Ultrasonic Imaging using Resolution Enhancement Compression and GPU-Accelerated Synthetic Aperture Techniques

Anthony Podkowa, Bradley University

***Abstract*—This study presents the preliminary results of an investigation in combining resolution enhancement compression (REC) and generic synthetic aperture ultrasound (GSAU) techniques to improve ultrasound image quality. REC is a coded excitation and pulse compression technique which has been shown to improve the axial resolution and signal to noise ratio (SNR) of ultrasound images. By exploiting the convolution equivalence principle, the characteristic response of the ultrasonic transducer may be artificially exchanged for a desired response with more favorable properties. To improve the lateral resolution and SNR, GSAU has proven effective. GSAU utilizes the location of the ultrasonic transducer elements to artificially focus ultrasound images at each pixel. Using MATLAB (Mathworks Inc., Natick, MA) in conjunction with Field II, software has been implemented to simulate the REC and GSAU techniques. Due to the massive amounts of computations associated with the GSAU technique, a general purpose graphics processing unit was used to reduce computation time by a factor of 116, down to a fraction of a second, thus allowing for the processing of images in real-time. Improving the resolution and SNR should allow a physician to observe smaller lesions in the tissue, potentially resulting in earlier detection of cancer.**

I. Introduction

ULTRASONIC image quality can be characterized by two common imaging metrics: spatial resolution and signal-to-noise ratio (SNR). Resolution refers to the extent an imaging system is able to disguish between details within the image. In the case of 2D images, resolution has two distinct components: axial resolution, which is parallel to the beam axis, and lateral resolution, which is perpendicular to the beam axis. SNR is a measure of the ratio of the power of a desired signal to corrupting noise.

Recent developments in ultrasonic imaging research have shown that resolution enhancement compression (REC) may be utilized to improve the axial resolution and SNR of ultrasound images [1, 2]. REC is a coded excitation and pulse compression technique that is based on the convolution equivalence principle, which states that a desired output may be represented as an infinite number of different convolutional pairs. By exploiting the convolution equivalence principle, it is possible to make a fixed system with a certain response behave as though it had a desired response with enhanced spectral properties.

To improve lateral resolution as well as SNR, a collection of synthetic aperture (SA) techniques may be used [3]. The simplest of these is the generic synthetic aperture ultrasound technique (GSAU). One drawback of SA techniques in general is the large amount of computing time required for processing the images. Due to the massively parallelizable nature of the GSAU algorithm, a general purpose graphics processing unit (GPGPU) may be used to overcome this problem. Conventional central processing units typically only have up to eight cores, meaning that only a small fraction of the pixels may be computed concurrently. Graphics processing units (GPUs) may have several hundreds of cores that may be used for parallel compution. GPGPU processing makes it feasible to utilize SA techniques such as the GSAU technique in real-time applications.

Based on knowledge of linear time-invariant system theory, it was hypothesized that combining REC with the GSAU technique would result in an overall improvement in spatial resolution and exhibit further increases in SNR than when taken individually. To test this hypothesis, a number of simulations were processed with the REC and GSAU techniques using Mathworks MATLAB (Mathworks Inc., Natick, MA) and Field II [4, 5]. To increase the speed of the GSAU processing, software was written that utilized the computational capabilities of MATLAB's Parallel Computing Toolbox and NVIDIA's CUDA C.

II. Methods

The complete system block diagram is shown in Fig. 1. Each subsystem will be analyzed in the following sections.

A. REC

Using the REC technique, it is possible to effectively replace the transducer's pulse-echo impulse response, $h_t(t)$, with a desired impulse response, $h_d(t)$. This desired impulse response is engineered to have more desirable properties, such as an increased bandwidth, which translates into an improvement in axial resolution [1, 2]. The driving idea behind REC is the convolution equivalence principle, which is illustrated in Fig. 2. By exciting the transducer with a pre-enhanced chirp, $v_{pc}(t)$, the resulting ultrasonic signal will have a response that is equivalent to a linear chirp, $v_{lc}(t)$, convolved with a desired impulse response. The block diagram for the REC subsystem is given in Fig. 3.

The primary task involved with the REC subsystem is the generation of the pre-enhanced chirp. The REC system takes an unwindowed linear chirp, $v_{ulc}(t)$, as an input. This chirp is passed through a Wiener filter which has a frequency response given by

$$G_1(f) = \frac{H_t^*(f)H_d(f)}{|H_t(f)|^2 + |H_t(f)|^{-2}}, \tag{1}$$

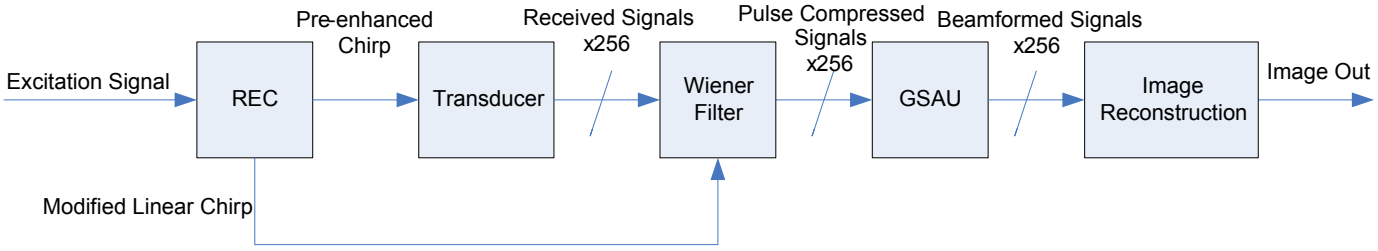

Fig. 1. Top Level System Block Diagram

Fig. 2. Illustration of convolution equivalence principle

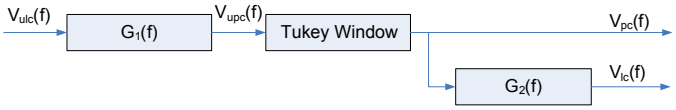

Fig. 3. REC Subsystem

where $H_d(f) = \mathscr{F}\{h_d(t)\}$, $H_t(f) = \mathscr{F}\{h_t(t)\}$, and $\mathscr{F}\{\cdot\}$ denotes the discrete Fourier transform. The expression shown in (1) is an approximation for the ideal inverse filter given by $G_1'(f) = H_d(f)/H_t(f)$, which in general cannot be used due to the potential for instability. This filtering operation will generate an unwindowed pre-enhanced chirp with spectrum denoted by $V_{upc}(f)$. The pre-enhanced chirp will then be windowed in the time domain by a Tukey-cosine window, $w(t)$, such that $v_{pc}(t)$ will be given by

$$v_{pc}(t) = v_{upc}(t)w(t). \tag{2}$$

The expression for a Tukey cosine window [1] of width T and rolloff factor α is given by

$$w(t) = \begin{cases} 1 & 0 \le |t| \le \frac{T}{2}(1+\alpha) \\ 0.5\left[\cos\left(\pi \frac{t-\frac{T}{2}(1+\alpha)}{T(1-\alpha)}\right)\right] & \frac{T}{2}(1+\alpha) \le |t| \le T. \end{cases} \tag{3}$$

However, because the Tukey window modifies the pre-enhanced chirp, convolution equivalence with $v_{ulc}(t)$ no longer holds true. Therefore, a modified linear chirp with frequency response $V_{lc}(f)$ must be generated from the pre-enhanced chirp by passing it through an inverse filter which has a frequency response given by

$$G_2(f) = \frac{H_t(f)}{H_d(f)}. \tag{4}$$

B. Transducer

The ultrasonic transducer is the component which converts electrical signals to ultrasonic pulses. For this project, a linear array transducer will be simulated. A linear array transducer is composed of several ultrasonic elements which may transmit and receive independently of one another, a diagram of which is depicted in Fig. 4. The transducer specifications are given in Table I.

TABLE I
TRANSDUCER SPECS

Center frequency:	8 MHz
Sampling frequency:	200 Mhz
Element height:	4 mm
Element width:	260 μm
Element kerf:	40 μm
Number of elements:	256
Focus:	20 mm

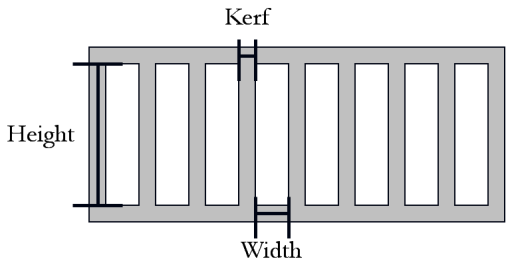

Fig. 4. Transducer Diagram

C. Wiener Filter

First, the received signal is passed through a Wiener filter to remove the spectrum of the linear chirp. The Wiener filter will be characterized by the transfer function in (5) [1].

$$\beta_{REC}(f) = \frac{V_{lc}^*(f)}{|V_{lc}(f)|^2 + \gamma \overline{eSNR}^{-1}(f)} \tag{5}$$

The γ factor is a parameter that can be tuned to fit the appropriate response, but was usually set to 1 in [1]. The $\overline{eSNR}(f)$ term describes the average echo SNR per frequency channel, which is equivalent to

$$\overline{eSNR}(f) = \frac{\text{PSD}_{sig}(f)}{\text{PSD}_{noise}(f)}, \tag{6}$$

where $\text{PSD}_{sig}(f)$ and $\text{PSD}_{noise}(f)$ are the power spectral densities of the signal and the additive noise respectively [1]. More information about the Wiener filter chosen can be found in [1, 2].

D. GSAU

With the GSAU technique, each ultrasonic transducer element is utilized one at a time for both the transmit and receive operations. This results in N_{xdc} transmit events, where N_{xdc} is the number of elements in the transducer. Modeling the tissue

sample as a collection of point scatterers, the received signal of the i^{th} element is given by

$$r_i(t) = \sum_p \sigma_p g(t - \tau_i) \tag{7}$$

where σ_p is the back-scattering coefficient of the tissue at the point $\vec{r}_p = (x_p, z_p)$, and $g(t)$ is the excitation signal. The delay τ_i is given by

$$\tau_i = 2\frac{|\vec{x}_i - \vec{r}_p|}{c}. \tag{8}$$

where x_i is the position of the transducer element and c is the speed of sound in tissue (approximately 1,540 m/s). The factor of 2 present in (8) is due to the fact that the pulse travels the same distance twice round trip. By compensating for these delays in each of the received signals and summing them together, it is possible to reconstruct the image at each point $\vec{r}_p$ as shown in (9).

$$\hat{f}_m(\vec{r}_p) = \sum_i r_i(\tau_i(\vec{x}_i, \vec{r}_p)) \tag{9}$$

This technique is depicted in Fig. 5. Because the system is

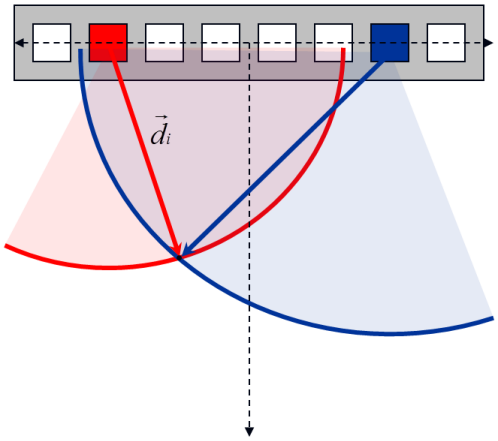

Fig. 5. Illustration of the delay calculation for the GSAU technique

purely digital, most of the calculated delays occur between samples. To simplify calculations, a nearest-neighbor interpolation is utilized by rounding the calculated delay value to match the nearest sample. The trade-off by using this type of interpolation is less accurate data points in exchange for less computation and simplicity of code.

E. Image Reconstruction System

Once the output has been fully beamformed using the GSAU technique described above, the resulting output may be processed through the image reconstruction system as shown in Fig. 6. This system consists of three operations: envelope detection, logarithmic compression, and hard-limiting. For the envelope detection, the magnitude of complex envelope of the received signal is calculated using Hilbert transform (HT) envelope detector which is described in [6, 7]. The HT envelope detector works by processing the discrete-time analytic signal

$$r_a(n) = r(n) + j\hat{r}(n) \tag{10}$$

for an input vector $r(n)$. The $\hat{r}(n)$ is the Hilbert transform of $r(n)$, which rotates all frequency components 90° by convolving $r(n)$ with $1/(\pi n)$. Assuming $r(n)$ is an AM signal of the form

$$r(n) = m(n)\cos(2\pi f_0 n + \theta) \tag{11}$$

where $m(n)$ is the modulating signal, then

$$\hat{r}(n) = m(n)\sin(2\pi f_0 n + \theta). \tag{12}$$

Therefore

$$r_a(n) = m(n)(\cos(2\pi f_0 n + \theta) + j\sin(2\pi f_0 n + \theta)) \tag{13}$$

$$= m(n)e^{j(2\pi f_0 n + \theta)} \tag{14}$$

The magnitude of $r_a(n)$ is then used as an output, to constrain the logarithmically compressed signal to real values.

The resulting output is logarithmically compressed into the decibel scale via $20\log_{10}(\cdot)$. This output is then limited to a dynamic range of -50 to 0 dB after normalizing all the signals to the highest intensity value of the image.

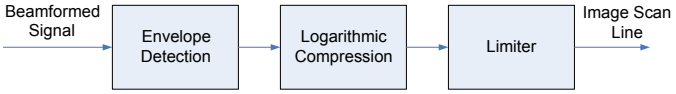

Fig. 6. Image reconstruction system

F. MATLAB Parallel Computing Toolbox

As of MATLAB R2010b, the MATLAB Parallel Computing Toolbox has incorporated many different features of GPGPU processing in over 180 of their most commonly used functions [8]. These include matrix operations, summations, and the `fft()` and `fft2()` functions. The Parallel Computing Toolbox enables GPGPU acceleration of these functions by means of the `gpuArray()` and `gather()` functions, which send the data to the GPU and CPU respectively. In addition to these capabilities, the `arrayfun()` function may be used to evaluate functions that are to be applied element-by-element, which can be ideal in massively parallel computations. In applications when more advanced manipulations are required, MATLAB supports calling CUDA kernel directly via its `feval()` function. Due to the inherent flexibility in the MATLAB Parallel Computing Toolbox, it has proven to be a useful tool for the acceleration of these simulations.

G. Serial GSAU Implementation

The serial GSAU algorithm was implemented by using two nested `for` loops in MATLAB, looping over the output scan lines and transmit events, respectively. The delay and sum calculation was then performed one output scan line at a time, using element-by-element operations to exploit MATLAB's optimized vector operations to ensure that the code ran as fast as possible. To ensure that no pixel attempted read beyond the data buffer, delays were saturated to the max edge of the buffer. This max edge was set to zero so that this saturation would not cumulatively distort the data.

H. GPU-Accelerated GSAU Implementation

In order to implement the GSAU beamformer, a CUDA kernel was written and imported into MATLAB. The GPU-accelerated GSAU algorithm was designed such that each thread spawned would evaluate a single output pixel. Each thread would loop over the number of transmit events, calculate the appropriate delay, and add the appropriately delayed signal to a local running sum. Once all the received signals were processed, each thread would export its local sum to the output array. The 20000×256 pixel image was partitioned into a 1250×16 grid of thread blocks, each of which containing 16×16 threads. Because the dimensions of the image did not evenly divide into 16×16 blocks, some of the edge blocks would extend beyond the output buffer of the image. As such, the kernel was designed such that any threads evaluating pixels outside the buffer would terminate to prevent a segmentation violation.

I. Test Hardware Specifications

The specifications for the test system are given in Table II [9]. To speed up the computations of the GSAU algorithm, an NVIDIA Quadro 5000 was used to perform parallel computations. As seen in Table III, the GPU performs much faster with single precision numbers, achieving double the amount of FLOPS (floating point operations per second). For this reason, the GPGPU GSAU implementation utilizes only single precision numbers.

TABLE II
TEST SYSTEM SPECIFICATIONS

Processor	Intel Core i7-2600K
Number of cores:	4 (8 logical)
Processor Clock:	3.40 GHz
L1 Cache Size:	32 KB
L2 Cache Size:	256 KB (shared)
L3 Cache Size:	8 MB (shared)
Theoretical GigaFLOPS	108.8
RAM:	16 GB

TABLE III
NVIDIA QUADRO 5000 SPECIFICATIONS

Number of CUDA cores:	352
Processor Clock:	1026 MHz
Dedicated RAM:	2560 MB GDDR5
Theoretical GigaFLOPS (Single Precision)	718.08
Theoretical GigaFLOPS (Double Precision)	359.04
Memory Bandwidth	120 GB/s

III. IMAGING QUALITY METRICS

To evaluate the quality of the ultrasonic imaging techniques, two fundmental imaging metrics must be calculated: spatial resolution and SNR.

A. Resolution

Both axial and lateral resolution are computed by evaluating the modulation transfer function (MTF) of the imaging signal. To compute the MTF, the ultrasonic system must first process the image of a point to determine the point spread function (PSF), or spatial impulse response of the system. The modulation transfer function may be calculated from the PSF by taking the magnitude of the spatial Fourier transform of the PSF. By convention, the MTF is normalized such that $MTF(0) = 1$. The resolution λ is then a function of the characteristic wavenumber k_0 which is defined as the point such that

$$MTF(k_0) = 0.1. \tag{15}$$

The resolution λ is then given by

$$\lambda = \frac{1}{2}\frac{2\pi}{k_0}. \tag{16}$$

B. SNR

The SNR is an imaging metric for comparing the strength of the pulse-echo signal power compared to a corrupting background noise. SNR is expressed in decibels (dB) which is shown below:

$$SNR_{dB} = 10\log_{10}\left(\frac{P_{sig}}{P_{noise}}\right) \tag{17}$$

For the purposes of this study, the SNR was computed from the axial scan line containing the imaged point.

IV. RESULTS

To test these techniques, images of a point centered 20 millimeters away from the transducer surface were collected. The REC and GSAU techniques were studied at each stage of the processing, both independently and together. For comparison purposes a conventional pulsing (CP) technique was also studied, with a delta function used as an excitation signal instead of a pre-enhanced chirp. Because a delta function input cannot be further compressed, the Wiener filter was omitted from the processing stage with the CP technique. For each of these images, axial resolution and SNR were computed using the axial center line of the image, corresponding to the point location. The lateral resolution was calculated from the lateral scan line containing the maximum point in the image. The results of imaging at each stage are shown in Fig. 7, and will be developed in the following sections.

A. REC Excitation

The desired impulse response was generated by windowing a sinusoid by windowing a 8 MHz sinusoid with a Hanning window of half length, resulting in a 200% -6 dB bandwidth system shown in Fig. 8(a). This system was excited with a linear chirp lasting 12.5 microseconds, sweeping from 0 Hz to 17.2 MHz. The resulting output was windowed with a Tukey cosine window with $\alpha = 0.8$ to generate the desired output in Fig. 8(c). The transducer impulse response was then modeled as a 8 MHz sinusoid with a full length Hanning window to generate a 100% -6 dB bandwidth system as shown in Fig. 8(d). Using the spectra of the desired system and the transducer, the desired output was Wiener deconvolved

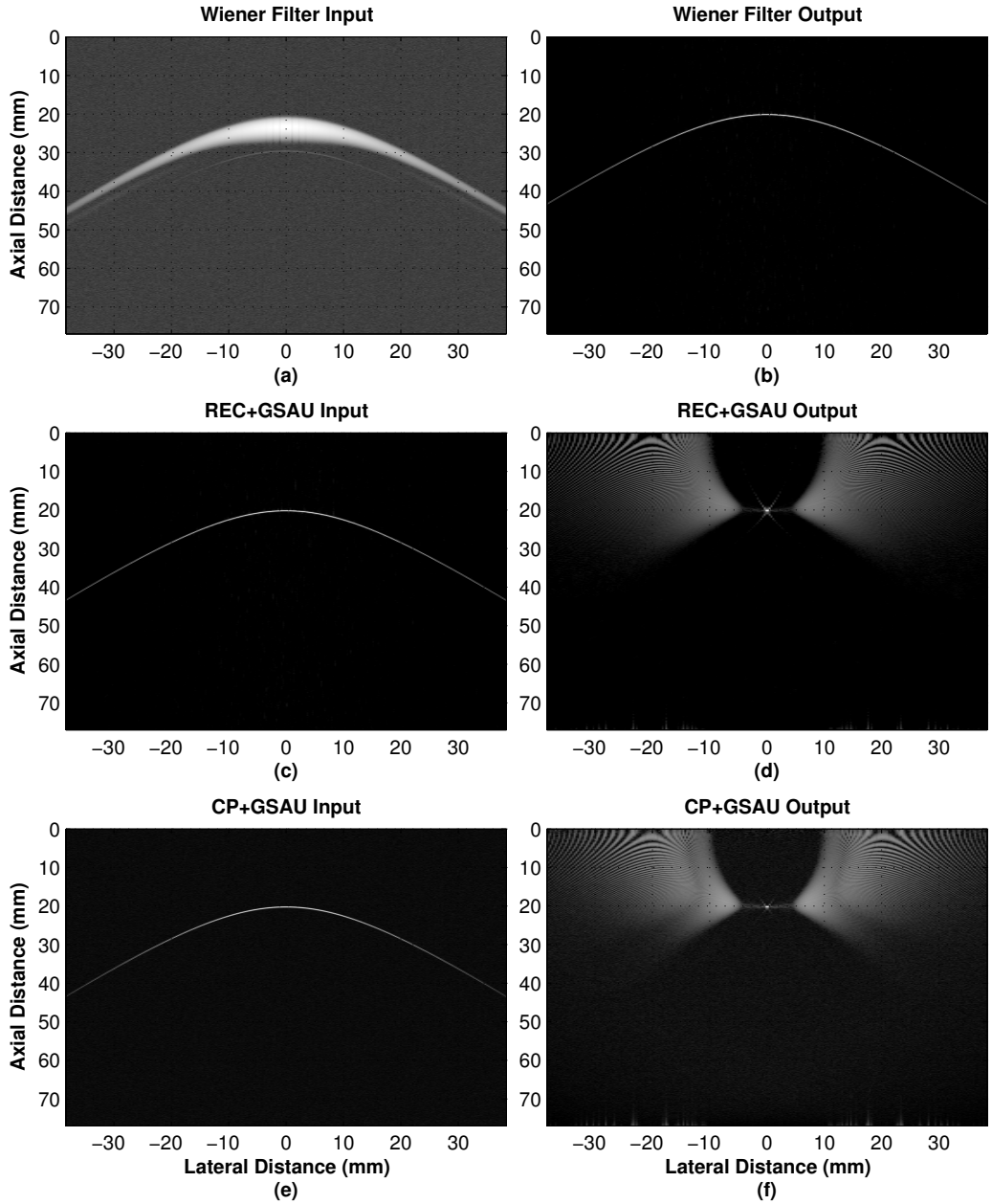

Fig. 7. (a) Uncompressed output from the ultrasonic transducer using REC excitation. (b)-(c) Pulse compressed output after Wiener filtering. (d) Final image output after beamforming with the GSAU technique using REC. (e) Received transducer output using CP excitation. (f) Final image output after beamforming with the GSAU technique using CP.

as discussed in section II-A to generate the pre-enhanced chirp shown in Fig. 8(e). The pre-enhanced chirp was then inverse filtered to generate the modified linear chirp shown in Fig. 8(b), which was then utilized in the Wiener filtering stage. Finally, the pre-enhanced chirp was applied to the transducer model to generate the actual output shown in Fig. 8(f).

The difference between the desired output and actual output was computed using a normalized mean square error (MSE) approach. The MSE is defined as

$$\text{MSE}(n) \equiv E\{|e(n)|^2\}, \tag{18}$$

where $e(n)$ is the difference between the actual and desired output [10]. To ensure that the MSE remains invariant under changes of scale, it was normalized to the variance of the actual output. The calculated normalized MSE for Fig. 8 was 1.36×10^{-6}.

B. Serial and GPU-Accelerated GSAU

A comparison of both the serial and GPU-accelerated GSAU beamformed outputs is shown in Fig. 9. Because of the parallel nature of GPU programming, there is a difference in the output. For the parallel implementation of this algorithm, the order of the additions associated with the GSAU technique is not preserved, resulting in different roundoff than the serial.

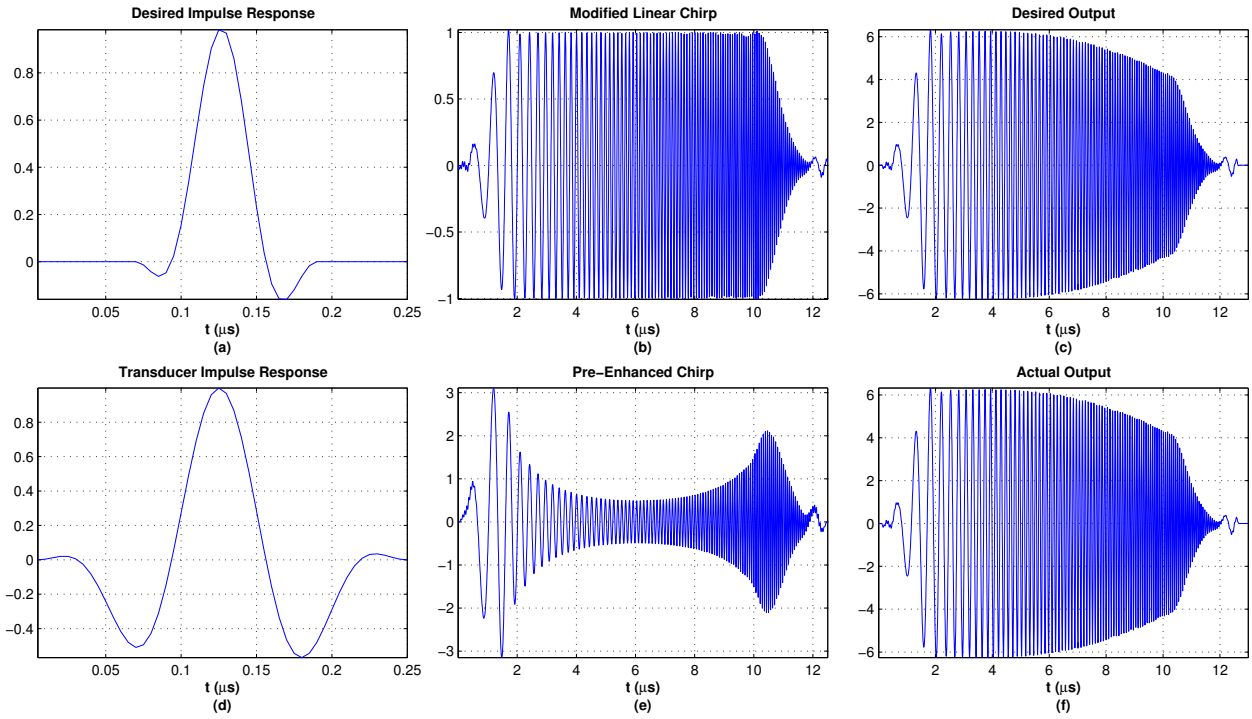

Fig. 8. (a) The desired impulse response. (b) The modified linear chirp. (c) The desired output of the system, which is equivalent to (a) convolved with (b). (d) The transducer impulse response. (e) The pre-enhanced chirp. (f) The actual output from convolving (d) with (e).

However, this only amounted to a $3.1 \times 10^{-4}\%$ difference of the signal power. This difference is largest in the regions with the strongest signal contributions, likely due to the fact that the most non-zero additions occur in those regions. The peak value of the difference is at a value of -15 dB. However, this effect is not particularly concerning, because it occurs at the peak value of the image, where it will be about 3% of the total value and will not be distinguishable on a logarithmically compressed image. The net benefit of using the GPU for this technique was a decrease in computation time from 29.25 seconds to 0.25 seconds, a speedup by a factor of 116.

C. Resolution

The axial MTF and PSF are depicted in Fig. 10 and the computed results are located in Table IV. Using the REC technique alone resulted in a 15.3% improvement in axial resolution. Using the GSAU technique, however resulted in degradation of axial resolution for both CP and REC, in the amounts of 4.02% and 23.9%, respectively.

The lateral MTF and PSF are depicted in Fig. 11 and the computed results are located in Table V. Without the GSAU technique, the REC technique appears to have degraded the lateral resolution by 4.76% over CP. With GSAU, lateral resolution improves by 63.3% and 64.5% for CP and REC, respectively.

TABLE IV
AXIAL RESOLUTION MEASUREMENTS

Measurement	Resolution (mm)
REC	0.440
REC after GSAU	0.645
CP	0.520
CP after GSAU	0.541

TABLE V
LATERAL RESOLUTION MEASUREMENTS

Measurement	Resolution (mm)
REC	0.294
REC after GSAU	0.099
CP	0.281
CP after GSAU	0.103

D. SNR

As shown in Table VI, REC demonstrates significant improvement in SNR over CP, both with and without GSAU. Without GSAU, over 5 dB improvement in the SNR is observed, with another 4 dB improvement after the GSAU technique is added. CP does not demonstrate any improvement in SNR with the GSAU technique.

TABLE VI
SNR MEASUREMENTS

Measurement	SNR (dB)
REC before Wiener Filter	19.9
REC after Wiener Filter	23.8
REC after Wiener Filter and GSAU	28.2
CP before GSAU	18.1
CP after GSAU	17.7

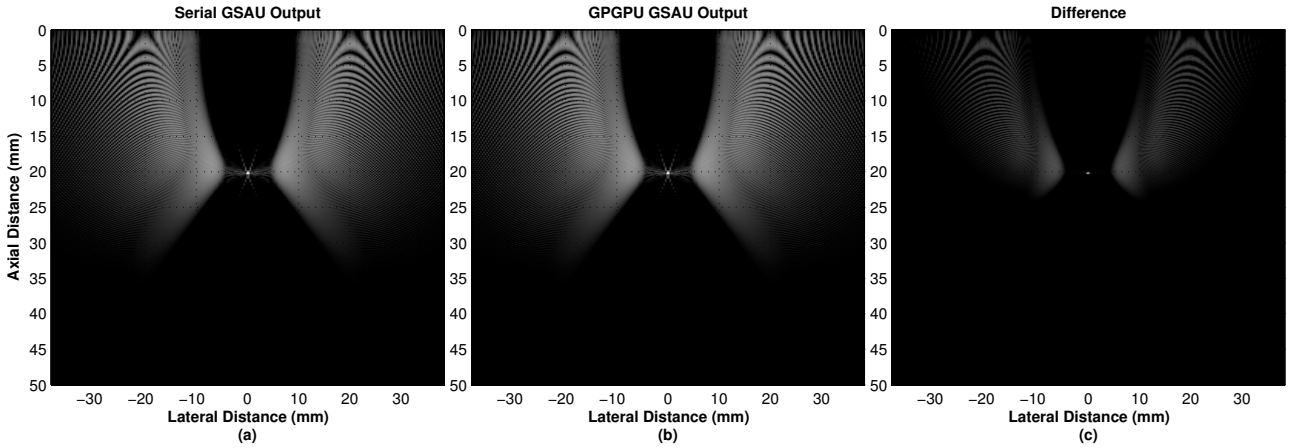

Fig. 9. (a) Serial GSAU output. (b) GPU GSAU output. (c) Difference between (a) and (b), normalized to the max of (a).

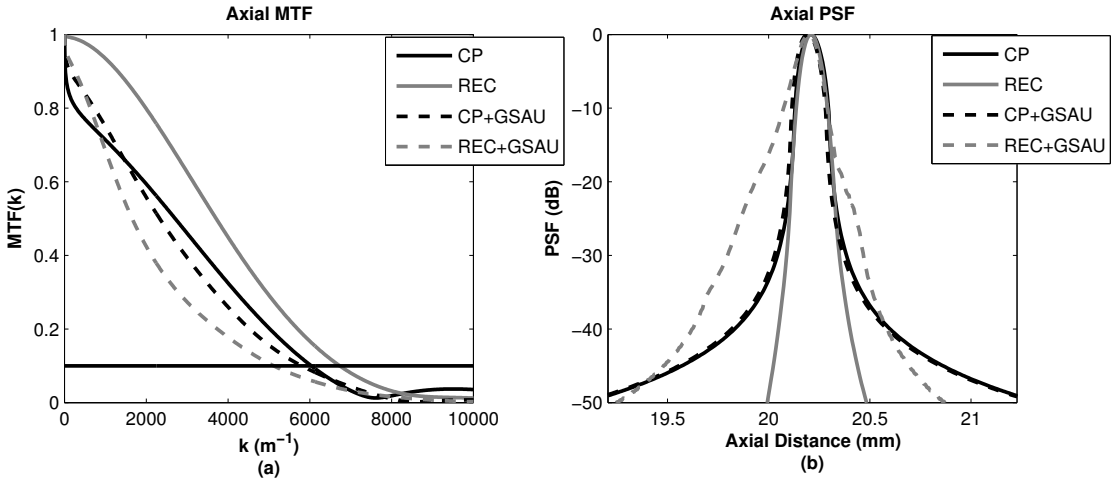

Fig. 10. (a) Axial MTF and (b) PSF of received images.

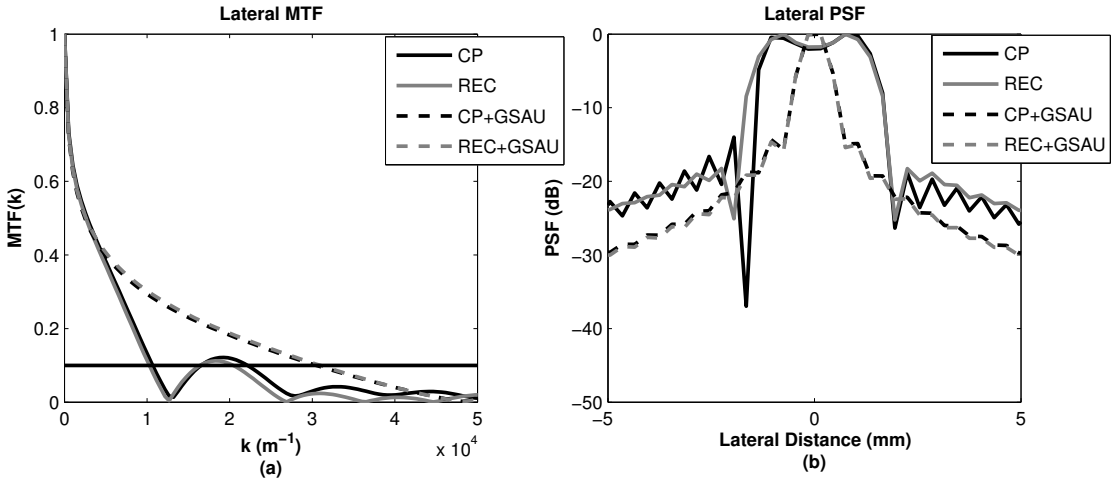

Fig. 11. (a) Lateral MTF and (b) PSF of received images.

V. Conclusion

As demonstrated in Section IV, using REC with the GSAU technique improves the lateral resolution and SNR of the image when compared to CP. However, the axial resolution with both REC and GSAU actually degrades significantly. Furthermore, the improvements in SNR were not close to the expected values for GSAU, because the theoretical improvement of GSAU should be approximately 24 dB for a 256 element transducer. Several things could contribute to this lack of expected performance. First, the GSAU technique studied utilized nearest neighbor interpolation of fractional delays, which could result in a loss of high frequency components as described in [3]. Using a linear or polynomial interpolation might result in more favorable results. The REC system can also be further tuned to improve performance by expanding the relative bandwidths of the desired and transducer impulse response and manipulating the γ factor.

VI. Acknowledgements

The author would like to thank Dr. José Sánchez for his continued support and guidance this year, Mr. Christopher Mattus for his technical support, Dr. Joseph Driscoll for his input regarding parallel programming of the GPU, and Bradley University Electrical and Computer Engineering Department for the use of their equipment.

VII. References

[1] M. Oelze, "Bandwidth and resolution enhancement through pulse compression," *IEEE Trans. Ultrason. Ferroelec. Freq. Control*, vol. 54, no. 4, pp. 768–781, Apr. 2007.

[2] J. Sanchez and M. Oelze, "An ultrasonic imaging speckle-suppression and contrast-enhancement technique by means of frequency compounding and coded excitation," *IEEE Trans. Ultrason. Ferroelec. Freq. Control*, vol. 56, no. 7, pp. 1327–1339, Jul. 2009.

[3] S. Nikolov, "Synthetic aperture tissue and flow ultrasound imaging," Ph.D. dissertation, Technical University of Denmark, 2001. [Online]. Available: https://svetoslavnikolov.wordpress.com/synthetic-aperture-ultrasound-imaging/

[4] J. Jensen, "Field: A program for simulating ultrasound systems," in *Medical & Biological Engineering & Computing*, vol. 34, 1996, pp. 351–353.

[5] J. Jensen and N. Svendsen, "Calculation of pressure fields from arbitrarily shaped, apodized, and excited ultrasound transducers," *IEEE Trans. Ultrason. Ferroelec. Freq. Control*, vol. 39, pp. 262–267, 1992.

[6] M. Schlaikjer, J. P. Bagge, O. Sorensen, and J. A. Jensen, "Trade off study on different envelope detectors for b-mode imaging," in *IEEE Symp. Ultrason.*, vol. 2. IEEE, 2003, pp. 1938–1941.

[7] W. Sangpisit, P. Wardkein, W. Kiranon, and C. Loescharataramdee, "A novel derivative envelope detector," *IEEE Trans. Consum. Electron.*, vol. 44, no. 4, pp. 1396–1405, 1998.

[8] MATLAB GPU Computing Support for NVIDIA CUDA-Enabled GPUs. [Online]. Available: http://www.mathworks.com/discovery/matlab-gpu.html

[9] "Intel i7-2600K data sheet," Intel, Santa Clara, California. [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/2nd-gen-core-desktop-vol-1-datasheet.pdf

[10] D. Manolakis, *Statistical and adaptive signal processing spectral estimation, signal modeling, adaptive filtering, and array processing*. Boston: Artech House, 2005.