents the processing frame rates of MV beamformer with different M and L. The frame rate has a relationship with the number of total pixels in the output image. As the amplitude estimate calculation of each pixel is identical, if the number of pixels in one image is changed to N, the output frame rate is then to be $\frac{N}{127\times1000} \times FrameRate_{127\times1000}$. Furthermore, since the amplitude estimate calculation of each pixel is independent with other pixels, the MV beamforming algorithm is easily scaled from single-GPU solution to multi-GPU solution. If K identical GPUs are used, the throughput is to be $K \times FrameRate_{singleGPU}$.

As discussed above, the image quality and the beamforming throughput vary with the value of M and L. Both for computation and imaging quality considerations, M is better to be a medium value in $[1, 128]$ and $\frac{L}{M}$ is better to be a medium value in $(0, \frac{1}{2}]$. As proved in our experiments, $M = 64$ and $L = 16$ provides a real-time MV beamformer with high image quality.

V. Conclusion

The implementation of real-time MV beamformer with video frame rate has not been shown in the ultrasound research community. This has prompted us to pursue such an effort. Our prototype implementation was able to achieve video frame rate performance without sacrificing the image quality. In order to obtain high quality ultrasound images with fast processing, the tradeoff between beamforming throughput and the image quality has been evaluated. It was determined that MV beamforming with $M = 64$ and $\frac{L}{M} = \frac{1}{4}$ seems to be a suitable tradeoff for achieving high throughput and high image quality.

Acknowledgements

This work is supported in part by Hong Kong Innovation and Technology Fund (ITS/492/09, InP/211/10), and in part by the Research Grants Council of Hong Kong, project GRF 716510.

References

[1] J.-F. Synnevag, A. Austeng, and S. Holm, "Adaptive beamforming applied to medical ultrasound imaging," *IEEE Trans. Ultrason. Ferroelec. Freq. Contr.*, vol. 54, no. 8, pp. 1606–1613, Aug. 2007.
[2] J.-F. Synnevag, A. Austeng, and S. Holm, "Benefits of minimum-variance beamforming in medical ultrasound imaging," *IEEE Trans. Ultrason. Ferroelec. Freq. Contr.*, vol. 56, no. 9, pp. 1868–1879, Sep. 2009.
[3] Z. Wang, J. Li, and R. Wu, "Time-delay- and time-reversal-based robust capon beamformers for ultrasound imaging," *IEEE Trans. Med. Imaging*, vol. 24, no. 10, p. 13081322, Oct. 2005.
[4] I. K. Holfort, F. Gran, and J. A. Jensen, "Broadband minimum variance beamforming for ultrasound imaging," *IEEE Trans. Ultrason. Ferroelec. Freq. Contr.*, vol. 56, no. 2, pp. 314–325, Feb. 2009.
[5] B. Asl and A. Mahloojifar, "Contrast enhancement and robustness improvement of adaptive ultrasound imaging using forward-backward minimum variance beamforming," *IEEE Trans. Ultrason. Ferroelec. Freq. Contr.*, vol. 58, no. 4, pp. 858–867, Apr. 2011.
[6] J. Capon, "High resolution frequency-wavenumber spectrum analysis," *Proc. IEEE*, vol. 57, no. 8, p. 14081418, Aug. 1969.
[7] S. Rosenzweig, M. Palmeri, and K. Nightingale, "Gpu-based real-time small displacement estimation with ultrasound," *IEEE Trans. Ultrason. Ferroelec. Freq. Contr.*, vol. 58, no. 2, pp. 399–405, Feb. 2011.
[8] B. Y. S. Yiu, I. K. H. Tsang, and A. C. H. Yu, "GPU-based beamformer: Fast realization of plane wave compounding and synthetic aperture imaging," *IEEE Trans. Ultrason. Ferroelec. Freq. Contr.*, vol. 58, no. 8, pp. 1698–1705, Aug. 2011.
[9] NVIDIA Co., *NVIDIA's Next Generation CUDA Compute Architecture: Fermi*. Santa Clara, USA: NVIDIA, 2010.
[10] J. A. Jensen and N. B. Svendsen, "Calculation of pressure fields from arbitrarily shaped, apodized, and excited ultrasound transducers," *IEEE Trans. Ultrason. Ferroelec. Freq. Contr.*, vol. 39, no. 2, pp. 262–267, Mar. 1992.