UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA
Departamento de Arquitectura de Computadores y Automática

TESIS DOCTORAL

Paralelización y mejora del proceso de conformación de haz en sistemas de apertura sintética para la generación de imagen ultrasónica en tiempo real

Parallelization and improvement of beamforming process in synthetic aperture systems for real-time ultrasonic image generation

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

David Romero Laorden

Directores

Óscar Martínez Graullera
Matilde Santos Peñas

Madrid, 2016

Paralelización y mejora del proceso de conformación de haz en sistemas de apertura sintética para la generación de imagen ultrasónica en tiempo real

Tesis doctoral presentada para obtener el grado de
Doctor en Informática

David Romero Laorden

Directores

Oscar Martínez Graullera
Matilde Santos Peñas

Departamento de Arquitectura de Computadores y Automática

Facultad de Informática

Universidad Complutense de Madrid

Noviembre 2015

Consejo Superior de Investigaciones Científicas

Parallelization and improvement of beamforming process in synthetic aperture systems for real-time ultrasonic image generation

PhD Thesis submitted for the degree of
Doctor in Computer Science

David Romero Laorden

Advisors

Oscar Martínez Graullera
Matilde Santos Peñas

Department of Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid
November 2015

A mi familia

Acknowledgements

2015 ha sido un gran año en todos los sentidos y este es el broche final a una etapa increíble de mi vida. El término *paralelizar* se ha convertido sin lugar a dudas en algo que utilizo de manera cotidiana en mi día a día, basta con preguntar a cualquiera que me conozca :) es por eso que quiero lanzar cientos de hilos en paralelo para daros las gracias a todos los que me habéis apoyado a lo largo de estos años!

Quiero expresar un agradecimiento muy especial a mis directores, a Oscar Martínez Graullera pues sin él este trabajo nunca hubiera sido posible gracias a su continuo esfuerzo y dedicación donde siempre ha tenido palabras de ánimo, crítica y motivación que han conseguido sacar lo mejor de mí; y a Matilde Santos Peñas por su interés, disposición y ánimos continuos para que esta tesis llegase a materializarse.

A los profesores Bruce W. Drinkwater de la Universidad de Bristol y a Andreas Austeng de la Universidad de Oslo, por su acogida y su colaboración durante los meses de trabajo en Inglaterra y Noruega así como a todos los compañeros y amigos de laboratorio que hicieron de esos meses un hogar. Y al profesor Ricardo Tokio de la Universidad de Ilha Solteira en Brasil por la colaboración en trabajos conjuntos.

A todos mis queridísimos *csicianos* sin excepción, tanto a los que siguen en el CSIC como a los que un día estuvieron conmigo y decidieron seguir su camino fuera de aquí. Llegué siendo David "El Nuevo" y me marcho siendo David "El Añejo". Gracias a todos, especialmente a los veteranos, Luis, Alberto, Montse y Marga, por su ayuda, ánimos y por rellenar el cajón de las chuches, a las señoritas Esther, Erika, Sofía, Andrea y Dalmay y a los caballeros Jesús, Ranz, Manolo, Xavi, Luis, Jorge, Nadir, Jordi, Juan Pablo, David M., David V., Damián y Vander por reírse de mis chistes malos, por su amistad y su apoyo en todos los momentos. Haremos una buena tómbola para celebrarlo! Gracias amigos.

Mención especial para Carlos y Javi, sin vosotros amigos esto no lo hubiera conseguido. Juntos somos el bueno, el malo y el feo ya lo sabéis, aquí no voy a decir quién es quién :) Gracias por soportar al tirano, por vuestra paciencia, nuestras aventuras, enseñarme tanto en estos años de amistad y por ser como sois. Estoy seguro que lograremos lo que nos propongamos.

A mis *friki-engineers* Enrique, Irene, Marina, Natalia, David, Victor y Elena. `01000111 01110010`

`01100001 01100011 01101001 01100001 01110011`. Y en especial a Almu ¡por fín lo conseguí! y a Pablo, por tantos años de amistad desde aquella primera palmera de chocolate en Biológicas en 2002 y su apoyo en los buenos y malos momentos.

Gracias a los *mostoleños* por todo lo que hemos recorrido juntos y esos “minutejos” que siempre me habéis dedicado y a los *safa-amigos* por hacerme sentir uno más y por los sabios consejos de los ya doctores Javi e Irene.

Cómo no a los *atuneros* María, Richi, Javi, Sara, Juan Carlos, Carmen y Gloria por tantas cosas vividas a lo largo de estos años y las que quedan por vivir! Gracias por vuestros ánimos que hicieron que no abandonase nunca a pesar de la “recochura”.

A toda mi *súper familia*, a los que dedico con todo mi corazón esta tesis. Porque son la mejor familia del mundo mundial. A mis padres, Mari Carmen y Jose, porque todo esto os lo debo a vosotros, por vuestro cariño, por enseñarme siempre el fruto del trabajo y apoyarme en cada paso que doy. A mis abuelos, sobre todo al que se sentía muy orgulloso de que su nieto trabajase para los espías. Y a mi hermana Nuria y mi cuñado Fran, por estar siempre pendientes de mí, que haría yo sin las recetas y consejos de una $doctora^2$, las risas y esas cenas reconfortantes para empezar con energía un nuevo día. A mis suegros Auxi y Jose, y a mis cuñados Jose, Susana y Enrique por sus palabras de aliento, sus dulces energéticos y su apoyo, y en especial a súper Patri por hacer que nos lo pasemos siempre pipa-pipa. Os quiero.

Por último, a Nataly, mi *artista* preferida, por su infinito cariño, su humor, su paciencia, su apoyo constante, por soportar esas noches tan largas de trabajo y por haberme dicho *oui* aquel sábado 31 de Mayo. Te quiero. Larga vida a los RomSom!

Resumen

La ecografía es hoy en día uno de los métodos de visualización más populares para examinar el interior de cuerpos opacos. Su aplicación es especialmente significativa tanto en el campo del diagnóstico médico como en las aplicaciones de evaluación no destructiva en el ámbito industrial, donde se evalúa la integridad de un componente o una estructura.

El desarrollo de sistemas ecográficos de alta calidad y con buenas prestaciones se basa en el empleo de sistemas multisensoriales conocidos como *arrays* que pueden estar compuestos por varias decenas de elementos. El desarrollo de estos dispositivos tiene asociada una elevada complejidad, tanto por el número de sensores y la electrónica necesaria para la adquisición paralela de señales, como por la etapa de procesamiento de los datos adquiridos que debe operar en tiempo real. Esta etapa de procesamiento de señal trabaja con un elevado flujo de datos en paralelo y desarrolla, además de la composición de imagen, otras sofisticadas técnicas de medidas sobre los datos (medida de elasticidad, flujo, etc).

En este sentido, el desarrollo de nuevos sistemas de imagen con mayores prestaciones (resolución, rango dinámico, imagen 3D, etc) está fuertemente limitado por el número de canales en la *apertura* del array. Mientras algunos estudios se han centrado en la reducción activa de sensores (*sparse arrays* como ejemplo), otros se han centrado en analizar diferentes estrategias de adquisición que, operando con un número reducido de canales electrónicos en paralelo, sean capaz por multiplexación emular el funcionamiento de una apertura plena. A estas últimas técnicas se las agrupa mediante el concepto de *Técnicas de Apertura Sintética* (SAFT). Su interés radica en que no sólo son capaces de reducir los requerimientos hardware del sistema (bajo consumo, portabilidad, coste, etc) sino que además permiten dentro de cierto compromiso la mejora de la calidad de imagen respecto a los sistemas convencionales.

La principal limitación de las técnicas de apertura sintética es el elevado coste computacional que tienen asociado. Disponer del conjunto de datos crudos de la adquisición independiente de las señales que componen una apertura de N elementos (conocido como *Full Matrix Capture*, FMC) proporciona un conjunto de datos de $N \times N$ señales (conocido como *Full Matrix Array*, FMA). Este conjunto de datos se combina de manera adecuada para generar una imagen de I píxeles mediante un proceso de *conformación de haz* o *beamforming*, que compensa los tiempos de propagación de todas las señales a cada uno de los píxeles de la imagen. Este proceso es conocido como *Total Focussing Method* (TFM) y utilizado en combinación con el FMC ha dado lugar a un nuevo estándar de calidad, pues proporciona la máxima resolución lateral posible para la apertura en todos los píxeles de la imagen. Desafortunadamente, esta técnica arrastra por un lado un alto coste computacional de $\mathcal{O}(N \cdot N \cdot I)$, que supone un difícil compromiso entre el paralelismo en hardware y el coste de la ejecución del FMC; y por otro lado tiene una baja relación señal a ruido como consecuencia de la baja energía acústica radiada durante la emisión en el medio.

Sobre toda esta problemática se desarrolla el trabajo aquí presentado, analizando el papel de los sistemas de procesamiento paralelo y evaluando su capacidad para generar imagen 2D y 3D en tiempo real, tanto en su configuraciones multi-core (CPUs) como en procesadores gráficos (GPUs).

La primera parte de este trabajo se desarrolla sobre la obtención de modelos de campo acústico. Las técnicas de modelización de transductores son muy importantes tanto para el diseño de aperturas dispersas, una de las estrategias apuntadas para el desarrollo de nuevos sistemas, como el desarrollo de espacios simulados que permiten modelar situaciones reales y probar nuevos algoritmos. En este sentido este tipo de problemas se adaptan muy bien al modelo de paralelización SIMD (*Single Instruction Multiple Data*), porque son algoritmos que requieren una gran potencia de cálculo, no tanto por la complejidad del modelo sino por el gran número de puntos sobre los que se debe realizar de manera independiente el cálculo y por los altos requerimientos de resolución.

La segunda parte está centrada en los procesos de conformación de complejidad $\mathcal{O}(N \cdot N \cdot I)$ como es el caso del TFM + FMC. Se han diseñado algoritmos con objeto de valorar la posibilidad de sustituir un conformador hardware tradicional por uno software. Se han analizado diferentes alternativas que muestran como una planificación adecuada del acceso a los datos y un buen uso de los recursos permiten mejorar los tiempos de ejecución de los algoritmos. Así, se ha demostrado que es posible conseguir un sistema de imagen en tiempo real de 25 imágenes por segundo para un array de $N = 64$ elementos y una imagen de 512×512 píxeles y hasta 65 imágenes por segundo para imágenes de 256×256 usando una tarjeta gráfica Nvidia Quadro K5000. Adicionalmente, esto ha dado lugar al desarrollo de una librería de procesamiento multiplataforma tanto para imagen 2D como 3D, que integra gran variedad de técnicas SAFT y facilita el prototipado de nuevas técnicas de imagen, permitiendo obtener imágenes o volúmenes en pocos milisegundos.

La tercera parte estudia aspectos más prácticos enfocados a desarrollar sistemas de bajo coste, alta calidad y alta velocidad de imagen. En este sentido una parte importante del trabajo se centra en la problemática del FMC y en desarrollar técnicas alternativas que exploten las soluciones de *mínima redundancia del coarray* (MRC). Para estas alternativas, que tienen un coste computacional de $\mathcal{O}(N \cdot I)$, se han desarrollado procedimientos de conformación de haz que implementan la técnica TFM + MRC. Asimismo se propone una nueva técnica para la composición del MRC, FAST-SAFT, que minimiza el número de número de disparos y acelera el proceso de captura de las señales.

Finalmente se ha estudiado el problema del rango dinámico a través del empleo de técnicas de compresión de pulsos, en particular con el uso de códigos de Golay.

Como conclusiones, los resultados obtenidos en el momento de las publicaciones han demostrado que es posible conseguir imagen a alta velocidad de 200 imágenes por segundo para imágenes de 256×256 y un array de $N = 64$ elementos. Actualmente, debido a mejoras en la organización de los datos y a nuevas tarjetas gráficas esta cifra se sitúa en 1000 imágenes por segundo bajo las mismas plataformas de cómputo que en el caso del FMC. Esto unido al desarrollo del concepto de FAST-SAFT que reduce el número de disparos en función del paralelismo del sistema. Dado un array de 64 transductores y con 32 canales en recepción esta técnica reduce los 64 disparos que se precisan para un 2R-SAFT a sólo 5, acelerando el proceso de captura y obteniendo una calidad de imagen comparable a la obtenida con TFM + FMC. Por tanto, las soluciones propuestas en este trabajo demuestran que es posible el desarrollo de sistemas compactos y alta calidad de imagen es algo real con un gran campo de aplicación en el sector de la medicina y de los ensayos no destructivos.

Abstract

Ultrasound is nowadays one of the most popular visualization methods to examine the interior of opaque objects. Its application is particularly significant in the field of medical diagnosis as well as non-destructive evaluation applications in industry.

The development of high performance ultrasound imaging systems is based on the use of multi-sensory systems known as *arrays*, which may be composed by dozens of elements. The development of these devices has associated a high complexity, due to the number of sensors and electronics needed for the parallel acquisition of signals, and for the processing stage of the acquired data which must operate on real-time. This signal processing stage works with a high data flow in parallel and develops, besides the image composition, other sophisticated measure techniques (measure of elasticity, flow, etc).

In this sense, the development of new imaging systems with higher performance (resolution, dynamic range, 3D imaging, etc) is strongly limited by the number of channels in array's *aperture*. While some studies have been focused on the reduction of active sensors (i.e. *sparse arrays*), others have been centered on analysing different acquisition strategies which, operating with reduced number of electronic channels in parallel, are able to emulate by multiplexing the behavior of a full aperture. These latest techniques are grouped under the term known as *Synthetic Aperture Techniques* (SAFT). Their interest is that they are able to reduce hardware requirements (low power, portability, cost, etc) and also allow to improve the image quality over conventional systems.

The main limitation of synthetic aperture techniques is the high computational cost required. Working with the raw-data derived from the independent acquisition of the signals which make an aperture of N elements (known as *Full Matrix Capture*, FMC) provides a set of $N \times N$ data signals (known as *Full Matrix Array*, FMA). This dataset is properly combined in order to generate an image of I pixels by a *beamforming* process, which compensates the propagation times of all signals to each image pixels. This process is known as *Total Focusing Method* (TFM) and when used in combination with the FMC it resulted in a new quality standard, as it provides the maximum possible lateral resolution in all pixels of the image. Unfortunately, this technique involves on one hand a high computational cost of $\mathcal{O}(N \cdot N \cdot I)$, which supposes a difficult compromise between the hardware parallelism and the execution cost of FMC; and on the other hand, it has a low signal to noise ratio as a result of the low acoustic energy radiated during the emission in the medium.

Above all these issues the work here presented is developed, analysing the role of parallel processing systems and assessing their ability to generate 2D and 3D imaging for real-time, both in multi-core configurations (CPU) and graphics processors (GPUs).

The first part of this work is focused on the development of acoustic field models. Modelling transducer techniques are very important for the design of sparse apertures, one of the strategies aimed to develop new systems, such as the development of simulated spaces to allow modelling real situations and test new algorithms. In this sense these problems are well suited to SIMD (*Single Instruction Multiple Data*) parallelization mode, because they are algorithms that require a great power calculation, not by the complexity of the model but for the large number of points on which must independently calculate and the high resolution requirements.

The second part is focused on the beamforming processes in the order of $\mathcal{O}(N \cdot N \cdot I)$ as the case of TFM + FMC. Algorithms are designed in order to assess the possibility of replacing a traditional hardware beamformer by a software beamformer. We analysed different alternatives which show how proper planning of access to data and a good use of resources can improve execution times of the algorithms. Thus, it has been shown it is possible to achieve an imaging system in real time of 25 images per second with an array of $N = 64$ elements and a 512×512 image pixels and up to 65 images per second for 256×256 images using an Nvidia Quadro K5000 GPU. Additionally, this has led to the development of a multi-platform library both 2D and 3D image processing, which integrates a variety of SAFT techniques and facilitates the prototyping of new imaging techniques, allowing to obtain images or volumes in few milliseconds.

The third part studies more practical aspects focused on developing low-cost systems, high quality and frame-rate. In this regard, an important part of the work is focused on FMC problem and to develop alternative techniques that exploit solutions of *minimum redundancy coarrays* (MRC). For these alternatives, which have a computational cost of $\mathcal{O}(N \cdot I)$, we have developed procedures that implement beamforming technique for TFM + MRC. Likewise, a new technique is proposed, FAST-SAFT, which enables the composition of the MRC minimizing of number of shots and accelerating the process of capturing signal. Finally, we have studied the problem of dynamic range through the use of techniques pulse compression, particularly with the use of Golay codes.

In conclusion, the results have shown that it is possible to achieve 200 images per second for 256×256 image pixels using an array of $N = 64$ elements. Nowadays, due to new improvements on data management and new graphics cards this number has reached 1000 imáges per second under same computational platforms used in FMC. This combined with the development of FAST-SAFT concept which reduces the number of shots depending on the system's parallelism. Given an array of 64 transducers and 32 channels in reception this technique reduces the 64 shots that are required by 2R-SAFT to only 5, accelerating the capture process and obtaining an image quality comparable to that obtained by TFM + FMC. Therefore, the solutions proposed along this thesis demonstrate that the development of compact and high image quality systems is possible and real with a large field of application in the medical and non-destructive testing fields.

Contents

Part I

Ultrasonography

1 Introduction

This chapter briefly introduces the concepts of ultrasound imaging. It can be englobed in three main sections. The first part introduces the fundamentals of ultrasonic array systems, imaging quality requirements (resolution, signal-to-noise ratio) and the procedures for collecting ultrasound data and generating (*beamforming*) images or volumes. The second part presents the state-of-art of the optimization of synthetic aperture beamforming and heterogeneous parallel computing, thus establishing the starting point of the thesis. The chapter finishes presenting the motivation, goals and main conclusions of the work.

1.1 Ultrasonic imaging

Ultrasound denotes mechanical vibrations with a frequency of oscillations above $20KHz$, the highest audible frequency. The term ultrasonic imaging or *ultrasonography* covers the area of remote sensing, where mechanical vibrations with known properties are generated, sent through a medium, and consecutively interpreted to characterize the medium. Thus, a classic ultrasound image represents the amplitude of the echoes produced by the reflectors or *scatterers*. The image shows the changes of *acoustic impedance* $\mathcal{Z}$ through the medium, which is defined as the ratio of the acoustic pressure p applied to a system and the velocity associated to the oscillating particle in the medium v:

$$\mathcal{Z} = \frac{p}{v} = \rho c \tag{1.1}$$

and it can also be expressed as the product of the density ρ and velocity of propagation c in the medium. The difference in the impedance at the interface between two materials is commonly referred to as the impedance mismatch. The greater the impedance mismatch, the greater the percentage of energy that will be reflected at the interface or boundary between one medium and another. The fraction of the incident wave intensity that is reflected or *reflection coefficient* can be calculated when the acoustic impedances of the materials on both sides of the boundary are known following this expression [78]:

$$K_R = \frac{\mathcal{Z}_2 - \mathcal{Z}_1}{\mathcal{Z}_2 + \mathcal{Z}_1} \tag{1.2}$$

Since the amount of reflected energy plus the transmitted energy must equal the total amount of incident energy, the transmission coefficient is then calculated by Equation 1.3. Figure 1.1 illustrates the changes of $\mathcal{Z}$ between two different mediums when a pulse is sent. Echo energy can be very low up to 40 dB below the emitted pulse.

$$K_T = \frac{2\mathcal{Z}_2}{\mathcal{Z}_2 + \mathcal{Z}_1} \tag{1.3}$$

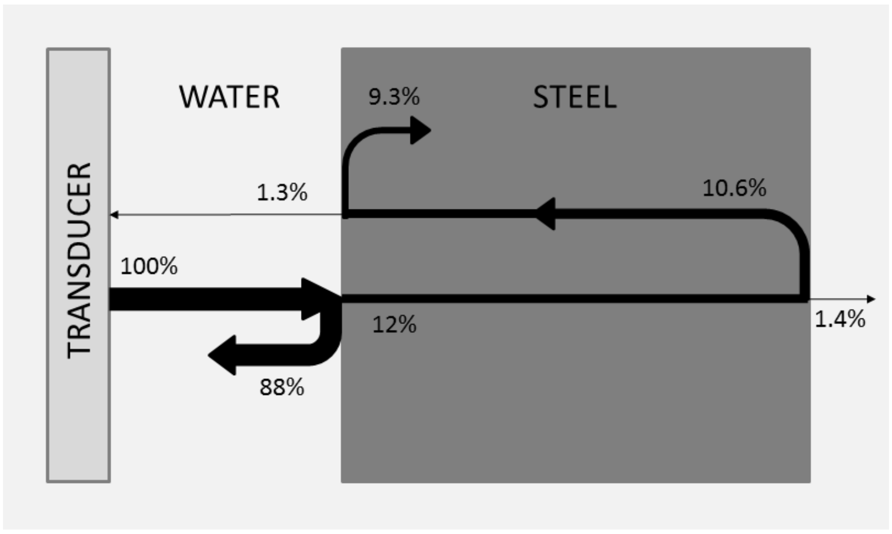


FIGURE 1.1: Acoustic impedance changes between two mediums. The sound energy leaves the transducer, travels through the water, encounters the front surface of the steel, encounters the back surface of the steel and reflects back through the front surface on its way back to the transducer. At the water steel interface (front surface), 12% of the energy is transmitted. At the back surface, 88% of the 12% that made it through the front surface is reflected. This is 10.6% of the intensity of the initial incident wave. As the wave exits the part back through the front surface, only 12% of 10.6% or 1.3% of the original energy is transmitted back to the transducer

Ultrasonography is currently one of the most popular visualization methods to examine the inside of opaque objects. Its application is especially significant in the medical diagnosis area, as well as in *non-destructive testing* (NDT) applications where the integrity of a component or structure is evaluated. In this sense, biological tissues present subtle changes in acoustic impedance, because both the density and velocity of propagation are similar in soft tissues, whereas in NDT a crack within a solid material has a significant change in acoustic impedance. All cases require good image resolution and contrast in order to facilitate the inspection and localization of multiple defects. Ultrasounds can generate quality images and, compared to other techniques like *X-Ray* or *Magnetic Resonance Imaging* (MRI), they have several advantages: they are cheap, relatively safe because they do not involve ionizing radiation, and they are non-invasive techniques. Moreover, they do not require expensive and special facilities for their use.

The development of ultrasonography has a long history. In 1928, the detection of defects in metals is discussed by the soviet physicist Sergei Y. Sokolov who is considered the initiator of non-destructive ultrasonic evaluation. Thus, early devices were based on simply registering the ultrasonic attenuation of a wave passing through the material (known as transmission technique), using quartz crystals with piezoelectric effect[1], discovered in 1880 by Pierre and Jacques Curie brothers. A greater attenuation in a region was indicative of the presence of an internal defect.

Research for medical diagnostic applications continued applying these techniques until, in 1969, the first ultrasound image generation system in real-time [77] was patented. It was based on a piezoelectric transducer capable of rotating, located at the focus of a parabolic mirror. In this way, the ultrasonic beam parallel rays were reflected to the patient's body. This principle was used by Siemens Vidoson devices since 1965, with a frame-rate from 15 to 30 frames per second [41]. Thereby, around mid 70's there is a complete knowledge-base for quality ultrasound in real-time scenarios [23]. In the industrial area, ultrasound was used for multiple applications, mainly focused on the *A-scan* analysis [86].

Although the physical principles of diffraction, which are the base of ultrasonic imaging, have been known for centuries (Trait de la Lumire by C. Huygens [46]), electronic technology and materials advances made possible the popularization of this technique from the 80's enabling the development of transducers with multiple elements called *arrays* [25]. This was a big evolution in ultrasonic imaging, giving rise to the current technology of ultrasound in clinical and industrial areas. Thus, in the case of medical diagnosis, having sold nearly 3000 Vidoson equipped with a single transducer, in 1980 Siemens waive this technology in favour of a more reliable and better performance array-based systems [156].

To develop ultrasound imaging systems a number of components is required: the ultrasonic array, which is in charge of generating pulses and sensing the echoes from the medium and determines the characteristics of the ultrasonic beam; signal emission and reception electronics; and signal processing and display subsystems. The purpose of all of them is to produce a high quality image in the shortest time possible, allowing the visualization of live organs, responding instantly to the action of the operator on the transducer or decreasing the time of automatic inspection systems in NDT.

1.2 Ultrasonic array systems

The performance of an ultrasonic imaging system is determined by the transducer, and it is reflected in its *radiation pattern* [3]. When a mono-element transducer is used the radiation pattern can be

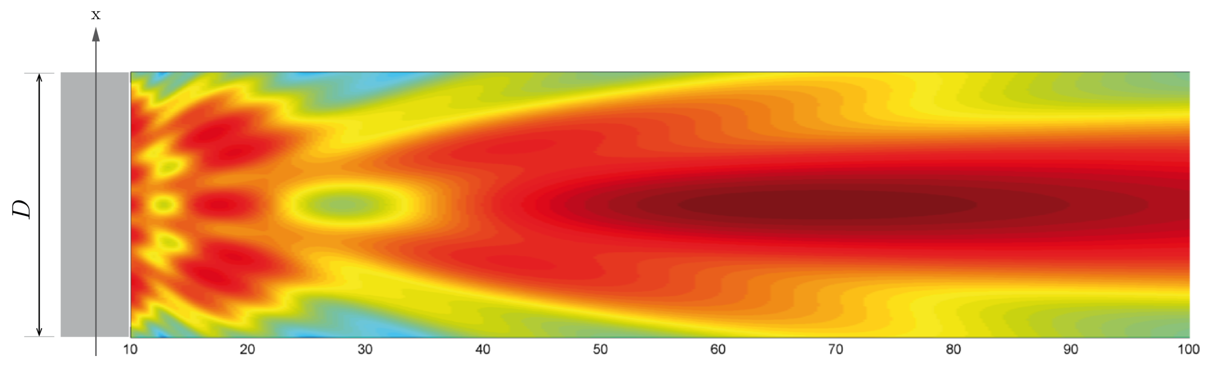
[1] Piezoelectricity is the electric charge which accumulates in certain solid materials (such as crystals, certain ceramics, and biological matter such as bone, DNA and various proteins) in response to applied mechanical stress

only modified by acting physically on the shape of the radiating element.

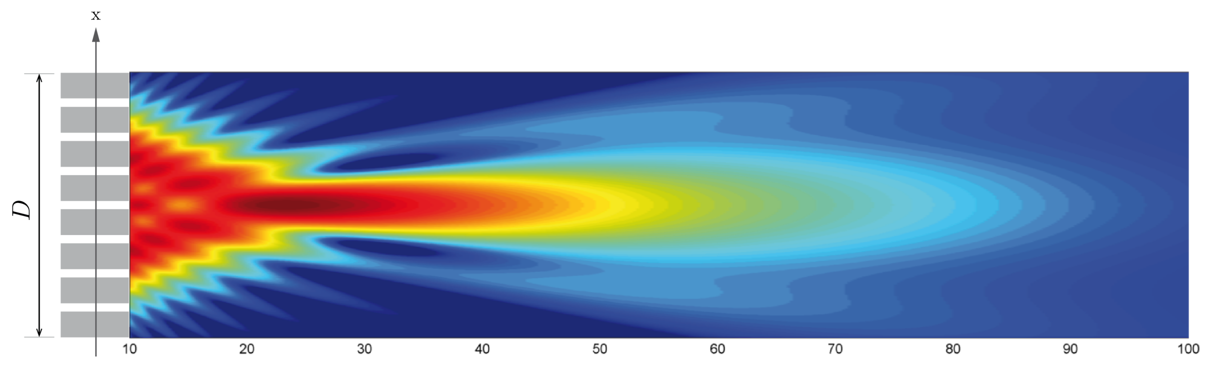
Generally, the area or volume to be inspected is located in the *near-field* or Fraunhofer zone of the transducer [3], [142] until it reaches a depth known as *far-field* distance (Z_L) calculated as:

$$Z_L = \frac{D^2}{4\lambda} \tag{1.4}$$

where D is the *aperture* of the transducer and λ is the wavelength. Here the *far-field* or Fresnel region begins, and the acoustic pressure field will suffer attenuation of $\frac{1}{z}$ due to the wave free space propagation. Figure 1.2(a) illustrates the acoustic field obtained when a mono-element transducer is employed, where it can be seen they have a limited field of application. Transducer frequency is $3MHz$, propagation medium is water ($c = 1480m/s$) and its size is $D = 10mm$, which means far-field is located at $Z_L = 50mm$. In the figure the depth of the *natural focus* of the transducer is represented, which coincides with the far-field distance. For depths below Z_L value, the propagation presents an oscillatory behaviour, and the beam can be *focused*.



(a)



(b)

FIGURE 1.2: Acoustic field was calculated using MTM method [50], [51] with 1.2(a) mono-element transducer without focus; 1.2(b) 40 element array with focus at $25mm$

In Figure 1.2(a) the acoustic field of a mono-element transducer with spherical surface is shown, with a focus located at $50mm$. Here, to focus the beam at point (R, θ), where R is the distance to the center of the transducer and θ the deflection angle, the delay that must be applied to the transducer in each direction is:

$$\tau = \frac{R - R_x}{c} \tag{1.5}$$

where R_x is the distance from de mono-element transducer to the focus point calculated by the Cosine Theorem as:

$$R_x = sqrt(x^2 + R^2 - 2xRsin(\theta)) \tag{1.6}$$

Substituting the value of these distances in the Equation 1.5 the final focusing delays are given by:

$$\tau = \frac{1}{c}(R - sqrt(x^2 + R^2 - 2xRsin(\theta))) \tag{1.7}$$

In order to control the near-field where high resolution beams can be achieved, the solution is to use transducers composed by multiple radiating elements or arrays. Thus, an array enables to divide the aperture of the transducer. Figure 1.2(b) shows the acoustic field generated by a linear array of 40 elements spaced half wavelength $d = \frac{\lambda}{2}$ which has been focused at $25mm$. By controlling the time of activation of the different elements of the array it is possible to deflect and focus the beam without moving the transducer in the region of interest [88]. It is also possible to modify the gain of each of the elements (*apodization* of the array), which allows more control of the radiation characteristics of the system [142].

Now, the focusing equation 1.5 can be rewritten as:

$$\tau_i = \frac{R - R_i}{c} \tag{1.8}$$

where R_i is the distance from the i^{th} array element to the focus point calculated as:

$$R_i = sqrt(x_i^2 + R^2 - 2x_iRsin(\theta)) \tag{1.9}$$

and the final focusing delays are:

$$\tau_i = \frac{1}{c}(R - sqrt(x_i^2 + R^2 - 2x_iRsin(\theta))) \tag{1.10}$$

Basically, depending on how the elements are distributed in the aperture, the arrays can be classified as *linear* (1D array) or *matrix* (2D array) [22] as they are presented in Figure 1.3. In linear arrays the elements are arranged in a row, enabling to obtain an image situated in front of the transducer. In the case of matrix arrays the elements are arranged in a plane, what allows to deflect and focus the beam on all directions and generate volumetric images [20], [30], [74], [89].

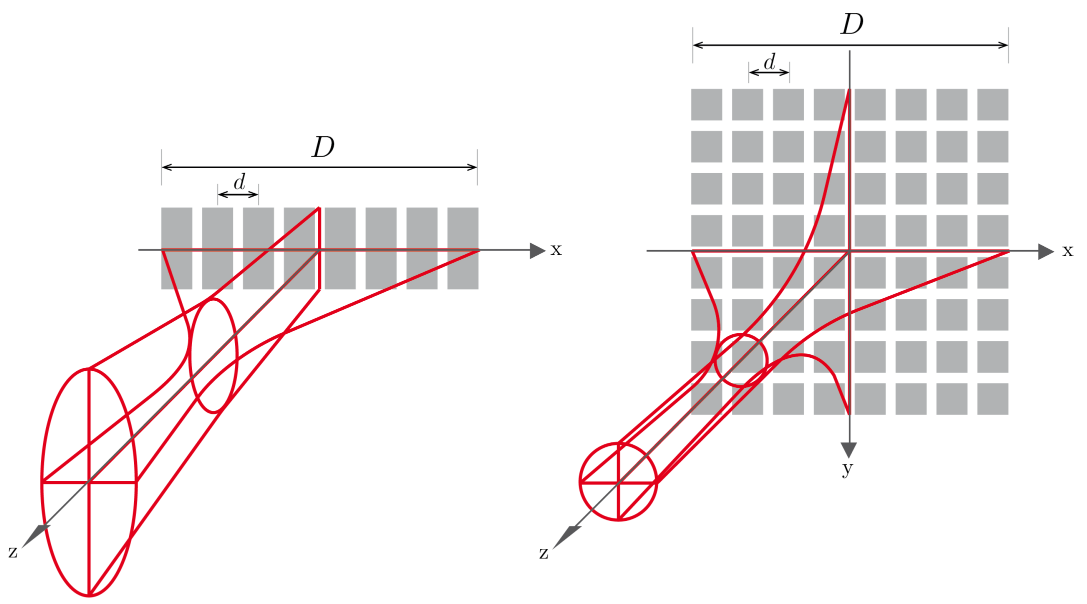


FIGURE 1.3: Different arrays configurations linear and matrix along the structure of the acoustic field generated by each one. Scheme of the geometries for the case of linear and matrix arrays as well as its position in the respective coordinate systems

For the case of one-dimensional configuration we always assume that the array will consist of N elements equally spaced at d intervals which results in an aperture of $D = N \cdot d$ size. For the two-dimensional configuration it will be generally distributed in a matrix structure of N rows and N columns where the spacing between the elements is constant and equal to d in both directions. Figure 1.3 shows both schemes.

The control of the coordinated action of the elements grouped in an array to change the acoustic beam at determined location is known as *beamforming*.

1.2.1 Beamforming with phased array systems

Considering a linear array with N elements with each array element located at $\vec{x}_i$ position and an ideal reflector located at position $\vec{x}_p$, the signal generated by a *phased array* (PA) imaging system can be implemented by two stages. The first stage of the beamforming is executed to compensate the propagation delay between elements in emission. This operation is called *focusing in transmission*. The signal in emission that reaches the focus can be written as:

$$S_E(t, \vec{x}_e) = \sum_{i=0}^{N-1} X(t) * \delta(t - T_i(\vec{x}_p) * \delta(t - \tau_i(\vec{x}_e)) \tag{1.11}$$

where $X(t)$ is the ultrasonic emitted pulse, $*$ is the convolution operator, $\delta(t)$ is the Dirac delta function, i is the index referred to the elements in emission of the aperture and $T_i(\vec{x}_p)$ is the propagation time from the i^{th} element to the spatial position defined as:

$$T_i(\vec{x}_p) = \frac{|\vec{x}_p - \vec{x}_i|}{c} \tag{1.12}$$

being c the medium sound velocity. Finally, $\tau_i(\vec{x}_e)$ is the focus delay in emission to focus at $|\vec{x}_e|(\sin\theta_e, \cos\theta_e)$ position, that is described by:

$$\tau_i(\vec{x}_p) = T_i(\vec{x}_p) - \max\{T_j(\vec{x}_p)\}, \quad \text{for} \quad j = 0, \ldots, N-1 \tag{1.13}$$

By considering all received signals, in the second stage, the reception beamforming process can be applied according to:

$$S_R(t, \vec{x}_e, \vec{x}_r) = \sum_{j=0}^{N-1} (S_E(t, \vec{x}_e) * \delta(t - T_j(\vec{x}_p))) * \delta(t - \tau_j(\vec{x}_r)) \tag{1.14}$$

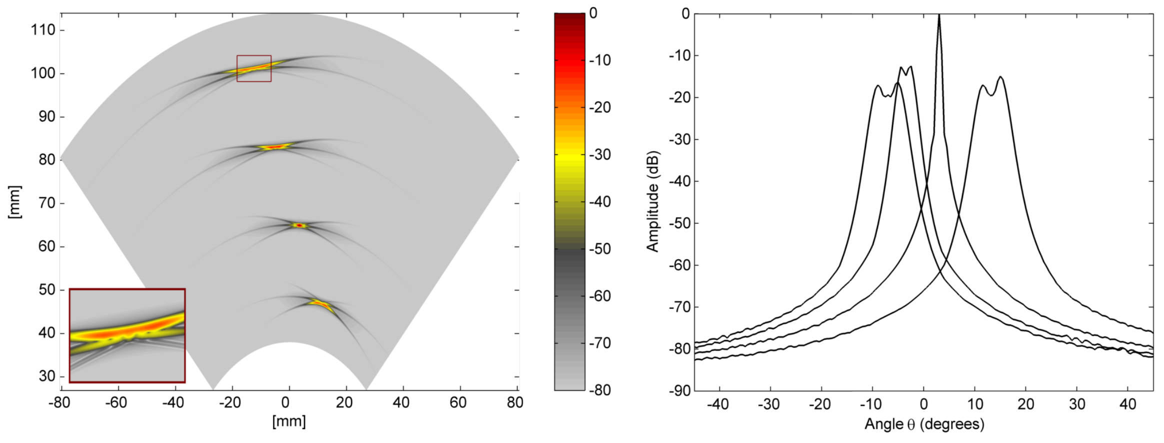
where the index j is referred to the transducers working in reception and $\tau_j(\vec{x}_r)$ is the reception focusing delay. The expression into the parenthesis in (1.14) can be considered as the signal received by the j^{th} channel. Although $\vec{x}_e$ and $\vec{x}_r$ can be different, it is a good practice to locate $\vec{x}_e$ at the same steering angle θ_e where the emitted energy is concentrated. When $\vec{x}_r = \vec{x}_e = \vec{x}_p$ the system produces at this position the best image possible.

From equation 1.14 an important analysis tool is the *Point Spread Function* (PSF). The PSF describes the response of an imaging system to a reflector point. That is, it determines the impulse response of the imaging system in terms of resolution, contrast and signal-to-noise ratio.

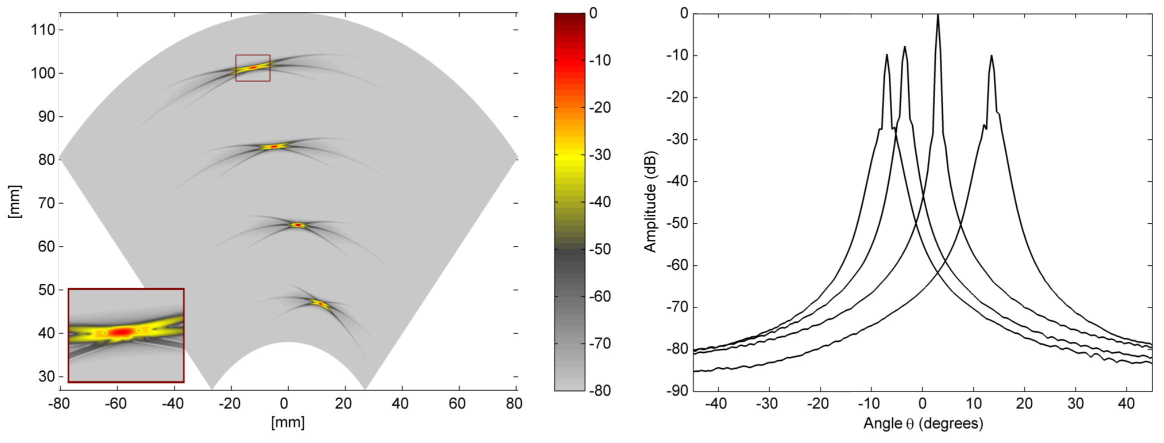
Left side of Figure 1.4 shows three different PSFs of 4 reflectors. Figure 1.4(a) presents the situation where a fixed focus on emission and reception has been applied at $60mm$ where amplitude and resolution of the other reflectors are highly reduced because of the elongation of the pulse caused by defocusing, whereas in Figure 1.4(b) dynamic focusing in reception has been applied which then improves the resolution. The Figure also includes enlargements of the fourth reflector's PSF obtained in each case. In the right side of Figure 1.4 the angular field values extracted from the images are presented.

Both beamforming operations perform the same action, by compensating the propagation delays between elements. However, while the focus on emission is limited to a region determined by the focusing law, the reception beamforming operates over the acquired data and can be dynamically adapted, providing dynamic focusing capability and compensating part of the degradation process at least in the steering angle θ_e determined by the emission focus. Figure 1.4(b) exposes this situation. Then, the process can be reformulated from Equation 1.14 as:

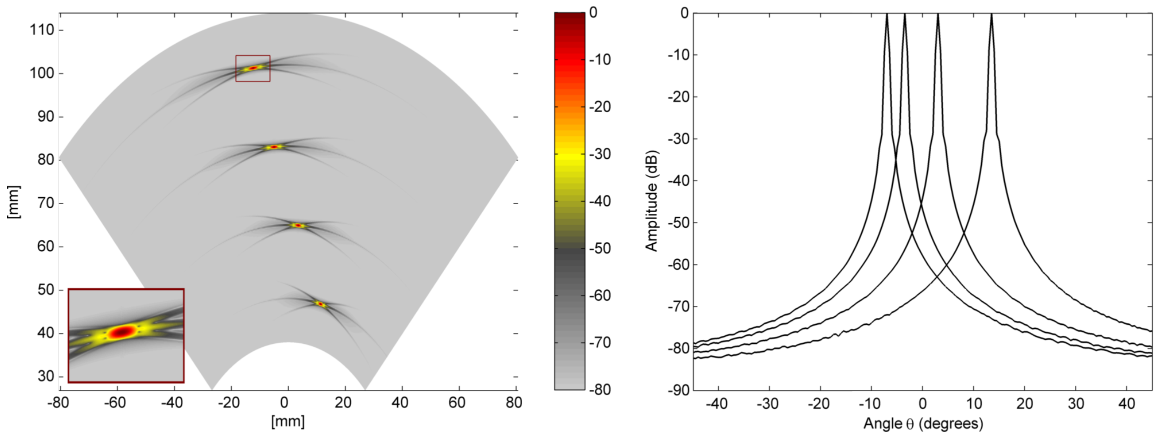
$$I(t, |x_e|, \theta_e) = S_R(t, \vec{x}_e, \vec{x}_r = ct(\sin\theta_e, cos\theta_e)) \tag{1.15}$$



(a)



(b)



(c)

FIGURE 1.4: Images and angular field values obtained with each of the reflectors to apply: 1.4(a) fixed focus on emission and reception (PA), 1.4(b) fixed focus on emission and dynamic focus in reception (PA) and 1.4(c) dynamic focus on emission and reception (SAFT) [90]

This operating mode is known as dynamic focusing on reception, and it yields the maximum quality that a parallel beamformer (PA) is able to produce.

1.2.2 Beamforming with synthetic aperture systems

From Figure 1.4 and Equation 1.15 it is observed that the image generation process in PA is mainly determined by the focussing process, and the quality of the image in regions far away from the emission focus is degraded. To extend the efficiency of the focusing process it is necessary to avoid the conventional emission focusing in the medium and to find an alternative in order to compensate the propagation emission delays dynamically.

Equation (1.14) can be rearranged to decompose the emission beamforming process defined by (1.11). The full beamforming process can now be described as:

$$S_R(t,\vec{x}_e,\vec{x}_r) = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1}(X(t) * \delta(t - T_i(\vec{x}_p)) * \delta(t - T_j(\vec{x}_p))) * \delta(t - \tau_i(\vec{x}_e)) * \delta(t - \tau_j(\vec{x}_r)) \quad (1.16)$$

where the term in brackets can be considered the wave acquired in reception by the j^{th} element with emission by the i^{th} element. In order to provide a more general expression, the reflected wave can be extended to an undetermined number of reflectors, each one with its own reflectivity coefficient a_k:

$$X_{ij}(t) = X(t) * \sum_k a_k \delta(t - \tau_i(\vec{x}_{p_k})) * \delta(t - \tau_j(\vec{x}_{p_k})) \quad (1.17)$$

In essence, this rearrangement means that all signals involved in the beamforming process could be acquired independently by pairs of emission-reception transducers. Thus, now the focusing process is not tied to any particular position and steering angle, and can be solved for any point in the image. Once all data has been stored, the beamforming operation can be addressed to compensate for the emission and reception delays simultaneously at any point in the region of interest, synthesizing the performance of a full aperture. This process is known as *Synthetic Aperture Focusing Technique* (SAFT) [10], [24], [58], [67] and it has significant implications for the electronic system and signal processing that has to be designed and performed. By considering a linear array of N elements and all combinations of transmitter and receiver data $X_{ij}(t)$, the emitted signal by element i and received by element j, the ultrasonic image at point $\vec{x}$ is obtained by:

$$I(\vec{x}) = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1} X_{ij}\left(\frac{|x|}{c} - (\tau_i(\vec{x}) + \tau_j(\vec{x}))\right) \quad (1.18)$$

As the waveforms are delayed by $\tau_i(\vec{x}) + \tau_j(\vec{x})$ and summed for every combination of emitter-receiver, this beamforming technique is also known as delay-and-sum (DAS).

When all N^2 possible combinations of emitter-receiver pairs are used in the image beamforming, a 3D data structure is created, where one axis corresponds to the emission channel, the second axis is the reception channel and the third one is the acquisition time. This data structure is known as *Full Matrix Array* (FMA) and the procedure to obtain it is usually referred to as *Full Matrix Capture* (FMC). The process of obtaining an image by beamforming over this FMA structure, described in (1.16) and (1.18), is known as the *Total Focusing Method* (TFM) and it is able to produce an image where all points scatterers are focused [22], [43]–[45], [98]. Compared to Figures 1.4(a) and 1.4(b), in Figure 1.4(c) this process has been modelled for the different targets, all of them focused dynamically on emission and reception.

1.2.3 Quality requirements

There is a set of requirements where ultrasonic system technicians and operators agree: an image is better as higher are its *resolution* (i.e. axial and lateral resolution), its *dynamic range* or *signal-to-noise ratio* (SNR) and its *contrast* (ability to detect anechoic areas such as cysts, in a field of strong echoes). It is generally considered that these aspects determine the quality of an ultrasound image [64], [139], [151]. Another crucial requirement is the acquisition and display cadences, which indicates the operability of the ultrasound systems to work in real-time environments.

Resolution

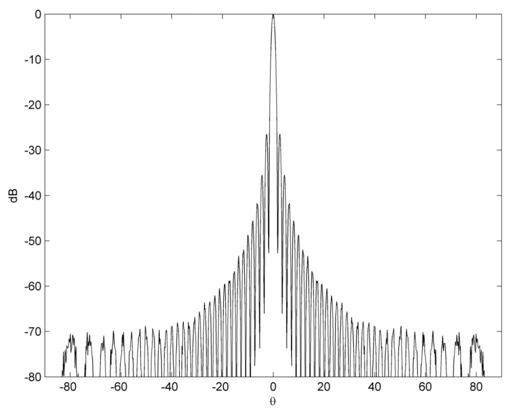
The spatial resolution is defined as the minimum distance at which two reflectors can be positioned to consider them separated. Its value is essential to quantify the ability of the system to spatially resolve small structures and to provide a good edge definition in the image. In 2D imaging, it has two components: the *lateral* resolution and *axial* resolution. The first one is measured perpendicular to the beam propagation direction. To calculate the lateral resolution of a system it is possible to apply different criteria, based on the ratio of the maximum amplitude of the main lobe and the amplitude at which its angular width is measured [70]. In our case, we will use the Rayleigh criterion [70], which establishes that two point sources are distinguishable if the maximum response of one of them is located at the first zero of the response of the other according to this expression:

$$RL = 2\ arcsin\left(\frac{\lambda}{D}\right) = 2\ arcsin\left(\frac{\lambda}{\frac{\lambda}{2}N}\right) = 2\ arcsin\left(\frac{2}{N}\right) \tag{1.19}$$

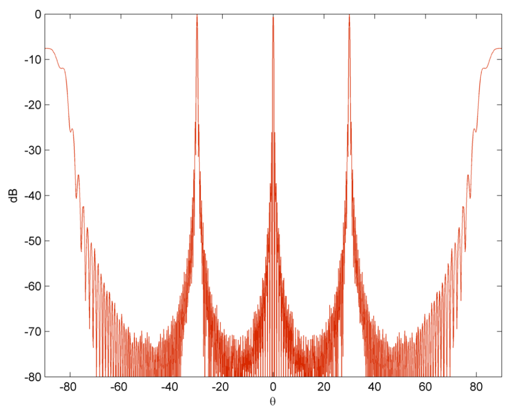
The axial resolution is measured in the ultrasound propagation direction and is proportional to the signal length, which is determined by the bandwidth of the transducer. Its best value would be λ. In the case of 3D imaging it must be added a third component which may be considered an extension of the lateral resolution of 2D images in a direction perpendicular to the image plane.

Extending the distance d between elements increases $D = N{\cdot}d$ and lateral resolution is improved.

If d is greater than $\frac{\lambda}{2}$ and the excitation signal is composed by multiple wavelengths, it is produced a constructive interference within the image field between consecutive cycles of different signals. This creates a replica of the main lobe and a fake lobe is introduced in specific angular distances. This interference lobe is known as *grating lobe*. As a consequence, this results in the appearance of high-energy areas in the images, which cause a strong reduction of their contrast and quality [142].



(a)



(b)

FIGURE 1.5: Beampattern (dB scale) for an steering angle $\theta = 0°$ in water $c = 1480m/s$, $N = 64$ with different element spacing: 1.5(a) $d = \frac{\lambda}{2}$ and 1.5(b) $d = 2\lambda$

As an illustrative example, Figure 1.5(a) shows the beampattern for a linear array $N = 64$ in water with $d = \frac{\lambda}{2}$ calculated at far-field distance. A main lobe of $0dB$ amplitude at $\theta = 0°$ and side lobes of small amplitudes in the other directions are observed. The main lobe is the result of the constructive interference of N signals, while the side lobes appear due to partial destructive interference between the signals emitted by each array element.

When the distance between elements is $d = 2\lambda$, Figure 1.5(b), in addition to the main lobe at $\theta = 0°$, grating lobes appears at $\theta = \pm 30°$ of the same amplitude as the main lobe, what makes impossible to distinguish the origin of the echoes as three beams are generated in different angles. Respect to the width of the main lobe in Figure 1.5(b) it is smaller than Figure 1.5(a) because the aperture D is four times bigger in this case, $D = N \cdot 2\lambda$ versus $D = N \cdot \frac{\lambda}{2}$.

TABLE 1.1: Array size, lateral resolution and number of elements needed for the construction of linear and matrix arrays with different aperture sizes [90]

Array size	Small		Medium		Big	
Aperture (D)	8λ	16λ	32λ	48λ	64λ	128λ
Lateral resolution (RL)	$7.2°$	$3.6°$	$1.8°$	$1.2°$	$0.9°$	$0.°5$
1D array (N)	16	32	64	96	128	256
2D array (N^2)	256	1024	4096	9216	16384	65536

Table 1.1 shows the lateral resolution achieved with different arrays sizes as well as the number of elements required in each case to avoid grating lobes for both 1D and 2D arrays. As it can be observed, the total number of elements necessary to obtain a good resolution is very high, especially in the case of matrix arrays [90].

Contrast

The *contrast* is determined by the ratio between the main lobe amplitude and the sidelobes of the radiation pattern, which are characteristic of the design of each particular array. The contrast is defined as the minimum difference of acoustic impedance of the medium which can be displayed with two different gray-levels in the image [114]. A good contrast is essential in medical imaging where the impedance differences can be pretty small. The main factors which determine the contrast are the spatial resolution, the dynamic range of the system, the level of the sidelobes, and the noise. The detection of objects with high contrast is limited by the spatial resolution while detection of low contrast objects is limited also by the noise of the image.

Signal-to-noise ratio

Besides contrast, there is noise inherent in an acquisition chain (thermal, electrical) which obviously has its implications. The *signal-to-noise ratio* or SNR is the relation between the information contained in the signal versus the noise which interferes with it. Sometimes it is referred as the dynamic range, defined as the ratio between the weakest and the strongest signal which the signal can provide. SNR improves as the amount of signals which are added in the process of beamforming is increased or using encoding techniques [14], [28], [97], [102], [111]. The response of a single transducer with noise can be defined as:

$$x(t) = s(t) + n(t) \tag{1.20}$$

where $s(t)$ represents the data and $n(t)$ represents the noise. The noise is mainly of two types: random electrical noise and grain noise. Electrical noise can be reduced by tending the electronic design. However, grain noise refers to artefacts that appear in the image in different ways, i.e. small variations in the brightness scale or small discontinuities produced by specular reflectors. Then, the SNR is defined as the ratio between the average signal power and the average noise power at the beamformer output, calculated as:

$$SNR_{Channel} = \frac{\bar{s}^2}{\bar{n}^2} \tag{1.21}$$

where $\bar{s}^2$ and $\bar{n}^2$ are, respectively, the mean square values of the signal and the noise measured at this point.

1.2.4 Hardware architecture

Two types of ultrasonic beamforming methods with arrays have been presented: Phased Array and Synthetic Aperture systems. Both are based on the same physical principles, but they differ substantially in the form in which the image is generated. Analysing these two types of imaging systems from the point of view of the hardware architecture there are significant differences which are necessary to remark.

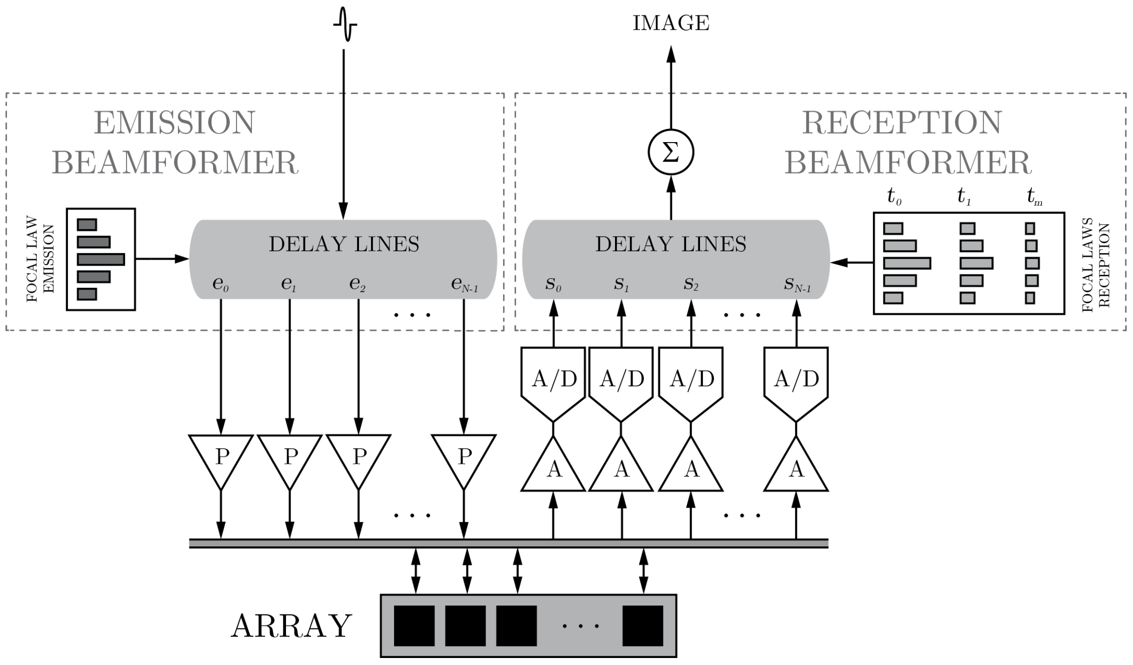


FIGURE 1.6: Parallel beamforming description of a phased array ultrasound system. Generic description of a beamforming system. Two beamforming stages are identified: emission beamforming that introduces delays in the emitted pulses; and the reception beamforming where the data of each channel can be rearranged in real time to compensate the delays in reception

Thus, a generic PA picture of the electronic system needed is represented in Figure 1.6 where it can be seen that it is composed by two processing stages:

1. The *emission beamformer* which is in charge of delaying the transmitted pulse of each transducer to compensate the differences in the path from the transmitters to a determined location or emission focus point according to Equation 1.11.

2. The *reception beamformer* that introduces at each array element a delay-line to compensate the differences in the back-path from a determined location or reception focus point as Equation 1.14 suggests.

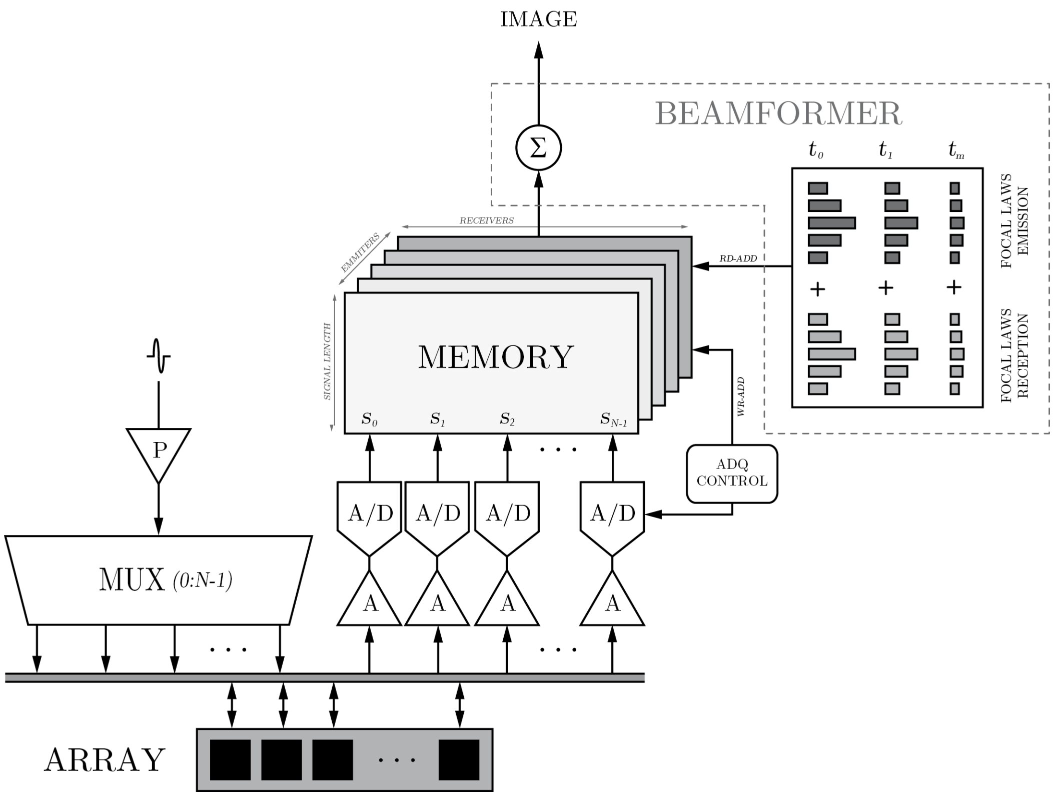


FIGURE 1.7: Generic description of a beamforming system for synthetic aperture imaging, with one pulser multiplexed over N channels and N receiver channels

By contrast, if the Equation 1.16 is applied to the parallel beamforming scheme of Figure 1.6 a significant reduction of the hardware can be achieved as Figure 1.7 shows. Firstly, the net of emission pulsers and the associated circuits for the focusing synchronization are substituted by only one emission pulser and a multiplexer system. Secondly, although in Figure 1.7 the receiver net has been maintained, it can be redesigned to operate with less channels at the cost of increasing the number of emissions needed to produce the image. In an extreme minimum configuration the acquisition process can be done by only one channel, multiplexing the transducers in reception, as it is done in transmission [66]. An illustrative scheme of a general multiplexed SAFT imaging system can be seen in Figure 1.7 where the parallelism can be observed.

Comparison of beamforming systems

Due to the differences between PA and SAFT systems, it is necessary to summarize the main advantages and disadvantages of both imaging modalities. With respect to the system hardware, in the case of PA systems one drawback arises from the need to have an electronic channel for the emission and another for reception, for every single array element. If the array has a small number of elements this problem is perfectly affordable with existing electronic systems. However, this is not usual, since the total number of elements required for a good quality array is high, usually few hundred elements (64 to 128 channels). For this reasons, it is still a complex, expensive and costly task to design an electronic system that allows control of an array with a large number of input-output channels.

SAFT techniques avoid these inconveniences and propose a compromise between the degree of hardware parallelism and signal acquisition procedure. This commitment is also linked to the number of signals used to generate the image. Thus, the simplest case is *Conventional-SAFT* where the maximum reduction is achieved [4], [71] because only one single array element is used both, in transmit and receive. This supposes for a system of N elements a total of N signals to acquire and process. The complex case is when TFM [43], [44], [67] is used. Here, the minimum reduction is obtained because reception hardware remains unchanged respect to a PA system, and N^2 signals are needed. In both configurations N firings are needed, and between these two cases, there are intermediate solutions.

In the case of SAFT systems it is possible to obtain images dynamically focused both on emission and reception. This is because, by emitting with a single element, it is not necessary to implement a real lens to regulate activation instants, as it happens in PA systems. In reception one or more receivers may be employed, but in all cases the received signals contain information for a single emitter. Once all signals have been acquired, an image can be generated dynamically focused at all image pixels. This operating principle of synthetic beamforming [17] offers great versatility respect to PA when composing images. The fact of not being subject to a specific image geometry formed by sectoral lines is also important. Hence, to form the image, whatever its form, we just need to look for the correspondence between its pixels and the spatial points. Then, properly offsetting and summing the signals stored, a focus on emission and reception is achieved in this position, obtaining the image pixel value. In this way, *scan conversion* problems relative to phased array systems disappear [5], [80], [119], [154], [155].

With respect to signal-to-noise ratio, PA systems uses all array elements during emission and therefore, more energy is used and major penetration capacity is achieved. However, in SAFT only few elements are used during emission what results in a significant loss of energy. Solutions to these kind of problems can be resolved using encoding techniques, which allow to increase the energy radiated to the medium improving image quality without increasing significantly the complexity of the systems [14], [28], [97], [111].

In ultrasonic imaging the frame-rate depends on the number of firings employed for acquisition. This fact means that a PA system need as many firings as image lines are desired. For SAFT systems that it is dependent with the number of elements in the aperture N_e and the number of reception channels N_r it is $N_{firings} = N_e * \frac{N_e}{N_r}$. In this sense, this compromise allows to accelerate the acquisition process respect to conventional PA systems.

Regarding acquisition rates, in the case of PA systems the emission focussing is a physical process, where a real wave-front is generated in the medium at a specific depth. However, the reception focussing is a logical process, because the sum of the contributions of every array element is done by properly offsetting the stored signals. On reception, the focus can be an off-line process, which can be on-line if there are electronics capable of delaying and summing the received signal samples as they are acquired. Therefore, to generate an image in a PA system is necessary to emit and receive as many times as there are scan lines in the final image. However, in Synthetic Aperture imaging systems the frame-rate depends on the signal acquisition procedure [17], [82]. Thus, if only one element is used to emit, the total time required is proportional to the size of the array, and not to the number of image lines. In this sense, this compromise allows to accelerate the acquisition process respect to conventional PA systems. As an example, Table 1.2 shows the number of lines and firings for different array sizes, and the frame-rate achieved supposing a water medium with $c = 1.48mm/\mu s$ and a maximum depth of $80mm$ and $N_r = N_e$. This means an acquisition time per line of $\approx 108\mu s$.

Nevertheless, despite of the great advantages of synthetic aperture techniques (reduction of the hardware, maximum image quality, dynamic focusing on emission and reception, acceleration of data capture), SAFT beamforming process (particularly TFM + FMC) involves a high computational cost. It is because, once all the signals have been acquired, it is necessary to handle all the data without introducing additional delays in the system. Thus, computational complexity is increased as more data/second needs to be processed. In addition, a large amount of physical memory is needed to store all the signals acquired by the system.

TABLE 1.2: Required lines, firings and acquisition frame-rate achieved for a $\theta = \pm 45$ scan for linear and matrix arrays in a PA and SAFT systems respectively

Array size	Small		Medium		Big	
Number of elements (N)	16	32	64	96	128	256
Lateral resolution (RL)	$7.2°$	$3.6°$	$1.8°$	$1.2°$	$0.9°$	$0.5°$
2D scanning lines	31	62	125	187	250	450
PA acquisition rate (img/sec)	298	149.3	74	49.5	37	20.5
3D scanning lines	961	3844	15625	34969	62500	202500
PA acquisition rate (vol/sec)	9.6	2.4	0.59	0.26	0.14	0.04
2D firings	16	32	64	96	128	256
SAFT acquisition rate (img/sec)	578	289	144	96	72	36
3D firings	256	1024	4096	9216	16384	65536
SAFT acquisition rate (vol/sec)	36	10	2.26	1	0.56	0.14

1.3 State-of-art

Until now overall concepts about SAFT imaging have been introduced particularly with the TFM method description. This section will be focused on the state-of-art of several optimization methods, and the current situation in terms of synthetic beamforming in ultrasound systems.

1.3.1 Optimization of synthetic aperture beamforming

Synthetic aperture techniques in ultrasound, referred sometimes as *Ultrasonic Synthetic Aperture* (SAU), arised as an attempt to apply on ultrasound systems the processing techniques developed in Synthetic Aperture Radar and Sonar (SAR and SAS) [35], [76], [138]. In such systems, the radar is located on a plane and is in charge of taking successive images of the same object. Once the data capture has finished, all snapshots are combined together in order to generate a final image, which has an equivalent resolution to that obtained with an antenna of size equal to the displacement made by the plane between the first and the latest acquisition.

To study the optimization of SAFT, the main solutions rely on simplifications in the total number of signals involved. They propose different acquisition schemes which produce reductions in hardware and/or the number of firings, thus establishing a trade-off respect to the final quality of the image. Kino et al. studied in the 80's [4], [71] the conventional SAFT which uses single-element subapertures that are activated sequentially to complete the array. Many research groups have worked with it since then. Mayer et al. [96], Busse [11] or Stepinski [143] presented methods for image reconstruction in frequency domain from the data obtained with bidimensional arrays; O'Donnell et al. has applied it to circular arrays to obtain intramuscular ultrasound images [110], and Ylitalo et al. discuss their possible use on different transducers geometries, analysing the achieved signal-to-noise ratio in each case [161]–[164]. Some other interesting works were published by Chiao et al. [13] and Passman et al. [115] and [6], [26], [81], [120], [153] where improvements and new applications were discussed.

However, transmitting and receiving with a single element has the main drawback of a low signal-to-noise ratio and the presence of grating lobes, and for this reason several authors have proposed to increase the size of the subapertures in order to have more radiated energy. The first author was Karaman in 1995 [67] proposing a new technique called *Multi-Element Synthetic Aperture* or M-SAF, where a set of overlapping subapertures in emission and reception. In emission, a defocusing lens is applied, which allows to consider each group of elements as a single virtual element which radiates energy proportional to the size of the subaperture. The idea was developed by Karaman in later works [66], [68] and it has been used by many authors, leading to a variety of techniques. Bilge et al. analysed in [7] a possible technique to correct the distortion introduced in the image during the acquisition process, due to the displacement of scanned objects; Ucar, et al. present a variant of the M-SAF [149]; Chiao et al. analyse the possibility of applying codes to improve the signal to noise ratio achieved [13]. Later, Tavh continue to develop the technique and in [147] a new method for correcting phase distortion is presented. More recently, authors like Bae and Jeong [2] or Robinson and Cooley [121], have been proposed acquisition methods based on the capture of multiple images with each subaperture, where each is focused at a different depth, achieving the resulting final image focused at all depths. In 2002, Gammelmark et al. presented in [28] a correction and compensation for images acquired using this technique. In same years, Johnson et al. published several works where the procedure to interpolate lines with the subimages acquired M-SAF to obtaining a high definition image is presented [60]–[63].

Parallel to the development of M-SAF, other authors have proposed several possible synthetic aperture techniques, each with their particular advantages and disadvantages. Trahey and Nock described in [108], [148] a new technique called *Synthetic Aperture Receive* (SRA) which uses all the elements of the array to emit and a set of subapertures to receive. Following this procedure it is possible to increase the energy radiated to the environment, improving the signal-to-noise ratio, while the cost of the system is reduced. As counterpoint to the SRA, in *Synthetic Transmit Aperture* (STA) subapertures are used in emission and the entire array is used for reception. Lockwood was the first to propose this solution in 90's [82], [84] where a new SAFT technique is presented. Likewise, Holm and Yao present a variant of this technique in [42]. These authors describe a firing scheme based on the emission with non-overlapping and pre-focused subapertures which enable to increase the energy radiated by the system. Nikolov has also published several works presenting a technique based on the STA, which allows to obtain a complete image after each emission [52], [100], [101]. Gran et al. show in [27], [34] experimental results with this technique and new improvements added to it.

In Section 1.2.2 it was introduced the denominated *Synthetic Focusing* (SF) technique by Nikolov [102], *Full Matrix Capture + Total Focusing Method* (FMC+TFM) by professors Wilcox and Drinkwater [43], [45], [98] or Gold-Standard by Jensen [67], which consists on the sequential emission with each one of the array elements in turn, and the reception in each shot with the full transducer aperture. The number of signals combined in this case is much higher compared to other SAFT techniques but it offers the highest quality [58] becoming a standard in ultrasound imaging.

Solutions for two-dimensional arrays and 3D imaging systems have been also proposed. Among the initial works published, Lockwood and Foster in 1996 [83] presented a detailed method where sparse apertures in emission and reception are chosen to minimize the number of signals. In [149] Ucar and Karaman show a possible implementation of the M-SAF for the two-dimensional case, and Lockwood holds a first approximation to the three-dimensional image [39], [85], applying the STA to a system composed of a linear array and a rotational mechanical system. Jensen and Nikolov follow a similar approach in [103] and in successive works [104], [106] both authors propose to apply STA method to a linear array which rotates over the volume to be inspected. Moreover, in [54] they present a new synthetic aperture technique designed to work with matrix transducers. Other interesting works are from Daher et al, and Yen [18], [19], [69]. Furthermore, Yen and Smith present in several works [157]–[160] different alternatives and acquisition methods to design an implement a 3D ultrasonography system in real-time.

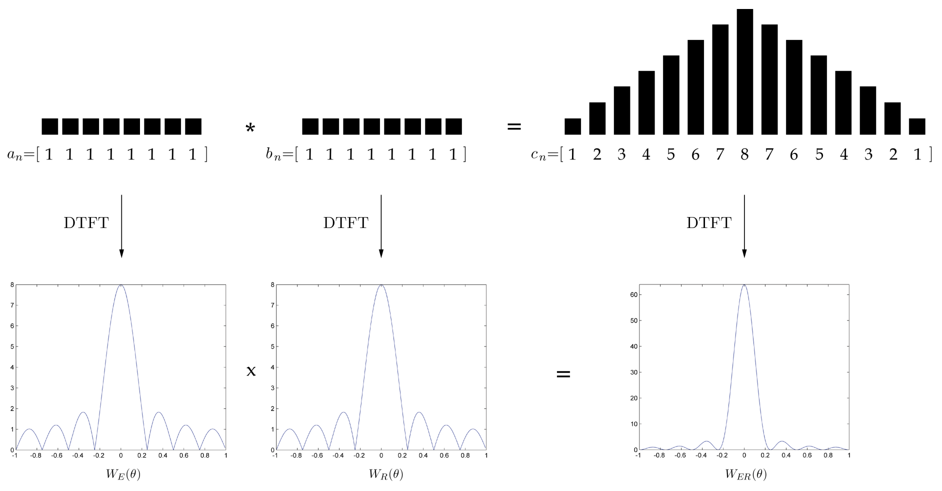
Coarray as a mathematical model

In order to study the equivalence between the phased array parallel beamforming and the synthetic aperture techniques a narrowband model can be used. This model is very useful because it allows for a rapid calculation of the far-field Point Spread Function (PSF) of the aperture. For the case of a linear array of N elements where the distance between them is d, it is well-known that the one-way response can be computed as the Fourier Transform of the aperture [141]:

$$F_E(\theta) = \sum_{i}^{N} exp^{-j\frac{2\pi}{\lambda} idsin(\theta)} \tag{1.22}$$

And the pulse-echo response is determined by the multiplication of emission and reception responses, and assuming that elements are spaced $\frac{\lambda}{2}$ (where λ is the wavelength that corresponds to the excitation frequency) given by the following expression:

$$F_{ER}(\theta) = \left(\sum_{i}^{N} exp^{-j\pi isin(\theta)}\right)\left(\sum_{l}^{N} exp^{-j\pi lsin(\theta)}\right) \tag{1.23}$$



a_n=[1 1 1 1 1 1 1 1]
*
b_n=[1 1 1 1 1 1 1 1]
=
c_n=[1 2 3 4 5 6 7 8 7 6 5 4 3 2 1]
DTFT
DTFT
DTFT
x
=
$W_E(\theta)$
$W_R(\theta)$
$W_{ER}(\theta)$

FIGURE 1.8: Narrowband diffraction pattern and coarray generation in phased array system

Using the relationship between the Fourier transform and the convolution it is possible to demonstrate that this response is equivalent to the one-way response produced by a linear array obtained by the convolution of emission and reception aperture:

$$c[i] = a[i] * b[i] = \sum_{m=0}^{N} a[i-m]b[m] \tag{1.24}$$

where $a[i]$ and $b[i]$ are functions which describe the distribution of the array elements in a $\frac{\lambda}{2}$ grid in emission and reception, and $c[i]$ is the resultant *coarray* distribution that its diffraction response is equivalent to the one produced by the original emission and reception aperture. This process is presented in Figure 1.8.

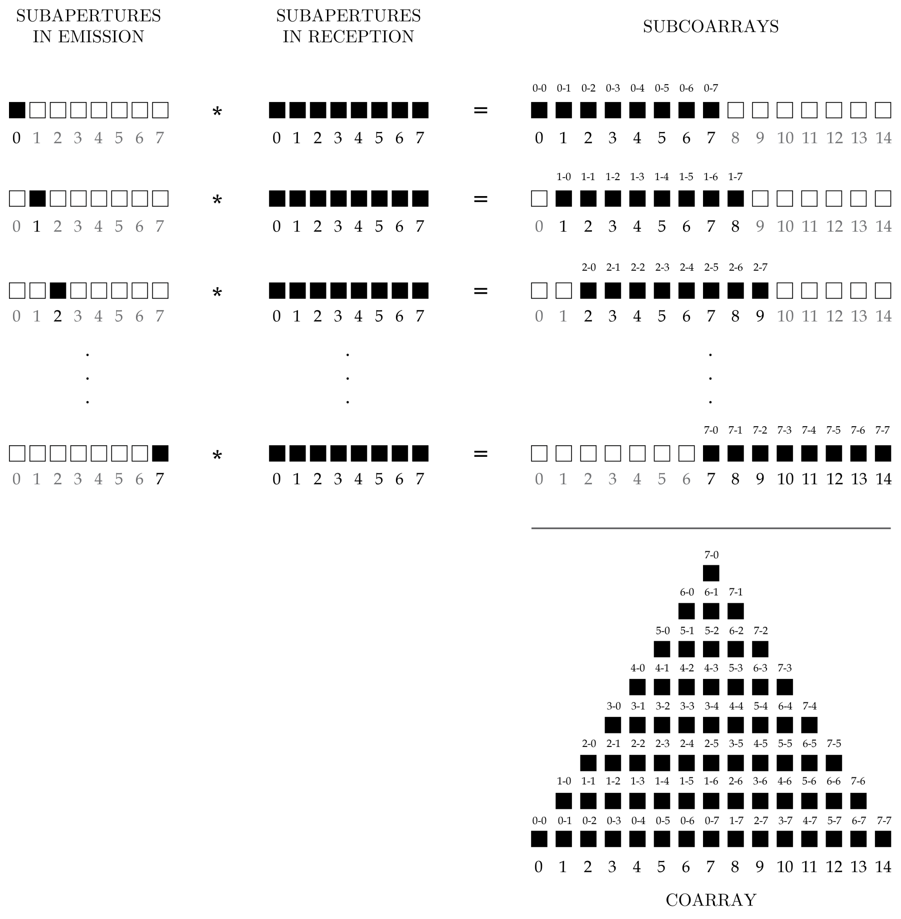


FIGURE 1.9: Shot sequence of the elements for FMC process and its corresponding generated coarray for $N = 8$

If this process is applied in the synthetic aperture scheme proposed by Equation 1.18 the full coarray described for the phased array case is composed progressively, introducing N new elements at each new shot. The more interesting relationship that can be obtained from Equation 1.24 is that the i^{th} position of $c[i]$ is composed by the sum of those elements in $a[k]$ and $b[l]$ that $k + l = i$. This relationship can be used to determine how each emission and reception pair $[k; l]$ is collaborating in the diffraction response of the aperture and, for instance, in the image quality. This can be seen in

Figure 1.8. A schematically diagram to illustrate the coarray generated by FMC can be observed in Figure 1.9, where it is also presented the firing sequence of the elements for $N = 8$.

The FMC beamformer can be seen as a spatial signal processor that exploits the *redundancy* in the signals to generate an image. The advantage of dispose of all the independent signals involved in the process is not limited to the full dynamic focusing. The nature of signals can be studied to reveal another kind of redundancies that can be used to improve the certainty of the image. This is the model in which primarily all compromise solutions designed for SAFT studied by different authors are based.

From the FMC coarray seen in Figure 1.9, it is possible to observe how some of the elements are formed by a single signal (the boundary elements) while the others increase progressively until reaching coarray centre with a value of N elements. Martín-Arguedas et al. works [90]–[94] introduced the concept of *minimum redundancy coarray* (MRC). Thus, MRC is defined as that coarray in which each element is composed of only one signal. Therefore, using the minimum possible number of signals the apertures diffraction properties can be improved by manipulating the gain of the elements.

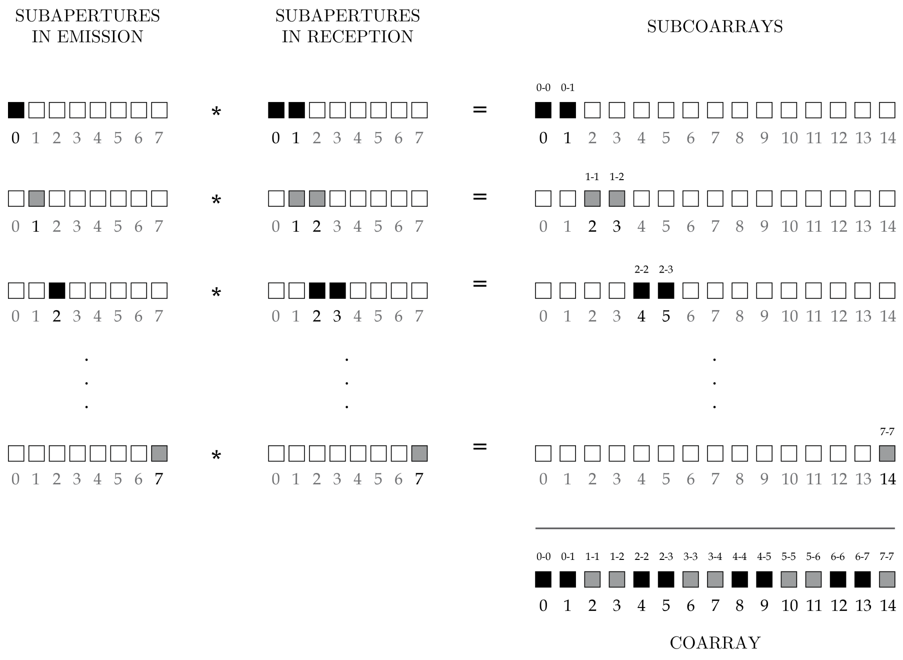
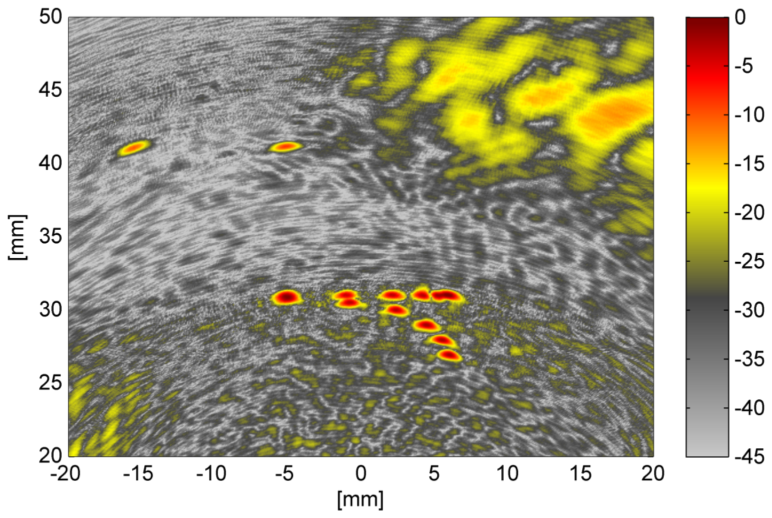
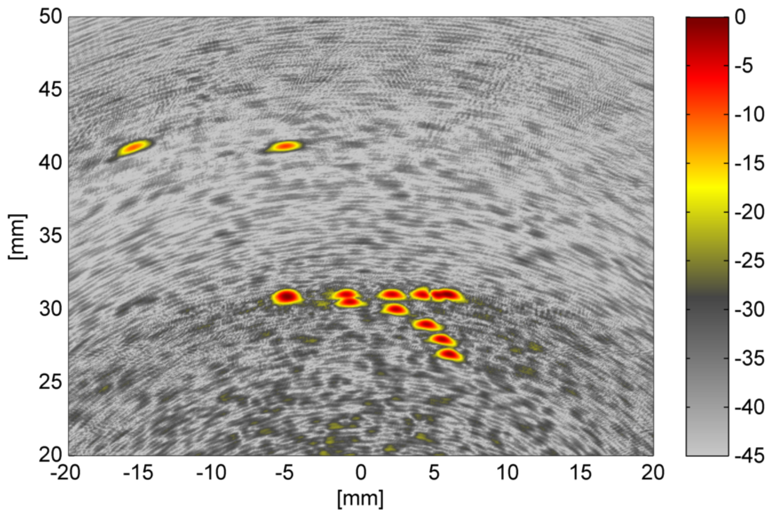


FIGURE 1.10: Shot sequence of the elements for 2R-SAFT technique and its corresponding generated coarray for $N = 8$

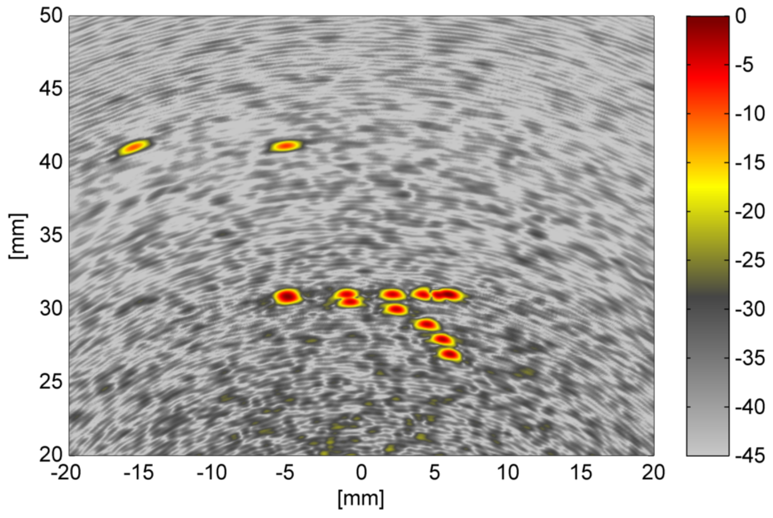
2R-SAFT technique for linear arrays was introduced by Martín-Arguedas in [91] and it has some particular advantages over Conventional-SAFT. 2R-SAFT uses only one element to transmit and two elements to receive. Therefore, all elements are consecutively activated as single emitters and at each shot, two consecutive channels are used as receivers requiring to store two signals per



(a)



(b)



(c)

FIGURE 1.11: Comparative between different SAFT techniques. 1.11(a)Conventional-SAFT with triangular apodization, 1.11(b) 2R-SAFT with triangular apodization and 1.11(c) FMC

emission. Figure 1.10 represents this operation mode. This technique has been reflected in several publications over the years [90], [92]–[94]. Figure 1.11 compares the use of Conventional-SAFT, 2R-SAFT and FMC imaging techniques on a experimental scenario composed by several wires grouped in a triangular area. From that it is observed the presence of grating lobes in Figure 1.11(a) when Conventional-SAFT is employed and how these artefacts are completely removed when the coarray is fully populated as it happens in the case of 2R-SAFT and TFM.

However, MRC techniques have as a main drawback the low dynamic range and signal to noise ratio, which can seriously limit the image contrast due to the low acoustic energy radiated during emission in the medium. This can be seen in Figures 1.11(b) and 1.11(c).

Finally, in Table 1.3 previously described strategies and their operate mode are summarized. Importantly, this refers to acquisition strategies and this lays constraints on how the hardware must be designed. With all these configurations it is possible to obtain both FMA (with enough firings) or MRC datasets.

TABLE 1.3: Description of SAFT acquisition strategies

SAFT strategy	Acquisition operating mode
Conventional-SAFT	One element emits - One element receives (the same)
2R-SAFT	One element emits - Two consecutive elements receive
*n*R-SAFT	One element emits - *n* consecutive elements receive
FMC	One element emits - All elements receive

1.3.2 Development of synthetic aperture imaging systems

In the last years, several studies have been done in parallel computing for ultrasound area, developing different types of solutions both in terms of hardware and/or software for simulation and imaging algorithms.

On one hand, solutions for TFM + FMC imaging algorithm by hardware using Field Programmable Gate Arrays (FPGAs)[8], [87], [102] have been published. Concretely, this has been materialized in 2014 by M2M company, which has launched a real-time TFM system with 64 sensors (4096 signals FMC) capable of achieving, using a set of pre-calculated delays, up to 25 images per second for an image of 256×256 pixels.

The main limitation of synthetic aperture techniques is the high computational cost associated in the order of $\mathcal{O}(N \cdot N \cdot I)$. In this sense, these algorithms are well suited for the SIMD (*Single Instruction Multiple Data*) parallelization model and therefore, alternatively to FPGA's works, there are an extensive number of studies which are focused on developing solutions for other computing platforms such as multi-core CPUs and many-core GPUs.

A brief introduction to heterogeneous computing. Microprocessors based on central processing units (CPUs), e.g. Intel Pentium or AMD families, allowed to reduce the cost and increase the performance in computer applications bringing billions of floating-point operations per second (GFLOPS) to consumer desktops and workstations in last decades. Consequently, software can be more functional and generate more useful results. Since 2003 the increase of the clock frequency within a single CPU has been limited because of energy consumption and heat dissipation issues. Thus, all microprocessor vendors switched to models with multiple processing units or *processor cores*, which are used in the chip in order to increase the computing power.

Traditionally, as von Neumman described in his report in 1945 [99], most of software applications are written as sequential programs. In order to increase their performance and speed, most developers have relied on the advances in hardware, i.e. running the same software in the new generation of processors. However, a sequential program will only run on one of the processor cores which will not become significantly faster, what results in making *parallel programs* in which multiple threads of execution cooperate to complete the work more efficiently. This has been referred in 2005 as the concurrency revolution [145] something which was associated to expensive computers in the past.

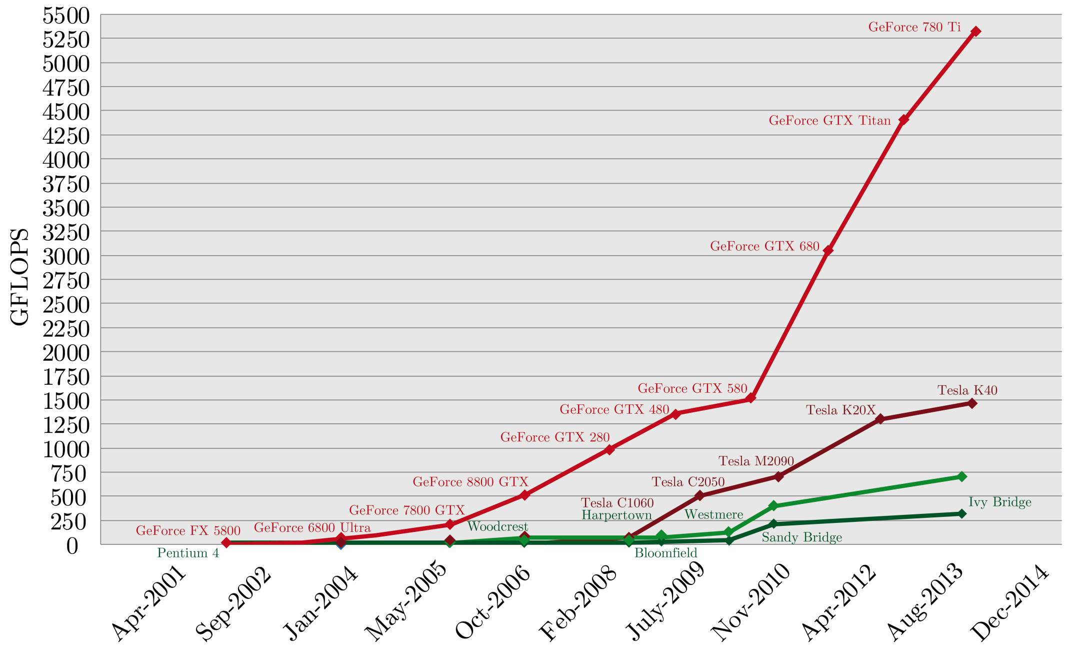


FIGURE 1.12: Theoretical GFLOPS achieved by NVIDIA GPUs in single (red) and double (dark red) precision and Intel CPUs in single (green) and double (dark green) precision

Since 2003, the semiconductor industry has settled on two trajectories for designing microprocessors [47]. On one hand the *multi-core* trajectory, focused in maintaining the execution speed of sequential programs while moving into multiple cores. On the other hand, the *many-thread* trajectory focuses more on the execution throughput of parallel applications.

Many-threads processors, especially the graphics processing units (GPUs), have led the race of floating-point performance. The ratio of peak floating-point calculation throughput between many-thread GPUs and multi-core CPUs is remarkable as we can observe in Figure 1.12. These are not necessarily application speeds, but are merely the raw speed that the execution resources can potentially support in these chips. This peak-performance gap between these two types of processors is due to the differences in the design philosophies between GPUs and CPUs as illustrated in Figure 1.3.2.

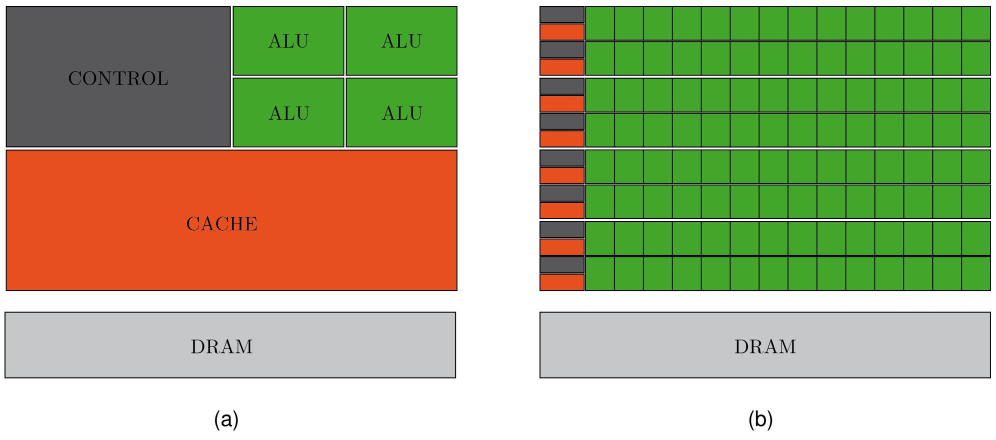


FIGURE 1.13: Multi-core CPUs 1.13(a) and many-thread GPUs 1.13(b) have fundamentally different design philosophies

A CPU is optimized for sequential code performance making use of sophisticated control logic to allow instructions from a single thread to execute in parallel or even out of their sequential order. Moreover, large last-level on-chip caches memories are provided in order to reduce the instruction and data access latencies for large complex applications. This design style is commonly referred to as *latency-oriented* design. Nowadays, it is typically to have multi-core microprocessors from eight to twelve processor cores and multiple megabytes of on-chip cache memories. Therefore, CPUs are designed to minimize the execution latency of a single thread.

On other hand, the design philosophy of GPU is shaped by the fast-growing video game industry which needs to perform a massive number of floating-point calculations per video frame. This demand motivated GPU vendors to maximize the chip area and power, and optimize for the execution throughput of massive numbers of threads. Small cache memories are provided to help control the bandwidth requirements of these applications so that multiple threads that access the same memory data do not need to all go to the DRAM. This design style is usually referred to as *throughput-oriented* design since it is focus on maximizing the total execution throughput of a large number of threads while allowing individual threads to take a potentially much longer time to execute.

Until 2006, graphics chips were very difficult to use, and OpenGL or Direct3D techniques were needed to program these chips. Thus, a computation had to be expressed as a function that paints a pixel in some way to be executed on these early GPUs. This technique was called GPGPU (*General-Purpose Programming using Graphics Processing Units*). Even with a higher-level programming environment, the underlying code still needs to fit into the APIs that are designed to paint pixels limiting the applications could be developed. In 2007, NVIDIA introduced the CUDA (*Compute Unified Device Architecture*) programming model, designed to support joint CPU-GPU execution of an application. This demand is further reflected in more recent programming models such as OpenCL, OpenACC, and C11AMP [49]. CUDA allows GPUs to be programmed directly in C [9], [48], [49], [109], and includes a unified shader pipeline, allowing each and every arithmetic logic unit on the chip to be used by a program intending to perform general-purpose computations.

These platforms can provide solutions in modelling systems and techniques, and the implementation of real-time imaging algorithms.

Parallelization in the development of simulation models

Works on transducer modelling and image simulators have also benefited from these technologies. In this way, ultrasonic techniques make use of different simulation models to obtain some image patterns. Several computational methods have been applied to obtain the response of targets in complex systems. These methods are based on the impulse response (IR) method [15], [55], [59], [116], [117], the finite element analysis [72], [118], optical geometry studies [112], [113] or multi-Gaussian beam models [36]. Using some of them, simulation tools have been developed for evaluation assessment in medical applications or other sophisticated software packages, such as the CIVA program [16] for industrial NDE, or FIELD II simulation library [56].

Parallelization in the optimization of beamforming processes

Some research has been focused in exploring parallel computing strategies both on CPUs and GPUs for the optimization of beamforming processes. CPU beamforming parallelization study began in the 2000 where J. A. Jensen et. al describes in [57] a DAS library for beamforming. With the boom and standardization of the cluster computing also Zhang et. al presents in [165] software processing to improve 2D/3D real-time ultrasound systems. All these previous works were implemented for a single core of CPU and, subsequently, they have still working to develop parallel versions of these libraries to exploit multicore CPUs [37], [75].

With respect to GPU based implementations, they were centred on the use of minimum redundancy coarray [90] for an array of 128 elements (255 signals) and 256 image points, producing a frame-rate of 30 TFM images per second in real time [125]. In succeeding works the frame-rate was increased until 160 img/s [127]. Furthermore, in [38] FMC (16 emision and 64 receptor elements) is considered at fixed focus, achieving 40 images per second for 512×512 pixels using an NVIDIA Fermi with 512 cores. Sutcliffe et. al, shows in [144] a system working in real time (acquisition through an Ethernet connection) with a frame rate of 8 images per second of 120×120 pixels for an array of 64 elements. In 2013 [137] several systems are compared (Xeon X5690 with 6 cores, GPU NVIDIA Fermi with 512 and 448 cores, and GPU ATI 1536 cores) using different developing programming languages (OpenCL, CUDA) achieving up to 5 images per second for 128 elements array. Finally, other works have used GPGPU techniques to implement more complex beamforming techniques, such as Capon Beamforming [1].

1.4 Motivation of the work

This work is a continuation of work done by Martín-Arguedas in [90]. This previous work develops a planning tool for the acquisition signal process for 2D and 3D imaging systems which reduces the cost of acquisition time by the selective reduction of signals. Although his solution opened the door to the development of low-cost portable systems and excellent image definition (resolution), several problems were left for future work. These issues are the high computational cost of generating the image that precludes its development in real-time and the loss of signal to noise ratio which reduced the contrast in the images.

On the basis of the above the motivation of this thesis is to evaluate whether the use of conventional technology (CPUs and GPUs) for the design and development of SAFT systems is possible for commercial environments. This work began in 2009 with the publication in IEEE Ultrasonics Symposium [125], proposing the use of a graphic processing unit to accelerate minimum redundancy beamformers in time domain. Since then, there is an active research line with great potential as it has been evidenced by many researchers in several papers [8], [37], [38], [107], [127], [136], [140].

In this thesis the TFM computational problem is analysed, and subsequently the compromise between hardware, frame-rate and quality of SAFT methodology will be studied in order to improve imaging processes. To summarize this will be addressed by two main objectives:

- Improve ultrasonic imaging generation for real-time applications. This involves three issues: to optimize the acquisition processes minimizing the number of firings and maintaining a reduced hardware commitment; decreasing the number of signals involved in the beamforming process; and to increase the processing speed associated to the SAFT beamforming. The beamforming stage will be focused on exploiting the Delay-and-Sum procedure in time domain. To this end, the capabilities of current parallel processing systems (HPC) are studied in order to develop real-time SAFT beamformers. Several strategies to implement the beamformer in software are studied, both in multi-core CPUs and many-core GPUs.

- Improving the quality of SAFT images. The main objective is to increase the dynamic range of the images and the signal to noise ratio. In order to obtain a compromise between the dynamic range and frame-rate, in this thesis the use of Golay's complementary series [79] and active sensing is evaluated and their integration in the coarray acquisition architectures in order to maintain the same performance and improve image quality.

The work is organized into three sections:

1.4.1 Development of models and simulation tools

This is aimed at developing efficient computational models to allow the development of simulated spaces (*phantoms*) in order to be capable of modelling real situations in a controlled environment for testing new imaging algorithms. Several methods to simulate ultrasound images have already been published, some aim to obtain the best accuracy possible without solving the full wave equation, like Abersim [150], which can accurately simulate an ultrasonic beam through heterogeneous tissues. Other well-known simulation methods are based on linearly calculating the impulse response at all scatterer positions, as is done by Field II [59], although they suffer from significant computational demands and memory intensive, particularly for dynamic images where the simulation grid must be very fine. To overcome these inconveniences, some approaches are based on diverse approximations and the use of parallel computing techniques to accelerate the simulation processes.

We have highlighted [126] and [131] as full-text publications in Chapter 1.4.1 but it also has the following list of related publications in journals and conference proceedings [124], [126], [128], [131], [152].

1.4.2 Parallelization of beamforming methods

The use of parallelism in the ultrasonic generation systems is discussed with a performance evaluation, and several tools are developed for 2D and 3D imaging cases. We describe the main guidelines on how to accomplish the implementation of the fundamental parts of a Synthetic Aperture Beamformer using parallel computing techniques. From an architectural analysis viewpoint, the beamforming algorithm is pretty interesting because it can be seen as a data parallel process, making its implementation on machines with diverse computational and I/O capabilities possible.

We have highlighted [134] and [122] as full-text publications in Chapter 1.4.2 but it also has the following list of related publications in journals and conference proceedings [127], [133], [135].

1.4.3 Strategies for hardware reduction on ultrasound imaging systems

The development of beamforming techniques in GPU is opening two new research fields for ultrasonic imaging. On one hand, the ability to implement in real-time more advanced processing techniques and complex beamforming [1], [12] and on other hand, a new type of imaging techniques is emerging based on what it has been named as *Ultrafast Imaging* [31], [146]. Here, dynamic range problems are studied and our own solutions are proposed to increase acquisition signal procedure and to improve capture frame-rates.

We have highlighted [125], [129] and [133] as full-text publications in Chapter 1.4.3 but it also has the following list of related publications in journals and conference proceedings [32], [73], [95], [123], [130], [132].

1.5 Conclusions and future work

This work has been focused on analysing the role of parallel processing systems and evaluate their ability to generate 2D and 3D images in real-time, both in multi-core configurations (CPU) and graphics processors (GPUs) for synthetic aperture techniques.

The advantages of Total Focusing Method have been clearly exposed along this thesis. Thus, a parallel beamformer design and a performance evaluation has been carried out in both architectures in order to accelerate the image processing. Different parallel schemes have been designed, showing how a proper planning data access and a good use of the resources can improve execution times of the algorithms.

However, the main problem is related to the tremendous amount of data which has to be acquired, transferred and manipulated. If the FMC process is used to obtain the complete set of signals it means we have to process N^2 signals with a computational cost of $\mathcal{O}(N \cdot N \cdot I)$. Thus, it requires more storage resources and processing capability than other techniques, since the data transfer accounts for 60% of the beamforming process. With respect to CPUs, computational times show that CPU parallel algorithms are not yet enough to reach an acceptable frame-rate suitable for real-time use [127], [133]–[135]. In contrast, GPU results in image composition process are promising, as they have proved that it is possible to achieve 25 frames per second for an array of $N = 64$ elements for an image of 512×512 pixels and 65 fps for images of 256×256, what equals or exceeds the results obtained with a commercial FPGA system [87]. The GPU is a cheap and flexible processor which can be easily integrated in current ultrasonics systems. Respect to 3D case in TFM + FMC, the design and implementation process is an extension from 2D case. On a standard PC equipped with a single NVIDIA Quadro 4000 (256 cores), the library is able to calculate $2,097,152$ voxels in ≈ 5 seconds using a matrix transducer with 121 elements when TFM + FMC is applied. Therefore, although the use of HPC techniques help greatly to reduce the computation time respect to conventional implementations the bottleneck is still in the signal acquisition and data transfer procedures.

As the years progressively pass, the development of new protocols and standards for ultra high speed will help to decrease the existing gap between data transferences, trying to reach the limit imposed by physical systems takeover. But it is clear that a reduction of data volume would be desirable.

Therefore, in order to address these issues and to obtain higher image rates without an impact into the image quality, we have developed our own solutions based on coarray model analysis. Therefore, the core of our discussion is focused on how Minimum Redundancy Coarrays are composed and the commitment that remains between acquisition speed and hardware complexity. In this sense, the work has been focused in resolve three issues.

In this regard, an important part of the work has been focused on the problem of FMC and to develop alternative techniques for MRC solutions. These alternatives have a computational cost of $\mathcal{O}(N \cdot I)$ and we have developed procedures to implement beamforming technique TFM + MRC. This has allowed to go from 65 img/sec [122], [134], [135] in FMC to 200 img/sec [32], [95], [123], [125], [130] for a 256×256 under the MRC formula using a Nvidia Quadro K5000 GPU. Nowadays this number is even high, reaching frame-rates of 1000 images per second when MRC techniques are employed.

By other hand, several MRC composition mechanisms has been studied in order to reduce the number of firings at the cost of increase the parallelism. We have named these techniques as FAST-SAFT [73], [133] and they allow to reduce the acquisition time in proportion to the number of channels in reception. As an example, in an concrete configuration, it is possible to acquire a complete image (MRC) with only two firings.

Finally, we have analysed one of the main problems of these techniques, which is the low level of the transmitted signals and the signal losses through the material which implies a reduction in the dynamic range in the images. As a solution, we have introduced pulse compression techniques based on complementary Golay codes has been proposed that requires the use of two different codes in emission, what doubles acquisition time. The conclusions drawn from the coarray study have allowed to design an implementation system which avoids double firing maintaining the frame rate and increasing considerably the quality of the images [129], [132], [133] compared with the TFM + FMC standard and verifying that the results are equivalent.

As future work lines, from a purely technological point of view, research lines are focused on the potential and capabilities of new ARM multi-core systems to run these algorithms. This would open the possibility of implementing low-cost imaging systems taking advantage of the current portable systems (such as tablets).

With respect to ultrasonic imaging issues there are several open research lines. On one hand it is necessary to shift these results to the 2D arrays case with the intention of developing algorithms for 3D imaging. On the other hand, this thesis opens the way for the implementation of *ultrafast imaging* systems, which is a new application field of great potential in medical imaging. Finally, the development of real-time solutions for complex beamforming algorithms with high computational cost in order to develop new image methods.

To sum up, the solutions proposed along this thesis demonstrate that the development of high image quality and compact systems for ultrafast imaging is a real option. They have a great potential and a large field of application in the medical field as well and non-destructive testing area.

Bibliography

[1] J. Asen, J. Buskenes, C.-I. C. Nilsen, A. Austeng, and S. Holm, "Implementing capon beamforming on a gpu for real-time cardiac ultrasound imaging.," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 61, no. 1, pp. 76–85, Jan. 2014, ISSN: 1525-8955. DOI: `10.1109/TUFFC.2014.6689777`. [Online]. Available: `http://www.ncbi.nlm.nih.gov/pubmed/24402897`.

[2] M.-H. Bae and M.-K. Jeong, "A study of synthetic-aperture imaging with virtual source elements in b-mode ultrasound imaging systems," *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 47, no. 6, pp. 1510–1519, 2000.

[3] C. A. Balanis, *Antenna Theory - Analysis and Design*. John Wiley & Sons, 1982.

[4] S. Bennett, D. K. Peterson, D. Corl, and G. S. Kino, "A real-time synthetic aperture digital acoustic imaging system," *Acoustic Imaging*, vol. 10, pp. 669–692, 1982.

[5] A. P. Berkhoff, H. Huisman, J. Thijssen, E. Jacobs, and R. Homan, "Fast scan conversion algorithms for displaying ultrasound sector images.," *Ultrasonic imaging*, vol. 16, no. 2, pp. 87–108, 1994.

[6] E. Biagi, L. Masotti, I. Rossi, and M. Scabia, "Synthetic aperture technique for echographic focusing based on pulse compression," in *IEEE International Ultrasonics Symposium*, vol. 00, 2004, pp. 1425–1428, ISBN: 0780384121.

[7] H. S. Bilge, M. Karaman, and M. O'Donnell, "Motion estimation using common spatial frequencies in synthetic aperture imaging hasan," in *IEEE International Ultrasonics Symposium*, 1996, pp. 1551–1554, ISBN: 0780336151.

[8] M. Birk, A. Guth, M. Zapf, M. Balzer, N. Ruiter, M. Hübner, and J. Becker, "Acceleration of image reconstruction in 3d ultrasound computer tomography: an evaluation of cpu, gpu and fpga computing," in *Conference on Design and Architectures for Signal and Image Processing (DASIP)*, Tampere, 2011, pp. 1–8, ISBN: 978-1-4577-0619-6. DOI: `http://dx.doi.org/10.1109/DASIP.2011.6136856`. [Online]. Available: `http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=6136856\&tag=1`.

[9] A. R. Brodtkorb, “Scientific computing on heterogeneous architectures,” PhD thesis, University of Oslo, 2010, p. 198.

[10] C. B. Burckhardt, P.-A. Grandchamp, and H. Hoffman, “An experimental 2 mhz synthetic aperture sonar system intended for medical use,” *IEEE Transactions on Sonics and Ultrasonics*, vol. 21, no. 1, pp. 1–6, 1974.

[11] L. J. Busse, “Three-dimensional imaging using a frequency-domain synthetic aperture focusing technique,” *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 39, no. 2, pp. 174–9, Jan. 1992, ISSN: 0885-3010. DOI: 10.1109/58.139112. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18263134.

[12] J. Camacho, M. Parrilla, and C. Fritsch, “Phase coherence imaging,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 56, no. 5, pp. 958–74, May 2009, ISSN: 1525-8955. DOI: 10.1109/TUFFC.2009.1128. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21622056.

[13] R. Y. Chiao and L. J. Thomas, “Aperture formation on reduced-channel arrays using the transmit-receive apodization matrix,” in *IEEE International Ultrasonics Symposium*, 1996, pp. 1567–1571, ISBN: 0780336151.

[14] R. Y. Chiao, L. J. Thomas, and S. D. Silverstein, “Sparse array imaging with spatially-encoded transmits,” in *IEEE International Ultrasonics Symposium*, vol. 2, 1997, pp. 1679–1682, ISBN: 0780341538.

[15] J.-H. L. S.-W. Choi, “A parametric study of ultrasonic beam profiles for a linear phased array transducer,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, p. 644, 2000.

[16] *Civa: simulation software for non destructive testing*. [Online]. Available: http://www-civa.cea.fr.

[17] C. R. Cooley and B. S. Robinson, “Synthetic focus imaging using partial datasets,” in *IEEE International Ultrasonics Symposium*, 1994, pp. 1539–1542.

[18] N. M. Daher and J. T. Yen, “Rectilinear 3-d ultrasound imaging using synthetic aperture techniques,” in *IEEE International Ultrasonics Symposium*, vol. 00, 2004, pp. 1270–1273, ISBN: 0780384121.

[19] N. M. Daher and J. T. Yen, “2-d array for 3-d ultrasound imaging using synthetic aperture techniques,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 53, no. 5, pp. 912–924, 2006.

[20] R. E. Davidsen and S. W. Smith, “Two-dimensional arrays for medical ultrasound using multilayer flexible circuit interconnection,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 45, no. 2, pp. 338–348, 1998.

[21] J.-L. Dillenseger, S. Laguitton, and E. Delabrousse, “Fast simulation of ultrasound images from a ct volume,” *Computers in biology and medicine*, vol. 39, no. 2, pp. 180–6, 2009, ISSN: 1879-0534. DOI: 10.1016/j.compbiomed.2008.12.009. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19181311.

[22] B. W. Drinkwater and P. D. Wilcox, “Ultrasonic arrays for non-destructive evaluation: a review,” *NDT & E International*, vol. 39, no. 7, pp. 525–541, Oct. 2006, ISSN: 09638695. DOI: 10.1016/j.ndteint.2006.03.006. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0963869506000272.

[23] K. R. Erikson, F. J. Fry, and J. P. Jones, “Ultrasound in medicine-a review,” *IEEE Transactions on Sonics and Ultrasonics*, vol. 21, no. 3, pp. 144–170, 1974. DOI: 10.1109/T-SU.1974.29810.

[24] J. J. Flaherty, K. R. Erikson, and V. M. Lund, *Synthetic aperture ultrasonic imaging systems*, 1967.

[25] G. Fleury and C. Gondard, “Improvements on ultrasonic inspections through the use of piezo-composite transducers,” in *European Conference on Non Destructive Testings*, 1995, pp. 4–7.

[26] C. H. Frazier and W. D. O’Brien, “Synthetic aperture techniques with a virtual source element,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 45, no. 1, pp. 196–207, 1998.

[27] G. Fredrik and J. J. Arendt, “Frequency division transmission imaging and synthetic aperture reconstruction,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 53, no. 5, pp. 900–11, May 2006, ISSN: 0885-3010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16764445.

[28] K. L. Gammelmark and J. A. Jensen, “Experimental study of convex coded synthetic transmit aperture imaging,” in *IEEE International Ultrasonics Symposium*, vol. 00, 2002, pp. 1613–1617, ISBN: 0780375823.

[29] H. Gao, H. F. Choi, P. Claus, S. Boonen, S. Jaecques, G. H. Van Lenthe, G. Van der Perre, W. Lauriks, and J. D’hooge, “A fast convolution-based methodology to simulate 2-d/3-d cardiac ultrasound images,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 56, no. 2, pp. 404–9, 2009, ISSN: 1525-8955. DOI: 10.1109/TUFFC.2009.1051. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19251529.

[30] J. F. Gelly and C. Maerfeld, “Proceedings for a 2d multiplexed array for acoustic imaging,” in *IEEE International Ultrasonics Symposium*, 1981, pp. 685–689.

[31] J.-L. Gennisson, J. Provost, T. Deffieux, C. Papadacci, M. Imbault, M. Pernot, and M. Tanter, “4-d ultrafast shear-wave imaging,” *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 62, no. June, pp. 1059–1065, 2015, ISSN: 0885-3010. DOI: `10.1109/TUFFC.2014.006936`.

[32] C. d. M. Giraldo, G. Lambert, O. Martínez-Graullera, L. Elvira, D. Romero-Laorden, L. Gómez-Ullate, and F. M. D. Espinosa, *Sistema de control de calidad ultrasonico on-line, especialmente aplicable a procesos de fabricación de estructuras de materiales compuestos de fibra de carbono y con curado fuera de autoclave*, 2012.

[33] O. Goksel and S. E. Salcudean, “B-mode ultrasound image simulation in deformable 3-d medium,” *IEEE Transactions on Medical Imaging*, vol. 28, no. 11, pp. 1657–69, 2009, ISSN: 1558-254X. DOI: `10.1109/TMI.2009.2016561`. [Online]. Available: `http://www.ncbi.nlm.nih.gov/pubmed/19278928`.

[34] F. Gran and J. A. Jensen, “Directional velocity estimation using a spatio-temporal encoding technique based on frequency division for synthetic transmit aperture ultrasound,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 53, no. 7, pp. 1289–99, Jul. 2006, ISSN: 0885-3010. [Online]. Available: `http://www.ncbi.nlm.nih.gov/pubmed/16889336`.

[35] H. D. Griffiths, “Synthetic aperture imaging with sonar and radar: a comparison,” in *World Congress on Ultrasonics*, Paris, 2003, pp. 511–518.

[36] L. W. S. J. H.-J. Kim and A. Sedov, “Generation of the basis sets for multi-gaussian ultrasonic beam models: an overview,” *J. Acoust. Soc. Am.*, pp. 1971–1978, 2006.

[37] J. M. Hansen, M. C. Hemmsen, and J. A. Jensen, “An object-oriented multi-threaded software beamformation toolbox,” in *SPIE Medical Imaging: Ultrasonic Imaging, Tomography, and Therapy*, J. D’hooge and M. M. Doyley, Eds., Mar. 2011, 79680Y–79680Y–9. DOI: `10.1117/12.878178`. [Online]. Available: `http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1350000`.

[38] J. M. Hansen, D. Schaa, and J. A. Jensen, “Synthetic aperture beamformation using the gpu,” in *IEEE International Ultrasonics Symposium*, Orlando, Florida, 2011.

[39] C. R. Hazard and G. R. Lockwood, “Effects of motion on a synthetic aperture beamformer for real-time 30 ultrasound,” in *IEEE International Ultrasonics Symposium*, 1999, pp. 1221–1224, ISBN: 0780357221.

[40] T. Hergum, S. Langeland, E. W. Remme, and H. Torp, “Fast ultrasound imaging simulation in k-space,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 56, no. 6, pp. 1159–67, Jun. 2009, ISSN: 1525-8955. DOI: `10.1109/TUFFC.2009.1158`. [Online]. Available: `http://www.ncbi.nlm.nih.gov/pubmed/19574124`.

[41] H. H. J. Holländer, "40 years of real-time ultrasound diagnostics," *Ultraschall in Medicin*, vol. 27, no. 3, pp. 214–216, 2006. DOI: 10.1055/s-2006-947050.

[42] S. Holm and H. Yao, "Improved framerate with synthetic transmit aperture imaging using prefocused subapertures," in *IEEE International Ultrasonics Symposium*, 1997, pp. 1535–1538, ISBN: 0780341538.

[43] C. Holmes, Bruce W. Drinkwater, and P. D. Wilcox, "Post-processing of the full matrix of ultrasonic transmitreceive array data for non-destructive evaluation," *NDT & E International*, vol. 38, no. 8, pp. 701–711, Dec. 2005, ISSN: 09638695. DOI: 10.1016/j.ndteint.2005.04.002. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0963869505000721.

[44] C. Holmes, B. W. Drinkwater, and P. D. Wilcox, "Advanced post-processing for scanned ultrasonic arrays: application to defect detection and classification in non-destructive evaluation," *Ultrasonics*, vol. 48, no. 6-7, pp. 636–42, Nov. 2008, ISSN: 1874-9968. DOI: 10.1016/j.ultras.2008.07.019. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18823924.

[45] A. J. Hunter, B. W. Drinkwater, and P. D. Wilcox, "The wavenumber algorithm for full-matrix imaging using an ultrasonic array.," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 55, no. 11, pp. 2450–62, Nov. 2008, ISSN: 1525-8955. DOI: 10.1109/TUFFC.952. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19942535.

[46] C. Huygens, *Traité de la lumière*. 1690.

[47] W.-M. Hwu, K. Keutzer, and T. G. Mattson, "The concurrency challenge," *IEEE Design and Test of Computers*, pp. 312–321, 2008.

[48] W.-M. W. Hwu, *GPU Computing Gems Emerald Edition*. 2011, p. 889, ISBN: 9780123814722.

[49] W.-M. W. Hwu and D. B. Kirk, *Programming Massively Parallel Processors : A Hands-on Approach*, 2nd. 2012, p. 519, ISBN: 9780123814722.

[50] A. Ibáñez, C. Fritsch, M. Parrilla, and J. Villazón-Terrazas, "Monochromatic transfer matrix method for acoustic field simulation thorough media boundaries," *Physics Procedia*, vol. 3, no. 2009, pp. 883–890, 2010, ISSN: 18753884. DOI: 10.1016/j.phpro.2010.01.113.

[51] A. Ibáñez, M. Parrilla-Romero, and J. Villazón-Terrazas, "Simulation of ultrasonic continuous wave fields in homogeneous media with soft curved interfaces," *ECNDT*, pp. 1–6, 2006.

[52] N. S. Ivanov and J. J. Arendt, "Velocity estimation using synthetic aperture imaging," in *IEEE International Ultrasonics Symposium*, vol. 2, Ieee, 2001, pp. 1409–1412, ISBN: 0-7803-7177-1. DOI: 10.1109/ULTSYM.2001.991985. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=991985.

[53] ——, “In-vivo synthetic aperture flow imaging in medical ultrasound,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 50, no. 7, pp. 848–856, 2003.

[54] ——, “Investigation of the feasibility of 3d synthetic aperture imaging,” in *IEEE International Ultrasonics Symposium*, vol. 00, 2003, pp. 1903–1906, ISBN: 0780379225.

[55] J. A. Jensen, “A model for the propagation and scattering of ultrasound in tissue,” *Journal of the Acoustical Society of America*, pp. 182–190, 1991.

[56] ——, “Field: a program for simulating ultrasound systems,” *Med. Biol. Eng. Comput.*, pp. 351–353, 1996.

[57] J. A. Jensen and S. I. Nikolov, “Fast simulation of ultrasound images,” *2000 IEEE Ultrasonics Symposium. Proceedings. An International Symposium (Cat. No.00CH37121)*, vol. 2, pp. 1721–1724, 2000. DOI: 10.1109/ULTSYM.2000.921654. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=921654.

[58] J. A. Jensen, S. I. Nikolov, K. L. k. Gammelmark, and M. H. g. Pedersen, “Synthetic aperture ultrasound imaging,” *Ultrasonics*, vol. 44, pp. 5–15, Dec. 2006, ISSN: 1874-9968. DOI: 10.1016/j.ultras.2006.07.017. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16959281.

[59] J. A. Jensen and N. B. Svendsen, “Calculation of pressure fields from arbitrarily shaped, apodized, and excited ultrasound transducers,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 39, no. March, pp. 262–267, 1992.

[60] J. A. Johnson, M. Karaman, and B. T. Khuri-Yakub, “Synthetic phased array image formation and restoration,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002, pp. 2885–2888.

[61] J. A. Johnson, M. Karaman, and B. T. Khuri-Yakub, “Coherent-array imaging using phased subarrays. part i: basic principles,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 52, no. 1, pp. 37–50, Jan. 2005, ISSN: 0885-3010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/15742561.

[62] J. A. Johnson, O. Oralkan, S. Ergun, U. Demirci, M. Karaman, and B. T. Khuri-Yakub, “Phased subarray imaging for low-cost, wideband coherent array imaging,” in *IEEE International Ultrasonics Symposium*, vol. 00, 2003, pp. 1875–1878, ISBN: 0780379225.

[63] J. A. Johnson, O. Oralkan, S. Ergun, U. Demirci, M. Karaman, and B. T. Khuri-Yakub, “Coherent array imaging using phased subarrays. part ii: simulations and experimental results,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 52, no. 1, pp. 51–64, 2005.

[64] R. Johnson, “Contrast response analysis for medical ultrasound imaging,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 44, no. 4, pp. 805–809, 1997, ISSN: 0885-3010. DOI: 10.1109/58.655195. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=655195.

[65] Jordi Glez-Llinas, “Paralelización de los modelos de difracción para el cálculo del campo acústico,” PhD thesis, 2013, p. 85.

[66] M. Karaman, H. S. Bilge, and M. O’Donnell, “Adaptive multi-element synthetic aperture imaging with motion and phase aberration correction,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 45, no. 4, pp. 1077–87, 1998, ISSN: 0885-3010. DOI: 10.1109/58.710591. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18244263.

[67] M. Karaman, P.-C. Li, and M. O’Donnell, “Synthetic aperture imaging for small scale systems,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 42, no. 3, pp. 429–442, May 1995, ISSN: 08853010. DOI: 10.1109/58.384453. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=384453.

[68] M. Karaman and M. O’Donnell, “Subaperture processing for ultrasonic imaging,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 45, no. 1, pp. 126–135, 1998.

[69] K.-S. Kim and T.-K. Song, “High volume rate 3-d ultrasound imaging using cross array based on synthetic transmit focusing,” in *IEEE International Ultrasonics Symposium*, vol. 00, 2004, pp. 1409–1412, ISBN: 0780384121.

[70] G. S. Kino, *Acoustic Waves: Devices, Imaging, and Analog Signal Processing*. Englewoods Cliffs, NJ: Prentice Hall Inc., 1987.

[71] G. S. Kino, D. Corl, S. Bennett, and D. K. Peterson, “Real time synthetic aperture imaging system,” in *IEEE International Ultrasonics Symposium*, 1980, pp. 722–731.

[72] N. N. Kishore, I. Sridhar, and N. G. R. Iyengar, “Finite element modelling of the scattering of ultrasonic waves by isolated flaws,” *NDT & E international*, pp. 297–305, 2000.

[73] C. Kitano, O. Martínez-Graullera, D. Romero-Laorden, R. T. Higut, V. T. Prado, and S. C. G. Granja, “Synthetic aperture imaging for ultrasonic non-destructive testing,” in *Signals and Images: Advances and Results in Speech, Estimation, Compression, Recognition, Filtering, and Processing*, C. C. C. Rosângela Fernandes Coelho, Vitor Heloiz Nascimento, Ricardo Lopes de Queiroz, João Marcos Travassos Romano, Ed., CRC Press, 2014.

[74] T. Kojima, “Matrix array transducer and flexible matrix array transducer,” in *IEEE International Ultrasonics Symposium*, 1986, pp. 649–654.

[75] J. Kortbek, S. I. Nikolov, and J. A. Jensen, "Effective and versatile software beamformation toolbox jacob," in *Medical Imaging 2007: Ultrasonic Imaging and Signal Processing*, S. Y. Emelianov and S. A. McAleavey, Eds., Mar. 2007. DOI: 10.1117/12.702996. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1299780.

[76] J. J. Kovaly, *Synthetic Aperture Radar*. Dedham, Massachusetts: Artech House Publishers, 1976.

[77] W. E. Krause, O. H. Kresse, and R. E. Soldner, *Ultrasound diagnostic apparatus*, 1969. [Online]. Available: http://www.google.com/patents/US3470868.

[78] J. Krautkrämer and H. Krautkrämer, *Ultrasonic Testing of Materials*, Third Edit. Springer, 1983, ISBN: 978-3-662-02359-4.

[79] J. B. Kruskal, "Golay's complementary series," *IRE Transactions on Information Theory*, pp. 273–276, 1961.

[80] M. H. Lee, J.-H. Kim, and S. B. Park, "Analysis of a scan conversion algorithm for a real-time sector scanner," *IEEE Transactions on Medical Imaging*, vol. 5, no. 2, pp. 96–105, 1986.

[81] M.-L. Li, W.-J. Guan, and P.-C. Li, "Technique with applications in high-frequency," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 51, no. 1, pp. 63–70, 2004.

[82] G. R. Lockwood and F. S. Foster, "Design of sparse array imaging systems," in *IEEE International Ultrasonics Symposium*, 1995, pp. 1237–1243, ISBN: 0780329406.

[83] G. R. Lockwood and F. S. Foster, "Optimizing the radiation pattern of sparse periodic two-dimensional arrays," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 43, no. 1, pp. 15–19, 1996, ISSN: 0885-3010. DOI: http://dx.doi.org/10.1109/58.484458. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=484458.

[84] G. R. Lockwood, P.-c. Li, M. O'Donnell, and F. S. Foster, "Optimizing the radiation pattern of sparse periodic linear arrays," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 43, no. 1, pp. 7–14, 1996, ISSN: 0885-3010. DOI: http://dx.doi.org/10.1109/58.484457. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=484457.

[85] G. R. Lockwood, J. R. Talman, and S. S. Brunke, "Real-time 3-d ultrasound imaging using sparse synthetic aperture beamforming," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 45, no. 4, pp. 980–8, 1998, ISSN: 0885-3010. DOI: 10.1109/58.710573. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18244252.

[86] L. C. Lynnworth, “Industrial applications of ultrasound - a review ii. measurements , tests , and process control using low-intensity ultrasound,” *IEEE Transactions on Sonics and Ultrasonics*, vol. 22, no. 2, pp. 71–101, 1974.

[87] M2M, *Gekko: advanced phased-array ut*, 2014.

[88] A. Macovski, *Ultrasonic array for reflection imaging*, 1975. [Online]. Available: `https://www.google.com/patents/US3918024`.

[89] M. G. Maginness, J. D. Plummer, W. L. Beaver, and J. D. Meindl, “State-of-the-art in two-dimensional ultrasonic transducer array technology,” *Medical physics*, vol. 3, no. 5, pp. 312–318, 1976.

[90] C. J. Martín-Arguedas, “Técnicas de apertura sintética para la generación de imagen ultrasónica,” PhD thesis, Universidad de Alcalá, 2010, p. 256, ISBN: 978-84-693-2364-9. DOI: `http://hdl.handle.net/10261/27225`. [Online]. Available: `http://digital.csic.es/handle/10261/27225`.

[91] C. J. Martín-Arguedas, O. Martínez-Graullera, L. Gómez-Ullate, A. Octavio, and G. Godoy, “Reduction of grating lobes in saft images,” in *IEEE International Ultrasonics Symposium*, Beijing, China, 2008, pp. 721–724. DOI: `10.1109/ULTSYM.2008.0172`.

[92] C. J. Martín-Arguedas, O. Martínez-Graullera, D. Romero-Laorden, R. Tokio Higuti, and L. Gómez-Ullate, “Linear scanning method based on the saft coarray,” in *36th Annual Review of Progress in Quantitative Nondestructive Evaluation*, Kingston (Rhode Island), 2009, pp. 2023–2030, ISBN: 978-0-7354-0748-0. DOI: `http://dx.doi.org/10.1063/1.3362366`.

[93] C. J. Martín-Arguedas, O. Martínez-Graullera, D. Romero-Laorden, M. Pérez-Lopez, and L. Gómez-Ullate, “Improvement of synthetic aperture techniques by means of the coarray analysis,” in *The International Congress on Ultrasonics*, Gdask, Poland, 2011, pp. 189–192, ISBN: 978-0-7354-1019-0. DOI: `http://link.aip.org/link/doi/10.1063/1.3703168`. [Online]. Available: `http://proceedings.aip.org/resource/2/apcpcs/1433/1/189\_1?isAuthorized=no`.

[94] C. J. Martín-Arguedas, D. Romero-Laorden, O. Martínez-Graullera, M. Pérez-Lopez, and L. Gómez-Ullate, “Estrategias de apertura sintética basadas en el coarray para la generación de imagen ultrasónica,” in *12 Congreso Español de END*, Valencia, 2011, ISBN: 978-84-694-4076-6.

[95] C. J. Martin-Arguedas, D. Romero-Laorden, O. Martinez-Graullera, M. Perez-Lopez, and L. Gomez-Ullate, “An ultrasonic imaging system based on a new saft approach and a gpu beamformer,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 59, no. 7, pp. 1402–1412, 2012, ISSN: 0885-3010. DOI: `http://dx.doi.org/10.1109/TUFFC.2012.2341`.

[96] K. Mayer, R. Marklein, K. Langenberg, and T. Kreutter, “Three-dimensional imaging system based on fourier transform synthetic aperture focusing technique,” *Ultrasonics*, vol. 28, pp. 241–255, 1990.

[97] T. X. Misaridis, “Ultrasound imaging using coded signals,” PhD thesis, 2001, p. 228.

[98] L. Moreau, B. W. Drinkwater, and P. D. Wilcox, “Transmission cycles for rapid nondestructive evaluation,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 56, no. 9, pp. 1932–1944, 2009.

[99] J. von Neumann, “First draft of a report on the edvac,” Tech. Rep., 1945.

[100] S. Nikolov, K. Gammelmark, and J. A. Jensen, “Velocity estimation using recursive ultrasound imaging and spatially encoded signals,” in *IEEE International Ultrasonics Symposium*, vol. 2, Ieee, 2000, pp. 1473–1477, ISBN: 0-7803-6365-5. DOI: `10.1109/ULTSYM.2000.921602`. [Online]. Available: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=921602`.

[101] S. Nikolov and K. L. k. Gammelmark, “Recursive ultrasound imaging svetoslav,” in *IEEE International Ultrasonics Symposium*, 1999, pp. 1621–1625, ISBN: 0780357221.

[102] S. I. Nikolov, “Synthetic aperture tissue and flow ultrasound imaging,” PhD thesis, Technical University of Denmark, 2001, p. 333.

[103] S. I. Nikolov and J. A. Jensen, “3d synthetic aperture imaging using a virtual source element in the elevation plane,” in *IEEE International Ultrasonics Symposium*, 2000, pp. 1743–1747, ISBN: 0780363655.

[104] S. I. Nikolov, J. A. Jensen, R. Dufait, and A. Schoisswohl, “Three-dimensional real-time synthetic aperture imaging using a rotating phased array transducer,” in *IEEE International Ultrasonics Symposium*, vol. 00, 2002, pp. 1585–1588, ISBN: 0780375823.

[105] S. I. Nikolov, J. Kortbek, and J. A. Jensen, “Practical applications of synthetic aperture imaging,” *Ultrasonics Symposium, 2010 IEEE*, pp. 350–358, 2010.

[106] S. I. Nikolov, B. G. Tomov, F. Gran, and J. A. Jensen, “Further development of synthetic aperture real-time 3d scanning with a rotating phased array,” in *IEEE International Ultrasonics Symposium*, vol. 00, 2003, pp. 1899–1902, ISBN: 0780379225.

[107] C.-I. C. Nilsen and I. Hafizovic, “Digital beamforming using a gpu,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Ieee, Apr. 2009, pp. 609–612, ISBN: 978-1-4244-2353-8. DOI: `10.1109/ICASSP.2009.4959657`. [Online]. Available: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4959657`.

[108] L. F. Nock and G. E. Trahey, "Synthetic receive aperture imaging with phase correction for motion and for tissue inhomogeneities. i. basic principles," *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 39, no. 4, pp. 489–95, Jan. 1992, ISSN: 0885-3010. DOI: `10.1109/58.148539`. [Online]. Available: `http://www.ncbi.nlm.nih.gov/pubmed/18267660`.

[109] NVIDIA, *CUDA C Programming Guide 6.0*, February 2014. 2014, p. 240. [Online]. Available: `www.nvidia.com`.

[110] M. O'Donnell and L. J. Thomas, "Efficient synthetic aperture imaging from a circular aperture with possible application to catheter-based imaging," *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 39, no. 3, pp. 366–380, 1992.

[111] M. O'Donnell and Y. Wang, "Coded excitation for synthetic aperture ultrasound imaging," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 52, no. 2, pp. 171–6, Feb. 2005, ISSN: 0885-3010. [Online]. Available: `http://www.ncbi.nlm.nih.gov/pubmed/15801306`.

[112] J. A. Ogilvy, "Computerized ultrasonic ray tracing in austenitic steel," *NDT Int*, pp. 67–77, 1985.

[113] A. L. P. Calmon and L. Paradis, "Modeling of ultrasonic fields and their interaction with defects," *Proc. IEEE Ultrason. Symp*, pp. 779–783, 1998.

[114] M. Parrilla, "Conformación de haces ultrasónicos mediante muestreo selectivo con codificación delta," PhD thesis, Universidad Politecnica de Madrid, 2004, p. 245.

[115] C. Passmann and H. Ermert, "A 100-mhz ultrasound imaging system for dermatologic and ophthalmologic diagnostics," *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 43, no. 4, pp. 545–552, 1996.

[116] P. A. Pinar Crombie and R. S.C.Cobbold, "Calculating the pulsed response of linear arrays accuracy versus computational efficiency," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, p. 997, 1997.

[117] B. Piwakowski and K. Sbai, "A new approach to calculate the field radiated from arbitrarily structured," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 46, no. 2, pp. 422–440, 1999.

[118] H. L. R. Lerch and H. T. Kaarmann, "Finite element modeling of the pulse-echo behavior of ultrasound transducers," *Proc. IEEE Ultrason. Symp., Cannes*, pp. 1021–1025, 1994.

[119] W. D. Richard and R. M. Arthur, "Real-time ultrasonic scan conversion via linear interpolation of oversampled vectors," *Ultrasonic Imaging*, vol. 16, no. 2, pp. 109–123, 1994. DOI: `10.1177/016173469401600204`.

[120] M. C. Robini, T. Rastello, D. Vray, and I. Magnin, “Space-variant deconvolution for synthetic aperture imaging using simulated annealing,” in *International Conference on Image Processing*, 1997, pp. 432–435.

[121] B. Robinson and C. Cooley, “Synthetic dynamic transmit focus,” in *IEEE International Ultrasonics Symposium*, 2000, pp. 1209–1214, ISBN: 0780363655.

[122] D. Romero-Laorden, C. J. Martín-Arguedas, J. Villazón-Terrazas, O. Martinez-Graullera, M. Santos-Peñas, and C. Gutierrez-Fernandez, “Cheetah : a library for parallel ultrasound beamforming in multi-core systems,” *Journal of Applied Mathematics and Physics*, vol. 3, no. August, pp. 1056–1061, 2015. DOI: `http://dx.doi.org/10.4236/jamp.2015.38131`.

[123] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, M. Pérez-Lopez, R. Tokio Higuti, and A. Octavio, “Aceleración del proceso de conformación de haz en imagen ultrasónica mediante gpus,” in *II Workshop en Aplicaciones de Nuevas Arquitecturas de Consumo y Alta Prestaciones*, 2009, ISBN: 978-84-692-7320-3.

[124] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, and L. Gómez-Ullate, “Gpgpu techniques to accelerate modelling in nde,” in *36th Annual Review of Progress in Quantitative Nondestructive Evaluation*, Kingston (Rhode Island), 2009, pp. 1991–1998, ISBN: 978-0-7354-0748-0. DOI: `10.1063/1.3362356`. [Online]. Available: `http://link.aip.org/link/APCPCS/v1211/i1/p1991/s1\&Agg=doi`.

[125] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, R. Tokio Higuti, and A. Octavio, “Using gpus for beamforming acceleration on saft imaging,” in *IEEE International Ultrasonics Symposium*, Rome, Italy: IEEE, 2009, pp. 1334–1337, ISBN: 9781424443895. DOI: `10.1109/ULTSYM.2009.5441790`. [Online]. Available: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5441790`.

[126] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, M. Pérez-Lopez, and L. Gómez-Ullate, “Field modelling acceleration on ultrasonic systems using graphic hardware,” *Computer Physics Communications*, vol. 182, no. 3, pp. 590–599, Oct. 2010, ISSN: 00104655. DOI: `10.1016/j.cpc.2010.10.032`. [Online]. Available: `http://linkinghub.elsevier.com/retrieve/pii/S0010465510004339`.

[127] D. Romero-Laorden, O. Martinez-Graullera, C. J. Martin-Arguedas, M. Perez-Lopez, and L. Gomez-Ullate, “Paralelización de los procesos de conformación de haz para la implementación del total focusing method,” in *12 Congreso Español de END*, Valencia, 2011, ISBN: 978-84-694-4076-6.

[128] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, M. Pérez-Lopez, and L. Gómez-Ullate, "Técnicas gpgpu para acelerar el modelado de sistemas acústicos," in *42 Congreso Español de Acústica (TECNIACUSTICA)*, Cáceres, Spain, 2011, ISBN: 978-84-87985-20-1.

[129] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, and M. Parrilla, "Application of golay codes to improve snr in coarray based synthetic aperture imaging systems," in *7th IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, Hoboken, New Jersey, EEUU, 2012, pp. 325–328, ISBN: 978-1-4673-1070-3. DOI: http://dx.doi.org/10.1109/SAM.2012.6250501. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=\&arnumber=6250501\&contentType=Conference+Publications\&searchField=Search_All\&queryText=Application+of+Golay+codes+to+improve+SNR+in+coarray+based+synthetic+aperture+imaging+systems.

[130] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, A. Ibáñez, and L. Gómez-Ullate, "Paralelización de los procesos de conformación de haz para imagen ultrasónica con técnicas gpgpu," *Revista Iberoamericana de Automática e Informática Industrial*, vol. 9, no. 2, pp. 144–151, Apr. 2012, ISSN: 1697-7912. DOI: http://dx.doi.org/10.1016/j.riai.2012.02.002. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1697791212000039http://www.sciencedirect.com/science/article/pii/S1697791212000039.

[131] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, M. Pérez-Lopez, and L. Gómez-Ullate, "Técnicas gpgpu para acelerar el modelado de sistemas ultrasónicos," *Revista Iberoamericana de Automática e Informática Industrial*, vol. 9, no. 3, pp. 282–289, Jul. 2012, ISSN: 16977912. DOI: 10.1016/j.riai.2012.05.002. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1697791212000325.

[132] D. Romero-Laorden, J. Villazón-Terrazas, J. Glez-Ilinas, and O. Martínez-Graullera, "Integration of golay pulse compression in a synthetic aperture imaging system based on minimum redundancy coarray," in *43 Congreso Español de Acústica (TECNIACUSTICA)*, Évora, Portugal, 2012, ISBN: 978-989-20-3284-9. [Online]. Available: http://www.sea-acustica.es/index.php?id=567.

[133] D. Romero-Laorden, J. Villazón-Terrazas, O. Martínez-Graullera, and A. Ibáñez, "Strategies for hardware reduction on the design of portable ultrasound imaging systems," in *Advancements and breakthroughs in Ultrasound Imaging*, G. P. P. Gunarathne, Ed., 2013, ISBN: 978-953-51-1159-7. [Online]. Available: http://www.intechopen.com/articles/show/title/strategies-for-hardware-reduction-on-the-design-of-portable-ultrasound-imaging-systems.

[134] D. Romero-Laorden, J. Villazón-Terrazas, O. Martinez-Graullera, A. Ibáñez, M. Parrilla, and M. Santos-Peñas, "Analysis of parallel computing strategies to accelerate ultrasound imaging processes," *IEEE Transactions on Parallel and Distributed Systems*, vol. Pending, 2015. DOI: `TPDS-2015-08-0564`.

[135] D. Romero-Laorden, J. Villazón-Terrazas, M. Santos-Peñas, M. A. G. Izquierdo, and O. Martínez-Graullera, "Análisis de la implementación de un conformador software de señales ultrasónicas para tiempo real," *Revista Iberoamericana de Automática e Informática Industrial*, vol. Pending, 2015. DOI: `RIAI-D-15-00060`.

[136] S. Rosenzweig, M. Palmeri, and K. Nightingale, "Gpu-based real-time small displacement estimation with ultrasound," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 58, no. 2, pp. 399–405, 2011, ISSN: 1525-8955. DOI: `10.1109/TUFFC.2011.1817`. [Online]. Available: `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3408661\&tool=pmcentrez\&rendertype=abstract`.

[137] G. Rougeron, J. Lambert, E. Iakovleva, L. Lacassagne, and N. Dominguez, "Implementation of a gpu accelerated total focusing reconstruction method within civa software," *40th Annual Review of Progress in Quantitative Nondestructive Evaluation*, vol. 1581, no. 1, pp. 1983–1990, 2013, ISSN: 1525-8955. DOI: `10.1063/1.4865067`.

[138] T. Sato, M. Ueda, and S. Fukuda, "Synthetic aperture sonar," *The Journal of the Acoustical Society of America*, pp. 799–802, 1973.

[139] S. Smith, R. Wagner, J. Sandrik, and H. Lopez, "Low contrast detectability and contrast/detail analysis in medical ultrasound," *IEEE Transactions on Sonics and Ultrasonics*, vol. 30, no. 3, pp. 164–173, 1983. DOI: `10.1109/T-SU.1983.31405`.

[140] H. K. H. So, J. Chen, B. Y. S. Yiu, and A. C. H. Yu, "Medical ultrasound imaging: to gpu or not to gpu," *IEEE Micro*, vol. 31, no. 5, pp. 54–65, 2011, ISSN: 0272-1732. DOI: `http://dx.doi.org/10.1109/MM.2011.65`. [Online]. Available: `http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=5959145`.

[141] B. D. Steinberg, *The Fourier Transform and its Applications*. McGraw-Hill, 1986.

[142] B. D. Steinberg, *Principles of Aperture and Array System Design*, Wiley, Ed. New York, 1976.

[143] T. Stepinski, "An implementation of synthetic aperture focusing technique in frequency domain," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 54, no. 7, pp. 1399–408, Jul. 2007, ISSN: 0885-3010. [Online]. Available: `http://www.ncbi.nlm.nih.gov/pubmed/17718329`.

[144] M. Sutcliffe, M. Weston, B. Dutton, P. Charlton, and K. Donne, "Real-time full matrix capture for ultrasonic non-destructive testing with acceleration of post-processing through graphic hardware," *NDT & E International*, vol. 51, pp. 16–23, 2012, ISSN: 09638695. DOI: 10.1016/j.ndteint.2012.06.005. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0963869512000850.

[145] H. Sutter and J. Larus, "Software and the concurrency revolution," *ACM Queue*, vol. 3, no. 7, pp. 56–62, 2005.

[146] M. Tanter and M. Fink, "Ultrafast imaging in biomedical ultrasound," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 61, no. 1, pp. 102–19, Jan. 2014, ISSN: 1525-8955. DOI: 10.1109/TUFFC.2014.6689779. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24402899.

[147] B. Tavli and M. Karaman, "Correlation processing for correction of phase distortions in subaperture imaging," *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 46, no. 6, pp. 1477–88, 1999, ISSN: 0885-3010. DOI: 10.1109/58.808871. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18244344.

[148] G. E. Trahey and L. F. Nock, "Synthetic receive aperture imaging with phase correction for motion and for tissue inhomogeneteities - part ii: effects of and correction for motion," *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 39, no. 4, pp. 496–501, 1992.

[149] F. N. Ucar and M. Karaman, "Beam space processing for low-cost systems," in *IEEE International Ultrasonics Symposium*, 1996, pp. 1349–1352, ISBN: 0780336151.

[150] T. Varslot and S.-E. Må sø y, "Forward propagation of acoustic pressure pulses in 3d soft biological tissue," in *Modeling, Identification and Control: A Norwegian Research Bulletin*, vol. 27, 2006, pp. 181–190. DOI: 10.4173/mic.2006.3.4. [Online]. Available: http://www.mic-journal.no/ABS/MIC-2006-3-4.asp.

[151] D. Vilkomerson, J. Greenleaf, and V. Dutt, "Towards a resolution metric for medical ultrasonic imaging," in *IEEE International Ultrasonics Symposium*, 1995, pp. 1405–1410. DOI: 10.1109/ULTSYM.1995.495818. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=\&arnumber=495818\&isnumber=10601.

[152] J. Villazón-Terrazas, A. Ibáñez, D. Romero-Laorden, and M. Parrilla, "A fast acoustic field simulator," in *43 Congreso Español de Acústica (TECNIACUSTICA)*, Évora, Portugal, 2012, pp. 1–9, ISBN: 978-989-20-3284-9. [Online]. Available: http://www.sea-acustica.es/index.php?id=567.

[153] D. Vray, T. Rastello, F. Schoofs, and P. Delachartre, “Improving the spatial resolution of intravascular ultrasound imaging with synthetic aperture fourier-based methods,” in *IEEE International Ultrasonics Symposium*, 1997, pp. 1531–1534, ISBN: 0780341538.

[154] K. Wall and G. R. Lockwood, “Modern implementation of a realtime 3d beamformer and scan converter system,” in *IEEE International Ultrasonics Symposium*, vol. 00, 2005, pp. 1400–1403, ISBN: 078039383X.

[155] L. Wang, D. Shi, A. Zhao, C. Tan, and D. C. Liu, “Real-time scan conversion for ultrasound imaging based on cuda with direct3d display,” in *Conference on Bioinformatics and Biomedical Engineering (iCBBE)*, Ieee, May 2011, pp. 1–4, ISBN: 978-1-4244-5088-6. DOI: 10.1109/icbbe.2011.5780361. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5780361.

[156] J. S. Woo, *A short history of the development of ultrasound in obstetrics and gynecology*, 1998.

[157] J. T. Yen and S. W. Smith, “Real-time curvilinear and improved rectilinear volumetric imaging,” in *IEEE International Ultrasonics Symposium*, 2001, pp. 1117–1122, ISBN: 0780371771.

[158] J. T. Yen and S. W. Smith, “Real-time rectilinear volumetric imaging,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 49, no. 1, pp. 114–24, Jan. 2002, ISSN: 0885-3010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18238521.

[159] ——, “Real-time rectilinear 3-d ultrasound with 4: 1 receive mode multiplexing,” in *IEEE International Ultrasonics Symposium*, vol. 00, 2003, pp. 954–959, ISBN: 0780379225.

[160] ——, “Real-time rectilinear 3-d ultrasound using receive mode multiplexing,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 51, no. 2, pp. 216–226, 2004.

[161] J. Ylitalo, “In-vitro study of a computed ultrasound imaging method,” in *IEEE International Ultrasonics Symposium*, Ieee, 1994, pp. 1577–1580, ISBN: 0-7803-2012-3. DOI: 10.1109/ULTSYM.1994.401893. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=401893.

[162] J. T. Ylitalo, “Synthetic aperture ultrasound imaging using a convex array,” in *IEEE International Ultrasonics Symposium*, vol. 2, Ieee, 1995, pp. 1337–1340, ISBN: 0-7803-2940-6. DOI: 10.1109/ULTSYM.1995.495803. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=495803.

[163] ——, “On the signal-to-noise ratio of a synthetic aperture ultrasound imaging method,” *European Journal of Ultrasound*, vol. 3, pp. 277–281, 1996.

[164] J. T. Ylitalo and H. Ermert, “Ultrasound synthetic aperture imaging: monostatic approach,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 41, no. 3, pp. 333–339, 1994.

[165] F. Zhang, A. Bilas, A. Dhanantwari, K. N. Plataniotis, R. Abiprojo, and S. Stergiopoulos, “Parallelization and performance of 3d ultrasound imaging beamforming algorithms on modern clusters,” in *Proceedings of the 16th international conference on Supercomputing*, 2002, pp. 294–304, ISBN: 1-58113-483-5. DOI: 10.1145/514191.514232.

Part II

Publications

2 | Related publications associated to Section 1.4.1

2.1 Introduction

Ultrasound simulation models are an important tool for testing and validating imaging techniques. Several methods to simulate ultrasound images have already been published, some aim to obtain the best accuracy possible without solving the full wave equation, like Abersim [150], which can accurately simulate an ultrasonic beam through heterogeneous tissues. Other well-known simulation methods are based on linearly calculating the impulse response at all scatterer positions, as is done by Field II [59], although they suffer from significant computational demands and memory intensive, particularly for dynamic images where the simulation grid must be very fine. To overcome these inconveniences, some approaches are based on diverse approximations and the use of parallel computing techniques to accelerate the simulation processes. [21], [29], [33] are some of them, where fast simulation of ultrasound images (even for deformable mediums) is presented for 3D simulations and their results are compared with exact solutions using Field II. Some other works like the proposed by T. Hergum [40] are not based in the spatial impulse response approach and they performs the convolution of the PSF and an object of sparsely distributed scatterers to simulate the ultrasound images.

Here, we have developed our own simulation model [65], [124], [126], [128], [131], [152] for the problem of diffraction, based on Piwakowski's impulse response method [117]. The solution is based primarily on the direct computation of the impulse response, which can be applicable to transducers and targets with arbitrary geometry and the use of GPUs to increase the speed of the algorithms as it is described in Papers 2.2 and 2.3. The computing time problem is especially important in the case of two-dimensional arrays which may contain hundreds or thousands of elements and have to be characterized in a three dimensional space.

2.1.1 Related publications

Apart from the full-text publications included in this chapter, in this section it is summarized all the contributions to this research topic published in journals and conference proceedings.

[124] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, and L. Gómez-Ullate, "Gpgpu techniques to accelerate modelling in nde," in *36th Annual Review of Progress in Quantitative Nondestructive Evaluation*, Kingston (Rhode Island), 2009, pp. 1991–1998, ISBN: 978-0-7354-0748-0. DOI: `10.1063/1.3362356`. [Online]. Available: `http://link.aip.org/link/APCPCS/v1211/i1/p1991/s1\&Agg=doi`.

[126] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, M. Pérez-Lopez, and L. Gómez-Ullate, "Field modelling acceleration on ultrasonic systems using graphic hardware," *Computer Physics Communications*, vol. 182, no. 3, pp. 590–599, Oct. 2010, ISSN: 00104655. DOI: `10.1016/j.cpc.2010.10.032`. [Online]. Available: `http://linkinghub.elsevier.com/retrieve/pii/S0010465510004339`.

[128] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, M. Pérez-Lopez, and L. Gómez-Ullate, "Técnicas gpgpu para acelerar el modelado de sistemas acústicos," in *42 Congreso Español de Acústica (TECNIACUSTICA)*, Cáceres, Spain, 2011, ISBN: 978-84-87985-20-1.

[131] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, M. Pérez-Lopez, and L. Gómez-Ullate, "Técnicas gpgpu para acelerar el modelado de sistemas ultrasónicos," *Revista Iberoamericana de Automática e Informática Industrial*, vol. 9, no. 3, pp. 282–289, Jul. 2012, ISSN: 16977912. DOI: `10.1016/j.riai.2012.05.002`. [Online]. Available: `http://linkinghub.elsevier.com/retrieve/pii/S1697791212000325`.

[152] J. Villazón-Terrazas, A. Ibáñez, D. Romero-Laorden, and M. Parrilla, "A fast acoustic field simulator," in *43 Congreso Español de Acústica (TECNIACUSTICA)*, Évora, Portugal, 2012, pp. 1–9, ISBN: 978-989-20-3284-9. [Online]. Available: `http://www.sea-acustica.es/index.php?id=567`.

2.2 Field modelling acceleration on ultrasonic systems using graphic hardware

Computer Physics Communications 182 (2011) 590–599

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Field modelling acceleration on ultrasonic systems using graphic hardware

D. Romero-Laorden *, O. Martínez-Graullera, C.J. Martín, M. Pérez, L.G. Ullate

Centro de Acústica Aplicada y Evaluación No Destructiva (CAEND) CSIC-UPM, Spain

ARTICLE INFO

Article history:
Received 8 February 2010
Received in revised form 19 August 2010
Accepted 30 October 2010
Available online 5 November 2010

Keywords:
Field modelling
GPGPU
Ultrasonics
NDE

ABSTRACT

Field modelling is a common practice in the area of ultrasonic non-destructive evaluation (NDE) because it is a useful tool for assessing NDE imaging. However, it is a very time consuming task because of its complexity and data volume, making difficult its use in systems demanding real time responses. Recently, graphics processing units (GPUs) have experienced an extraordinary evolution in both computing performance and programmability, leading to greater use on non-rendering applications. This work shows that the use of GPU technology, which has a high level of parallelism, accelerates the ultrasonic field simulation, reducing the computing time in more than one order of magnitude respect to CPU implementations.

1. Introduction

Nowadays, non-destructive evaluation (NDE) techniques have an important role in both the detection and characterization of flaws in materials and structures. Ultrasonic NDE techniques are based on a basic principle: an electronic system excites one or more ultrasonic transducers to generate high frequency mechanical waves, which propagate in the material under inspection. The reflected waves are again received by the transducers and they are digitally processed in order to extract several characteristics, such as the presence of discontinuities that can be identified as flaws [1,2].

The trend nowadays is focused on the inspection of structures with complex geometries using transducers arrays with hundreds of elements. In order to resolve the difficulties associated to the analysis of the data in the evaluation, it is necessary to use diverse simulation techniques in order to obtain a reference of the result. In this way, ultrasonic NDE techniques make use of different simulation models to obtain some image patterns, which are used as a base to determine the state of the component under analysis. In this sense, several computational methods have been applied to obtain the response of targets in complex systems. These methods are based on the impulse response (IR) method [3–10], the angular spectrum technique [11–13], the finite element analysis [14,15], optical geometry studies [16,17], multi-Gaussian beam models [18] and so on. Using some of them, simulation tools have been developed for evaluation assessment in medical applications [19] or other sophisticated software packages, such as the CIVA program [20] for industrial NDE. Furthermore, modelling methods are also useful to design new inspections in ultrasonic imaging [21] and they present a common feature: the computing algorithm is iteratively applied to a huge amount of points, increasing the execution time and therefore preventing its use on real time systems. However, as every point can be handled independently, these modelling processes are perfect candidates and suitable for parallel computing.

In last years, one of the dominant trends in microprocessor architectures has been the continuous increment of the chip-level parallelism. Multicore CPUs providing 2–4 scalar cores are now commonplace, and there is no doubt that the trend towards increasing the parallelism will continue on "manycore" chips. In contrast, GPUs (Graphics Processor Units) have been at the leading edge of this drive towards increased chip-level parallelism, and nowadays they mainly are manycore processors. In fact, the current NVIDIA GPUs contain up to 240 scalar processing elements per chip [22] and, in contrast to earlier generations of GPUs, they can be programmed directly in C using CUDA [23–25].

In this paper, the application of GPU computation techniques to field modelling in ultrasonic NDE is investigated, with the objective of facilitating their implementation in real time systems. To do that, we have employed one of the most known approaches based on the impulse response method [5], which can be applicable to transducers and targets with arbitrary geometry. Two examples of calculus are presented: the first one based on a linear array transducer of 128 elements (A1 array); the second one using a two-dimensional square array transducer with 128×128 elements (A2 array). In both cases, a 3 MHz transducer with 50% of relative bandwidth has been used. The pitch is $d = \lambda/2$ ($\lambda = 0.5$ mm in

* Corresponding author at: Carretera de Campo Real, km 0.200, Arganda del Rey, Madrid E-28500, Spain. Tel.: +34 91 871 19 00.
E-mail address: dromerol@caend.upm-csic.es (D. Romero-Laorden).

0010-4655/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2010.10.032

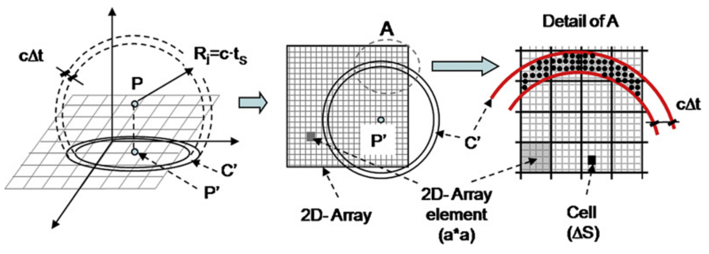


Fig. 1. Example of the aperture discretization. Number of cells that integrate an element.

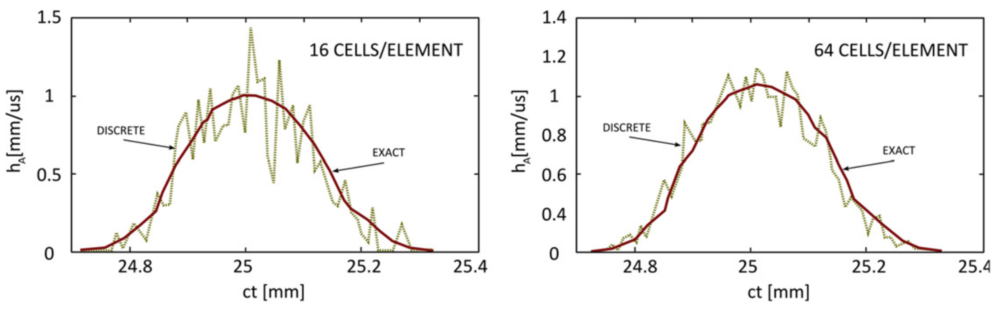


Fig. 2. Comparative between exact and discrete impulse responses for the A2 array.

water) and the size of the elements is $a = \lambda/2$, $b = 16\lambda$ in the linear array, and $a = b = \lambda/2$ in the squared array. The kerf has been neglected.

2. Field modelling algorithm

The general expression of the acoustic pressure generated by an array working in emission-reception is given by [8]:

$$p(\vec{x},t) = \frac{1}{c^2} * \frac{\partial^2 v(t)}{\partial t^2} * h_E(\vec{x},t) * h_R(\vec{x},t) \tag{1}$$

where h_E and h_R are, respectively, the impulse responses in emission and reception of the array:

$$h_E(\vec{x},t) = \sum_{i=1}^{N} a_i^E h_i(\vec{x}, t - T_i^E) \tag{2}$$

$$h_R(\vec{x},t) = \sum_{i=1}^{N} a_i^R h_i(\vec{x}, t - T_i^R) \tag{3}$$

In the former expressions h_i is the impulse response of the ith array element and a_i is its gain. T_i^E and T_i^R are, respectively, the focusing delays in emission and reception. For a single point $P(\vec{x})$, they are given by the following expression:

$$T_p(i) = \frac{|\vec{x} - \vec{x}_i| - |\vec{x}|}{c} \tag{4}$$

where c is the speed of sound, and $\vec{x}_i$ the position vector of the ith array element. In order to calculate the acoustic field it has been followed the Piwakowski method [5], which makes a direct computation of the Rayleigh integral by mean of transducer surface discretization into squared cells of elemental area ΔS, and time sampling with intervals of Δt (Fig. 1). Hence, the contribution of every element to the array impulse response at a given instant t_S is obtained by adding the contributions of the cells contained between two concentric spherical waves, which are separated by the discretization interval $c\Delta t$ [5]. Thus, the contribution of the ith array element at the instant $t = t_s$ is given by:

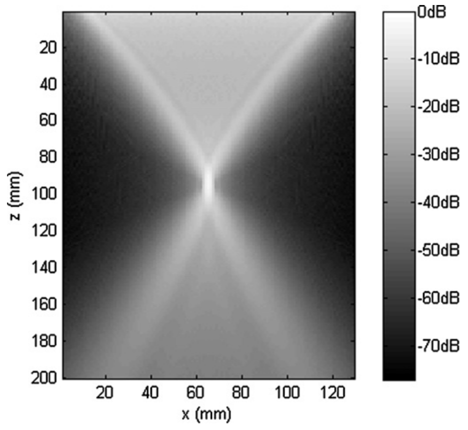


Fig. 3. PSF of A1 array focusing at the distance $R = 52$ mm.

$$h_i(t = t_S) = \frac{1}{\Delta t} \sum_j b_j, \quad t_S - \frac{\Delta t}{2} \leqslant t_j \leqslant t_S + \frac{\Delta t}{2} \tag{5}$$

$$b_j = \frac{a_i \Delta S}{2\pi(R_j - T_i)} \quad \text{and} \quad R_j = ct_j \tag{6}$$

where T_i denotes the focusing delay in emission or reception and R_j is the distance from each cell to the field point $P(\vec{x})$. An example of discretization of a 2D squared array is presented in Fig. 1. One element of the array will integrate all the cells falling within its silhouette. The increment of the spatial and temporal resolutions contributes to reduce the discretization error but, at the same time the computational cost is heavily increased. The differences between the exact impulse response and the impulse responses obtained with the Piwakowski method, considering 16 or 64 cells per element, are also shown in Fig. 2. The computing error between the discrete pattern and the exact analytic solution was studied in [5], concluding that there is very high accuracy when spatial and temporal sampling rates are in the order of $\Delta x/\lambda = 0.01$ and $\Delta tf = 0.01$ respectively. Figs. 3 and 4 show

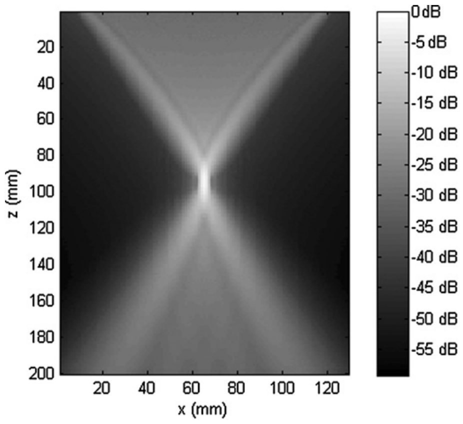


Fig. 4. PSF of A2 array focusing at the distance $R = 52$ mm.

the two-way pressure field generated by the linear array (A1) and the 2D squared array (A2) respectively. Simulations have been done for 12 900 field points between $-16\text{ mm} \leqslant x \leqslant 16\text{ mm}$ and $5\text{ mm} \leqslant z \leqslant 105\text{ mm}$. In both cases a fixed focus on emission and reception has been implemented at $x_F = 0$ and $z_F = 52$ mm.

2.1. Computational algorithm

Fig. 5 shows the computational algorithm based on Piwakowski method, which has been used for acoustic field computation. More in detail, the figure presents the necessary steps to calculate the field at a single point P. It is important to remark that the algorithm is an iterative process that is applied to all the field points. These are the computation steps for every point:

1. In the first step, the distances from the emission aperture cells to every field point P are calculated. Moreover, other data, such as focusing delays (Eq. (4)) and apodization weights, are calculated and associated to their respective cells (Eq. (6)).
2. In the second step the same procedure is applied to the reception aperture if it is different from the emission.
3. Then, the array impulse response in emission $h_E(P,t)$ is calculated (Eqs. (2) and (5)).
4. In the fourth step, the array impulse response in reception $h_R(P,t)$ is obtained (Eqs. (3) and (5)).
5. Later on, the convolution between h_E and h_R is computed, obtaining the two-way impulse response of the array aperture $h_{ER}(P,t)$ (Eq. (1)).
6. Finally, the convolution between $h_{ER}(P,t)$ and the emitted signal is done, obtaining the two-way acoustic field value (Eq. (1)).

3. Parallel computing

Before discussing the design of the algorithm, we shall comment briefly some details about NVIDIA's GPU architecture and CUDA parallel programming model. As their name implies, GPUs came about as accelerators for graphic applications, predominantly those using OpenGL and DirectX programming interfaces. Due to the huge parallelism inherent to the graphics, GPUs have long been massively parallel machines. Although originally they were pure fixed-function devices, the market demand for real time and 3D graphics made themshelves to evolve into increasingly flexible highly parallel, multithreaded, manycore processors, with extremely high computational power and very high memory bandwidth.

Modern NVIDIA GPUs, beginning with the GeForce 8800 GTX, are nowadays fully programmable manycore chips built around an array of parallel processors. In this sense, a comparison between CPU and GPU architectures is shown in Fig. 6. As it can seen, both CPU and GPU manage their own memory, which is known as host and device memory respectively. In fact, a GPU is composed by N multiprocessors (MPs), from 1 to 30 in the current NVIDIA architecture, each one composed by 8 scalar processor (SPs). Furthermore, it is equipped with an on-chip memory with a very low access latency and high bandwidth, similar to the "level one" CPU cache.

In order to provide to the developers the means to execute parallel programs on the GPU, NVIDIA has developed an API called CUDA (*Compute Unified Device Architecture*) [23,24]. The main concept of CUDA parallel programming model is to operate with thousands of lightweight threads, which execute the same function with different data. All thread management, including creation, scheduling and barrier synchronization, is performed entirely in hardware using a SIMT (*Single Instruction, Multiple Thread*) architecture. Threads are executed in groups of 32, called *warps*, that are executed on separate scalar processors, which share a single multithreaded instruction unit. This SIMT architecture allows the hardware to achieve substantial efficiencies during the execution of non-divergent dataparallel codes. Therefore, an application is organized as a sequential host program that can execute parallel functions, known as *kernels*, in a parallel device (Fig. 7). A kernel is a SPMD style (*Single Program, Multiple Data*) computation, executing a scalar sequential program across a set of parallel threads. Basically, the programmer organizes these threads into thread blocks

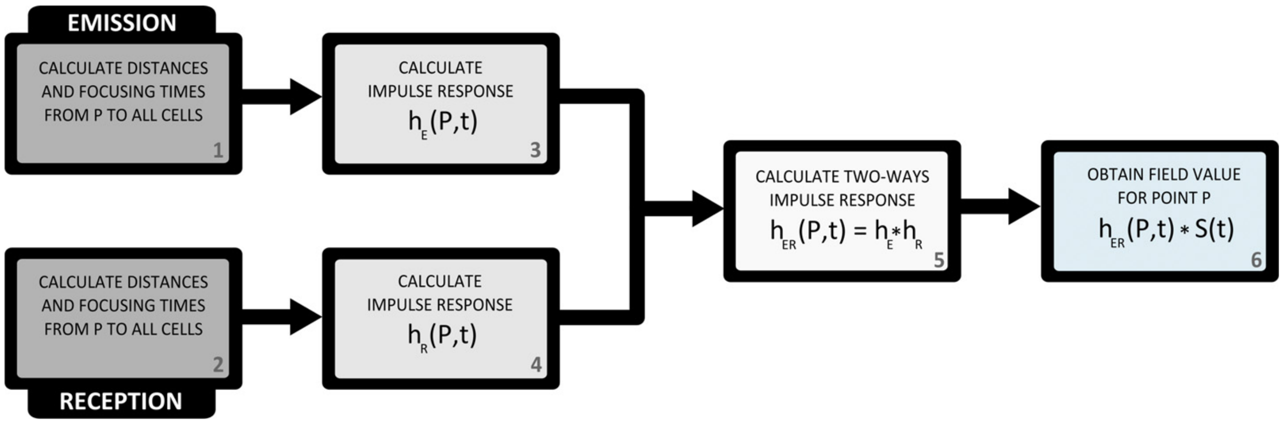

EMISSION
CALCULATE DISTANCES AND FOCUSING TIMES FROM P TO ALL CELLS
1
CALCULATE IMPULSE RESPONSE
$h_E(P,t)$
3
CALCULATE TWO-WAYS IMPULSE RESPONSE
$h_{ER}(P,t) = h_E * h_R$
5
OBTAIN FIELD VALUE FOR POINT P
$h_{ER}(P,t) * S(t)$
6
CALCULATE DISTANCES AND FOCUSING TIMES FROM P TO ALL CELLS
2
CALCULATE IMPULSE RESPONSE
$h_R(P,t)$
4
RECEPTION

Fig. 5. Iterative algorithm of Piwakowski method. The calculus of acoustic field on emission-reception require six steps for each point.

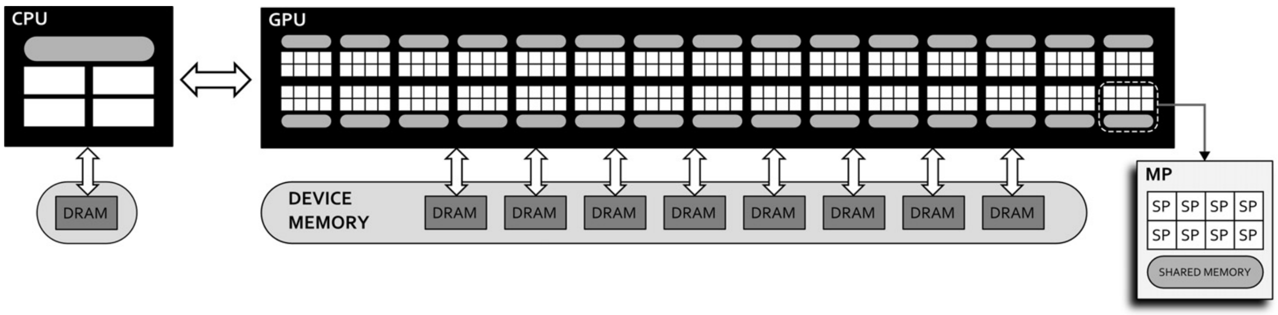


Fig. 6. Four-cores CPU versus GeForce GTX 295 GPU with 240 SPs organized in 30 MPs.

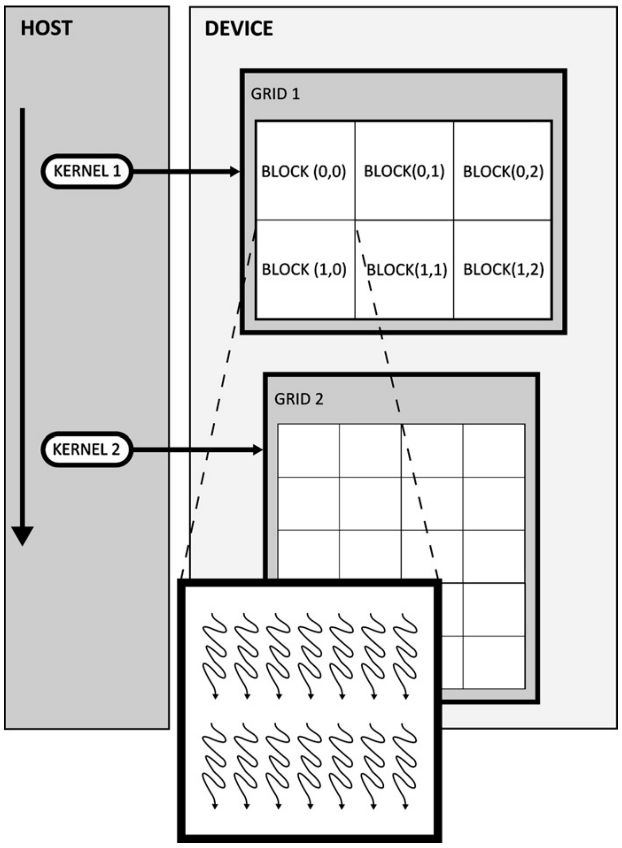


Fig. 7. Threads configuration (blocks and grids). Serial code executes on CPU, while parallel code (kernels) executes on GPU.

$(B_0, B_1, \ldots, B_N)$ and grids of one or more blocks $(NB_x \times NB_y)$, which are identified with a unique identifier (*id*) inside the kernel (Fig. 7). A thread block is a group of concurrent threads that can cooperate among themselves through barrier synchronization and a per-block shared memory space. When a kernel is invoked, it is necessary to specify both the number of blocks and the number of threads per block.

The optimum usage of the GPU resources is partly conditioned by some parameters: the number of threads per block, the number of registers and the amount of shared memory used by a kernel. These parameters determine the occupancy of the multiprocessors. The programmer has to deal with these parameters to maximize the parallelism as much as possible, taking advantage of the power of the graphic hardware (the use of some type of profiling tools such as CUDA Occupancy Calculator [24] for example can help).

There are several types of memory defined in CUDA, each one designed for different uses in kernels. In that way, the computation performance can been increased. The global memory is essentially the random access video memory of the GPU. It can be read or written at any time and any location by any thread. The access to global memory should be coalesced in order to achieve high performance. This means that the threads must follow a specific memory access pattern [23]. A kernel has access to two cached, memories: the constant memory, used to store constants during kernel execution; and the texture memory, used to store program data in one, two, or three dimensions. There is also a parallel data cache available for reading and writing. This is the shared memory, which is divided into banks. This memory makes possible the cooperative work of threads in a block. Different kernels should be written in a way to avoid bank conflicts. This means that the threads executed physically at the same time should access to different banks. More information about how to achieve this and other resources can be found in the Programming Guide [23].

3.1. Considerations

The notable differences between CPU and GPU architectures and CUDA programming model make necessary to discuss some considerations about the GPU computing. It is important to emphasize that memory transactions between CPU and GPU are pretty slow and they must be minimized as much as possible in order to improve the general performance. Furthermore, the device memory is limited. Hence, it could be necessary to reduce the volume of data to be processed in parallel. The reading time from global memory is very slow, so it is recommendable to use some other mechanisms: texture, shared or constant memories. Finally, it is necessary to simplify the thread operations, in order to homogenize the execution time of all threads, and to limit the write access to GPU memory. Unfortunately, the direct implementation on GPU of the original algorithms designed for CPU usually is not the best option.

4. Parallel algorithms

Executed functions many times on different data are prime candidates for GPU computing and, in this way, the features of the ultrasonic image modelling algorithms (that usually imply the operation with a huge amount of data and the calculus of diverse results in a repetitive way) make them to be excellent candidates for parallel computing. Furthermore, maximizing the parallelism to obtain an efficient algorithm on this kind of architectures is fundamental, because using a small number of threads to execute a kernel is very inefficient. For these reasons, the major problem of GPU computing is to find the best strategy to parallelize the algorithms in order to obtain the maximum benefit.

In this work several alternatives have been evaluated. Due to the fact that the computational algorithm is applied per space

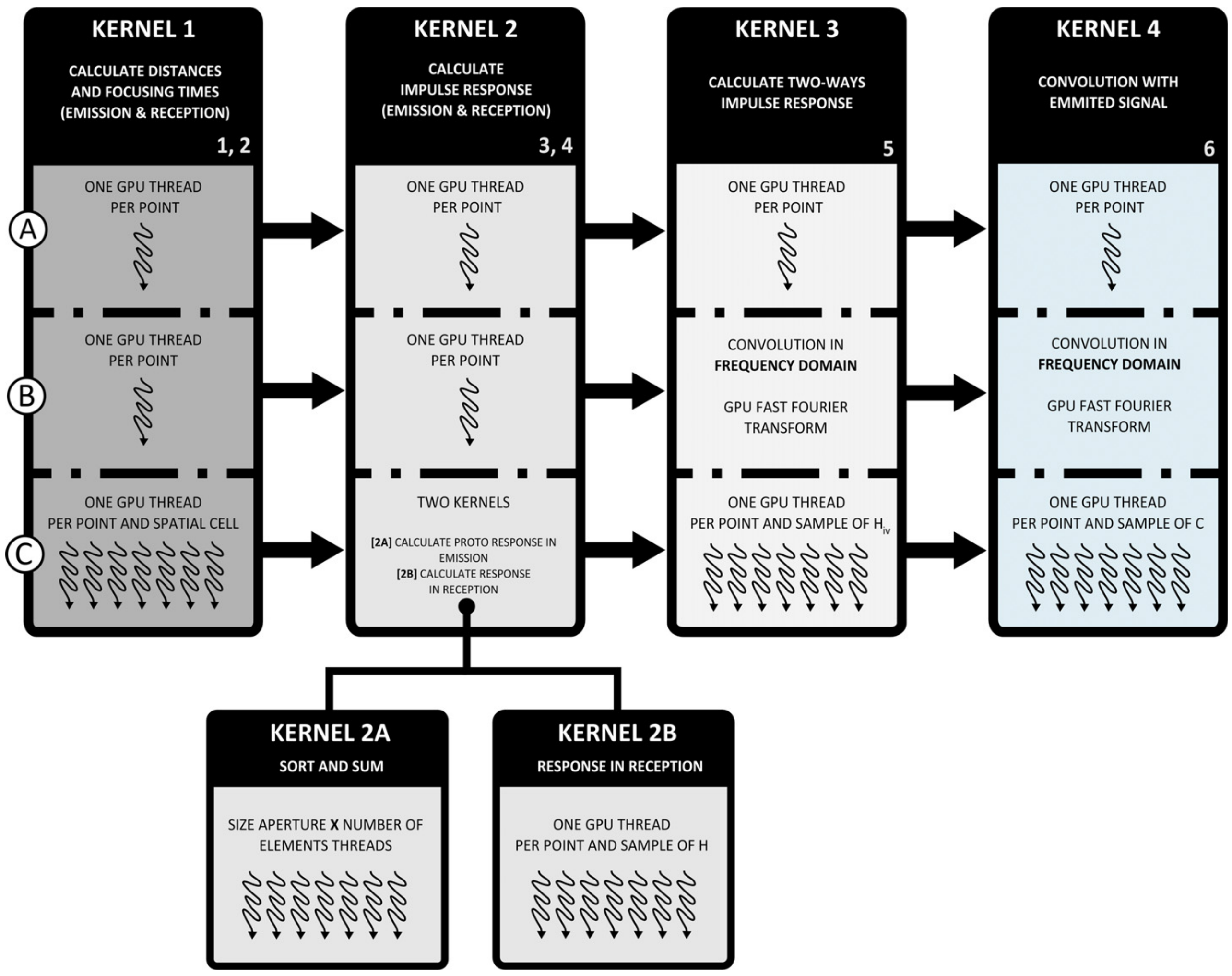

KERNEL 1
CALCULATE DISTANCES AND FOCUSING TIMES (EMISSION & RECEPTION)
1, 2
KERNEL 2
CALCULATE IMPULSE RESPONSE (EMISSION & RECEPTION)
3, 4
KERNEL 3
CALCULATE TWO-WAYS IMPULSE RESPONSE
5
KERNEL 4
CONVOLUTION WITH EMMITED SIGNAL
6
A
B
C
ONE GPU THREAD PER POINT
CONVOLUTION IN FREQUENCY DOMAIN
GPU FAST FOURIER TRANSFORM
ONE GPU THREAD PER POINT AND SPATIAL CELL
TWO KERNELS
[2A] CALCULATE PROTO RESPONSE IN EMISSION
[2B] CALCULATE RESPONSE IN RECEPTION
ONE GPU THREAD PER POINT AND SAMPLE OF H_iv
ONE GPU THREAD PER POINT AND SAMPLE OF C
KERNEL 2A
SORT AND SUM
SIZE APERTURE X NUMBER OF ELEMENTS THREADS
KERNEL 2B
RESPONSE IN RECEPTION
ONE GPU THREAD PER POINT AND SAMPLE OF H

Fig. 8. Schemes of the three strategies followed in this work: (A) One thread per field point, convolution in the time domain, (B) One thread per field point, convolution in frequency domain, (C) Threads adapted to each step.

point, the trivial parallelization method could be by points, using one thread for each point and calculating the maximum of their values simultaneously. Nevertheless, there are other possibilities that must be taken into account. The different stages of the algorithm imply different computational requirements, so it should be tried to parallelize by spatial cells in the aperture, sampling the emission-reception responses, or maybe others. We have to consider that the set of spatial points must be divided into groups, adapting the volume of data stored in the GPU memory and calculating the acoustic field by groups of points.

4.1. Parallelization strategies

We have analyzed three strategies. They are schematically presented in Fig. 8. For the development of the parallel algorithms several kernels are defined, which are associated with a specific stage of the computational algorithm.

4.1.1. A strategy

The first strategy is a direct implementation of the computational algorithm seen in Section 2.1. Four kernels are defined to cover each step of the algorithm. It consists on launching an execution thread in GPU (drawn as a wavy arrow in Fig. 8) for every field point and step of the algorithm. Therefore, each kernel is

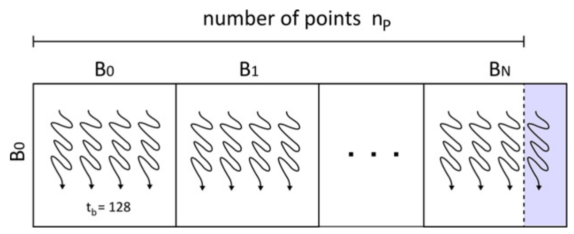


Fig. 9. Common structure of threads blocks and grid defined for strategies A and B.

made up by a grid of $ceil(n_p/t_b) \times 1$ blocks, where t_b is the block-size and n_p the number of points. In this particular case $t_b = 128$ and $n_p = 12\,900$ (Fig. 9). All partial results, such as distances, delays and impulse responses, are copied to texture memory. Despite of doubling the velocity achieved with the CPU, this strategy is too slow, due to the huge memory requirements and the very large time consumed for the convolution functions in the two last stages.

4.1.2. B strategy

This second strategy tries to accelerate the calculus of the convolutions. As the previous one, it also launches one thread per field

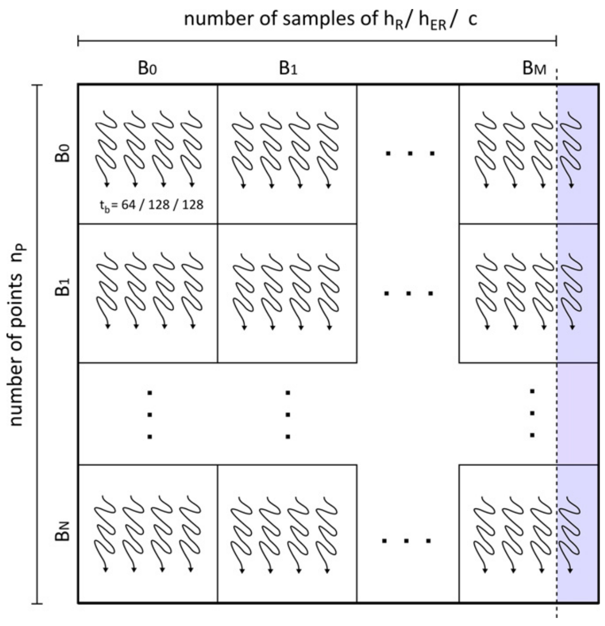


Fig. 10. Thread blocks and grid used in strategy C.

point to calculate the stages 1, 2, 3 and 4 (kernels 1 and 2), using the texture memory to store the results, and maintaining the same configuration of grids and blocks (Fig. 9). Instead of doing convolutions in time domain, it uses the NVIDIA CUFFT library, which provides specific functions to do convolutions in frequency domain. Despite FFTs are optimized in GPU, it is required a huge amount of memory to calculate them and store the results, decreasing the global performance. Hence, the velocity is near five times faster than conventional implementations. Moreover, the homogenization required by FFTs is inefficient, because the spatial impulse responses have diverse lengths.

4.1.3. C strategy

In this third strategy, we improve the use of memory and accelerate the computing time of each kernel, developing a specific solution for each step and a new data structures to increase the use of hardware resources. Thus, the algorithm is fitted to the power of the GPU. The optimized procedure is as follows:

- Stages 1 and 2 (Kernel 1). The strategy of parallelization by points is changed by other one that consists on parallelizing per points and cells (Fig. 8). Now, the kernel is made up by a grid of $ceil(n_p/t_b) \times n_c$ blocks, where n_c is the number of cells and t_b and n_p have the same values as before (Fig. 10). In this way, the distances and delays are calculated very fast (square root, sums and multiplications), and are stored in a data structure of pairs (distance, delay) in texture memory.
- Stages 3 and 4 (Kernel 2). With the goal of reducing the amount of memory required to store the array impulse responses (h_E, h_R), the second kernel was divided into two parts. The first one (Kernel 2A) sorts the structure of pairs from minor to major distance, evaluating the first field of the pair and creating a new data structure defined as *proto_response*, which contains only essential information of h_E (null values are excluded). This is a great advantage, because impulse responses usually are extremely long and they could contain hundreds of gaps (zeros), forcing to allocate more memory than available. In this way, the *proto_response* in emission is calculated very fast and with minimum resources. The second part (Kernel 2B) calculates the complete response in reception, launching a thread per point and sample of h_R (Fig. 10) and storing the result in texture memory.
- Stage 5 (Kernel 3). In this step, the kernel uses the *proto_responses* in emission and reception calculated in the previous one to calculate the first convolution ($h_E * h_R$). To do that, a parallel convolution algorithm based on the *Input Side* type [27] has been developed. It consist on launching a thread per field point and sample of the convolution result. This stage is very efficient because data structures are sorted, facilitating thus the search of samples.
- Stage 6 (Kernel 4). For the last convolution between h_{ER} and the wideband signal it has been developed an optimized parallel algorithm based on the *Output Side* type [27]. It allows to obtain the final value of the sample at the end of the kernel execution. This means it is not necessary to store the full result of the convolution, only the maximum. Instead of storing the whole convolution and making a search for the maximum, like the strategies A and B, in this case a thread per point and sample of the convolution structure is launched (Fig. 10). The value for this sample is stored in shared memory. Once all threads of a block have calculated the value for the sample, the maximum per block is obtained. Thus, after all maximums per block have been obtained, a reduction process looking for the global maximum per point is carried out. Finally, the result is copied back to global memory, and then it is transferred to the host.

5. Modelling results

To test our parallel algorithms a NVIDIA GeForce GTX 295 graphic card has been used. This card has 2 GPUs inside altough only one has been used. There are 240 cores per GPU with 1 GB of global memory each. It has been installed in a computer with a four-core 2.66 GHz Intel Q9450 processor and 4 GB of RAM. In summary, GPU resources exceed CPU resources in approximately 60 times.

Two different implementations in CPU have been tested. Both optimize the calculus of convolutions. The first implementation (CPU_TEMP) computes the convolutions in time domain using the same type of algorithms as in GPU (convolutions *Input* and *Output Side*). The second implementation (CPU_FREC) uses the FFTW library [26] to make the convolutions in frequency domain. Both applications are compared with the implementation of the C strategy (GPU_TEMP) because it is the most interesting.

5.1. Computational cost

Therefore, a comparison between the computing speed with GPU and CPU has been evaluated, considering several examples and using simple floating precision. The times given in Tables 1 and 2 are the total computation times from the start to the end of the entire program in all cases.

A linear array with 128 elements separated $\lambda/2$ and 16λ of height for a set of 12 900 field points has been used in the first example. The second example includes a two-dimensional array of 128×128 elements with the same parameters. The spatial and temporal resolutions have been varied from 16 to 4096 and from 1 to 256 cells per element respectively.

Table 1 shows a comparison between CPU and GPU computing time for the first case. CPU_TEMP implementation is obviously the slowest (first column). The computing time is more or less increased linearly with the temporal resolution and with the number of cells. The second implementation (CPU_FREC) appears in the second column of the Table 1. As it can be observed, there are not significant differences when the sampling intervals are modified, due to CPU algorithms make use of FFTs and normalizes the

Table 1
Time in μseconds per field point for computing the acoustic field in CPU and GPU for the A1 array of 128 elements.

Spatial resolution		CPU_TEMP			CPU_FREC			GPU_TEMP		
Cell size	Cells per element	Temporal resolution			Temporal resolution			Temporal resolution		
		$\frac{\lambda}{32c}$	$\frac{\lambda}{64c}$	$\frac{\lambda}{128c}$	$\frac{\lambda}{32c}$	$\frac{\lambda}{64c}$	$\frac{\lambda}{128c}$	$\frac{\lambda}{32c}$	$\frac{\lambda}{64c}$	$\frac{\lambda}{128c}$
$\frac{\lambda}{2}$	16	10 759.6	21 407.7	42 815.5	6678.2	7183.7	13 866.6	215.5	376.7	804.6
$\frac{\lambda}{4}$	64	27 331.1	54 662.1	109 324.2	312 23.25	31 938.7	32 901.2	720.1	996.9	1840.3
$\frac{\lambda}{8}$	256	95 023.5	190 046.5	380 093.1	80 651.1	81 325.8	82 271.3	1711.8	2096.2	3226.3
$\frac{\lambda}{16}$	1024	387 248.5	774 519.3	1 549 038.7	302 751.9	303 937.9	305 054.26	7293.2	7676.4	8807.7
$\frac{\lambda}{32}$	4096	1 665 829.4	3 331 658.8	6 663 317.7	1 218 232.5	1 218 302.3	1 230 596.8	45 123.25	45 506.97	46 637.9

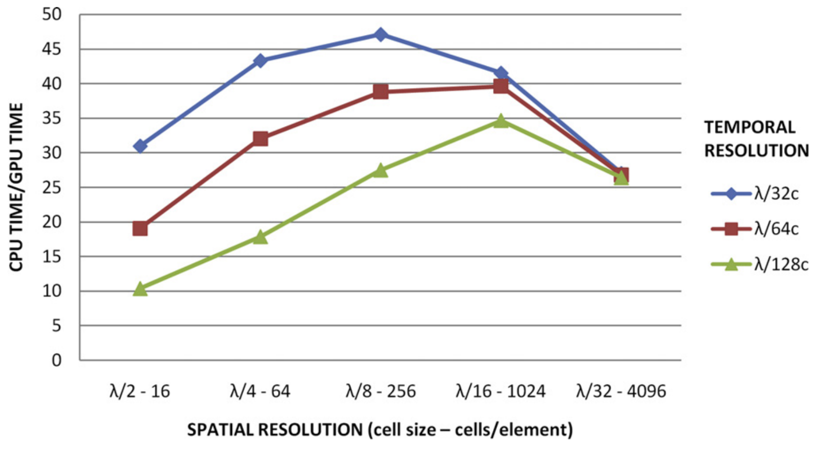


Fig. 11. Speed up achieved in the calculus of the acoustic field for the A1 array of 128 elements (12 900 field points).

Table 2
Time in μseconds per field point for computing the acoustic field in CPU and GPU for the A2 array of 128 × 128 elements.

Spatial resolution		CPU_TEMP			CPU_FREC			GPU_TEMP		
Cell size	Cells per element	Temporal resolution			Temporal resolution			Temporal resolution		
		$\frac{\lambda}{32c}$	$\frac{\lambda}{64c}$	$\frac{\lambda}{128c}$	$\frac{\lambda}{32c}$	$\frac{\lambda}{64c}$	$\frac{\lambda}{128c}$	$\frac{\lambda}{32c}$	$\frac{\lambda}{64c}$	$\frac{\lambda}{128c}$
$\frac{\lambda}{2}$	1	39 273.2	78 138.3	156 276.5	24 440.3	25 358.1	26 827.2	787.6	1151.9	2296.8
$\frac{\lambda}{4}$	4	99 758.1	199 516.6	399 033.3	120 976.7	121 782.9	123 705.4	3365.8	3824.1	5386.8
$\frac{\lambda}{8}$	16	346 835.7	693 669.7	1 425 348.7	313 720.6	316 279.1	317 674.4	7304.6	8127.9	10 079.1
$\frac{\lambda}{16}$	64	1 413 455.2	2 826 994.3	5 731 443.1	1 191 007.7	1 192 403.1	1 195 038.7	32 188.7	34 252.7	36 683.7
$\frac{\lambda}{32}$	256	6 080 275.8	12 160 554.6	23 987 943.7	4 787 906.9	485 348.8	4 846 976.7	212 031.1	213 023.25	220 232.5

lengths of h to the nearest power of two value. By contrast, if the cells per element increases, the computing time is incremented.

The third column shows the time achieved using GPU. Here, differences in computing time are more evident respect to CPU implementations. All the results include transactions time to global memory of GPU and the time required to copy the results to the CPU. The relation between temporal rates decreases as the number of cells per element increases, due to the major use of the GPU power.

Fig. 11 also presents a graphic with the total speed up achieved (CPU_FREC vs GPU_TEMP). The right and left parts of the graphic correspond with a high number (spatial oversampling) and a small number (spatial undersampling) of cells per element respectively, which produces a small reduction of the computation efficiency. Considering a temporal resolution of $\lambda/128c$ (triangular marker), which corresponds with very short time intervals (time oversampling), the demand of memory is very high. Therefore, it forces to divide the set of points into smaller groups, decreasing the computation efficiency (about 10 times faster). However, in most cases the performance of the algorithm on GPU is above 30 times faster than the performance on CPU, reaching in some cases a speed up near 50×.

The computing time obtained with CPU and GPU for the 2D array case is shown in Table 2. If we compare these results with the linear array case, a rate of around 4 can be observed, which match with the rate of the area of both arrays. On the other hand, the effect of increasing the temporal and spatial resolutions is similar in both examples, demanding a higher volume of memory. Therefore, it forces to divide the set of points into smaller groups, decreasing the computation efficiency.

If the spatial resolution is $\lambda/2$, element impulse responses only have a single sample facilitating the calculations. However, when the spatial resolution is increased, the effect of the very high number of cells per element (N^2) presents difficulties for the memory management and the computing time increases (Fig. 12).

Despite of this, a remarkable aspect in both examples is that GPUs efficiency is notably higher than CPU, being more than 20 times faster in all cases.

5.2. Quality of results

Once the comparison between the execution times obtained with CPU and GPU methods has been exposed, it is necessary to verify the results of both implementations. Thus, to measure prop-

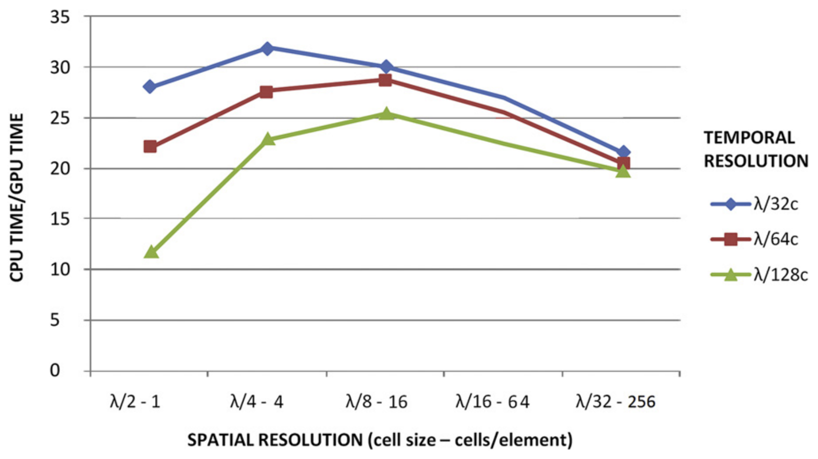


Fig. 12. Speed up achieved in the calculus of the acoustic field for the A2 array of 128 × 128 elements (12 900 field points).

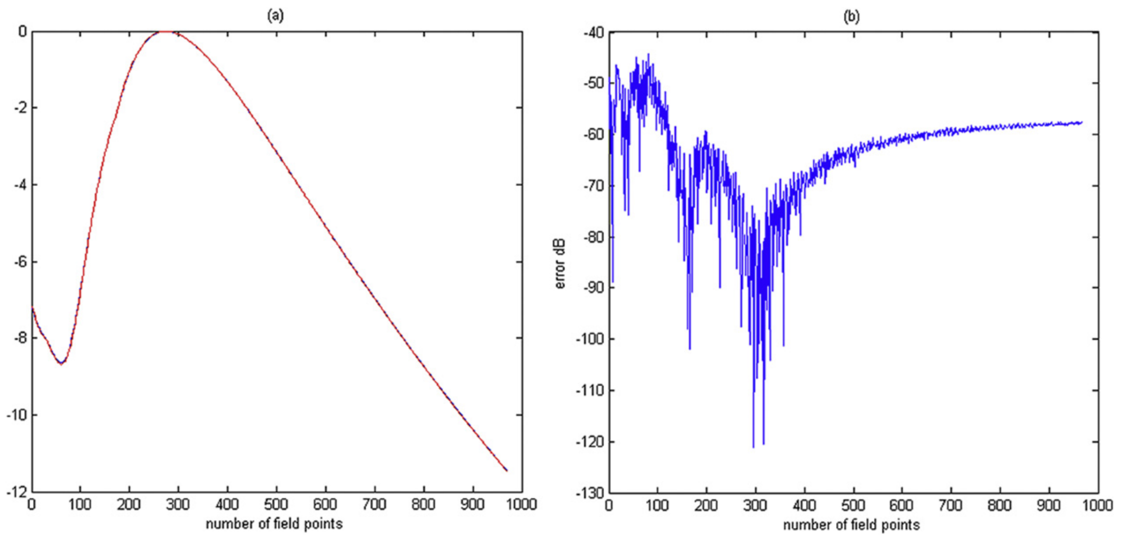


Fig. 13. Pressure field obtained with CPU and GPU implementations (blue and red lines, respectively) and its associated error. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

erly the errors, CPU results have been taken as a reference using the following equation:

$$error = 20 * \log 10\left(\left|\frac{CPU_{result} - GPU_{result}}{CPU_{result}}\right|\right) \quad (7)$$

Fig. 13a shows the pressure field obtained using CPU and GPU implementations. As we can see, both curves do not present relevant differences, being the maximum error around −50 dB and tending to stabilize around −60 dB. More in detail, we can observe that there is a great fluctuation in the initial area, which corresponds with the near field zone. Here, the differences between the various wave fronts in which the aperture has been decomposed are more significant and higher than half wavelength.

Due to the differences in the data precision (CPU and GPU use double and single precision respectively) we have evaluated its effects in the quality of the results as a cause of these errors. Because of the particularities of the GPU implementation, the CPU algorithm has been rewritten to be able to run the same operations in a similar way to the parallel algorithm designed for GPU. This new implementation is not optimal and its computation time has not been taken into account in the comparison of the results.

We have been detected some singularities in a small percentage of situations which produce errors in the partial results. However, there are not significant differences in the final result. Fig. 14 shows an example of one of these singularities for a 2D transducer array with 10 × 10 elements. Figs. 14a, 14c and 14e, corresponds to the one-way impulse response, two-way impulse response and pressure wave, respectively. The associated errors, according to Eq. (7), are shown on the right side of the figure.

We can conclude that these errors are introduced in the rounding process during the calculation of the one-way impulse response. These singularities produce a variation of only one sample up or down respect to the implementation in CPU. In Fig. 14a we can appreciate the ripple effect in the impulse response due to the Piwakowski approximation. As we see in the detail, there are two minor errors (samples 92 and 230) which are accumulated when the remaining computations are done (calculus of the two-way impulse response and pressure wave). However, these rounding errors have no importance in relation to the calculation of the acoustic field (Fig. 13).

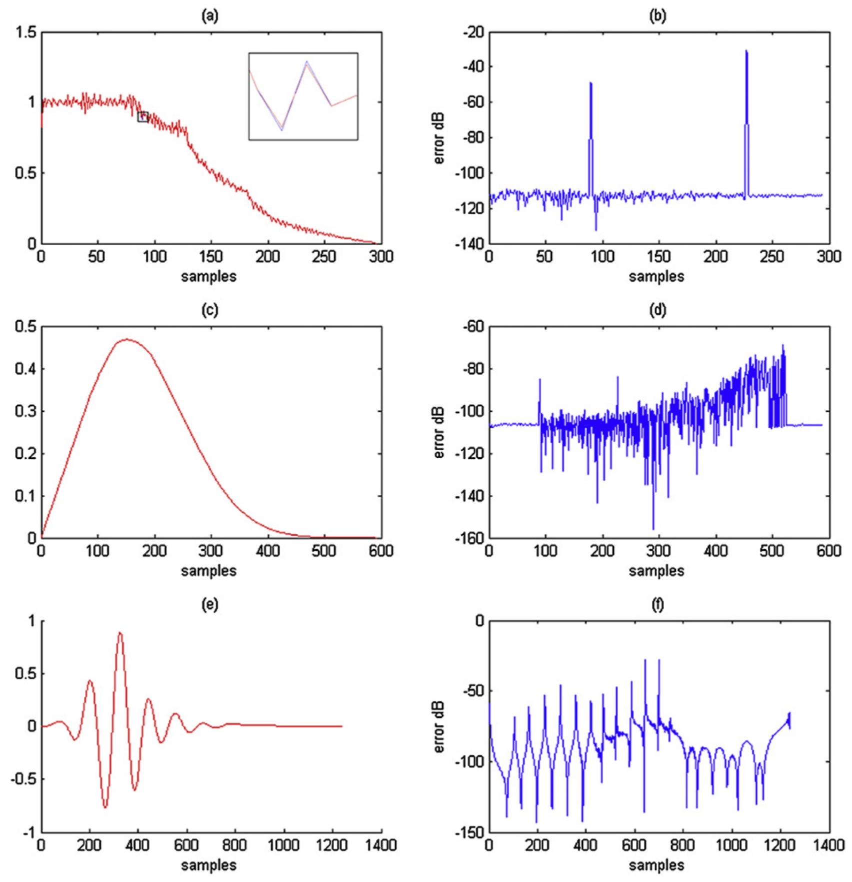


Fig. 14. Example of singularities and associated errors: one-way impulse response (figures a and b); two-way impulse response (figures c and d) and pressure wave (figures e and f).

6. Conclusions

This work is focused on exploring the GPU parallelism for NDE modelling algorithms. Using a simple graphics card equipped with NVIDIA CUDA technology, the experimental results demonstrate that the field computing time can be drastically reduced in relation to the CPU implementations, achieving in some cases a performance near fifty times faster than the conventional CPU. Furthermore, it has been demonstrated that the designed parallel algorithms are faster. Therefore, this is a very cheap solution with a good relation between computational power and price, and it is a scalable solution that can be used in most cases to accelerate modelling problems in the area of ultrasonic NDE. Similar strategies can be applied to other cases of study in order to increase the performance. In summary, shifting the computation of acoustic fields from CPU to programmable GPU can be a very good solution for field computing in ultrasonic NDE.

Acknowledgements

This work was supported in part by the Spanish Ministry of Science and Innovation under Grant BES-2005-7704, BES-2008-008675 and the projects DPI2007-65408-C02-01 and TRA2007-67711/AUT.

References

[1] B.D. Steinberg, Principles of Aperture and Array System Design, Wiley-Interscience, 1976.
[2] G.S. Kino, Acoustic Waves, Devices, Imaging and Analog Signal Processing, Prentice-Hall, 1987.
[3] S. McLaren, J.P. Weight, Transmit-receive mode response from finite-sized targets in fluid media, J. Acoust. Soc. Amer. 82 (1987) 2102–2112.
[4] A. Lhkmery, R. Raillon, Impulse-response method to predict echo responses from targets of complex geometry. Part II. Computer implementation and experimental validation, J. Acoust. Soc. Amer. 95 (1994) 1790–1800.
[5] B. Piwakowski, K. Sbai, A new approach to calculate the field radiated from arbitrarily structured transducer arrays, IEEE Trans. Ultrason. Ferroelect. Freq. Contr. 46 (2) (1999) 422–439.
[6] L. Wan, P.C. Pedersen, S.K. Jespersen, Modeling of received signals from annular array ultrasound transducers due to extended reflectors, IEEE Trans. Ultrason. Ferroelect. Freq. Contr. 50 (11) (2003) 1525–1541.
[7] F. Buiochi, O. Martínez, L.G. Ullate, F. Montero de Espinosa, A computational method to calculate the longitudinal wave evolution caused by interfaces between isotropic media, IEEE Trans. Ultrason. Ferroelect. Freq. Contr. 51 (2) (2004) 181–192.
[8] J.A. Jensen, A model for the propagation and scattering of ultrasound in tissue, J. Acoust. Soc. Amer. N. 89 (1) (1991) 182.
[9] Joon-Hyun Lee, Sang-Woo Choi, A parametric study of ultrasonic beam profiles for a linear phased array transducer, IEEE Trans. Ultrason. Ferroelect. Freq. Contr. 47 (May 2000) 644.
[10] Pinar Crombie, Peter A.J. Bascom, Richard S.C. Cobbold, Calculating the pulsed response of linear arrays Accuracy versus computational efficiency, IEEE Trans. Ultrason. Ferroelect. Freq. Contr. 44 (September 1997) 997.
[11] N.F. Haines, D.B. Langston, The reflection of ultrasonic pulses from surfaces, J. Acoust. Soc. Amer. 67 (5) (1980) 1443–1454.
[12] G.S. Garrett, J.N. Tjotta, R.L. Rolleigh, S. Tjotta, Reflection of parametric radiation from a finite planar target, J. Acoust. Soc. Amer. 75 (5) (1984) 1462–1472.
[13] P.C. Pedersen, D.P. Orofino, Modeling of received ultrasound signals from finite planar targets, IEEE Trans. Ultrason. Ferroelect. Freq. Contr. 43 (2) (1996) 303–311.
[14] R. Lerch, H. Landes, H.T. Kaarmann, Finite element modeling of the pulse-echo behavior of ultrasound transducers, in: Proc. IEEE Ultrason. Symp., Cannes, France, 1994 pp. 1021–1025.

[15] N.N. Kishore, I. Sridhar, N.G.R. Iyengar, Finite element modelling of the scattering of ultrasonic waves by isolated flaws, NDT E Int. 33 (5) (2000) 297–305.
[16] J.A. Ogilvy, Computerized ultrasonic ray tracing in austenitic steel, NDT Int. 18 (2) (Apr. 1985) 67–77.
[17] P. Calmon, A. Lhkmery, L. Paradis, Modeling of ultrasonic fields and their interaction with defects, in: Proc. IEEE Ultrason. Symp., Sendai, Japan, 1998 pp. 779–783.
[18] H.-J. Kim, L.W. Schmerr Jr., A. Sedov, Generation of the basis sets for multi-Gaussian ultrasonic beam models: An overview, J. Acoust. Soc. Amer. 119 (2006) 1971–1978.
[19] J.A. Jensen Field, A program for simulating ultrasound systems, Med. Biol. Eng. Comput. 34 (1) (1996) 351–353, Suppl. 1.
[20] CIVA: Simulation software for Non-Destructive Testing, http://www-civa.cea.fr.
[21] B.W. Drinkwater, P.D. Wilcox, Ultrasonic arrays for non-destructive evaluation: A review, NDT E Int. 39 (7) (2006) 525–541.
[22] E. Lindholm, J. Nickolls, S. Oberman, J. Montrym, NVIDIA Tesla: A unified graphics and computing architecture, IEEE Micro 28 (March–April 2008) 39.
[23] NVIDIA Corporation, NVIDIA CUDA 2.1 Programming Guide, February 2009.
[24] Cuda Zone, courses, resources, tools, etc., http://www.nvidia.com/object/cuda_home.html.
[25] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable parallel programming with CUDA, Queue 6 (March–April 2008) 40.
[26] Fastest Fourier Transform in the West, http://www.fftw.org.
[27] Steven W. Smith, Digital Signal Processing, Analog Devices, 1998, pp. 110–123.

2.3 Técnicas GPGPU para acelerar el modelado de sistemas ultrasónicos

SciVerse ScienceDirect
Disponible en www.sciencedirect.com

Revista Iberoamericana de Automática e Informática industrial 9 (2012) 282–289

R I A I
www.elsevier.es/RIAI

Técnicas GPGPU para acelerar el modelado de sistemas ultrasónicos

D. Romero-Laorden*, O. Martínez-Graullera, C.J. Martín-Arguedas, M. Pérez, L.G. Ullate

Centro de Acústica Aplicada y Evaluación No Destructiva (CAEND CSIC-UPM) Arganda del Rey, Madrid, España.

Resumen

El desarrollo de sistemas de simulación de campo acústico en tiempo real para aplicaciones de Evaluación no Destructiva ultrasónica constituiría una herramienta muy útil tanto para la planificación de las inspecciones como para la interpretación de los resultados de evaluaciones in-situ. Sin embargo, son algoritmos que requieren una alta capacidad de cómputo, no tanto por su complejidad sino por el gran número de puntos a analizar, lo que limita su uso al laboratorio sobre estaciones de trabajo de altas prestaciones. Los recursos de paralelización que actualmente ofrecen los sistemas informáticos, como son los procesadores multicore o las técnicas GPGPU, constituyen una oportunidad muy interesante para el desarrollo de este tipo de aplicaciones. Este trabajo analiza el modelo de paralelización de ambas alternativas con objeto de desarrollar un sistema portable de simulación de campo ultrasónico para tiempo real. Se describen por tanto los cambios en el algoritmo de cálculo de campo acústico para adaptarlo a una estrategia GPGPU y se valora el coste computacional de ambas implementaciones. *Copyright © 2012 CEA. Publicado por Elsevier España, S.L. Todos los derechos reservados.*

Palabras Clave:
algoritmos paralelos, sistemas ultrasónicos, GPGPU

1. Introducción

Actualmente las técnicas de Evaluación No Destructiva o END juegan un rol importante tanto en la caracterización como en la detección de defectos en materiales y estructuras. Su objetivo es usar algún tipo de fenómeno que al interaccionar con el objeto de interés permita medir algunas propiedades en el mismo sin alterarlo. Las técnicas de END basadas en ultrasonidos se basan en un principio básico: un sistema electrónico excita uno o más transductores ultrasónicos que generan pulsos de ondas mecánicas de alta frecuencia que se introducen y propagan en el material a inspeccionar. Del análisis de los cambios que se producen sobre esta misma onda o sobre sus reflexiones es posible extraer diversas características como la presencia de discontinuidades, que pueden ser identificadas como defectos (Steinberg, 1976; Kino, 1987).

Hoy en día se tiende a inspeccionar estructuras de geometría cada vez más compleja, mediante agrupaciones de transductores (conocidos como arrays) que pueden contener cientos de elementos. Para resolver las dificultades asociadas al análisis de los datos de la evaluación se precisa del apoyo de técnicas de simulación que permitan obtener una referencia del resultado de la misma. En este sentido, las técnicas de END basadas en ultrasonidos hacen uso de distintos modelos de simulación (Piwakowsky and Sbai, 1999; Choi, 2000; Pinar Crombie, 1997) tanto para la planificación de inspecciones como para la obtención de "huellas sónicas"que son usadas como base a la hora de determinar el estado del componente bajo análisis. Sofisticados paquetes de software como el CIVA (Cea, 2003) han sido desarrollados con este objetivo. Sin embargo su uso esta principalmente limitado al laboratorio, ya que la simulación del campo acústico es un proceso que requiere de una gran potencia de cálculo y tiene un coste computacional alto. No tanto por la complejidad del modelo de simulación sino por el gran número de puntos sobre los que se debe calcular y por los altos requerimientos de resolución que exige la zona de campo cercano. Este hecho limita la eficacia de los modelos de simulación como herramienta de campo ya que el evaluador no puede hacer uso de ella durante la inspección, donde le permitiría adaptar la planificación a las circunstancias reales o facilitaría la interpretación de los resultados. Se han propuesto distintos modelos basados en aproximaciones que reducen el tiempo de computación (Jensen, 1991) pero los resultados aún están fuera del tiempo real. Sin embargo, la mayoría de los métodos de modelado en END presentan características comunes: el algoritmo computacional utilizado se aplica de manera iterativa a una gran cantidad de puntos, aumentando el tiempo de ejecución y haciendo por tanto muy difícil su uso en sistemas que demandan tiempo real. Sin

*Autor en correspondencia.
Correo electrónico: `david.romero@csic.es` (D. Romero-Laorden)

http://dx.doi.org/10.1016/j.riai.2012.05.002

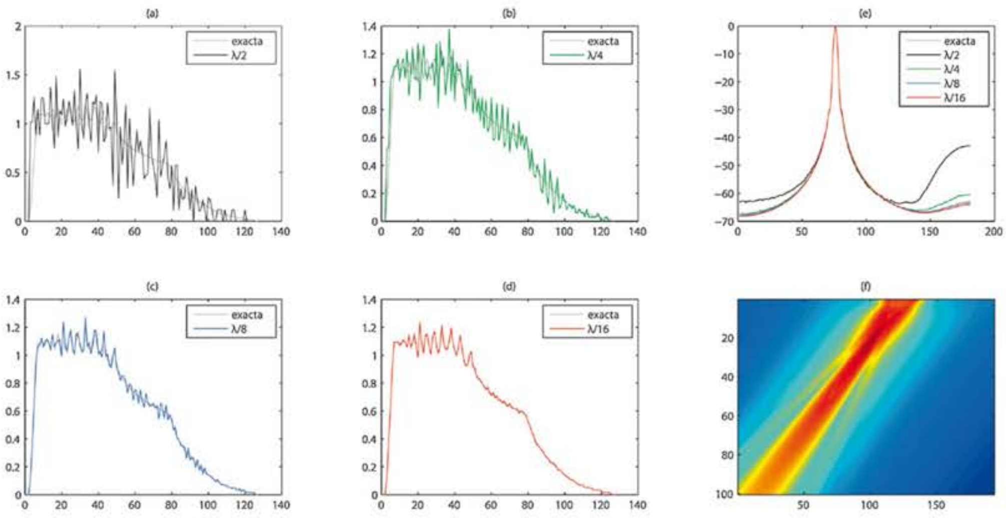


Figura 1: Ejemplo de la discretización de la apertura. Comparativa de la respuesta exacta con (a) la respuesta discreta con resolución espacial $\frac{\lambda}{2}$ (b) la respuesta discreta con resolución espacial $\frac{\lambda}{4}$ (c) la respuesta discreta con resolución espacial $\frac{\lambda}{8}$ (d) la respuesta discreta con resolución espacial $\frac{\lambda}{16}$ (e) valor del campo acústico para todos los casos expuestos (f) campo de presión generado sobre un plano situado en frente del array. Resolución temporal en todos los casos es de $\frac{\lambda}{32c}$.

embargo, estas características los hacen idóneos para la computación paralela.

La progresiva popularización de sistemas que explotan el paralelismo como vía para aumentar la capacidad de procesamiento, como los procesadores multicore o las técnicas GPGPU, suponen alternativas interesantes para el desarrollo de este tipo de problemas. Ambas plataformas plantean soluciones para aplicaciones que precisen alta capacidad de procesamiento a bajo coste. En nuestro caso nos permiten desarrollar soluciones para cálculo en tiempo real sacando la simulación del laboratorio y llevándolo a las zonas de trabajo, aumentando la capacidad del evaluador para determinar la gravedad del problema detectado. Sin embargo, debido a la naturaleza de ambos procesadores los estilos de paralelización son radicalmente distintos e involucran modelos de programación muy diferentes.

El modelo multicore basado en la integración de procesadores de altas prestaciones en un sólo chip, cada uno con su propia memoria cache L1 y una serie de recursos compartidos entre todos (cache L2, etc). Estos sistemas, diseñados para el acceso aleatorio a memoria, permiten implementar estructuras de datos complejas y un modelo de desarrollo que explota el paralelismo a nivel de *thread* (thread-level parallelism), que en nuestro caso es fácil desarrollar a nivel de punto espacial.

En el modelo GPGPU (*General Purpose Computation on Graphics Processing Units*) donde el número de procesadores es mucho mayor, pudiendo llegar a centenares, pero con unos recursos mucho más reducidos (pocos registros internos, limitaciones aritméticas) y particulares modos de acceso a memoria (E.Lindholm and Montrym, 2008), la paralelización no es un proceso sencillo. Para adaptar los accesos a memoria y las estructuras de datos a las características de la GPU es necesario un análisis complejo del problema a paralelizar. Aunque ya existen herramientas que facilitan tanto el desarrollo como la portabilidad del código (Nvidia, 2012b; J. Nickolls and Skadron, 2008), un conocimiento de esta arquitectura permite una mejor optimización del algoritmo.

Pese a que cualquier algoritmo que es implementable en una CPU lo es también en una GPU, éstas implementaciones no son igual de eficientes. Este trabajo analiza ambos modelos de desarrollo para el problema de cálculo de campo acústico, basándonos en el modelo de la respuesta espacial al impulso desarrollada por Piwakowsky (Piwakowsky and Sbai, 1999). Se analiza en profundidad la solución para GPU, por sus peculiaridades y por el cambio que supone frente al modelo de paralización convencional. El objetivo es reducir el alto coste asociado a los algoritmos y facilitar su implementación en sistemas que permitan a la larga el desarrollo de soluciones de tiempo real, lo que de alguna forma permitan desarrollar aplicaciones para operar en campo.

2. Modelado en END

La expresión general de la presión acústica generada por un array que trabaja en emisión y recepción viene dada por (Jensen, 1991):

$$p(\vec{x},t) = \frac{1}{c^2} * \frac{\partial^2 v(t)}{\partial t^2} * h_E(\vec{x},t) * h_R(\vec{x},t) \tag{1}$$

donde h_E y h_R son, respectivamente, las respuestas al impulso en emisión y recepción del array:

$$h_E(\vec{x},t) = \sum_{i=1}^{N} a_i^E h_i(\vec{x}, t - T_i^E) \tag{2}$$

$$h_R(\vec{x},t) = \sum_{i=1}^{N} a_i^R h_i(\vec{x}, t - T_i^R) \tag{3}$$

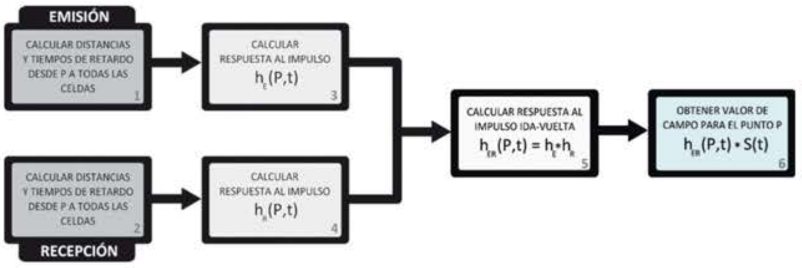

EMISIÓN
CALCULAR DISTANCIAS Y TIEMPOS DE RETARDO DESDE P A TODAS LAS CELDAS 1
CALCULAR RESPUESTA AL IMPULSO $h_E(P,t)$ 3
CALCULAR DISTANCIAS Y TIEMPOS DE RETARDO DESDE P A TODAS LAS CELDAS 2
CALCULAR RESPUESTA AL IMPULSO $h_R(P,t)$ 4
RECEPCIÓN
CALCULAR RESPUESTA AL IMPULSO IDA-VUELTA $h_{ER}(P,t) = h_E * h_R$ 5
OBTENER VALOR DE CAMPO PARA EL PUNTO P $h_{ER}(P,t) * S(t)$ 6

Figura 2: Algoritmo iterativo de Piwakowsky. El cálculo del campo acústico en emisión y recepción requiere seis pasos

En la expresión anterior, $h_i()$ es la respuesta al impulso del i-ésimo elemento y a_i es su peso asociado. T_i^E y T_i^R son los retardos de focalización en emisión y recepción respectivamente, que para un punto $p(\vec{x})$ vienen dados por la expresión:

$$T_p(i) = \frac{|\vec{x} - \vec{x_i}| - |\vec{x}|}{c} \tag{4}$$

donde c es la velocidad del sonido en el medio inspeccionado, y $\vec{x_i}$ es el vector de posición del i-ésimo elemento. Para poder calcular el campo acústico se ha seguido el método de Piwakowsky (Piwakowsky and Sbai, 1999), que hace una computación directa de la integral de Rayleigh por medio de muestras temporales a intervalos Δt y discretiza las superficie del transductor en celdas cuadradas de área ΔS. La contribución de cada elemento a la respuesta al impulso en el instante dado t_S se obtiene añadiendo las contribuciones de cada una de las celdas contenidas entre dos ondas esféricas concéntricas de radio inferior a $c\Delta t_S$, que están separadas por el intervalo de discretización $c\Delta t$ (Piwakowsky and Sbai, 1999). Así, la contribución a la respuesta al impulso correspondiente al i-ésimo elemento del array en el instante $t = t_s$ es:

$$h_i(t = t_S) = \frac{1}{\Delta t}\sum_j b_j \qquad t_S - \Delta t/2 \leq t_j \leq t_S + \Delta t/2 \tag{5}$$

$$b_j = \frac{a_i \Delta S}{2\pi R_j} \text{ y } t_j = \frac{R_j}{c} + T_i \tag{6}$$

donde T_i denota el retardo de focalización en emisión o recepción y $\frac{R_j}{c}$ es el tiempo de propagación. De este modo, el valor del campo resultante viene dado por la siguiente ecuación:

$$P(\vec{x}) = max_t(abs(p(\vec{x}, t))) \tag{7}$$

A pesar de ser un modelo aproximado, el método de Piwakowsky tiene la peculiaridad de poder aplicarse a cualquier tipo de aperturas y geometrías. El error de cómputo entre los patrones discretos y la solución analítica exacta ha sido estudiado en (Piwakowsky and Sbai, 1999), concluyendo que hay una alta precisión cuando tanto la resolución espacial como temporal son del orden de $\frac{\Delta x}{\lambda} = 0{,}01$ y $\Delta t f = 0{,}01$ respectivamente, donde f es la frecuencia central de la onda ultrasónica.

Los efectos acerca del problema de la discretización para el caso de un array de 32 elementos (tabla 1) pueden apreciarse en la figura 1. En concreto, las figuras (a), (b), (c) y (d) muestran las diferencias existentes entre las respuestas obtenidas en el cálculo de la respuesta al impulso por el método de Piwakowsky para una resoluciones de $\frac{\lambda}{2}$, $\frac{\lambda}{4}$, $\frac{\lambda}{8}$ y $\frac{\lambda}{16}$ respectivamente con respecto a la respuesta propuesta por Piwakowsky calculadas sobre un punto situado a una profundidad de 10mm para una focalización deflectada 15° en elevación. Como vemos, el incremento de la resolución espacial ayuda a reducir el error introducido por la discretización, aunque al mismo tiempo el coste computacional aumenta sustancialmente. La mayoría de los errores introducidos por la discretización son ruido de alta frecuencia, y por tanto son eliminados por la onda al actuar ésta como un filtro. Este efecto de filtro se hace patente sobre los resultados de la figura 1, donde se muestra que una resolución espacial de $\frac{\lambda}{16}$ es adecuada y ofrece una solución de compromiso entre el número de puntos en la apertura y la exactitud del resultado. Esto es así ya que la propuesta de Piwakowsky es fuertemente restrictiva al definir estos límites para determinar la exactitud sobre la respuesta espacial al impulso y no sobre la onda de presión resultante. Por otro lado, en la figura 1(e) se muestra el valor de campo para todas la resoluciones en decibelios donde los errores de la discretización se hacen evidentes principalmente para el caso de $\frac{\lambda}{2}$. Por último, el campo de presión sobre un plano situado en frente del array se muestra en la figura 1(f) como resultado de la simulación.

2.1. *Algoritmo computacional*

La figura 2 describe esquemáticamente el algoritmo computacional utilizado para el cálculo del campo acústico usando el método de Piwakowsky. Más en detalle, se muestran las etapas necesarias para calcular el valor del campo en cada punto espacial P, siendo por tanto, un proceso iterativo que se aplica sobre todo el conjunto de puntos del espacio. A continuación se describen cada una de las etapas:

- En los pasos 1 y 2, se calculan las distancias desde todas las celdas pertenecientes a la apertura a cada punto del espacio P en emisión y recepción respectivamente. Además, se calcula información adicional como son los tiempos de retardo de focalización (ec. 4) y los pesos de la apodización o filtro espacial, y se asocian a sus respectivas celdas (ec. 6).

- Después, en los pasos 3 y 4, se calcula la respuesta al impulso del array en emisión $h_E(P, t)$ y en recepción $h_R(P, t)$ (ecs. 2 y 5).
- A continuación, se hace la convolución entre h_E y h_R, obteniendo la respuesta al impulso en ida y vuelta del array $h_{ER}(P, t)$ (ec. 1).
- Finalmente, se obtiene el campo acústico en ida y vuelta, realizando la convolución entre $h_{ER}(P, t)$ y la señal emitida (ec. 1).

Por último, cabe remarcar un aspecto importante. La longitud de las respuestas al impulso h depende fundamentalmente de la distribución de los elementos del array en el tiempo. Estos valores pueden ser corregidos mediante la aplicación de lentes de focalización, de tal manera que acortarán la longitud de las respuestas al impulso en el foco mientras que se verán alargadas fuera de él.

2.2. *Modelo de cálculo*

Para poder evaluar las capacidades y ventajas de la computación paralela, se ha simulado un ejemplo de cálculo del campo generado en emisión y recepción con un array lineal emitiendo en un medio homogéneo (en este caso, agua). Las características del modelo están resumidas en la tabla 1. Como vemos, el array tiene $N = 32$ elementos con una separación entre elementos (*pitch*) $d = \frac{\lambda}{2}$. El ancho de los elementos es $a = d$ y su altura es $b = 10\lambda$. Los elementos emiten pulsos de 50 % de ancho de banda. Para la evaluación de los resultados se ha calculado el máximo de presión en 19300 puntos pertenecientes a un plano situado frente al array.

Tabla 1: Parámetros del modelo de simulación

Medio	Agua
Velocidad del medio	1540 m/s
Tamaño del array	32 elementos
Ancho de los elementos (*pitch*)	$\frac{\lambda}{2}$mm
Altura de los elementos	10λmm
Frecuencia central	3 MHz
Ancho de banda	50 %
Número de puntos de campo	19300 (48.25mm×101mm)
Resolución espacial	(res_x, res_z)=(0.25, 1.01)

3. Infraestructuras de cómputo paralelo

Antes de discutir el diseño de nuestro algoritmo, es necesario comentar brevemente algunos detalles relativos a las arquitecturas multicore en CPU como en GPU. En estos últimos años, una de las tendencias dominantes en las arquitecturas de microprocesadores ha sido el continuo incremento del paralelismo a nivel de chip. Hoy en día, vemos algo común trabajar con CPUs compuestas de varios núcleos de proceso (*cores*) y probablemente la tendencia seguirá incrementando aún más el paralelismo con nuevos *chips* formados por múltiples núcleos. Un procesador multicore CPU, está compuesto por dos o más procesadores independientes, encargados de leer y ejecutar las instrucciones del programa. Estas instrucciones son las instrucciones típicas de una CPU (como sumar, mover datos o instrucciones condicionales) pero con la ventaja de que los distintos núcleos son capaces de ejecutar varias instrucciones al mismo tiempo aumentando considerablemente la velocidad general de los programas que son susceptibles de computación paralela.

En cambio, las unidades de procesamiento gráficos o GPUs (*Graphics Processor Units*) son arquitecturas muy poderosas computacionalmente hoy día que pueden estar formadas por centenares de *cores*. De hecho, es común tratar con GPUs de 128, 240, 256 o 512 *cores* (arquitectura Fermi) aunque las actuales NVIDIA GPUs (Nvidia, 2012a) ya pueden contener hasta 1358 *cores* (arquitectura Kepler) y éstas pueden ser programadas directamente en código C usando las tecnologías CUDA (Nvidia, 2012b; J. Nickolls and Skadron, 2008) u OpenCL.

Con el objetivo de obtener el mayor provecho posible de una GPU, se ha considerado apropiado describir las características generales de la arquitectura hardware de una tarjeta gráfica para su programación en CUDA. Para tal fin, una comparativa entre las arquitecturas de una CPU multicore y una GPU se muestra esquemáticamente en la figura 3. Como vemos, una GPU se compone de una serie de multiprocesadores (MPs) (desde 1 a 30 dependiendo de la arquitectura considerada) que a su vez están compuestos por varios procesadores escalares (SPs) (varían entre 8 a 192 según el tipo de GPU). Además, cada MP contiene una memoria on-chip (memoria compartida) con muy baja latencia y alto ancho de banda, similar a una cache L1 de una CPU. Las diferencias son apreciables y tanto la CPU como la GPU se encargan de gestionar su propia memoria, siendo la primera conocida como memoria del *host* (que es directamente la memoria RAM del PC) mientras que la segunda, llamada memoria del dispositivo, *device* o global, es la memoria de la tarjeta gráfica que es usada por todos los multiprocesadores de la GPU.

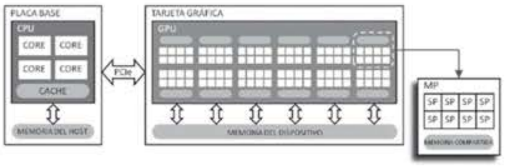


Figura 3: CPU de cuatro-cores frente a una tarjeta gráfica de 240 cores (30 MPs de 8 SPs cada uno)

El concepto fundamental del modelo de programación CUDA es trabajar con cientos de hilos que ejecutan la misma función pero con diferentes datos. Por tanto, una aplicación se organiza como un programa secuencial en la CPU que incluye funciones especiales paralelas, llamadas *kernels*, que son ejecutadas varias veces sobre diferentes datos. El programador organiza los datos a procesar por un *kernel* en estructuras lógicas que consisten en bloques de hilos y rejillas de bloques. Así, una rejilla o *grid* es una estructura 1D, 2D o 3D de bloques y un bloque es una estructura 1D, 2D o 3D de hilos. Únicamente se puede definir un *grid* dentro de un *kernel*. Asociado a cada función paralela existen una serie de parámetros que es necesario conocer si se quiere maximizar el rendimiento, siendo los más importantes el número de registros utilizados por cada hilo y la ocupación de los MPs. Para una completa descripción sobre la arquitectura y el modelo de programación puede consultarse la extensa documentación disponible (Nvidia, 2012b).

4. Algoritmos paralelos

Los candidatos perfectos para la computación paralela son aquellas funciones que son ejecutadas muchas veces independientemente sobre diferentes datos y en este sentido, las características propias de los algoritmos utilizados en END normalmente implican operar con un gran volumen de datos y el cálculo de distintos resultados de manera repetitiva, lo que les hace ser unos candidatos idóneos para la computación paralela.

En este trabajo se evalúan tres implementaciones distintas, siendo dos de ellas desarrolladas exclusivamente en CPU. La primera de las soluciones que se ha desarrollado es la puramente tradicional, que utiliza un único núcleo como cerebro para el cálculo el campo acústico. En todos los casos, los algoritmos se han programado en C y ejecutados en Matlab® por simplicidad para el desarrollo.

4.1. *Paralelización multi-core*

Trasladar el algoritmo de cómputo de campo a una arquitectura multicore en CPU es un proceso relativamente sencillo. Por ello, la paralelización ha consistido en utilizar un hilo de ejecución en CPU por cada uno de los puntos del espacio, distribuyendo el trabajo entre el máximo de procesadores disponibles en nuestro equipo. Para tal fin, y puesto que todas las librerías de campo habían sido desarrolladas con anterioridad utilizando el entorno de cálculo matemático Matlab®, la paralelización se ha realizado usando las herramientas para cómputo paralelo que éste entorno nos proporciona, consiguiendo de una manera rápida una buena distribución de los recursos disponibles y de la memoria.

4.2. *Paralelización en GPU*

Desafortunadamente, una implementación directa de los algoritmos diseñados para CPU a GPU no es normalmente la mejor opción por lo que es necesario definir una estrategia de paralelización apropiada de los algoritmos para poder obtener el máximo beneficio. Las notables diferencias que existen entre las arquitecturas CPU y GPU hacen necesario comentar brevemente algunas consideraciones sobre la computación en GPU.

En primer lugar, las transacciones entre CPU y GPU son bastante lentas, por lo que deben ser minimizadas lo máximo posible para mejorar el rendimiento global. Además, la memoria del dispositivo es limitada por lo que en muchos casos puede ser necesario tener que reducir el volumen de datos que se debe procesar en paralelo. El tiempo de lectura desde la memoria del dispositivo es lento y es recomendable por tanto usar algunos otros mecanismos de acceso, como la memoria de textura (que es cacheable y rápida) o la memoria compartida (disponible en cada MP y por tanto accesible únicamente entre hilos de un mismo bloque) siempre que sea posible. Finalmente, simplificar las operaciones por hilo y homogeneizar el tiempo de ejecución de todos los hilos puede resultar muy beneficioso así como minimizar las escrituras a memoria de la GPU.

En este trabajo se han evaluado distintas alternativas. Debido al hecho de que el algoritmo se aplica a cada punto del espacio, la paralelización directa sería por puntos al igual que sucede en el caso multicore calculando todos los puntos simultáneamente. Pero la arquitectura es distinta, y tenemos mucho más paralelismo, por lo que hay otras posibilidades que pueden ser tomadas en cuenta. Las diferentes etapas del algoritmo implican diferentes requisitos computacionales, así que quizás podría intentarse una paralelización por celdas espaciales de la apertura, muestras de las respuestas en emisión-recepción u otras.

Tabla 2: Configuraciones de los distintos *kernels* para cada estrategia, tamaño de bloque t_b, número de bloques en x e y n_{bx} y n_{by}, número de puntos n_p, número de registros por *kernel* n_{reg}, ocupación de los multiprocesadores $ocup$, número de celdas n_c, número de muestras de las respuestas en ida-vuelta $m_{h_{iv}}$ y número de muestras de la convolución con la onda m_c.

Kernels	A	B
1	$t_b = 128$	$t_b = 128$
	$n_{bx} = n_p/t_b$	$n_{bx} = n_c/t_b$
	$n_{by} = 1$	$n_{by} = n_p$
	$n_{reg} = 15$	$n_{reg} = 13$
	$ocup = 100\,\%$	$ocup = 100\,\%$
2	$t_b = 128$	$t_b = 64$
	$n_{bx} = n_p/t_b$	$n_{bx} = n_h/t_b$
	$n_{by} = 1$	$n_{by} = n_p$
	$n_{reg} = 19$	$n_{reg} = 5$
	$ocup = 75\,\%$	$ocup = 100\,\%$
3	$t_b = 64$	$t_b = 128$
	$n_{bx} = n_p/t_b$	$n_{bx} = m_{h_{iv}}/t_b$
	$n_{by} = 1$	$n_{by} = n_p$
	$n_{reg} = 17$	$n_{reg} = 12$
	$ocup = 93\,\%$	$ocup = 100\,\%$
4	$t_b = 128$	$t_b = 128$
	$n_{bx} = n_p/t_b$	$n_{bx} = m_c/t_b$
	$n_{by} = 1$	$n_{by} = n_p$
	$n_{reg} = 19$	$n_{reg} = 10$
	$ocup = 75\,\%$	$ocup = 100\,\%$

Para el desarrollo de los algoritmos paralelos en GPU se han diseñado 4 *kernels* bien diferenciados que cubren cada una de la etapas definidas anteriormente: *Kernel 1*, para calcular las distancias y tiempos de retardo desde cada punto espacial a todas las celdas en emisión y recepción (etapas 1 y 2); *Kernel 2*, que computa la respuesta espacial al impuso con una implementación paralela del método de Piwakowsky en emisión y recepción (etapas 3 y 4); *Kernel 3*, que realiza la convolución en ida y vuelta (etapa 5); y *Kernel 4* obtiene el valor final de campo (etapa 6).

A partir de la definición operativa de estos *kernels*, la paralelización se puede realizar a dos niveles que corresponden con las dos estrategias identificadas en la figura 4. La primera sigue el patrón convencional seguido para la implementación multicore (paralelización por punto espacial), lo que hemos definido como un modelo de grano grueso. La segunda, es de grano más fino y la paralelización se lleva a cabo a partir de las celdas espaciales de la apertura y los tamaños de los vectores. Asimismo, en la tabla 2 se muestran los distintos parámetros asociados a ambos niveles, relativos al tamaño de bloque, número de bloques, la ocupación de los multiprocesadores, registros, etc.

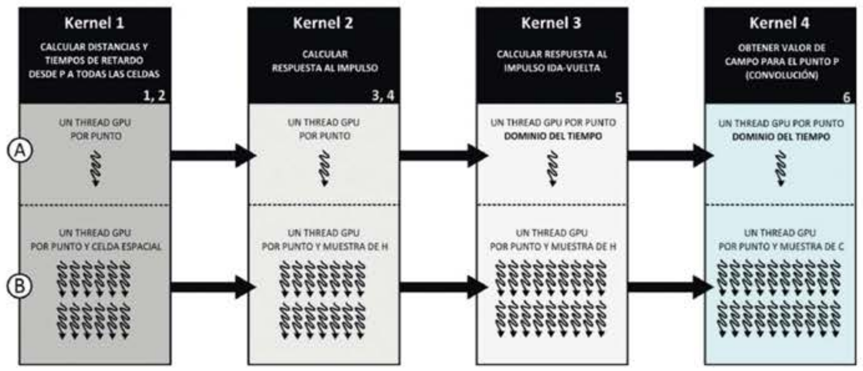

Kernel 1
CALCULAR DISTANCIAS Y TIEMPOS DE RETARDO DESDE P A TODAS LAS CELDAS
1, 2
Kernel 2
CALCULAR RESPUESTA AL IMPULSO
3, 4
Kernel 3
CALCULAR RESPUESTA AL IMPULSO IDA-VUELTA
5
Kernel 4
OBTENER VALOR DE CAMPO PARA EL PUNTO P (CONVOLUCIÓN)
6
A
UN THREAD GPU POR PUNTO
UN THREAD GPU POR PUNTO
UN THREAD GPU POR PUNTO DOMINIO DEL TIEMPO
UN THREAD GPU POR PUNTO DOMINIO DEL TIEMPO
B
UN THREAD GPU POR PUNTO Y CELDA ESPACIAL
UN THREAD GPU POR PUNTO Y MUESTRA DE H
UN THREAD GPU POR PUNTO Y MUESTRA DE H
UN THREAD GPU POR PUNTO Y MUESTRA DE C

Figura 4: Esquemas de las dos estrategias estudiadas para GPU en este trabajo: (A) un thread por punto de campo, convolución en el dominio del tiempo, (B) solución paralela específica adaptada a cada etapa del algoritmo

4.3. *Paralelización de grano grueso*

Como se ha comentado antes, está primera estrategia consiste en lanzar un *thread* (simbolizado con una flecha ondulada en la figura 4) para cada punto del campo y cada paso del algoritmo. El tamaño del *grid*, que en este caso es idéntico a todos los *kernels*, tiene unas dimensiones de $(ceil(\frac{n_p}{t_b}) \times 1)$ bloques de tamaño $t_b = 128$. Esto proporciona una buena ocupación de los multiprocesadores pero que no llega a ser del 100 % en todos los casos (ver tabla 2). Cada resultado parcial (distancias, retardos, respuestas al impulso) es transferido a memoria de textura (que es cacheable) para usarlo en las etapas posteriores. De este modo, se optimizan los accesos a memoria y se gana en velocidad. A pesar de doblar la velocidad obtenida en CPU, los requisitos de memoria eran muy grandes y se comprobó que era demasiado lento debido al gran tiempo consumido por las funciones de convolución de las dos últimas etapas como factores más determinantes.

4.3.1. *Paralelización de grano fino*

A raíz de los problemas identificados se intentó mejorar el uso de memoria y acelerar el tiempo de cómputo de cada *kernel*, explotando el poder de la GPU, diseñando una solución paralela a más bajo nivel (esto es, para cada etapa del algoritmo de la figura 2.1) y desarrollando nuevas estructuras de datos que permitieran incrementar el uso de recursos del hardware. A continuación se detallan las optimizaciones y cambios llevados a cabo en cada en cada *kernel*:

- *Kernel 1*. La paralelización se hace por punto y por celda como se observa en la figura 4. El tamaño de bloque se ha fijado en $t_b = 128$ en un *grid* de dimensión $(ceil(\frac{n_p}{128}) \times n_c)$. En ese sentido, el *kernel* se ha simplificado hasta la mínima expresión, calculando las distancias y los retardos muy rápido (raíz cuadrada, sumas y multiplicaciones). Los resultados se almacenan en vectores de tuplas (distancia, retardo) en memoria de textura.

- *Kernel 2*. Con el objetivo de reducir la memoria requerida para almacenar las respuestas al impulso en los pasos 3 y 4 - figura 2.1, el segundo *kernel* tiene dos funciones: la primera, tiene como objetivo ordenar las distancias de menor a mayor en función del primer campo de la tupla (ordenación bitónica) y sumar los retardos cuando las distancias son iguales, generando una estructura que llamaremos $Proto_{respuesta}$ que contiene únicamente la información esencial de h (se excluyen los valores nulos). De esta manera, la $Proto_{respuesta}$ en emisión se calcula muy rápido. A continuación, la respuesta completa en recepción se computa lanzando un *thread* por punto y por muestra de la h. Ambos resultados se almacenan en memoria de textura para beneficiarnos del acceso a memoria más eficiente.

- *Kernel 3*. La primera convolución utiliza las $Proto_{respuesta}$ en emisión y las respuestas en recepción calculadas en la etapa anterior. En este caso, el algoritmo desarrollado sigue el modelo *Input Side* (Smith, 1998) paralelo, donde se lanza un *thread* por punto y por muestra de la estructura que almacena la respuesta al impulso en ida y vuelta. De este modo, la calculamos de manera eficaz, gracias a que los vectores están ordenados, lo que permite acceder directamente el valor correspondiente para la muestra.

- *Kernel 4*. Para el último paso, se ha desarrollado un algoritmo paralelo optimizado en GPU y que está basado en el modelo de convolución *Ouput Side* (Smith, 1998). Calcula por tanto las convoluciones con la onda y permite obtener el resultado final para esa muestra al final de la ejecución del *kernel*, no siendo necesario almacenar el resultado pues sólo nos interesa obtener el máximo. Para ello, se lanza un *thread* por punto y por muestra de la convolución en ida-vuelta, se computa el valor (minimizando los accesos de lectura y escritura a memoria) y se almacena en memoria compartida. Una vez todos los *threads* del mismo bloque han terminado su tarea, se obtiene el máximo por bloque y se realiza una reducción del conjunto de máximos por bloque para cada punto, calculando el máximo global y obteniendo así el valor final del campo.

Tabla 3: Tiempo en segundos para calcular el campo acústico en 19300 puntos de campo para distintas configuraciones en CPU y GPU

Resolución Espacial		**CPU (1 core)**			**CPU (4 cores)**			**GPU**					
Tamaño de la celda	Puntos en la apertura	Resolución Temporal			Resolución Temporal			Resolución Temporal			Iteraciones		
		$\frac{\lambda}{32c}$	$\frac{\lambda}{64c}$	$\frac{\lambda}{128c}$	$\frac{\lambda}{32c}$	$\frac{\lambda}{64c}$	$\frac{\lambda}{128c}$	$\frac{\lambda}{32c}$	$\frac{\lambda}{64c}$	$\frac{\lambda}{128c}$	$\frac{\lambda}{32c}$	$\frac{\lambda}{64c}$	$\frac{\lambda}{128c}$
$\frac{\lambda}{2}$	1280	31.42	32.50	38.12	9.58	11.25	12.32	0.51	1.44	4.94	1	1	1
$\frac{\lambda}{4}$	2560	71.70	74.60	83.10	20.25	21.89	23.18	1.15	2.34	6.55	2	4	4
$\frac{\lambda}{8}$	5120	224.35	231.30	240.14	68.75	69.76	77.62	3.27	5.75	10.66	8	12	12
$\frac{\lambda}{16}$	10240	875.00	889.45	912.36	261.30	268.64	284.28	14.38	15.32	31.80	20	24	28
$\frac{\lambda}{32}$	20480	3241.42	3256.10	3289.51	984.10	1001.80	1019.60	91.12	105.7	178.60	48	56	64

Cabe por último comentar en ambos casos, que los datos correspondientes al modelo de simulación (esto es, el array y los puntos de campo) se almacenan en memoria constante de la GPU para conseguir un acceso lo más rápido posible. Respecto a la tabla 2, como vemos la paralelización de grano fino hace que los multiprocesadores estén al 100 % de actividad, no como sucedía en la solución directa. Y además, el número de registros usados en cada *kernel* también se ha visto reducido, dando como efecto una mejor optimización de cada una de las etapas que se refleja significativamente en el tiempo de cómputo.

5. Discusiones y resultados

Para testear los algoritmos paralelos desarrollados, se ha utilizado una tarjeta NVIDIA Quadro 4000 que contiene 256 cores de proceso con 2GB de memoria global. Se ha instalado en un PC Core 2 Quad y procesador Intel Q9450 2.66 GHz con 4GB de RAM.

Las implementaciones de los diversos algoritmos se han realizado desde el entorno de programación MATLAB® en su versión R2010a gracias a la interoperabilidad que CUDA proporciona con estos entornos de trabajo. Se ha llevado a cabo una comparación entre el tiempo de cómputo usando las tres configuraciones descritas: CPU (1 core), CPU (4 cores) y GPU para el modelo de simulación descrito en la sección 2.2. Se han calculado por tanto 19300 puntos de campo. Las resoluciones espaciales y temporales se han variado desde $\frac{\lambda}{2}$ a $\frac{\lambda}{32}$ y desde $\frac{\lambda}{32c}$ a $\frac{\lambda}{128c}$ respectivamente. Los cálculos se han realizado utilizando precisión simple. Un análisis exhaustivo acerca de la precisión en el cálculo de campo acústico sobre hardware gráfico está descrito en (Romero-Laorden, 2011) concluyendo que no hay diferencias apreciables entre el uso de precisión simple o doble en este tipo de aplicaciones.

La tabla 3 muestra los tiempos obtenidos para la solución monocore, multicore y GPU. Los tiempos obtenidos usando la GPU incluyen las transferencias a memoria global y la copia de resultados al *host*. Además, el límite de memoria impuesto por la GPU Quadro 4000 nos obliga a dividir nuestro cómputo lanzando grupos más pequeños de puntos, puesto que la memoria necesaria para almacenar los resultados parciales puede llegar a ser muy grande. Esto significa que irremediablemente tendremos una o más iteraciones dependiendo del caso como se puede ver en la última columna de la tabla 3.

Con respecto a las dos primeras implementaciones - CPU (1 core) y CPU (4 cores) - la primera es obviamente la más lenta al no existir ningún tipo de paralelización. En general, la tabla 3 indica que los tiempos de la solución multicore dividen entre 3 y 4 veces el tiempo de ejecución de la solución monocore. Para las soluciones que explotan el paralelismo por punto espacial, el aumento del número de puntos en la apertura supone un aumento significativo en el coste mucho más importante que al incrementar la resolución temporal. Esto se debe en parte al hecho de que las FFTs de los algoritmos en CPU trabajan con respuestas al impulso cuyas longitudes son normalizadas al valor más cercano potencia de dos. Sin embargo, esto no se produce en la solución GPU donde el aumento más importante en el coste se produce al incrementar el tiempo, disminuyendo el rendimiento global debido a un mayor número de ejecuciones.

Para los valores de resolución propuestos por Piwakowsky, los tiempos son de más de 7 horas en monocore y más de 2 en el caso multicore. Dado el filtro que introduce el pulso gaussiano de excitación mostrado en la sección 2 para el ejemplo desarrollado, estos requerimientos son excesivos siendo posible obtener un buen resultado para $\frac{\lambda}{16}$ y $t = \frac{\lambda}{32c}$ que corresponde con tiempos de 875.00, 261.30 y 15.32.

Finalmente, las figuras 5(a) y 5(b) muestran la ganancia en velocidad conseguida. La parte derecha de ambos gráficos corresponde con un alto número celdas en la apertura (más de 15000 celdas) que produce un decremento en la eficiencia de método. La causa principal se debe a que el número de iteraciones en estos casos es mayor debido a las limitaciones de memoria. Por otro lado, la parte izquierda de las figuras corresponde con un submuestreo espacial, donde no se obtiene tanto provecho en GPU. Respecto a la resolución temporal, la línea con cuadrados corresponde a intervalos de tiempo muy cortos (sobremuestreo temporal), demandando una mayor cantidad de memoria que aumenta el tiempo de ejecución. Las líneas con triángulos y círculos corresponden a valores más lógicos, siendo los más adecuados para resoluciones de $\frac{\lambda}{8}, \frac{\lambda}{16}$ y $\frac{\lambda}{32c}$ y $\frac{\lambda}{64c}$. Como podemos observar, para estos valores el rendimiento del algoritmo en GPU llega a ser de hasta 20 veces más rápido respecto a una implementación multicore en CPU.

6. Conclusiones

Este trabajo se ha centrado en explorar el paralelismo que se ofrecen en los sistemas de consumo para el desarrollo de algoritmos de modelado en END con vía a la implementación de sistemas de tiempo real. Así, se han comparado tres implementaciones de cálculo de campo acústico para el modelo de la

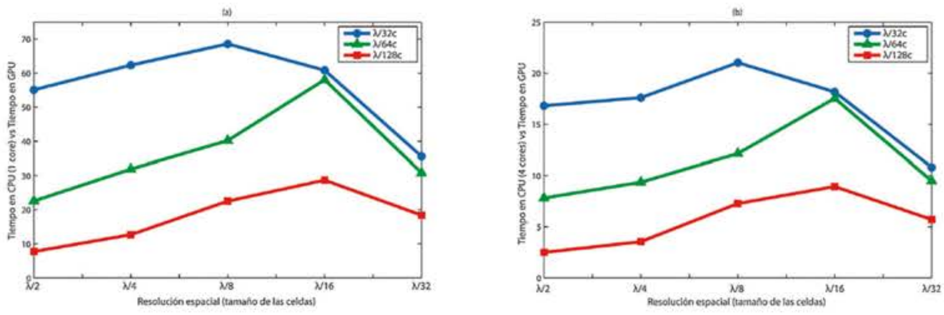


Figura 5: (a) Ganancia en velocidad CPU (1 core) vs GPU (b) Ganancia en velocidad CPU (4 cores) vs GPU

respuesta espacial al impulso basándonos en la solución aproximada de Piwakowsky. La solución convencional que correspondería a un sólo núcleo; la paralelización de cálculo a nivel de puntos de campo que corresponde a un desarrollo multicore; y por último la solución para GPU que explota el paralelismo dentro de las cuatro etapas del algoritmo. La paralelización multicore es particularmente sencilla ya que permite utilizar el algoritmo original respecto al monocore, y produce una mejora directamente proporcional al número de cores involucrados. Utilizando únicamente una tarjeta gráfica sencilla equipada con tecnología NVIDIA CUDA, nuestros resultados experimentales demuestran que el tiempo de cálculo para el campo acústico se ve drásticamente reducido con respecto a implementaciones en CPU (monocore y multicore), consiguiendo en algunos casos una ganancia de hasta 70 veces más rápida para el sistema monocore y de 20 para el sistema multicore. Los resultados demuestran que la GPU se adapta particularmente bien a los algoritmos con un alto grado de paralelismo, que no precisan necesidad de estructuras de datos complejas, y que exigen una alta intensidad aritmética. Ésta es una solución económica con muy buena relación entre potencia y coste. Además, es escalable y puede ser utilizada para acelerar muchos otros problemas de modelado en END, facilitando el desarrollo de implementaciones para tiempo real. El incremento en velocidad conseguido por la técnicas GPGPU es especialmente significativo si el número de hilos de ejecución y los recursos asociados se mantienen dentro de los límites físicos de la tarjeta y el número de ejecuciones es reducido.

English Summary

Using GPGPU techniques to accelerate modelling of ultrasonic systems.

Abstract

The development of acoustic field simulation in real time for non destructive ultrasonic evaluation applications would be an useful tool for both the planning and evaluation of inspections in-situ. However, they are algorithms which require high computing power, not due to their complexity but because of the large number of points to be analysed, which limits their use to laboratory workstations for high performance. The parallelization resources currently available in computer systems, such as multicore processors and GPGPU techniques, are a very interesting chance for the development of such applications. This work analyses the parallelization model of both alternatives in order to develop a portable ultrasonic field simulation system for real-time. The changes for both algorithms are described, in order to adapt it to GPGPU philosophy and a estimation of the computational cost of both implementations is given.

Keywords:

parallel algorithms, ultrasonic systems, GPGPU

Agradecimientos

Este trabajo está apoyado por el Ministerio de Economía y Competitividad a través del proyecto DPI2010-19376 y la beca BES-2008-008675.

Referencias

Cea, 2003. Civa: Simulation software for non destructive testing.
Choi, J.-H. L. S.-W., 2000. A parametric study of ultrasonic beam profiles for a linear phased array transducer. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control 47, 644–650.
E.Lindholm, J.Nickolls, S., Montrym, J., 2008. Nvidia tesla: A unified graphics and computing architecture. IEEE Micro 28 (2), 39–55.
J. Nickolls, I. Buck, M. G., Skadron, K., 2008. Scalable parallel programming with cuda. Queue 6 (2), 40–53.
Jensen, J., 1991. A model for the propagation and scattering of ultrasound in tissue. J. Acoust. Soc. Am. N. 89 (1), 182–190.
Kino, G. S., 1987. Acoustic Waves, devices, imaging and analog signal processing. Prentice-Hall.
Nvidia, 2012a. Gtx680 kepler white paper.
Nvidia, Enero 2012b. Guía de Programación de CUDA 4.1.
Pinar Crombie, e., 1997. Calculating the pulsed response of linear arrays accuracy versus computational efficiency. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control 44, 997–1009.
Piwakowsky, B., Sbai, K., 1999. A new approach to calculate the field radiated from arbitrary structuredtransducer arrays. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control 46 (2), 422–439.
Romero-Laorden, David, e., Mar. 2011. Field modelling acceleration on ultrasonic systems using graphic hardware. Computer Physics Communications 182 (3), 590–599.
DOI: 10.1016/j.cpc.2010.10.032
Smith, S. W., 1998. Digital Signal Processing. Analog Devices.
Steinberg, B. D., 1976. Principles of Aperture and Array System Design. Wiley-Interscience.

3 Related publications associated to Section 1.4.2

3.1 Introduction

As a case of study the technique known as Total Focusing Method has been chosen due to its high quality and the requirement of a huge volume of data which is necessary to process. It consists on the sequential emission with each one of the array elements in turn, and the reception in each shot with the full transducer aperture. The area under inspection is defined as the array elements are activated one by one, and all received signals need to be stored in a memory buffer. This dataset has a total size of $N^2 \times L$, being N the number of array elements and L the signal's length. Likewise, this method enables to obtain images perfectly focused at all points in the region of interest because of the implementation of dynamic focusing in emission and reception [43]–[45], [58].

In this sense, these papers describe the main guidelines on how to accomplish the implementation of the fundamental parts of a Synthetic Aperture Beamformer using parallel computing techniques. From an architectural analysis viewpoint, the image generation algorithm is pretty interesting because it can be seen as a data parallel process, making its implementation on machines with diverse computational and I/O capabilities possible. Thereby, in Paper 3.2 a detailed description about the fundamentals of Full Matrix Capture + Total Focusing Technique are presented followed by a thorough analysis about how to accomplish the parallelization on multi-core CPUs and many-thread GPUs in order to analyse whether the technology is mature for implementation of SAFT imaging systems in real time scenarios. On other hand, Paper 3.3 describes a GPU library where TFM with different acquisition schemes (FMC, MRC, etc) have been implemented for the fast generation of ultrasound SAFT images in order to assist on development of new ultrasound beamforming strategies for 2D and 3D imaging.

3.1.1 Related publications

Apart from the full-text publications included in this chapter, in this section it is summarized all the contributions to this research topic published in journals and conference proceedings.

[122] D. Romero-Laorden, C. J. Martín-Arguedas, J. Villazón-Terrazas, O. Martinez-Graullera, M. Santos-Peñas, and C. Gutierrez-Fernandez, "Cheetah : a library for parallel ultrasound beamforming in multi-core systems," *Journal of Applied Mathematics and Physics*, vol. 3, no. August, pp. 1056–1061, 2015. DOI: `http://dx.doi.org/10.4236/jamp.2015.38131`.

[127] D. Romero-Laorden, O. Martinez-Graullera, C. J. Martin-Arguedas, M. Perez-Lopez, and L. Gomez-Ullate, "Paralelización de los procesos de conformación de haz para la implementación del total focusing method," in *12 Congreso Español de END*, Valencia, 2011, ISBN: 978-84-694-4076-6.

[133] D. Romero-Laorden, J. Villazón-Terrazas, O. Martínez-Graullera, and A. Ibáñez, "Strategies for hardware reduction on the design of portable ultrasound imaging systems," in *Advancements and breakthroughs in Ultrasound Imaging*, G. P. P. Gunarathne, Ed., 2013, ISBN: 978-953-51-1159-7. [Online]. Available: `http://www.intechopen.com/articles/show/title/strategies-for-hardware-reduction-on-the-design-of-portable-ultrasound-imaging-systems`.

[134] D. Romero-Laorden, J. Villazón-Terrazas, O. Martinez-Graullera, A. Ibáñez, M. Parrilla, and M. Santos-Peñas, "Analysis of parallel computing strategies to accelerate ultrasound imaging processes," *IEEE Transactions on Parallel and Distributed Systems*, vol. Pending, 2015. DOI: `TPDS-2015-08-0564`.

[135] D. Romero-Laorden, J. Villazón-Terrazas, M. Santos-Peñas, M. A. G. Izquierdo, and O. Martínez-Graullera, "Análisis de la implementación de un conformador software de señales ultrasónicas para tiempo real," *Revista Iberoamericana de Automática e Informática Industrial*, vol. Pending, 2015. DOI: `RIAI-D-15-00060`.

3.2 Analysis of parallel computing strategies to accelerate ultrasound imaging processes

IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. X, NO. X, JULY 2015 1

Analysis of parallel computing strategies to accelerate ultrasound imaging processes

D. Romero-Laorden, *Student Member, IEEE,* J. Villazón-Terrazas, *Student Member, IEEE,* O. Martínez-Graullera, *Member, IEEE,* A. Ibáñez, M. Parrilla and M. Santos Peñas

Abstract

This work analyses the use of parallel processing techniques in synthetic aperture ultrasonic imaging applications. In particular, the Total Focussing Method, which is a $O(N^2 \times P)$ problem, is studied. This work presents different parallelization strategies for multicore CPU and GPU architectures. The parallelization process in both platforms is discussed and optimized in order to achieve real-time performance.

Index Terms

Ultrasound imaging, GPGPU, multicore, signal processing, parallel computing, CUDA

1 INTRODUCTION

Ultrasonic imaging is currently one of the most popular visualization methods to examine the inside of opaque objects. It is mainly based on the pulse-echo response, where the propagation of an induced mechanical perturbation inside an object is analysed from the received echoes, that are caused by discontinuities in the material. Its application is especially relevant in the medical diagnosis area, as well as in *non-destructive testing* (NDT).

Most of ultrasonic imaging systems are based on array of transducers and they have to manage hundreds of independent elements that yield multiple input data signals which are

• *D. Romero-Laorden, J. Villazón-Terrazas, O. Martínez-Graullera A. Ibáñez and M. Parrilla are with the Instituto de Tecnologías Físicas y de la Información (ITEFI), Spanish National Research Council (CSIC), C/Serrano 144, 28006, Madrid (Spain) - Email: david.romero@csic.es, M. Santos Peñas is with Dpto. Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid*

This work has been supported by the Spanish Government under projects DPI2010-19376 and FIS2013-46829R.

processed, to form images, using some of the several known modalities of beamforming [1]. This is, in general, a computationally demanding task, so almost all commercial systems employ expensive tailored hardware designs, with parallel processing capabilities, to attain the framerate required by each application. Nevertheless, real time applications of some computationally intensive techniques, as Synthetic Aperture Focusing Techniques (SAFT) with the Total Focussing Method (TFM), which are object of interest, push systems to the limit. However nowadays due to the popularization of multicore systems and General-Purpose Computing on Graphics Processing Units (GPGPU) techniques, parallel signal processing can be developed in these platforms at very affordable prices. Several works have been published addressing the possibility of using conventional hardware for ultrasonic beamforming, such as CPU cluster [2] or as GPU [3], [4].

Synthetic Aperture Focusing Techniques are based on the independent acquisition of signals from pairs of emission-reception array elements to apply beamforming on them. For an array of N elements this set of signals can be composed at most of N x N signals, which is referred to as the Full Matrix Capture (FMC). For its part, the TFM performs beamforming at each image point with every signal in the FMC. Although this process produces high quality images whit all its points focussed both in emission and reception[5], it also has a very high computational cost. For a P points image TFM is an $O(N^2 \times P)$ algorithm, so it is mainly used as a research tool.

In the last years, some research has been addressed to implement FMC + TFM by hardware using Field Programmable Gate Array (FPGA)[6], [7], [8]. This work has been recently materialized by M2M company, which has launched a real-time TFM system with 64 sensors (4096 signals FMC) capable of achieving, using a set of pre-calculated delays, up to 25 images per second for a 256×256 image.

Other studies have been centred in exploring parallel computing strategies on CPUs and GPUs to address this problem. Thus, CPU beamforming parallelization is a study area that began in the 2000, with the boom and standardization of the cluster computing [2] in order to improve real-time 2D as well as 3D ultrasound imaging systems. Moreover, with the advent of faster and more powerful PCs the requirements in dedicated hardware were reduced being replaced by software processing. J. A. Jensen et. al describe in [9] and [10] a DAS library that implements specific functions for the beamforming. All these previous works were implemented for a single core of CPU and, subsequently, they have still working to develop parallel versions of these libraries to exploit multicore CPUs. Thus, Munk et al. shows in [11] toolboxes for beamforming computation over multi-core CPUs, where the time consumption takes around 142.3 seconds for a $300,000$ pixels TFM image from an array of 192 sensors $(35,000$ signals) running in a 16-core high standard workstation.

Initial work on GPGPU based implementations of SA were centred on the use of minimum redundancy coarray [12] for an array of 128 elements (255 signals) and 256 image points, producing a frame-rate of 30 TFM images per second in real time [13]. In succeeding works the frame-rate was increased until 150 img/s [14]. More recently some works have been presented working with FMC in post-processing but without TFM. Then in [15] FMC (16 emision and 64 receptor elements) is considered at fixed focus, it achieves a frame rate of 40 for a 512×512 image (NVIDIA Fermi with 512 cores). Other works have presented the GPGPU use to implement more complex beamforming techniques, such as Capon Beamforming [16].

Also, more specific works centred in FMC + TFM can be found. In [17] several systems are compared (Xeon X5690 with 6 cores, GPU NVIDIA Fermi with 512 and 448 cores, and GPU ATI 1536 cores) using different developing systems (OpenCL, CUDA) achieving up to 5 images per second for a 200×200 image with an array of 128 elements. Also, in [18] working now in real time (acquisition through an Ethernet connection) a frame rate of 8 is achieved for a 120×120 image (NVIDIA Fermi with 384 cores) were obtained for an array of 64 elements.

Following these works this paper analyses the implementation of the TFM in multi-core CPUs and many-thread GPUs. For both cases an incremental improve implementation of the basic algorithm is followed, helping us to identify bottlenecks. In the next section, we review the theory behind of the Total Focusing Method algorithm and how the beamforming is applied in SAFT systems. Next, a parallel beamformer design is carried out in both architectures. Finally, experimental results are presented along with a performance evaluation of both platforms to establish the main conclusions of this work.

2 Total Focussing Method

Figure 1 describes the general procedure to compose an image by the FMC + TFM technique. Here it is presented the linear transducer array of N elements, the Region of Interest (ROI) that has been discretized in a grid of points $x[k,l]$, and the final image $a[k,l]$, and an ideal reflector located at $\vec{x}_p$ ($x[k_p, l_p]$ in the discretized space).

Thus, assuming that e_i is the emitter transducer and e_j and e_{j+1} are two of the receivers transducers, the presence of a reflector at $\vec{x}_p$ introduces an echo at time $t_i + t_j$ for the signal $s_{i,j}(t)$ (echo-pulse response where i-index is the emitter element and the j-index is the receiver), and at time $t_i + t_{j+1}$ for the signal $s_{i,j+1}(t)$. Then, for any $x[k,l]$ space point, we can obtain from each signal $s_{i,j}(t)$ the echo response originated at this point, and sum all together to the image point $a[k,l]$. If this value becomes significant compared to other image points it is possible to assure the presence of a reflector on it.

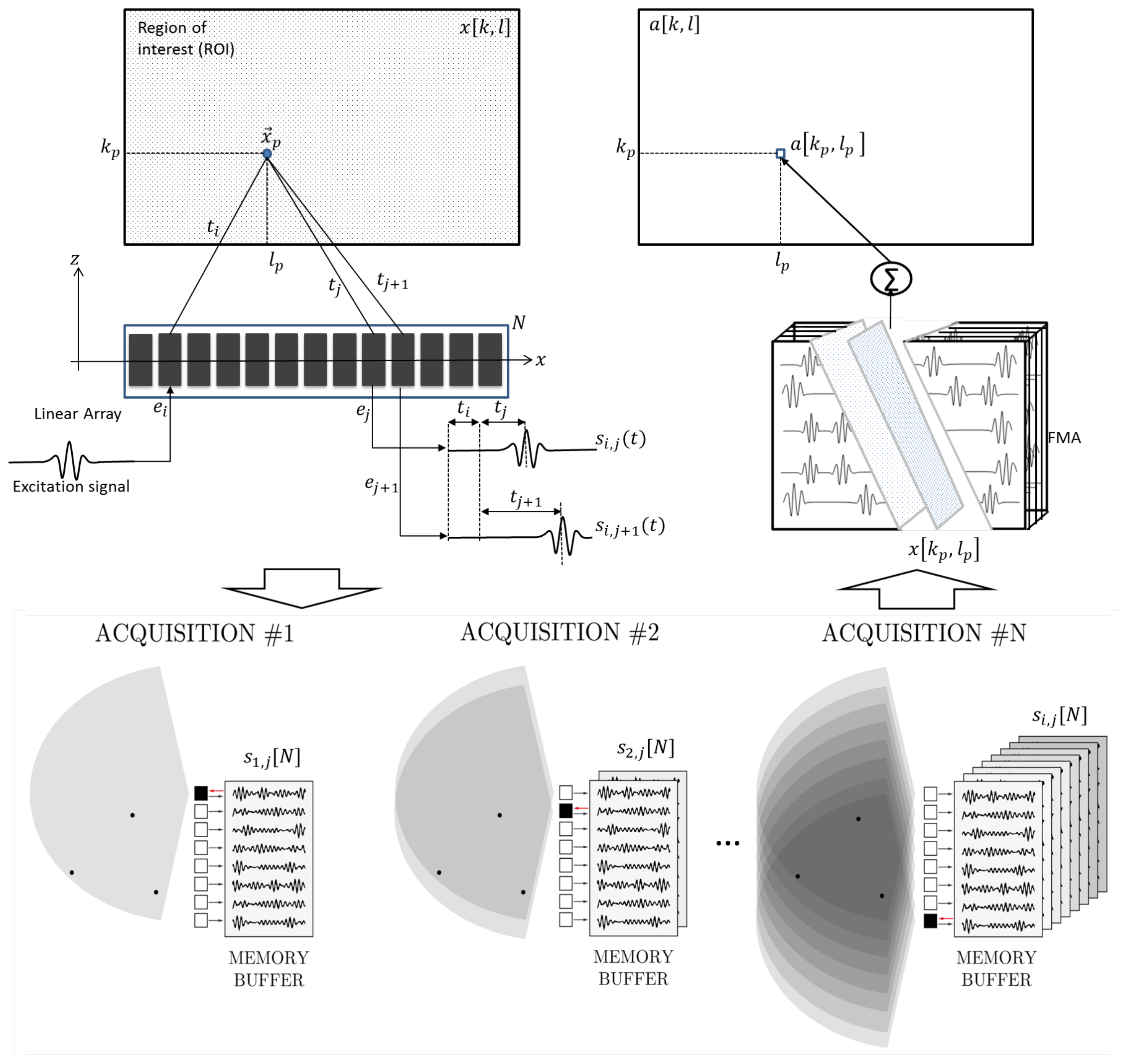


Fig. 1. Schematic diagram illustrating the TFM with a FMC acquisition procedure. An linear array of N elements, a reflector point $\vec{x}_p$ and the travel time to several elements array e_i, e_j and e_{j+1} from this point is presented. The echo signals received at both elements is also presented. The FMC is composed through the acquisition process were for each emision element a matrix of N received signal is obtained. In order to focus the data at $x[k_p, l_p]$ a slice of data from the FMC can selected and summed to compose the $a[k,l]$ value in the image.

2.1 The Beamformer

Based on this description, and assuming that the acquisition system has produced the a FMC of sampled signals, a generic signal can de described as:

$$s_{i,j}[n] = s_{i,j}(t) \cdot \delta(t - n\tau_s) \tag{1}$$

where $s_{i,j}[n]$ is the signal which corresponds to the e_i emitter and e_j receiver and $1/\tau_s$ is the sampling rate.

Then if we want to obtain the echo response of a virtual reflector at $x[k,l]$ point, the time of flight from e_i to e_j is

$$t_{i,j}[k,l] = t_i[k,l] + t_j[k,l] - t_0 = \frac{|\vec{x}_i - x[k,l]| + |\vec{x}_j - x[k,l]|}{c} - t_0 \tag{2}$$

being c the medium velocity, $\vec{x}_i$ and $\vec{x}_j$ the locations of elements e_i and e_j, and t_0 the initial acquisition time. To fit this value in the sampled signal $s_{i,j}[n]$ they should be mapped to memory by decomposing in to factor

$$m[i,j,k,l] = floor\left(\frac{t_{i,j}[k,l]}{\tau_s}\right) \tag{3}$$

$$\Delta m[i,j,k,l] = \frac{t_{i,j}[k,l]}{\tau_s} - floor\left(\frac{t_{i,j}[k,l]}{\tau_s}\right) \tag{4}$$

where $m[i,j,k,l]$ is an index value and the $\Delta m[i,j,k,l]$ corresponds to an interpolation factor. So the echo response for the $x[k,l]$ point in the signal $s_{i,j}$ can be obtained by linear interpolation as:

$$S[i,j,k,l] = s_{i,j}[m[i,j,k,l]] + \Delta m[i,j,k,l]\,(s_{i,j}[m[i,j,k,l]+1] - s_{i,j}[m[i,j,k,l]]) \tag{5}$$

If all $s_{i,j}$ are considered the image value at the point of interest is obtained as:

$$a[k,l] = \sum_{i=1}^{N}\sum_{j=1}^{N} b_{i,j} S[i,j,k,l] \tag{6}$$

where $b_{i,j}$ are the coefficients of the spatial filter that is implemented by the array.

The beamformed points $a[k,l]$ are computed in radiofrequency, so in order to smooth the image it should be rectified by envelope filtering. This process can be done by means of the Hilbert Transform [19] to obtain the analytic representation of $a[k,l]$:

$$a[k,l] \to a_I[k,l] + ja_Q[k,l] \tag{7}$$

where $a_I[k,l]$ and $a_Q[k,l]$ are the phase and quadrature components. Then the envelope can be computed as:

$$A[k,l] = \sqrt{a_I^2[k,l] + a_Q^2[k,l]} \tag{8}$$

and use this value to determine the colour of each pixel in the image.

2.2 Practical implementation

Before to proceed to study the implementation of the equation 8 some practical issues should be addressed.

The first question is addressed to reduce the size FMC. This is achieved through a technique known as *half-matrix* [20], [21], [18]. Analysing the times of flight for two reciprocal signal it easy to show that both signals are beamformed by the same time, that is $t_{i,j}[k,l] = t_{j,i}[k,l]$. Then, due to $b_{i,j} = b_{j,i}$, instead of maintain both signals it is posible to build a new one $\hat{s}_{i,j}[n]$ as:

$$\hat{s}_{i,j}[n] = s_{i,j}[n] + s_{j,i}[n] \tag{9}$$

to obtain a new data set formed by $N(N+1)/2$ signals that is almost a half of FMC. If 9 is performed by the system front end or, in the case of a GPU beamformer, by the CPU in addition to reduce data movements, the operations for beamforming are also reduced by two.

The second issue is referred to the envelope computation. Due to periodicity condition assumed by the FFT, the Hilbert Transform may introduce artifacts in the border of the image [19]. In order to avoid these artifacts the analytic decomposition can be done in the acquired signals (now $\hat{s}_{i,j}[n]$). Then, the distortion is produced in the extreme of the signals, that usually is far away from the ROI or if it is not the case they are smoothed by the beamforming process. Then, in terms of data size, a complex data matrix is created and $s_{i,j}(t)$ can now be expressed by its analytic form:

$$\hat{s}_{i,j}[n] \to \hat{s}_{Ii,j}[n] + j\hat{s}_{Qi,j}[n] \tag{10}$$

Then the echo values at $x[k,l]$ point for each signal in the analytic decomposition is:

$$Iv_{i,j}[k,l] = \hat{s}_{Ii,j}[m_{i,j}[k,l]] + \Delta m_{i,j}[k,l]\left(\hat{s}_{Ii,j}[m_{i,j}[k,l]+1] - \hat{s}_{Ii,j}[m_{i,j}[k,l]]\right) \tag{11}$$

$$Qv_{i,j}[k,l] = \hat{s}_{Qi,j}[m_{i,j}[k,l]] + \Delta m_{i,j}[k,l]\left(\hat{s}_{Qi,j}[m_{i,j}[k,l]+1] - \hat{s}_{Qi,j}[m_{i,j}[k,l]]\right) \tag{12}$$

The ultrasonic image is now obtained as:

$$A_I[k,l] = \sum_{i=1}^{N}\sum_{j=i}^{N} b_{i,j} Iv_{i,j} \tag{13}$$

$$A_Q[k,l] = \sum_{i=1}^{N}\sum_{j=i}^{N} b_{i,j} Qv_{i,j} \tag{14}$$

$$A[k,l] = \sqrt{A_I[k,l]^2 + A_Q[k,l]^2} \tag{15}$$

Although this solution forces to double the processing chain in practice

its cost is mainly measured in memory resources. However, the additional memory resources needed are allocated in the beamformer no transferred to it, which is especially relevant for GPU beamformers.

2.3 The algorithm.

The box **Algorithm 1** describes the process and presents the first implementation CPU1. As we can appreciate, there are massive data-read operations from memory, computation operations for the delays, convert them to indexes, index into FMC matrix to recover signal values, interpolate, sum and write to memory every pixel value for every combination of emitter and receiver, etc. Two parts of this process are subject of being parallelized: in the analytic decomposition (pre-processing) and the beamforming itself.

3 EXPERIMENTAL SET-UP

To illustrate the process a medical phantom (physical model which approximates the scattering and attenuation properties of biological tissue) has been used. Concretely, the $040GSE$ model from CIRS Inc. company (Figure 2(a)). Also, in the figure 2(a) an area that has been chosen to compose our reference image has been delimited (figure 2(b)). The image has been composed with three different image sizes: 256×256, 512×512 and 1024×1024.

The array transducer used is a PA 2.75/64-1093 model from VERMON company. It is a linear array of 64 elements with an elementary pitch of $d = 0.28mm$. Each element emits a Gaussian pulse with a center frequency of $2.6MHz$ and a relative bandwidth of 65%. The acquisition

Algorithm 1 CPU Beamforming. CPU1

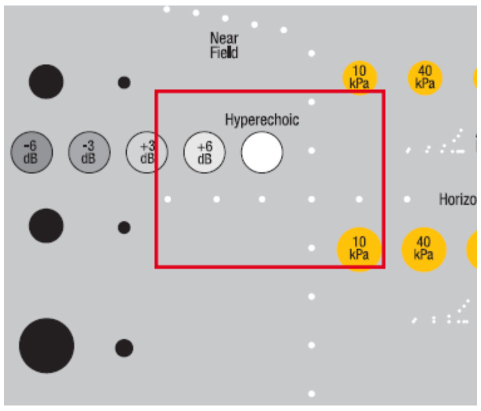
```
1: Pre-processing
2: $\hat{s}_{i,j}[n] \leftarrow s_{i,j}[n] + s_{j,i}[n]$                              Data reduction
3: $\hat{s}_{Ii,j}[n] + j\hat{s}_{Qi,j}[n] \leftarrow \hat{s}_{i,j}[n]$                   Analytic signal
4: Beamforming
5: for $k = 0$ to $R_H$ do
6:   for $l = 0$ to $R_V$ do
7:     $A_I = 0$, $A_Q = 0$                                             Initialize pixel
8:     for $i = 1$ to $N$ do
9:       for $j = i$ to $N$ do
10:        $m$, $\Delta m$                                              Compute index
11:        if $0 \leq m \leq L$ then
12:          $I \leftarrow (1-\Delta m)\hat{s}_{Ii,j}[m] + \Delta m\hat{s}_{Ii,j}[m+1]$   Interpolated sample
13:          $Q \leftarrow (1-\Delta m)\hat{s}_{Qi,j}[m] + \Delta m\hat{s}_{Qi,j}[m+1]$   Interpolated sample
14:          $A_I \leftarrow A_I + b_{i,j}I$      Multiply by $b_{i,j}$ and accumulate sum in $A_I$ pixel
15:          $A_Q \leftarrow A_Q + b_{i,j}Q$      Multiply by $b_{i,j}$ and accumulate sum in $A_Q$ pixel
16:        end if
17:      end for
18:    end for
19:    $A[k,l] \leftarrow \sqrt{A_I^2 + A_Q^2}$                              Envelope calculation
20:  end for
21: end for
22: return $A$                                                          Final image
```

equipment utilized is a SITAU system [22] from DASEL S.L. company (www.daselsistemas.es). The data sets of $64 \times 64 \times 4096$ that form the FMC have been acquired with a data resolution of 12 bits and transferred. The results givens in the following only consider processing time. Time for data transference from the acquisition system has not been considered.

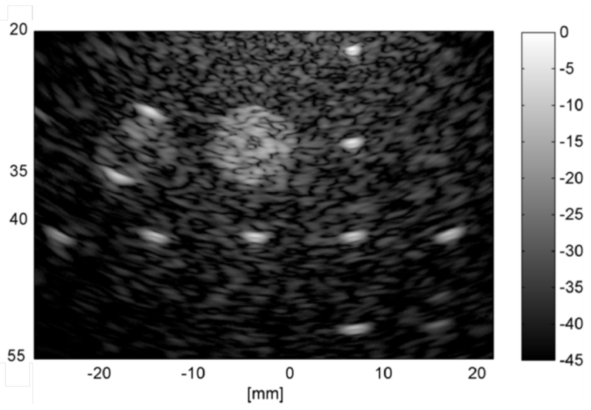
As imaging system we have chosen a diversity of computing platforms which are presented in Table 1. Platform # 1 is a notebook, platform #3 and #4 are low-level workstation and platform #2 and #5 are medium-level workstations. GPUs of platforms #1, #2 and #3 are based on Fermi architecture [23] (GeForce 540M, Quadro 2000 and Quadro 4000 with 96, 192 and 256 cores respectively). Platforms #4 and #5 are based on latest Nvidia's arquitecture, Kepler [23] (Quadro K2000 and Quadro K5000 equipped with 384 and 1536 cores each).

All platforms are running Microsoft® Operating System Windows® 7. In all cases the FMC data and the output ultrasonic image are stored as single-precision floating-point numbers (4 bytes). The CUDA tools are used for GPGPU development.

Although our final analysis involves several platforms, in order to improve legibility we have focused this section in computing platform #3. This is a low-level workstation (Intel Core 2 Quad Q9450 with 4 cores) with a professional an NVIDIA GPU (Quadro 4000 graphics card with 256 cores, the best of our Fermi boards).



(a)



(b)

Fig. 2. 2(a) Multi-Tissue ultrasound phantom scenario 2(b) TFM image from Multi-Tissue ultrasound phantom

TABLE 1
Imaging computing platforms. *DDR: Double Data Rate; GDDR: Graphics Double Data Rate

Computing Platform	CPU & GPU	Memory*	# Cores
1	Intel Core i7 3632QM	DDR3, 6 Gbytes	8
	GeForce 540M (Fermi optimus)	GDDR3, 1 Gbytes	96
2	Intel Xeon E51650v2	DDR3, 32 Gbytes	12
	Quadro 2000 (Fermi)	GDDR5, 1 Gbytes	192
3	Intel Core 2 Quad Q9450	DDR3, 4 Gbytes	4
	Quadro 4000 (Fermi)	GDDR5, 2 Gbytes	256
4	Intel Core 2 Quad Q9450	DDR3, 4 Gbytes	4
	Quadro K2000 (Kepler)	GDDR5, 2 Gbytes	384
5	Intel Xeon E51650	DDR3, 8 Gbytes	12
	Quadro K5000 (Kepler)	GDDR5, 6 Gbytes	1536

4 Analytic Decomposition

The parallelization of the analytic decomposition is tied to the paralellization of the FFT through the Hilbert transformation. This is a very well studied process whose implementation has been optimized in several libraries. For this work, in order to implement it in the CPU the FFTW3 [24] in float type variable has been chosen. In this case the parallelization has been done creating as many threads as available cores and dividing the whole number of signal btween them. Single core implementation has also computed as a reference.

In the case of the GPU the Hilbert Transform use the fastest FFT algorithm provided by CUFFT libraries [23]. Furthermore, CUDA provides fast intrinsic maths routines which provide better performance at the price of IEEE compliance. In our case, because our data is 12 bit integer type and current converters handle up to 16 bits, the use of these routines causes an insignificant numerical difference who has no influence on the final output ultrasound image quality. However the performance is increased by 150%. So we decided to use them throughout all the algorithms implemented for GPU.

TABLE 2
Computing times in x64 OS measured in seconds for the FFT in CPU and GPU over the all platforms

Computing Platform	FFT Single CPU	FFT Multi CPU	FFT Multi GPU
# 1	0.234	0.056	0.016
# 2	0.185	0.025	0.011
# 3	0.312	0.081	0.006
# 4	0.324	0.095	0.009
# 5	0.187	0.026	0.003

The Table 2 shows the computing time of these analytic decomposition based on the three FFT implementation in all the platforms. Although these times are not indicative of the final performance, they show the upper limits that each platform can achieve.

The improvement in the CPU due to the parallelization is clearly shown, increasing the limit to the maximum frame rate from 5 to 40. For those platforms that have four cores (#3 and #4) the speed-up due to parallelization is almost 4 times, close to the number of cores. However when the number of cores grows this improvement is not so good. In fact, platform # 1 (increasing from 1 to 8 cores) shows a similar seed-up than the obtained for platforms # 3 and #4 (4 cores); and for 12 cores (platforms #5, #2) the improvement achieved is only 7.4 times faster.

In the case of GPU, analytic decomposition limits the frame rate to 333 in the best platform (#5, Kepler) and to 65 in the worst case (#1, Fermi Optimus). However some considerations should be done due to the differences between both GPU architectures. For instance, although it has been claimed that there is no difference in the device compute capability from Fermi to Kepler, when the number of cores is comparable (256 in #3 and 384 in #4) the Fermi platforms shows better performance than Kepler (1.5 times faster). Also, meanwhile in Fermi Quadro platforms the increment in performance is proportional to the number of cores, in Kepler architecture the increment is lower.

As it was awaited, better results are obtained by GPU platforms, that are at least 8 times faster than the CPUs.

In next tables the cost of this operation is always included in the figures.

5 CPU BEAMFORMING

The implementation of the beamforming algorithm in a multicore CPU is based in a common multi-threading scheme described in Figure 3. This is a *pixel-oriented* parallelization where each CPU thread is responsible to calculate the value of a set of pixels. Thus, image pixels are divided among the number of CPU cores N_{CPU} establishing a total of N_{SP} pixels per CPU thread, where:

$$N_{SP} = \frac{N_H \times N_V}{N_{CPU}} \tag{16}$$

being N_H and N_V are the number of pixels in both dimensions of the image ($N_I = N_H \times N_V$).

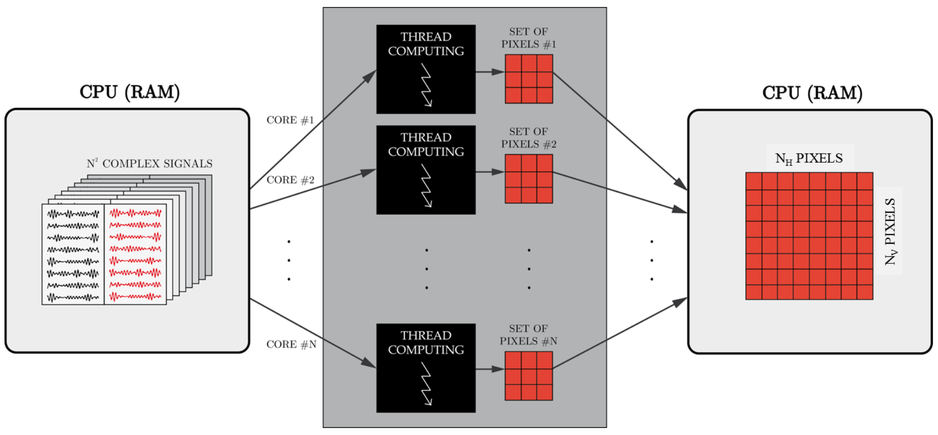


Fig. 3. CPU multi-threading implementation scheme

DRAFT

5.1 First implementation

The first implementation follows the guidelines of **Algorithm 1**. That means that each pixel $[k, l]$ of the image can be solved by one isolated thread that iterates over all the array elements, jumping from signal to signal over the FMC structure following the sample index provided by $m[i, j, k, l]$. Two separate variables are maintained to beamform *in phase* and *in quadrature* values that finally are used to obtain $A[k, l]$ is obtained.

The time consumption (Table 3) of this implementation for different images sizes, 256×256, 512×512 and 1024×1024 pixels, are respectively 3.638, 8.68 and 23.262 second. That is far away from the real time requirements.

The main advantage of this process is that each pixel is solved using a register-oriented implementation. Then, writing of temporary results memory is avoided. Its main drawback is that read memory accesses to data are not coalesced due to sample values are retrieved from non-contiguous memory spaces. This fact has been considered as the main reason of this poor performance.

5.2 Second implementation

The solution to this problem is to change the direction of the DAS beamformation process. Thus, the second implementation iterates first over the array elements and then over every image pixel as it is shown in the **Implementation 2**. Each core solves different set of N_{SP} points of the image.

Implementation 2 CPU Beamforming. CPU2/CPU2SSE — Notes

```
1:  $A_I[k,l] = 0$, $A_Q[k,l] = 0$                                   Initialize subimage region
2:  for $i = 1$ to $N$ do
3:    for $j = i$ to $N$ do
4:      for $[k,l] = 0$ to $N_{SP}$ do
5:        $m$, $\Delta m$
6:        if $0 \le m \le L$ then
7:          $I \leftarrow (1-\Delta m)\hat{s}_{Ii,j}[m] + \Delta m \hat{s}_{Ii,j}[m+1]$
8:          $Q \leftarrow (1-\Delta m)\hat{s}_{Qi,j}[m] + \Delta m \hat{s}_{Qi,j}[m+1]$
9:          $A_I[k,l] \leftarrow A_I[k,l] + b_{i,j} I$
10:         $A_Q[k,l] \leftarrow A_Q[k,l] + b_{i,j} Q$
11:       end if
12:     end for
13:   end for
14: end for
15: $A[k,l] = \sqrt{A_I[k,l]^2 + A_Q[k,l]^2}$
16: return $A[k,l]$                                                  Final subimage region
```

The results of the computational cost of the different implementations are compared in the

Table 3. The $CPU2$ reduce the computing time to by two respect to $CPU1$, achieving 0.869, 4.576 and 12.062 seconds respectively for the different image sizes but maintains the same dependence with the number of image points than $CPU1$. Although now each $A_I[k,l]$ and $A_Q[k,l]$ have to be written and read several times, the cost is less than in the direct implementation of **Algorithm 1**. This is because now we can achieve benefit from cache memory and spatial locality what improves substantially the computational time.

In order to make a faster implementation, the use of *Streaming SIMD Extensions 2* (SSE2) [25] have been considered. Using SSE2 instructions an optimized version of CPU2, CPU2SSE, has been designed. In this case, at each cicle the operation register works with *in phase* and *in quadrature* components of two pixels at the same time .

TABLE 3
Parallelization of TFM algorithm running on Platform #3. CPU strategies comparison for different image sizes

#3	256 × 256	512 × 512	1024 × 1024
CPU1	3.63	8.68	23.26
CPU2	0.86	4.57	12.06
CPU2SSE	0.46	2.245	5.95

With the use of the SSE instructions the results are improved drastically. $CPU2SSE$ obtains 0.469, 2.245 and 5.95 seconds respectively for each case, what is ≈ 5 times faster than CPU1 implementation.

5.3 Third implementation

In order to improve performance another consideration can be made. If our platform has enough memory resources and the region of interest is well defined, time-delay computation and index transformation can be done as a previous stage and stored as a table rather than computed on the fly. The total space needed is $N_X \times N_y \times N^2$, which in case of big images it can be over the system memory capacity, so it has only been tested in platform # 2 (with 32GB of memory). This new implementation is described in **Implementation 3** and only the CPU3SSE has been studied. In order to obtain a measurement of the performance, previous implementations CPU1, CPU2 and CPU2SSE have also been computed.

Results are presented in Figure 4. The most significant result is that CPU3SSE is able to produce more than 13 images per second for the 256 case, increasing the performance to 168% from CPU2SSE. In the case of 512 the performance has been increased to 140%. However for

Implementation 3 CPU3/CPU3SSE — Notes

```
1: $A_I[k,l] = 0,\ A_Q[k,l] = 0$                                  Initialize subimage region
2: for $i = 1$ to $N$ do
3:   for $j = i$ to $N$ do
4:     for $[k,l] = 0$ to $N_{SP}$ do
5:       $m,\ \Delta m$
6:       if $0 \leq m \leq L$ then
7:         $I \leftarrow (1-\Delta m)\hat{s}_{Ii,j}[m] + \Delta m \hat{s}_{Ii,j}[m+1]$
8:         $Q \leftarrow (1-\Delta m)\hat{s}_{Qi,j}[m] + \Delta m \hat{s}_{Qi,j}[m+1]$
9:         $A_I[k,l] \leftarrow A_I[k,l] + b_{i,j} I$
10:        $A_Q[k,l] \leftarrow A_Q[k,l] + b_{i,j} Q$
11:      end if
12:    end for
13:  end for
14: end for
15: $A[k,l] = \sqrt{A_I[k,l]^2 + A_Q[k,l]^2}$
16: return $A[k,l]$                                               Final subimage region
```

the case of 1024 it has only been improved a slightly 8%. This degradation of the performance is due to the increase of transactions with the memory that use a maximum of 4 access channels to serve the 12 threads.

#2	256 × 256	512 × 512	1024 × 1024
CPU1	0.42	1.27	4.75
CPU2	0.26	0.95	3.64
CPU2SSE	0.12	0.36	1.35
CPU3SSE	0.07	0.26	1.24

TABLE 4
Parallelization of TFM algorithm running on Platform #2. CPU strategies comparison for different image sizes

6 GPU BEAMFORMING

The GPGPU development involves a paradigm shift due to the many-thread SIMD model of GPUs. Furthermore, the integration of different memory access allows to develop alternative implementations for the **Algorithm 1** that are not feasible for CPU case. However the amount of memory is more limited in GPU than in CPU, so the delay precomputation would be only posible for small images and it is not considered for this platform.

6.1 First implementation

The starting point to improve the performance of the beamforming is the **Algorithm 1**. Then, **Implementation 4** shows how it can be adapted to the GPU architecture. GPU1 is carried out by launching a thread per image pixel. To this end, a computational grid with $B_X = \lceil \frac{N_H}{T_{B_X}} \rceil$ and $B_Y = N_V$ blocks of T_{B_X} threads is defined to launch the kernel. This implementation maintains all the intermediate results in registers of the scalar processors avoiding write operations to memory.

Implementation 4 GPU1 kernel — GPU optimization resources and notes. GPU1op

```
$s_{Ii,j}[n], s_{Qi,j}[n]$                                  Signal stored in textures memory
$x[k,l]$                                                    Space coordinates stored in textures memory
$b_{i,j}$                                                   Spatial filter stored in constant memory
$x_i$                                                       Sensor coordinates stored shared memory
$k \leftarrow threadIdx.x + blockIdx.x * blockDim.x$        Calculate $k$ coordinate by thread index
$l \leftarrow blockIdx.y$                                   Calculate $l$ coordinate by block index

1:  for $i = 1$ to $N$ do
2:    for $j = i$ to $N$ do
3:      $\hat{m} = (t_j[k,l] + t_i[k,l] - t_0)/\tau_s$               Compute interpolation factor
4:      if $0 \leq m(x,z) \leq L$ then
5:        $I_v \leftarrow texture\{\hat{s}_{Ii,j}, \hat{m}\}$          Interpolation by GPU texture hardware
6:        $Q_v \leftarrow texture\{\hat{s}_{Qi,j}, \hat{m}\}$          Interpolation by GPU texture hardware
7:        $A_I \leftarrow A_I + b_{i,j} I_v$
8:        $A_Q \leftarrow A_Q + b_{i,j} Q_v$
9:      end if
10:   end for
11: end for
12: $A[k,l] \leftarrow \sqrt{A_I^2 + A_Q^2}$
13: return $A[k,l]$
```

In Figure 5, computing times achieved for this implementation are presented. Concretely, it can be appreciated that this strategy $GPU1$, where resource optimization has not been done, achieves 0.325, 0.68 and 1.95 seconds respectively for images of 256×256, 512×512 and 1024×1024 pixels. Although it shows non-coalescing data reads and it does not maximizes the occupancy of the multiprocessors, it produces better performance than the CPU implementation.

If the application is aimed at optimizing the resources of the GPU, the performance can be improved. Once identified the data elements of the process, they can be arranged between the different memory resources according to their use. Generally, this task can be approached using different mechanisms: shared memory which is small in size but it has fast access speed because it is on-chip; texture memory which is cached and can be used for write and read operations ensuring all data reads are coalesced; and constant memory which is fast but only can be used for data read operations [23].

Attending to these memory considerations, the first implementation can be optimized in $GPU1op$, where the use of texture memory to store the data matrix and the space coordinates, and constant memory has been used to store the apodization values. Likewise, the interpolation operation is no longer needed explicitly, because texture mechanisms provide linear interpolations at no cost directly by hardware when the sample value is retrieved. All the changes are reflected in GPU1op (see **Implementation 4**). As we can see in Figure 5 this data organization improves processing time. Now, the running times are 0.18, 0.36 and 1.10 seconds respectively for each image size.

6.2 Second implementation

However, despite the good result of the previous implementation there is another approach which is more suitable for the GPU model. This is based on the Nikolov SAFT implementation proposed to be implemented in multiple FPGA [7]. This implementation can be easily parallelized and optimized to GPU.

The basic principle is simple. The data which belongs to any single element (the emitter) and all the receivers can be used for making a low resolution image (LRI). Then, combining (adding together) the N low resolution images obtained, a high resolution image (the final image) is obtained with fully dynamic focusing at all points in the image. Although it forces to increase the memory needed to store the results in N times, it also solves the coalescence problem identified in previous implementation. The process is illustrated in Figure 4.

Therefore, the parallelization strategy in $GPU2$, is composed of two different kernels (see **Implementation 5**). The first one is responsible of creating the low resolution image for each element. To this purpose, a thread per image pixel an array element is launched, and a three-dimensional computational grid is defined in the kernel, with $B_X = \lceil \frac{N_H}{T_{B_X}} \rceil$, $B_Y = \lceil \frac{N_V}{T_{B_Y}} \rceil$ and $B_Z = N$ blocks of $T_{B_X} \times T_{B_Y}$ threads in each dimension. It is important to remark, that now we are creating 3D blocks in that way less blocks are launched per multiprocessor what increase performance. The block size must be equal to the number of elements in the array in order to cover each element on emission. Each thread within a block is charged of calculating the partial sum of each emission-reception combination.

Once all low resolution images have been computed, the second kernel is responsible to combine all LRI images together (implementation 5). To achieve this, a second grid with $B_X = \lceil \frac{N_H}{T_{B_X}} \rceil$ and $B_Y = \lceil \frac{N_V}{T_{B_Y}} \rceil$ blocks of $T_{B_X} \times T_{B_Y}$ is defined where each thread is responsible of calculate the sum for a given pixel across the multiple LRI images.

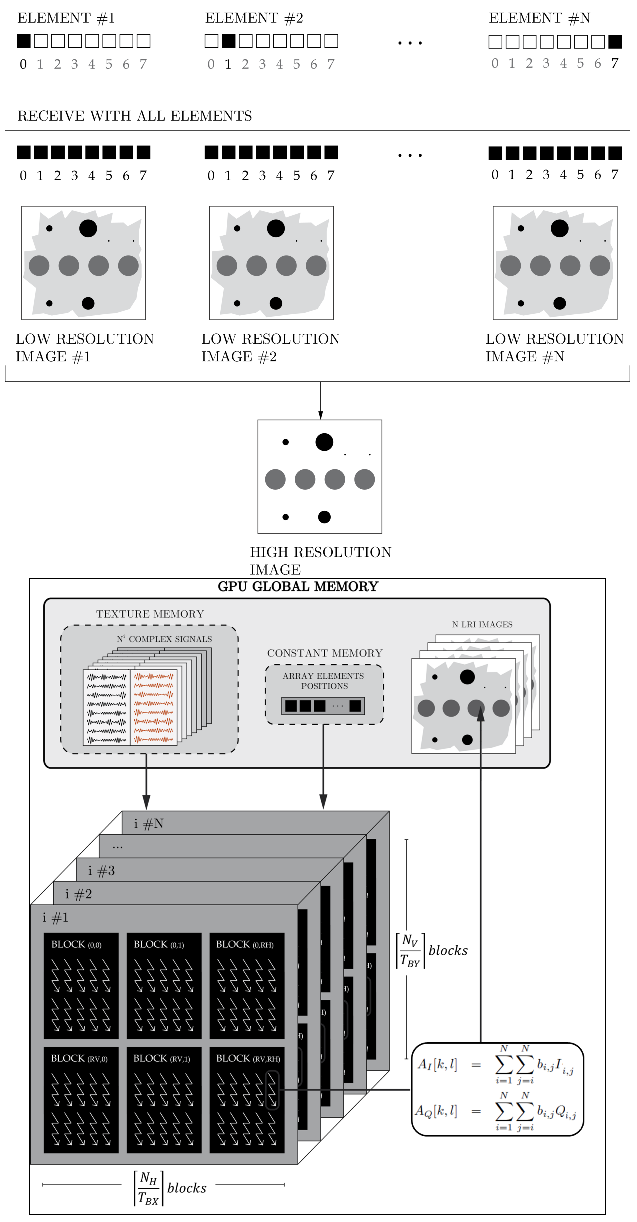


Fig. 4. Each combination of emitter and all the receivers is used to create N LRI images which are combined to compose the final high resolution image. One thread is responsible for a pixel and an element of the array producing a set of low resolution images which belongs to each array element

Implementation 5 $GPU2$ kernel one

```
$s_{Ii,j}[t], s_{Qi,j}[t], x[k,l]$                      Store in textures memory
$b_{i,j}$                                               Store in constant memory
$k \leftarrow threadIdx.x + blockIdx.x * blockDim.x$    Calculate $k$ coordinate by thread/block index
$l \leftarrow threadIdx.y + blockIdx.y * blockDim.y$    Calculate $l$ coordinate by thread/block index
$i \leftarrow threadIdx.z + blockIdx.z * blockDim.z$    Calculate $l$ coordinate by thread/block index

1: $LRI_I[i,k,l] = 0$, $LRI_Q[i,k,l] = 0$               Initialization
2: for $j = 1$ to $N$ do
3:     $\hat{m} = (t_j[k,l] + t_i[k,l] - t_0)/\tau_s$    Compute interpolation factor
4:     if $0 \leq m(x,z) \leq L$ then
5:         $I_v \leftarrow texture\{\hat{s}_{Ii,j}, \hat{m}\}$    Interpolation by GPU texture hardware
6:         $Q_v \leftarrow texture\{\hat{s}_{Qi,j}, \hat{m}\}$    Interpolation by GPU texture hardware
7:         $LRI_I[i,k,l] \leftarrow LRI_I[i,k,l] + b_{i,j}I_v$    Store as GPU texture
8:         $LRI_Q[i,k,l] \leftarrow LRI_Q[i,k,l] + b_{i,j}Q_v$    Store as GPU texture
9:     end if
10: end for
11: return $LRI_I$, $LRI_Q$                               Low resolution images
```

Implementation 5 $GPU2$ kernel two

```
$k \leftarrow threadIdx.x + blockIdx.x * blockDim.x$    Calculate $k$ coordinate by thread/block index
$l \leftarrow threadIdx.y + blockIdx.y * blockDim.y$    Calculate $l$ coordinate by thread/block index

1: $A_I[k,l] = 0$, $A_Q[k,l] = 0$                        Initialization
2: for $i = 1$ to $N$ do
3:     $A_I[k,l] \leftarrow A_I[k,l] + texture\{LR_I[i,k,l]\}$    Read from texture memory
4:     $A_Q[k,l] \leftarrow A_Q[k,l] + texture\{LR_Q[i,k,l]\}$    Read from texture memory
5: end for
6: $A[k,l] \leftarrow \sqrt{A_I[k,l]^2 + A_Q[k,l]^2}$
7: return $A$                                            Final image
```

The Figure 5 show the evolution in performance for platform #3 in all the considered implementations. Although the optimization of the resources can double the performance, the most significant increase is achieved when the coalescence problem is solved. In fact, for the last implementation the computing times are 0.028, 0.09 and 0.355 seconds for each image size. These results, depending on the image size, are between 6 and 3 times better than previous results.

7 PERFORMANCE EVALUATION

In order to maintain the comparative between platforms, the implementation CPU2opt and GPU2 are considered. The time needed for generating each image at each platform is presented in Table 1. they have been computed using an average of 16 measurements. It should be

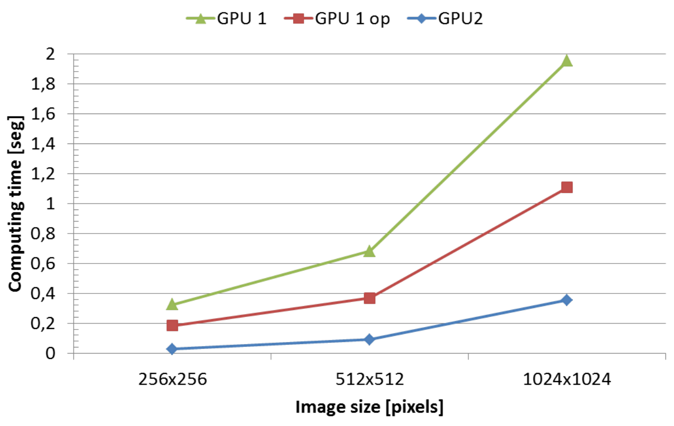


Fig. 5. Parallelization of TFM algorithm running on Platform #3. GPU strategies comparison for different image sizes

mentioned that GPU times also includes the copy time from CPU RAM to GPU RAM. The analysis of the performance of the GPGPU algorithm with *NVIDIA profiler* has shown that almost the 40% of the computational cost is due to the data transfer from the acquisition system to the GPU through the CPU.

As it can be observed, in all cases the time consumed by the GPU is less than the time required by the CPU. If the best CPU performance (platform # 2) is compared with the best GPU (platform # 5), GPU performance is about 9 times better than the CPU for all image sizes. If the different combinations between GPU and CPU platforms are considered, the Table reveals that move the beamforming process from CPU to GPU depending on the platform can improve the performance between 1.6 to 60 times. This reveals the diversity of the platforms and the need of a previous analysis of the hardware capabilities before to proceed to any optimization.

If frame rate is considered (Figure 6), it is noticeable that the CPU frame rates are up to 8 img/s (remember it achieves 13 for CPU3opt in #3), whereas in GPU-based systems all of them are able to obtain rates above 12 frames per second. The platforms #3 GPU and #5 GPU can be highlighted due to both are capable for real-time generating 35 and 65 frames per second for images of 256×256. For CPU it is interesting to remark, that when platforms with the same number of cores are used (#2 CPU and #5 CPU) changes in the SSE instructions provide a more efficient improvements (Ivy vs Sandy Intel architectures). This fact is more noticeable when the size of the images increases.

TABLE 5
Computing times in x64 OS measured in seconds

Platform		256 × 256	512 × 512	1024 × 1024
#1	CPU	0.29	1.02	3.94
	GPU	0.072	0.23	0.851
#2	CPU	0,13	0,37	1,35
	GPU	0.046	0.133	0.479
#3	CPU	0.45	2.25	5.95
	GPU	0.028	0.09	0.355
#4	CPU	0.49	2.35	5.99
	GPU	0.042	0.103	0.420
#5	CPU	0.15	0.51	1.72
	GPU	0.015	0.038	0.14

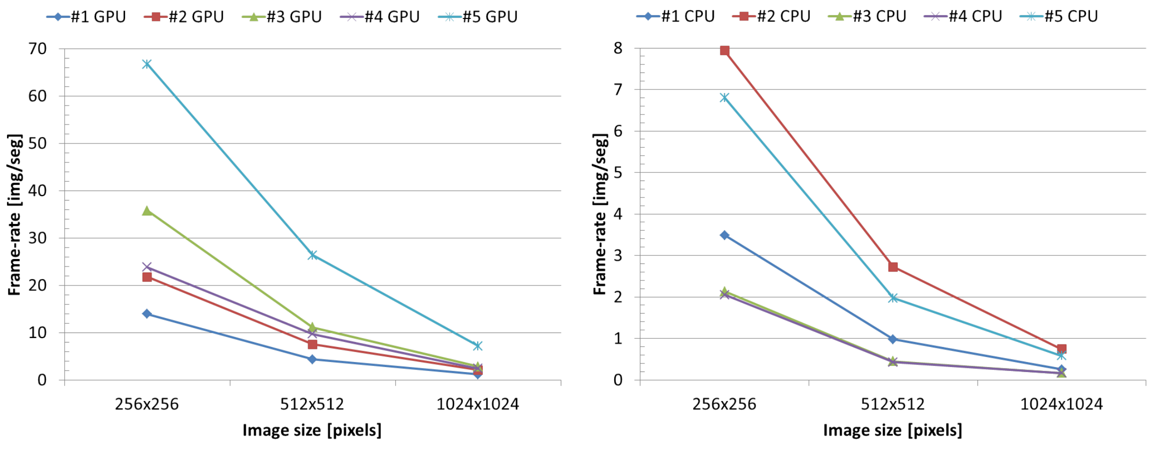


Fig. 6. Frame-rate in GPU an CPU

8 CONCLUSIONS AND FUTURE WORK

This work has been focused on evaluating whether the CPU/GPU technology is mature for the implementation of a generic/scalable TFM imaging system in real-time. Thus, a parallel beamformer design and a performance evaluation is carried out in both architectures. Differences in the parallelization process between both platforms have been shown. Rather than in the computational process itself, the optimization of the basic algorithm has shown that the bottlenecks are mainly produced by data transference and the memory used. In particular, memory access has been identified as the main problem in both implementations.

The results are encouraging because they show that conventional hardware is able to execute high computational demand signal processing algorithms in order to compose high quality ultrasonic image in real time. This result shows that the acquisition system could be simplified avoiding the implementation of complex signal processing systems. In fact, the results for GPGPU show that in the same conditions the GPU is able to obtain grater frames rates than those obtained with an FPGA. For the most powerful GPU boards, depending on the considered application, frame rate could also be several times higher than the acquisition rate.

In our opinion, the developed model for the GPU is better than the multicore CPU because it provides a more accurate control of the resources. We also believe that the use of CUDA and the special libraries provided by the manufactures help to increase the performance. In this sense the performance of #3 (Fermi 256 cores), similar to the boards cited in the literature if not worse, is able to improve the performance in 4 times over other published solutions for a 256×256 image. In addition, it is interesting to remark that our best result in CPU is better than some of the results that have been found in the literature for GPU.

These results show that the development of systems oriented to FMC acquisition is interesting and possible, even for real time applications. This fact, in the signal processing field, allows to develop new real time algorithms based on the FMC to improve the quality of ultrasonic imaging.

Acknowledgments

This work has been supported by the Spanish Government under projects DPI2010-19376 and FIS2013-46829R.

References

[1] T. L. Szabo, *Diagnostic Ultrasound Imaging*. Elsevier, 2004.

[2] F. Zhang, A. Bilas, A. Dhanantwari, K. N. Plataniotis, R. Abiprojo, and S. Stergiopoulos, "Parallelization and Performance of 3D Ultrasound Imaging Beamforming Algorithms on Modern Clusters," in *Proceedings of the 16th international conference on Supercomputing*, 2002, pp. 294–304.

[3] C.-I. C. Nilsen and I. Hafizovic, "Digital beamforming using a GPU," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. Ieee, Apr. 2009, pp. 609–612.

[4] H. K. H. So, J. Chen, B. Y. S. Yiu, and A. C. H. Yu, "Medical Ultrasound Imaging: To GPU or not to GPU," *IEEE Micro*, vol. 31, no. 5, pp. 54–65, 2011.

[5] C. Holmes, Bruce W. Drinkwater, and P. D. Wilcox, "Post-processing of the full matrix of ultrasonic transmitreceive array data for non-destructive evaluation," *NDT & E International*, vol. 38, no. 8, pp. 701–711, Dec. 2005.

[6] "Gekko: Advanced phased-array ut by m2m." [Online]. Available: http://www.m2m-ndt.com/products/Gekko_features.htm

[7] S. I. Nikolov, "Synthetic aperture tissue and flow ultrasound imaging," Ph.D. dissertation, Technical University of Denmark, 2001.

[8] M. Birk, A. Guth, M. Zapf, M. Balzer, N. Ruiter, M. Hübner, and J. Becker, "Acceleration of image reconstruction in 3D ultrasound computer tomography: An evaluation of CPU, GPU and FPGA computing," in *Conference on Design and Architectures for Signal and Image Processing (DASIP)*, Tampere, 2011, pp. 1–8.

[9] J. r. A. Jensen and S. I. Nikolov, "Fast simulation of ultrasound images," *2000 IEEE Ultrasonics Symposium. Proceedings. An International Symposium (Cat. No.00CH37121)*, vol. 2, pp. 1721–1724, 2000.

[10] J. Kortbek, S. I. Nikolov, and J. r. A. Jensen, "Effective and versatile software beamformation toolbox Jacob," in *Medical Imaging 2007: Ultrasonic Imaging and Signal Processing*, S. Y. Emelianov and S. A. McAleavey, Eds., Mar. 2007.

[11] J. M. Hansen, M. C. Hemmsen, and J. r. A. Jensen, "An object-oriented multi-threaded software beamformation toolbox," in *SPIE Medical Imaging: Ultrasonic Imaging, Tomography, and Therapy*, J. D'hooge and M. M. Doyley, Eds., Mar. 2011, pp. 79 680Y–79 680Y–9.

[12] C. J. Martín-Arguedas, "Técnicas de apertura sintética para la generación de imagen ultrasónica," Ph.D. dissertation, Universidad de Alcalá, 2010.

[13] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, R. Tokio Higuti, and A. Octavio, "Using GPUs for beamforming acceleration on SAFT imaging," in *IEEE International Ultrasonics Symposium*. Rome, Italy: IEEE, 2009, pp. 1334–1337.

[14] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, M. Pérez-Lopez, and L. Gómez-Ullate, "Paralelización de los procesos de conformación de haz para la implementación del Total Focusing Method," in *12 Congreso Español de END*, Valencia, 2011.

[15] J. M. Hansen, D. Schaa, and J. r. A. Jensen, "Synthetic Aperture Beamformation using the GPU," in *IEEE International Ultrasonics Symposium*, Orlando, Florida, 2011.

[16] J. Asen, J. Buskenes, C.-I. C. Nilsen, A. Austeng, and S. Holm, "Implementing capon beamforming on a GPU for real-time cardiac ultrasound imaging." *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 61, no. 1, pp. 76–85, Jan. 2014.

[17] G. Rougeron, J. Lambert, E. Iakovleva, L. Lacassagne, and N. Dominguez, "Implementation of a GPU Accelerated Total Focusing Reconstruction Method within CIVA Software," *40th Annual Review of Progress in Quantitative Nondestructive Evaluation*, vol. 1581, no. 1, pp. 1983–1990, 2013.

[18] M. Sutcliffe, M. Weston, B. Dutton, P. Charlton, and K. Donne, "Real-time full matrix capture for ultrasonic non-destructive testing with acceleration of post-processing through graphic hardware," *NDT & E International*, vol. 51, pp. 16–23, 2012.

[19] A. V. Oppenheim, W. R. Schafer, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Upper Saddle River, New Jersey: Prentice-Hall, 1989, vol. 23, no. 2.

[20] C. Holmes, B. W. Drinkwater, and P. D. Wilcox, "Advanced post-processing for scanned ultrasonic arrays: application to defect detection and classification in non-destructive evaluation," *Ultrasonics*, vol. 48, no. 6-7, pp. 636–42, Nov. 2008.

[21] A. J. Hunter, B. W. Drinkwater, and P. D. Wilcox, "The wavenumber algorithm for full-matrix imaging using an ultrasonic array." *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 55, no. 11, pp. 2450–62, Nov. 2008.

[22] J. Camacho, O. Martinez, M. Parrilla, R. Mateos, and C. Fritsch, "A strict-time distributed architecture for digital beamforming of ultrasound signals," in *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, Oct 2007, pp. 1–6.

[23] NVIDIA, *CUDA C Programming Guide 6.0*, 2014, no. February 2014. [Online]. Available: www.nvidia.com

[24] "Fftw website." [Online]. Available: www.fftw.org

[25] "Intel intrinsics guide - sse instructions." [Online]. Available: https://software.intel.com/sites/landingpage/IntrinsicsGuide/

PLACE
PHOTO
HERE

David Romero-Laorden was born in Talavera de la Reina, Toledo, Spain, in 1984. He received the M.S. degree in computer science from the Complutense University of Madrid, Spain, in 2008. He is currently pursuing a Ph.D. degree at the Instituto de Tecnologias Fisicas y de la Informacion of the Spanish National Research Council (ITEFI-CSIC), Madrid, Spain. His research interests include ultrasonic imaging systems, synthetic aperture systems, digital signal processing and parallel programming on heterogeneous architectures.

PLACE
PHOTO
HERE

Javier Villazon-Terrazas was born in Cochabamaba (Bolivia). He belongs to Ultrasonic Evaluation Group of the Applied Acoustic and Nondestructive Evaluation Department of the Spanish National Research Council (CSIC). He graduated with honours in Electronic Engineering from Universidad Del Valle Bolivia in 2002. He is currently a PhD. Student in the Signal, System and Radio communications Department at Universidad Politcnica de Madrid. His research activities include Evolutionary Algorithms, Acoustic Field, Ultrasonic Array design and Computer Science.

PLACE
PHOTO
HERE

Oscar Martínez-Graullera was born in Valencia, Spain. He received the Telecommunication Engineer degree in 1995 from the Polytechnic University of Valencia, and his Ph.D. degree from the Polytechnic University of Madrid in 2000. Since 2004, he is working at the Spanish National Research Council (CSIC, Madrid) as a Tenured Scientist. He is involved with ultrasonic imaging, ultrasonic arrays, digital signal processing, and real-time architectures.

PLACE
PHOTO
HERE

Alberto Ibañez-Rodriguez was born in Benavente, Zamora (Spain). In 1984 he becomes Degree in Physics from the University of Valladolid in the field of Electronics and then Ph.D. in Electronics from the Complutense University of Madrid (Spain). Since 1984, he is working at the Spanish National Research Council (CSIC, Madrid) as a Tenured Scientist. His reseach interests are centered in the area of integration of sensors, real-time systems, signal processing and its applications in nondestructive evaluation.

PLACE
PHOTO
HERE

Montserrat Parrilla Romero was born in Ciudad Real, Spain. She obtained the Universidad Politcnica de Madrid B. Tech. and M.S. degrees in information technology in 1990 and 1992, respectively. She has also received a degree in computing from the same university. She obtained a Ph.D. degree in computer science from the Departmento de Arquitectura y Tecnologa de Sistemas Informticos (Computer Architecture and Information Technology Department) in 2004 from the same university. Her research interests include 2-D and 3-D real-time ultrasonic imaging for clinical and industrial applications and automatic defect detection and characterization in ultrasonic nondestructive testing applications.

PLACE
PHOTO
HERE

Matilde Santos Peñas was born in Madrid, Spain. She received her B.Sc. and M.Sc. degrees in Physics (Computer Engineering) and her Ph.D in Physics in 1994, from the University Complutense of Madrid (UCM). Since 1986 she has been with the Department of Computer Architecture and Systems Engineering at the UCM, where she currently is a Full Professor in System Engineering and Automatic Control. Her major research interests are: Intelligent Control, Signal Processing, Modelling and Simulation.

3.3 Cheetah: A library for parallel ultrasound beamforming in multi-core systems

Journal of Applied Mathematics and Physics, 2015, 3, 1056-1061
Published Online August 2015 in SciRes. http://www.scirp.org/journal/jamp
http://dx.doi.org/10.4236/jamp.2015.38131

Cheetah: A Library for Parallel Ultrasound Beamforming in Multi-Core Systems

David Romero-Laorden[1], Carlos Julián Martín-Arguedas[2], Javier Villazón-Terrazas[1], Oscar Martinez-Graullera[1], Matilde Santos Peñas[3], César Gutierrez-Fernandez[2], Ana Jiménez Martín[2]

[1]ITEFI, Spanish National Research Council, Madrid, Spain
[2]Department of Electronics, University of Alcalá, Alcalá de Henares, Spain
[3]Department of Computer Architecture and Automation, Complutense University of Madrid, Madrid, Spain
Email: cj.martin@uah.es

Received 15 August 2015; accepted 19 August 2015; published 26 August 2015

Abstract

Developing new imaging methods needs to establish some proofs of concept before implementing them on real-time scenarios. Nowadays, the high computational power reached by multi-core CPUs and GPUs have driven the development of software-based beamformers. Taking this into account, a library for the fast generation of ultrasound images is presented. It is based on Synthetic Aperture Imaging Techniques (SAFT) and it is fast because of the use of parallel computing techniques. Any kind of transducers as well as SAFT techniques can be defined although it includes some pre-built SAFT methods like 2R-SAFT and TFM. Furthermore, 2D and 3D imaging (slice-based or full volume computation) is supported along with the ability to generate both rectangular and angular images. For interpolation, linear and polynomial schemes can be chosen. The versatility of the library is ensured by interfacing it to Matlab, Python and any programming language over different operating systems. On a standard PC equipped with a single NVIDIA Quadro 4000 (256 cores), the library is able to calculate 262,144 pixels in ≈105 ms using a linear transducer with 64 elements, and 2,097,152 voxels in ≈ 5 seconds using a matrix transducer with 121 elements when TFM is applied.

Keywords

GPGPU, Ultrasound Image, Beamforming, Array Transducer

1. Introduction

During recent years, computing industry has opened a way to parallel computing. First, dual-core processors (CPUs) were introduced in personal systems at the beginning of 2005, and it is currently common to find them in laptops as well as 8 and 16-core workstation computers, which means that parallel computing is not relegated to big supercomputers. On the other hand, Graphics Processor Units (GPUs), as their name suggests, came about as accelerators for graphics applications, predominantly those using the OpenGL and DirectX programming interfaces. Although originally they were pure fixed-function devices, the demand for real-time and 3D graphics made them evolve into small computational units, multithreaded processors with extremely high computational power and very high memory bandwidth that are now available to anyone with a standard PC or laptop.

Since 2006 GPUs can be programmed directly in C/C++ using CUDA or OpenCL [1], allowing each and every arithmetic logic unit on the chip to be used by programs intended for general-purpose computations (GPGPU). A CUDA program consists of one or more stages that are executed on either the host (CPU) and a NVIDIA GPU. The stages that exhibit little or no data parallelism are implemented in CPU code whereas those that exhibit rich amount of data parallelism are implemented in the GPU code. These parallel functions are called kernels, and typically generate a large number of threads to exploit data parallelism

From an architectural analysis viewpoint, beamforming techniques are pretty interesting in particular because they can be seen as a data parallel process, making possible its implementation on machines with diverse computational and I/O capabilities. Some previous works as [2] show toolboxes for beamforming computation over CPUs and multi-core CPUs obtaining very good timing results. Our research group has been working on GPUs applied for field modeling acceleration [3] [4] as well as for fast beamforming [5]-[7] achieving speed ups of 150× over conventional CPU beamforming. Nowadays, GPU beamformers are a reality and there are lot of research groups working on solutions for NDT and medical applications [8].

The aim of this work is to present CHEETAH, a fast ultrasonic imaging library to assist on fast development of new ultrasound beamforming strategies (currently, only SAFT methods are supported) making possible to generate 2D and 3D images on a standard PC or laptop in just few milliseconds. The input data can originate from either a simulation program or from an experimental setup. The library is composed by several routines written in CUDA for fast execution, thus a NVIDIA© GPU is required at the present time. Nowadays, the 1.0 version (Windows and Linux OS) will be soon available.

2. CHEETAH Core Features

CHEETAH has been designed as a free multi-platform library written in C++ and CUDA which can handle multitude of focusing methods, interpolation schemes and apodizations, to generate images from real RF signals obtained from any application and any acquisition process. The main features currently supported are:

- **Custom transducers.** Commands for defining linear and matrix arrays are given. Likewise, arbitrary geometry transducers such as sparse arrays can be also specified. Thereby different transducers can be used depending of the concrete application.
- **Custom SAFT techniques.** Commands for easily defining specific SAFT sequences of emission/reception are given. Anyway, the library comes with some predefined techniques, like 2R-SAFT [6], and TFM [9].
- **2D/3D imaging.** Commands for composing bidimensional and volumetric images (also C-Scan images) are given what makes possible to span a wide range of applications.
- **Coherence factor.** Commands for the application of coherence factors are given [10].
- **Matlab©/Python bindings.** As the library is written in C++ and CUDA, its functionality is available in Windows or Linux OS. Likewise, we have developed specific bindings to connect it to Matlab© what allows to reuse existing code or utilize specific toolbox functionalities.

3. TFM Design on Cheetah Library

In this section, TFM method has been chosen as the case of study to analyze the main implementation aspects of the beamforming algorithm on the library.

3.1. TFM Imaging Principles

Synthetic Aperture Focusing Techniques (SAFT) are based on the sequential activation of the array elements in emission and reception, and the separate acquisition of all the signals involved in the process. Then, a beamforming algorithm is applied to focus the image dynamically in emission and reception obtaining the maximum quality at each image point. One of the most common beamforming methods is Total Focusing Method (TFM) [9] [11] and it is based on Full Matrix Array (FMA), which is the complete data matrix $X(t)$ created by any transmitter-receiver combination:

$$\boldsymbol{X}(t) = \left\{ X_{tx,rx}(t) \mid \forall 1 \le (tx, rx) \le N \right\} \tag{1}$$

where $X_{tx,rx}(t)$ is the corresponding signal to *tx* transmitter and *rx* receiver, and *N* is the number of array elements. For ease the envelope computation, acquired signals are decomposed into their analytic signals form (in-phase *I* and quadrature components *Q*) applying the Hilbert Transform being now expressed as:

$$\mathbf{X}(t) = \mathbf{I}(t) + j\mathbf{Q}(t) \tag{2}$$

According to the Hilbert transformation applied in Equation (2), a complex data matrix has been created. Then, for the case of a 2D scenario 2 a Delay-and-Sum (DAS) beamforming process is used to calculate two images in the following way:

$$A_I(x,z) = \sum_{i=1}^{N}\sum_{k=1}^{N} I_{tx_i,rx_k}\left(D(x,z)\right) \tag{3}$$

$$A_Q(x,z) = \sum_{i=1}^{N}\sum_{k=1}^{N} Q_{tx_i,rx_k}\left(D(x,z)\right) \tag{4}$$

where $A_I(x, z)$ and $A_Q(x, z)$ are the in-phase and quadrature images respectively, and $D(x, z)$ is the delay corresponding to the focus point (x_{fp}, z_{fp}) in the space which is calculated as follows:

$$D(x,z) = \frac{\sqrt{(x_{fp}-x_{tx})^2 + z_{fp}^2} + \sqrt{(x_{fp}-x_{rx})^2 + z_{fp}^2}}{c} \tag{5}$$

being x_{tx} and x_{rx} the coordinates of the transducer elements *tx* and *rx*, respectively. Over this general scheme it is possible to introduce any type of apodizations.

$$A = \sqrt{A_I^{\,2} + A_Q^{\,2}} \tag{6}$$

Finally, the envelope is calculated according to the equation6 to obtain the final image.

3.2. Parallel GPU Implementation

The background behind CHEETAH library comes from the knowledge that our group has on beamforming acceleration using GPUs [5]-[7]. Nevertheless, it has been considered to briefly review the main concepts involved in the parallel implementation of the TFM process on the library.

During the processing the input raw data and the output image pixels are stored as single-precision floating-point numbers (4 bytes). We have also used fast intrinsic math routines which provide better performance at the price of IEEE compliance (the precision is slightly reduced). In our case, this causes a minimal numerical difference who has little influence on the final output ultrasound image quality.

Designed parallel strategies perfectly fit the SIMD (Single Instruction Multiple Data) model of GPU architecture [1]. The general parallelization scheme can be resumed as **Figure 1** suggests:

a) The process starts when the Full Matrix Array $X(t)$ defined in Equation (1) is transferred from CPU memory to GPU global memory via PCI Express bus.

b) According to Equation (2) the Hilbert Transform is applied to every signal using CUFFT libraries [1]. The parallelism strategy is signal-oriented which properly splits the algorithm and computes the FFT of the data. This means that *NxN* threads work concurrently. The result of these operations is a complex data matrix $I(t)$ and $Q(t)$ which is stored in GPU texture memory.

c) DAS *kernel* is applied to the complex data matrix to calculate *N* low resolution images (LRI_I and LRI_Q) each one corresponding to an emitter-all receivers [8] as Equations (3) and (4) suggest. The parallelization is performed launching as many threads as image pixels what, supposing image dimensions of $S_X \times S_Z$, means $N \times S_X \times S_Z$ threads. Several optimizations (reuse computed values, symmetries, shared/constant memories [7]) have been used to maximize performance. The focusing delay is calculated on-the-fly and it is indexed in the $I(t)$ and $Q(t)$ complex signals interpolating real and imaginary parts. Finally, the complex samples are multiplied by the corresponding apodization gains and added together to beamform each of the images.

d) A new kernel is defined to calculate the final A_I and A_Q images performing the sum of the (LRI_I and LRI_Q) images respectively. Once the final values for both images are computed, the envelope calculation is carried out. This strategy follows a pixel-oriented parallelism launching $S_X \times S_Z$ threads.

We must mention that, this scheme has been further optimized for those SAFT strategies based on the coarray

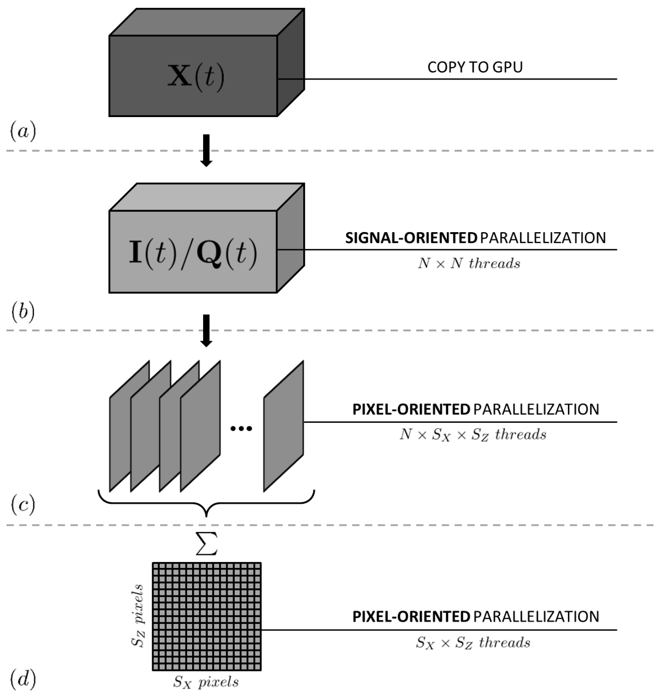

$\mathbf{X}(t)$
COPY TO GPU
(a)
$\mathbf{I}(t)/\mathbf{Q}(t)$
SIGNAL-ORIENTED PARALLELIZATION
$N \times N$ threads
(b)
PIXEL-ORIENTED PARALLELIZATION
$N \times S_X \times S_Z$ threads
(c)
$\sum$
S_Z pixels
PIXEL-ORIENTED PARALLELIZATION
$S_X \times S_Z$ threads
(d)
S_X pixels

Figure 1. Parallel scheme for beamforming acceleration.

model and their particular characteristics as well as for 2D and 3Dimaging, in order to squeeze the maximum performance and speed [6] [7].

4. Library Performance Evaluation

To evaluate the performance of the library we have chosen two experimental scenarios, a Multi-Tissue ultrasound phantom (040 GSE model by CIRS Inc. company) to evaluate the 2D-imaging; and a methacrylate piece with five drills for 3D imaging.TFM has been chosen as the beamforming method for both cases. Details are given in **Table 1**.

As a 2D imaging example, we use a linear 64 elements transducer which results in a FMA matrix of 4096 signals. The result of the processing is shown in **Figure 2(a)**.

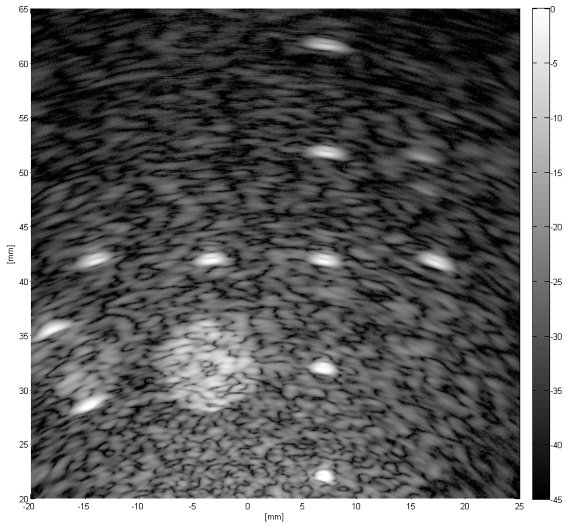
In **Table 2** and **Table 3** computing times can be observed for several CPUs (testing in mono-core and multi-core). Respect to GPUs, they are equipped with different number of cores.

Given that a standard PC cannot use a NVIDIA Tesla, we underline the results of NVIDIA Quadro 4000, which has 256 cores and 2 GB RAM and took around 105 ms for an image of 512 × 512 pixels, as well as, the results of the NVIDA QuadroK5000, which took less than 50 ms.

As a 3D imaging example we use a matrix array of 11 × 11 elements. 3D-TFM method has been applied, using an FMA of 14,641 signals and the same GPUs to generate a volume of 128 × 128 × 128 pixels which is shown in **Figure 2(b)**. The results of the imaging generation took less than 5 seconds in the NVIDA Quadro 4000, and less than 2 seconds in the NVIDA Quadro K5000, more details are described in **Table 4**.

5. Conclusions and Future Work

A fast and versatile library for the generation of ultrasound images has been created which exploits GPU technology to implement the beamformer via software. We have made a detailed description of the library features and it has been quantified the benefits of using the GPU as a processing tool. So by using a simple graphics card equipped with NVIDIA CUDA technology now is possible to accelerate the development of new imaging techniques.


(a)

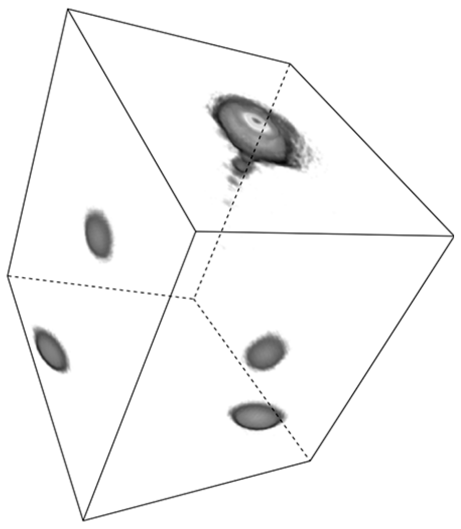

(b)

Figure 2. (a) TFM image from Multi-Tissue ultrasound phantom; (b) Volumetric image from 5 drills using 3D-TFM method.

Table 1. Beamformation scenarios tested for 2D and 3D imaging.

Parameters	Image	
	2D Imaging	3D Imaging
Scenario	Multi-Tissue Phantom	Methacrylate piece
Medium velocity	1540 [m/s]	2690 [m\s]
Array size	64 elements	121 elements
Array pitch	0.28 [mm]	1.0 [mm] both directions
Imaging frequency	2.6 [MHz]	3.16 [MHz]
Sample frequency	40 [MHz]	40 [MHz]

Table 2. Performance of CPU-based TFM algorithm.

CPU processor	RAM	# Cores	TFM time [seg]
Intel Core 2 Quad Q9450	4 GB	1	24.13
		4	6.96
Intel Core i7 3632Q M	8 GB	1	6.91
		8	2.08
Intel Xeon E51650	30 GB	1	6.84
		4	2.23
		8	1.80
		12	1.64

Table 3. Performance of GPU-based TFM algorithm using CHEETAH.

NVIDIA GPU	RAM	# Cores	TFM time [mseg]
GeForce 540 M	1 GB	96	265.57
GeForce 9800 GTX+	512 MB	128	215.01
GeForce 635 M	1 GB	144	239.27
Quadro 4000	2 GB	256	105.07
Quadro K2000	2 GB	384	119.23
Quadro K5000	6 GB	1536	45.11

Table 4. Performance of GPU-based TFM-3D algorithm using CHEETAH.

NVIDIA GPU	RAM	# Cores	TFM time [mseg]
GeForce 540 M	1 GB	96	7972.70
GeForce 9800 GTX+	512 MB	128	7056.01
GeForce 635 M	1 GB	144	7587.25
Quadro 4000	2 GB	256	4874.28
Quadro K2000	2 GB	384	6697.05
Quadro K5000	6 GB	1536	1929.13

We are currently working on the implementation of more beamforming algorithms (e.g. adaptative beamforming and PA), on supporting CPU capabilities for parallel computing (OpenACC, OpenMP, MPI) and multi-GPU processing and on improving the overall performance. We are open to receive collaboration/feedback of any group that will be interested in using our library in their own research.

Acknowledgements

This work has been supported by the Spanish Government and the University of Alcalá under projects DPI2010-19376 and CCG2014/EXP-084, respectively.

References

[1] Hwu, W.-M.W. and Kirk, D.B. (2010) Programming Massively Parallel Processors: A Hands-On Approach.

[2] Hansen, J.M., *et al*. (2011) An Object-Oriented Multi-Threaded Software Beamformation Toolbox, SPIE Medical Imaging: Ultrasonic Imaging, Tomography, and Therapy. In: D'hooge, J. and Doyley, M.M., Eds., 79680Y.

[3] Romero-Laorden, D., *et al*. (2011) Field Modelling Acceleration on Ultrasonic Systems Using Graphic Hardware. *Computer Physics Communications*, **182**, 590-599. http://dx.doi.org/10.1016/j.cpc.2010.10.032

[4] Villazón-Terrazas, J., *et al*. (2012) A Fast Acoustic Field Simulator. In 43o Congreso Español de Acústica (TECNIACUSTICA), Evora, 1-9.

[5] Romero-Laorden, D., *et al*. (2009) Using GPUs for Beamforming Acceleration on SAFT Imaging. *IEEE International Ultrasonics Symposium*, Rome, 1334-1337.

[6] Martín-Arguedas, C.J., *et al*. (2012) An Ultrasonic Imaging System Based on a New SAFT Approach and a GPU Beamformer. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, **59**, 1402-1412. http://dx.doi.org/10.1109/TUFFC.2012.2341

[7] Romero-Laorden, D., *et al*. (2013) Strategies for Hardware Reduction on the Design of Portable Ultrasound Imaging Systems. Advancements and Breakthroughs in Ultrasound Imaging, G. P. P. Gunarathne, Ed., 2013. http://dx.doi.org/10.5772/55910

[8] So, H.K.H., *et al*. (2011) Medical Ultrasound Imaging: To GPU or Not to GPU. *IEEE Micro*, **31**, 54-65. http://dx.doi.org/10.1109/MM.2011.65

[9] Holmes, C., *et al*. (2008) Advanced Postprocessing for Scanned Ultrasonic Arrays: Application to Defect Detection and Classification in Non-Destructive Evaluation. *Ultrasonics*, **48**, 636-642. http://dx.doi.org/10.1016/j.ultras.2008.07.019

[10] Martínez-Graullera, O., *et al*. (2011) A New Beamforming Process Based on the Phase Dispersion Analysis. *The International Congress on Ultrasonics*, Gdansk, 185-188.

[11] Hunter, A.J., *et al*. (2008) The Wavenumber Algorithm for Full-Matrix Imaging Using an Ultrasonic Array. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, **55**, 2450-2462. http://dx.doi.org/10.1109/TUFFC.952

4 Related publications associated to Section 1.4.3

4.1 Introduction

The development of beamforming techniques in GPU is opening two new research fields for ultrasonic imaging. On one hand, the ability to implement in real-time more advanced processing techniques and complex beamforming (Capon [1], phase coherence imaging [12], etc) which are opening new applications and their implementation in real scenarios. On the other hand, a new type of imaging techniques is emerging based on what it has been named as *Ultrafast Imaging*. Its aim is to reach 1000 images/second and it is expected to be the basis of new type of applications especially in the biomedical field (ultrasound-based screening, diagnosis, and therapeutic monitoring). Thus, the work of Tanter [31], [146] makes a detailed analysis of these techniques basing their work on the plane-wave imaging and plane-wave compounding, while other studies are based on conventional techniques as Jensen [53], [102], [105] and Lockwood [39], [85].

This chapter is focused in resolve three key problems: MRC signals reduction and the speed-up in image generation process extending the parallel beamforming scheme applied in TFM + FMC, as explained in Paper 4.2; a new technique for synthetic aperture called FAST-SAFT which is based on the coarray model and uses a single element in emission and a set of subapertures in reception in order to reduce the acquisition time to at least in $\frac{N}{k}$ firings presented in Paper 4.3; and the improvement of dynamic range described in Book Chapter 4.4, a solution based on Golay pulse encoding with its integration in the FAST-SAFT imaging architecture.

4.1.1 Related publications

Apart from the full-text publications included in this chapter, in this section it is summarized all the contributions to this research topic published in journals and conference proceedings.

[32] C. d. M. Giraldo, G. Lambert, O. Martínez-Graullera, L. Elvira, D. Romero-Laorden, L. Gómez-Ullate, and F. M. D. Espinosa, *Sistema de control de calidad ultrasonico on-line, especialmente aplicable a procesos de fabricación de estructuras de materiales compuestos de fibra de carbono y con curado fuera de autoclave*, 2012.

[73] C. Kitano, O. Martínez-Graullera, D. Romero-Laorden, R. T. Higut, V. T. Prado, and S. C. G. Granja, "Synthetic aperture imaging for ultrasonic non-destructive testing," in *Signals and Images: Advances and Results in Speech, Estimation, Compression, Recognition, Filtering, and Processing*, C. C. C. Rosângela Fernandes Coelho, Vitor Heloiz Nascimento, Ricardo Lopes de Queiroz, João Marcos Travassos Romano, Ed., CRC Press, 2014.

[95] C. J. Martin-Arguedas, D. Romero-Laorden, O. Martinez-Graullera, M. Perez-Lopez, and L. Gomez-Ullate, "An ultrasonic imaging system based on a new saft approach and a gpu beamformer," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 59, no. 7, pp. 1402–1412, 2012, ISSN: 0885-3010. DOI: `http://dx.doi.org/10.1109/TUFFC.2012.2341`.

[123] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, M. Pérez-Lopez, R. Tokio Higuti, and A. Octavio, "Aceleración del proceso de conformación de haz en imagen ultrasónica mediante gpus," in *II Workshop en Aplicaciones de Nuevas Arquitecturas de Consumo y Alta Prestaciones*, 2009, ISBN: 978-84-692-7320-3.

[125] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, R. Tokio Higuti, and A. Octavio, "Using gpus for beamforming acceleration on saft imaging," in *IEEE International Ultrasonics Symposium*, Rome, Italy: IEEE, 2009, pp. 1334–1337, ISBN: 9781424443895. DOI: `10.1109/ULTSYM.2009.5441790`. [Online]. Available: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5441790`.

[129] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, and M. Parrilla, "Application of golay codes to improve snr in coarray based synthetic aperture imaging systems," in *7th IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, Hoboken, New Jersey, EEUU, 2012, pp. 325–328, ISBN: 978-1-4673-1070-3. DOI: `http://dx.doi.org/10.1109/SAM.2012.6250501`. [Online]. Available: `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=\&arnumber=6250501\&contentType=Conference+Publications\&searchField=Search\_All\&queryText=Application+of+Golay+codes+to+improve+SNR+in+coarray+based+synthetic+aperture+imaging+systems`.

[130] D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, A. Ibáñez, and L. Gómez-Ullate, "Paralelización de los procesos de conformación de haz para imagen ultrasónica con técnicas gpgpu," *Revista Iberoamericana de Automática e Informática Industrial*, vol. 9, no. 2, pp. 144–151, Apr. 2012, ISSN: 1697-7912. DOI: `http://dx.doi.org/10.1016/j.riai.`

2012.02.002. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1697791212000039http://www.sciencedirect.com/science/article/pii/S1697791212000039.

[132] D. Romero-Laorden, J. Villazón-Terrazas, J. Glez-Ilinas, and O. Martínez-Graullera, “Integration of golay pulse compression in a synthetic aperture imaging system based on minimum redundancy coarray,” in *43 Congreso Español de Acústica (TECNIACUSTICA)*, Évora, Portugal, 2012, ISBN: 978-989-20-3284-9. [Online]. Available: http://www.sea-acustica.es/index.php?id=567.

[133] D. Romero-Laorden, J. Villazón-Terrazas, O. Martínez-Graullera, and A. Ibáñez, “Strategies for hardware reduction on the design of portable ultrasound imaging systems,” in *Advancements and breakthroughs in Ultrasound Imaging*, G. P. P. Gunarathne, Ed., 2013, ISBN: 978-953-51-1159-7. [Online]. Available: http://www.intechopen.com/articles/show/title/strategies-for-hardware-reduction-on-the-design-of-portable-ultrasound-imaging-systems.

4.2 Using GPUs for beamforming acceleration on SAFT imaging

10.1109/ULTSYM.2009.0323

Using GPUs for beamforming acceleration on SAFT imaging

D. Romero, O. Martínez-Graullera, C. J. Martín
Dep. Sistemas
IAI - CSIC
Arganda del Rey (Madrid), Spain
dromerol@iai.csic.es

R. T. Higuti
Dep. Electrical Engineering
UNESP
Universidade Estadual Paulista
Ilha Solteira, SP, Brasil

A. Octavio
Dep. Señales, Sistemas y Tecnologías
Ultrasónicas
IA - CSIC
Madrid (Spain)

***Abstract*—SAFT techniques are based on the sequential activation, in emission and reception, of the array elements and the post-processing of all the received signals to compose the image. Thus, the image generation can be divided into two stages: (1) the excitation and acquisition stage, where the signals received by each element or group of elements are stored; and (2) the beamforming stage, where the signals are combined together to obtain the image pixels. The use of Graphics Processing Units (GPUs), which are programmable devices with a high level of parallelism, can accelerate the computations of the beamforming process, that usually includes different functions such as dynamic focusing, band-pass filtering, spatial filtering or envelope detection. This work shows that using GPU technology can accelerate, in more than one order of magnitude with respect to CPU implementations, the beamforming and post-processing algorithms in SAFT imaging.**

I. Introduction

The use of synthetic aperture techniques (SAFT) to reduce the volume and complexity of the imaging systems has been a topic profusely studied in several application areas, such as radar [1], sonar [2] or ultrasonic imaging [3], [4]. These techniques are based on sequentially activating, one by one, the array elements in emission-reception. Once all signals have been stored, they are combined together to obtain the image pixels [5]. The complexity of the algorithms and the high number of signals and image points involved, justify that the processing stage can result too slow when a typical PC-based system is used.

In last years, one of the dominant trends in microprocessor architectures has been the continuous increment of the chip–level parallelism. Multicore CPUs providing 2–4 scalar cores are now commonplace, and there is no doubt that the trend towards increasing the parallelism will continue on "manycore" chips. However, Graphics Processor Units (GPUs) have been at the leading edge of this drive towards increased chip–level parallelism, and nowadays they mainly are manycore processors. As their name implies, GPUs came about as accelerators for graphic applications, predominantly those using the OpenGL and DirectX programming interfaces. Due to the tremendous parallelism inherent in graphics, GPUs are massively parallel machines. Although originally they were pure fixed-function devices, the demand for real time and 3D graphics made them to evolve into increasingly flexible highly parallel, multithreaded, manycore processors with extremely high computational power and very high memory bandwidth. In fact, the current NVIDIA GPUs contain up to 240 scalar processing elements per chip [6] and, in contrast to earlier generations of GPUs, they can be programmed directly in C using CUDA [7], [8].

CUDA (*Compute Unified Device Arquitecture*) is an API (*Application Programming Interface*) where an application is organized into a sequential host program that may execute parallel programs, referred as *kernels*, on a parallel device. It provides to the developers the means to execute parallel programs on the GPU.

In this paper it is presented an approach to beamforming acceleration using GPGPU techniques. The GPU is able to support several thousands of concurrent threads, being necessary to maximize the parallelism to obtain an efficient algorithm on this kind of architectures. This solution is a good compromise between price and performance for ultrasonic imaging.

II. System Overview

The proposed hardware/software solution for real time SAFT imaging system consists mainly of two subsystems, as it is shown in figure 1.

A) The first subsystem, dedicated to the excitation and the data acquisition, is schematically represented on the left side of figure 1. Essentially, it requires a reduced hardware: multiplexer, pulser, analog/digital converters, FPGA and memory components. The pulser excitation, multiplexer control, acquisition of the signals and preprocessing are some of the actions that take place in this part. The acquisition strategy is based on the coarray tool, concretely in the 2R-SAFT method [9], which requires only one emission channel and two parallel receiving channels. This technique is explained in detail in the section III. All the received signals are stored and then they are transferred to the host by means of a USB connector.

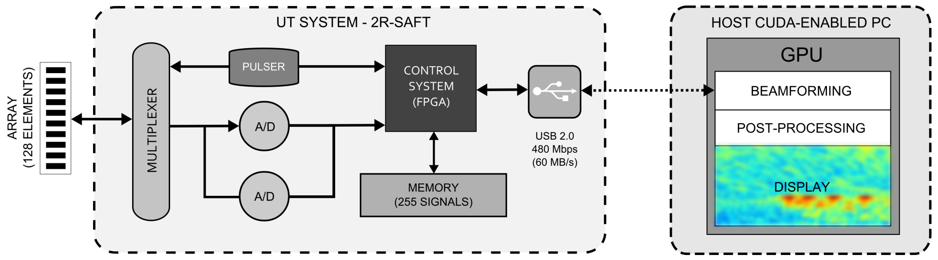


Figure 1: Proposed hardware/software solution: (left) UT SYSTEM 2R-SAFT and (right) CUDA-ENABLED PC

B) The second subsystem, which is shown in the right side of figure 1, is centered in the image generation. It makes use of the great parallel potential of a GPU in order to develop the beamforming, post-processing and display stages. These are computationally expensive algorithms which require a high capacity of processing.

III. Coarray Saft: Acquisition

The coarray is a mathematical tool used to study the beam pattern generated by pulse-echo systems. Basically, it is the virtual aperture of a system that produces, in one way, the same beam pattern than the original system working on emission and reception.

Suppose a linear array with N elements, being an the complex weights of the transducers. In far-field, and assuming very narrow band signals, the radiation pattern could be written

$$f(u) = \sum_{n=0}^{N-1} a_n e^{jkx_n u} = \sum_{n=0}^{N-1} a_n e^{jkndu} = \sum_{n=0}^{N-1} a_n (e^{jkdu})^n$$

where $u = sin(\theta)$, and θ is the angle measured from the perpendicular to the array. Substituting e^{jkdu} by the complex variable z, the radiation pattern can be expressed as a polynomial, that corresponds with the Z-Transform of the sequence a_n. Thus, considering a pulse-echo system, the complex radiation pattern will be the product of two polynomials with degree $N-1$.

$$f_{total}(z) = Z\{c_n\} = \sum_{n=0}^{2N-2} c_n z^n = \sum_{n=0}^{N-1} a_n z^n \;.\; \sum_{n=0}^{N-1} b_n z^n$$

where a_n and b_n are the gains applied to the transducers in emission and reception, and c_n is the coarray ($Z\{c_n\}$ represents the Z-Transform of the sequence c_n). Thus, the radiation pattern of the system in continous wave is directly the DFT of the coarray [10].

In synthetic aperture systems, each scanned image is obtained after several firing sequences of the elements as it shown in Figure 2. According to this, the coarray may be expressed like a sum of several sub-coarrays, each one obtained as the convolution of two sub-apertures that represent the weights of the active elements used to emit and receive the signals each time.

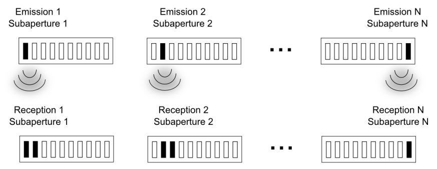

Emission 1 Subaperture 1
Emission 2 Subaperture 2
Emission N Subaperture N
Reception 1 Subaperture 1
Reception 2 Subaperture 2
Reception N Subaperture N

Figure 2: Firing sequences of the elements in 2R-SAFT

The 2R-SAFT method [9] has some advantages that are particularly useful to the aims of the present work. With this configuration, $2N-1$ signals are received, which are associated, one by one, to every virtual element of the coarray, producing thus a dense coarray of 2N-1 elements as it is observed in figure 3. As a consecuence, 2R-SAFT reduces the hardware cost (only one channel in emission and two channels in reception are needed) and generates a complete coarray, allowing to obtain images with good performance, high lateral resolution and grating lobes free [9].

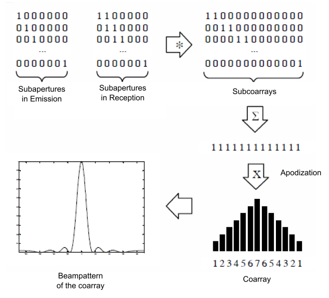


Figure 3: Coarray synthesized by 2R-SAFT

IV. Gpu Beamforming

Shifting an algorithm from CPU to be executed in GPU is not an easy task. Functions that are executed many times independently on different data are prime candidates. Essentially, the algorithms used in beamforming usually imply the operation with a big volume of signals to calculate diverse results in a repetitive way and, therefore, they are good candidates for GPU computation. The major problem of the GPU computing is to find the best strategy to parallelize the algorithms, in order to obtain the maximum benefit.

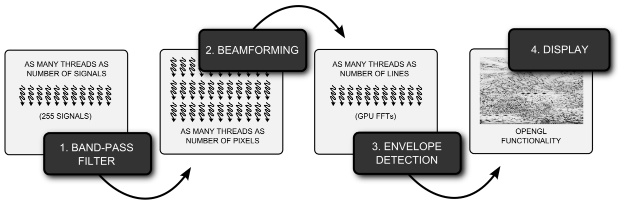


Figure 4: Schemes of the strategy followed: (1) One thread per signal, (2) One thread per image pixel, (3) One thread per image line, (4) Display using OpenGL

In the present work we have designed a specific solution for every different stage of the algorithm, in order to maximize GPUs efficiency. To understand the problem in detail we analyze the strategy followed, which is schematically presented in Figure 4. These are the computation steps on GPU:

① On the first step, a pass band filter is applied to all the signals received in the acquisition stage. This step is fundamentally beneficial to remove the different off-sets introduced by the capture stage and to reduce the noise. In this case, the strategy of parallelism consists on launch one thread of execution on GPU (drawn as wavy arrows in figure 4) per store signal, applying the filtering in parallel.

② The second step is focused on the beamforming algorithm to generate the image. In this case, the parallelization have been carried out launching a thread per image pixel. This thread calculates the corresponding value for this point of the image, allowing a very good parallelization and getting the result very fast. The beamforming stage is composed of several sub-steps:
   a) Calculate the focusing times. Here, dynamic focusing on emission and reception are applied.
   b) Image composition. An interpolation function is applied to calculate the value of the pixel, being able to choose between linear or polynomic interpolation with no differences in computational cost between both options. After that, several spatial filters can be applied, such as blackman-harris, triangular, hamming or boxcar apodizations.

③ Later on, the Hilbert Transform is applied to every line of the image, in order to detect the envelope, launching one thread per image line and using very fast optimized algorithms (GPU FFTs provided by a NVIDIA library).

④ Finally, the generated image is displayed directly using the OpenGL libraries, which provides specific functionality for graphics representation.

Due to the notable differences between CPU and GPU architectures, it is necessary to do some considerations about the GPU computing. It is important to emphasize that the memory transactions between CPU and GPU are pretty slow and they must be minimized as much as possible in order to improve the general performance. Furthermore, the device memory is limited so, in most cases, it could be necessary to reduce the volume of data to be processed in parallel. The reading time from global device memory is very slow so it is recommendable to use some other mechanisms, such as the texture memory or the shared memory. Finally, it is necessary to simplify the thread operations, in order to homogenize the execution time of the threads and to limit the write access to GPU memory. Unfortunately, the direct implementation on GPU of the algorithms originally designed by CPU usually is not the best option.

V. RESULTS

To test our parallel algorithms a NVIDIA GeForce GTX 295 graphic card has been used. This card has 2 GPUs inside but, at the moment, only one of them has been employed. There are 240 cores per GPU with 1GB of global memory. It has been installed in a computer with a four-core 2.66 GHz Intel Q9450 processor and 4GB of RAM. An evaluation about the times measured on GPU in the image reconstruction stage was made using simple floating precision.

Table I shows the times obtained using the GPU. The differences between the diverse image sizes are evident because of a mayor use of GPU resources. The first row shows the acquisition time plus the transmission time of 255 signals with 1000 samples through a bus of 26MB/s. In the second row, the pass-band filter presents a homogenous time in all the cases because it is applied to the same amount of signals. By contrast, in the beamforming step, which is the kernel of the image reconstruction stage (third row), a high increment of the computing time is appreciated as the image size grows up. The envelope detection makes use of GPUs FFTs and normalizes the length of the lines to the nearest value power of two, in order to increase the global performance; for this reason, the computed times are very similar. Finally, the time used to display the results is very fast, being not necessary to copy the image from GPU to CPU because OpenGL provides mechanisms to use the GPU directly for displaying.

The framerate achieved, supposing that acquisition and beamforming stages are executed sequentially, is shown in the last row. Otherwise, considerating a parallel execution of both stages the framerate would be determined by the slowest task (usually the acquisition), being about 50 images/second.

Figure 5 shows several snapshots of images calculated on GPU at different instants. As it can be observed on the top of the figure, the active zone of analysis is a region of 300x300 pixels, which is calculated dynamically each time. It is important to remark that is possible to make scroll over the whole image to change the active zone and explore more in detail. Furthermore, the zoom function is able to increase the detail of the images without resolution loss and without

Table I: Times achieved on GPU by 2R-SAFT imaging system

Actions	Image size (pixels)			
	100x100	200x200	300x300	500x500
Acquisition and copy to host	20 ms			
Pass-band filtering	1.05 ms	1.03 ms	1.07 ms	1.06 ms
Beamforming	0.69 ms	2.52 ms	4.47 ms	10.92 ms
Envelope detection	0.45 ms	0.58 ms	0.84 ms	2.05 ms
Displaying	0.07 ms	0.10 ms	0.13 ms	0.20 ms
Global beamforming time	2.70 ms	4.53 ms	6.51 ms	14.23 ms
Framerate	44 img/s	41 img/s	38 img/s	30 img/s

increasing the computational time, operating always in real time.

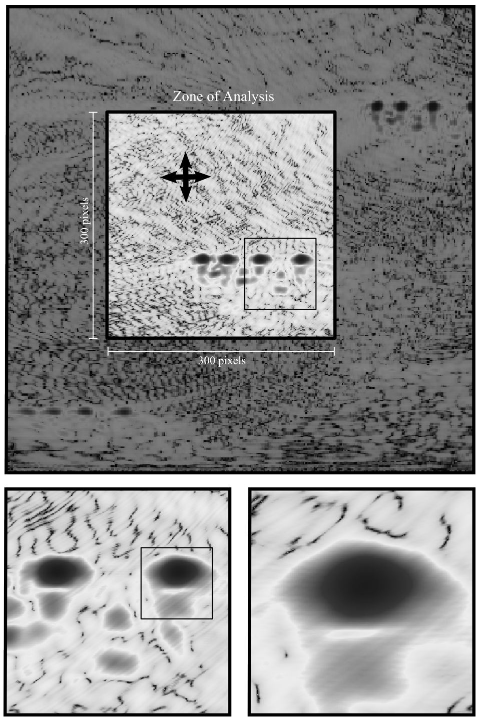


Figure 5: Images obtained with the 2R-SAFT system. Size of images: 300x300 pixels. Dynamic change of the representation range: zoom and scroll functions allow obtaining images with more detail without resolution loss and no increment of computational time. Dynamic focusing on emission and reception is applied

VI. Conclusion

One of the characteristics of the ultrasonic imaging generation is the requirement of high parallelism, as it needs applying a set of processing algorithms to a great amount of digitized signals and a dense grid of spatial points. In this work we have explored the GPU parallelism, in order to reduce processing time during the beamforming process. Using a simple graphics card equipped with NVIDIA CUDA technology, the experimental results show that the beamforming computing time per pixel (dynamic focussing, interpolation, apodization, envelope) is 0.53 μs. That supposes 14 ms for an image of 500x500 pixels. Thus, the framerate of the beamforming process in this case is 70 img/s. Therefore, GPUs parallelization constitutes an excellent method of accelerating the imaging formation at very low cost and complexity, and it can be applied to more cases of study. It has been demonstrated that shifting the beamforming process from CPU to programmable GPU is a very good solution for beamforming acceleration on SAFT imaging.

Acknowledgment

This work was supported in part by the Spanish Ministry of Science and Innovation under Grant BES-2005-7704, BES-2008-008675 and the projects DPI2007-65408-C02 and TRA2007-67711/AUT.

References

[1] N. A. Goodman and J. M. Stiles, “The information content of multiple receive aperture sar systems,” *Proceedings of IEEE Geoscience and Remote Sensing Symposium*, vol. 4, pp. 1614–1616, 2001.
[2] N. C. Yen and W. Carey, “Application of synthetic-aperture processing to towed-array data,” *The Journal of the Acoustical Society of America*, pp. 158–171, 1989.
[3] J. A. Jensen, S. I. Nikolov, K. L. Gammelmark, and M. H. Pedersen, “Synthetic aperture ultrasound imaging,” *Ultrasonics*, vol. 44, pp. e6–e16, 2006.
[4] P. D. Corl, P. M. Grant, and G. S. Kino, “A digital synthetic focus acoustic imaging system for nde,” *IEEE Ultrasonic Symposium*, pp. 263–268, 1978.
[5] G. R. Lockwood, J. R. Talman, and S. S. Brunke, “Real-time 3-d ultrasound imaging using sparse synthetic aperture beamforming,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 45, pp. 980–988, 1998.
[6] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, “Nvidia tesla: A unified graphics and computing architecture,” *IEEE Micro*, vol. 28, no. 2, pp. 39–55, Mar/Apr 2008.
[7] N. Corporation, *NVIDIA CUDA 2.1 Programming Guide*, February 2009.
[8] J. Nickolls, I. Buck, M. Garland, and K. Skadron, “Scalable parallel programming with cuda,” *Queue*, vol. 6, no. 2, pp. 40–53, Mar/Apr 2008.
[9] C. J. Martin, O. Martinez, and L. G. Ullate, “Reduction of grating lobes in saft images,” *IEEE International Ultrasonics Symposium*, pp. 721–724, 2008.
[10] B. D. Steinberg, *Principles of Aperture and Array System Design*. Wiley-Interscience, 1976.

4.3 Application of Golay codes to improve SNR in coarray based synthetic aperture imaging systems

2012 IEEE 7th Sensor Array and Multichannel Signal Processing Workshop (SAM)

Application of Golay codes to improve SNR in coarray based synthetic aperture imaging systems

David Romero-Laorden, Oscar Martinez-Graullera, Carlos Julián Martín-Arguedas, Montserrat Parrilla-Romero
CAEND (UPM-CSIC), C/Serrano 144, Madrid 28006, Spain
Email: david.romero@csic.es

***Abstract*—Minimum redundancy synthetic aperture solutions (based on the coarray concept) are usually limited by poor signal to noise ratio. The use of pulse coding based on Golay codes can help in order to improve the signal to noise-ratio. Two solutions (2R-SAFT and 4R-SAFT) for data reduction in the beamforming process of multi-element synthetic aperture imaging are used here to reduce the system's architecture, cost and requisites. Alongside with the application of Golay codes it is experimentally demonstrated the improvement of the dynamic range and SNR, achieving an image quality comparable to the Total Focusing Method and establishing its use for suitable real-time medical applications.**

I. INTRODUCTION

Phased array techniques are commonly used for ultrasonic imaging applications. The use of arrays with a high number of elements significantly increase the quality of the images (both resolution and contrast), and consequently also the complexity and cost of the imaging system. Synthetic Aperture Focusing Techniques (SAFT) have been a topic deeply applied in several areas such as radar, sonar or ultrasonic imaging. These techniques are a suitable solution to reduce the hardware requisites and therefore the consumption and cost of the systems, making possible their miniaturization and portability [1], [2].

SAFT techniques are based method on the sequential activation in emission and reception of every array element. Once all the RF signals has been stored in memory, a beamforming process is applied to focus the image dynamically in emission and reception, thus obtaining the maximum quality at each image point. According to the description of teachers Wilcox and Drinkwater in the literature [3], a Full Matrix Array capture (FMA) is used to obtain the complete set of signals, and after the acquisition stage, a Total Focusing Method (TFM) is applied to generate an image [3]. Supposing an array of N elements it means to process N^2 signals. Thereby, TFM requires a high number of signals which increase the cost involved in the acquisition system, the data transfer and the computational cost.

In order to develop real-time imaging systems, the solution relative to this problem is to employ acquisition methods based on solutions of minimum redundancy (coarray-based). Coarray-based implementations with GPGPU techniques have been proposed [4] achieving an speed up of 150x over CPU beamforming thus establishing the basis for their use in-vivo medical systems.

The simplest coarray-based technique is known as 2R-SAFT (two receivers SAFT [5]) and it uses one emitter and two receivers, thus reducing the number of electronic channels and providing images with good quality and grating lobes free [5]. One of its main drawbacks is the low signal-to-noise ratio. The technique 4R-SAFT [6] is also based on the coarray principle, although in this case one emitter and four receivers centred around the active element are used improving the SNR, because two signals per coarray element are used instead of a signal as it happens in 2R-SAFT method.

However, signal-to-noise ratio is also compromised both 2R-SAFT and 4R-SAFT methods because they involve only $2N-1$ and $4N-3$ signals (which can be reduced to $2N-1$ in the acquisition system [6]) respectively, compared to the TFM which uses N^2. The signal level decreases not only due to the tissue attenuation but also because of the nature of the emissions. This problem can be addressed in several ways: (1) using multiple elements to create virtual sources and (2) generating long pulses with temporal encoding. The use of a long waveform is a well known and used technique in the radar systems involving either sending series of short pulses with phase encoding. There are numerous binary sequences that can be used for encoding ultrasonic signals, like Barker or Chirp codes [7].

This work is focused on the use of Golay complementary pairs [8], [9] in conjunction with minimum redundancy solutions in order to enhance the dynamic range and SNR, reaching the image quality of the TFM method. At the end of the paper B-mode images from measurements on a clinical phantom will be presented.

II. GOLAY ENCODING FOR ULTRASONIC EXCITATION

Golay complementary pairs have been deeply used for transducer excitation because the sum of its auto-correlation function has a main peak and zero side-lobes [8]. A pair is composed of two binary sequences, $A[n] = [a_0, a_1, \ldots, a_{N-1}]$ and $B[n] = [b_0, b_1, \ldots, b_{N-1}]$, of the same length N such that $a_i, b_i \in \{-1, +1\}$. The auto-correlation functions of $A[n]$ and $B[n]$ have side lobes equal in magnitude but opposite in sign. The sum of these independent auto-correlation functions provides an ideal delta function according to:

$$C_A[n] + C_B[n] = \begin{cases} 0, & n = 0 \\ 2N, & otherwise \end{cases} \tag{1}$$

where $C_A[n]$ and $C_B[n]$ are the auto-correlation functions of $A[n]$ and $B[n]$, respectively, for any integer n satisfying. The construction of Golay code pairs is done by recursively operating on shorter Golay code pairs with the *"negate and concatenate"* method, a technique used by Golay [8] to create longer pairs from the shorter, hand-constructed pairs given. Specifically, if $A[n]$ and $B[n]$ are the N-digit binary representations of a complementary pair of codes, then a new pair of complementary codes $A'[n]$ and $B'[n]$ of length $2N$ can be formed by concatenating $B[n]$ to $A[n]$ and concatenating $\sim B[n]$ to $A[n]$ where $\sim B[n]$ is the complement of $B[n]$. Thus, $A'[n] = A[n] \mid B[n]$, and $B'[n] = A[n] \mid\sim B[n]$.

One of the major drawbacks of Golay codes is associated with the firings of the array elements. Using this approach, two shots are needed for each element in order to complete both A and B codes respectively. In our work, Golay codes of length equal to 8 bits have been used, being $A[8] = [+1+1+1+1+1-1-1+1]$ and $B[8] = [+1-1+1-1+1+1-1-1]$, producing consequently a gain of 16dB according to equation 1. Figures 3 and 5 shows the results for an experimental test using the Golay codes of 8 bits which have been previously mentioned, and their corresponding images (with and without encoding) are analysed in section IV.

III. COARRAY BASED SAFT METHODS

The simplest SAFT method is based on the sequential activation in emission and reception of every array element. Although this method requires very few resources, its main drawbacks are the low signal-to-noise ratio (SNR) and the existence of grating lobes which can seriously limit the image contrast.

As we previously introduced, the two techniques used in this work are based on the coarray concept, which is also usually known as *"effective aperture"* in ultrasonic literature [10]. The first technique is used here (2R-SAFT) [5] utilizes only a transmitter and two receivers for each shot. The system is very similar to the single-SAFT, being the main difference the increment of the number of receiving channels (from one to two), and the size of the memory buffer to $2N-1$ signals (derived from the sequence of activation receivers). Thus, a full coarray grating lobes free is obtained and signal to noise ratio is improved in 3dB, because it doubles the total number of signals involved in the image generation ($2N-1$ against N in single-SAFT).

A second reduction method called 4R-SAFT [6] is also used in this work, where one element is utilized to transmit and a sub-aperture of 4 elements, centred around the active element, to receive. Therefore, two signals per coarray element are now obtained at different shots, thus doubling the number of signals of 2R-SAFT. In [6], it has been mathematically demonstrated that if far field is assumed, the signals involved in the formation of the same element of the coarray can be considered practically in phase, being no needed to separately transmit them to the beamformer. Thus, the signals which belong to the same element of the coarray can be added directly without focusing in the acquisition stage [6]. This fact reduces to $2N-1$ the total number of signals to be transferred from the acquisition stage to the buffer memory when 4R-SAFT is employed.

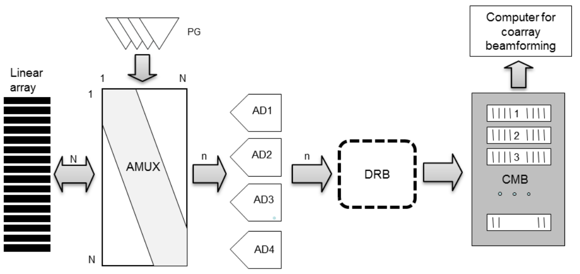


Fig. 1. Schematic electronic architecture of the 4R-SAFT system

Figure 1 shows how is the electronic architecture necessary to implement 4R-SAFT. As we observe, it includes blocks for analog multiplexing (AMUX), electronic circuits for analog processing and four analog to digital conversion (ADi) and pulse generation (PG). Later on, a data reduction block (DRB) has been inserted into the architecture, where the signals of identical spatial frequency are added in phase just when they are acquired.

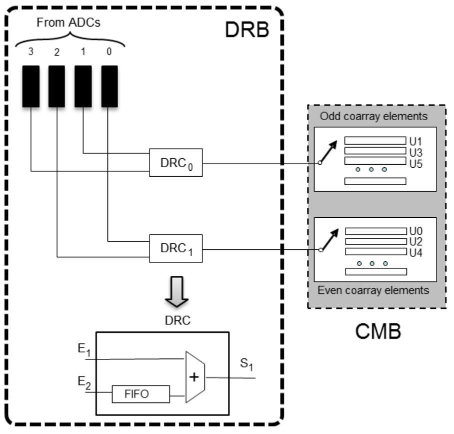


Fig. 2. Schematic architecture of the DRB and coarray memory block (CMB)

Detailed DRB block is shown in figure 2 and as we see, it has four data inputs, which are connected to the ADCs. The addition in phase of the signals is made by two data-reducing circuits (DRC), which are formed by an adder and a FIFO memory, where the data produced in the previous acquisition are stored. The DRCs are the output ports of the block, which produce the data for the coarray memory block (CMB), where only $2N-1$ signals are stored. CMB is divided into two parts, where the signals corresponding to the even and odd coarray elements are stored separately. It is not difficult to prove that the data reduction architecture here exposed executes the additions of the signals and stores the results in the coarray memory block, from which the data can be transmitted to a conventional computer for fast digital beamforming using a GPU [4].

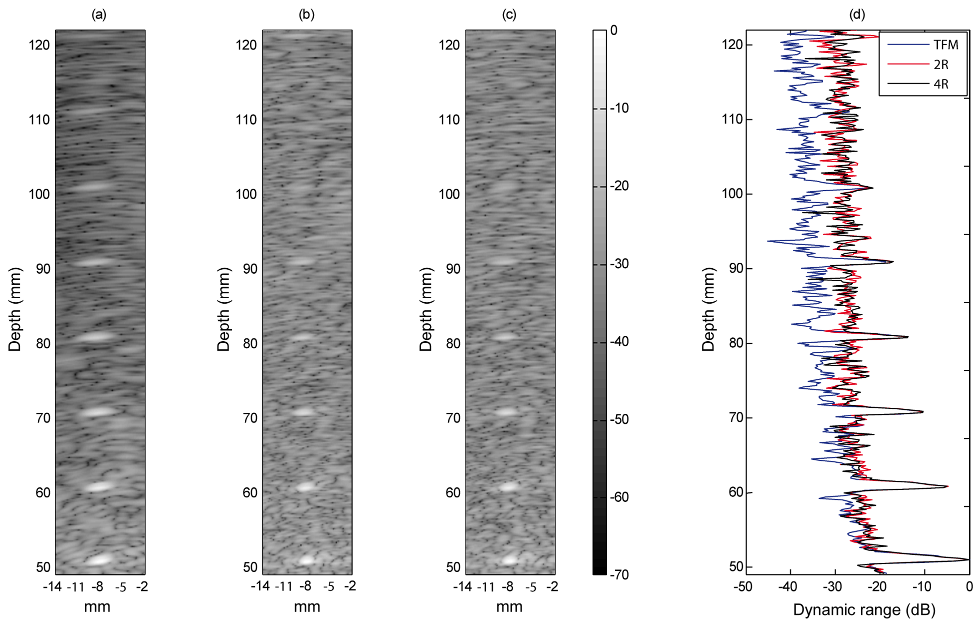


Fig. 3. Experimental results without Golay coded excitation for (a) TFM, (b) 2R-SAFT, (c) 4R-SAFT and (d) Lateral profiles in depth (50mm to 122mm). Dynamic range in all images is -70dB

A. Application of Golay codes to increase signal-to-noise ratio

The application of the Golay codes described in section II in conjunction with 4R-SAFT method can be easily integrated considering the electronic architecture presented in the previous section. Taking into account, as we have seen, that Golay encoding needs to fire twice, 2R-SAFT and 4R-SAFT architectures allow to make the deconvolution process inside the DRC integrating the excitation codes into it. For this purpose, figure 4 shows the new data-reducing circuit (DRC$_G$) proposed in this work. Two FIR filter blocks have been added in each one of the channels to perform the Golay code deconvolution which is alternated between each shot:

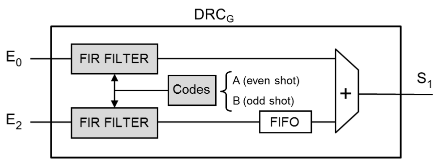


Fig. 4. DRC$_G$ block proposed for Golay encoding in 4R-SAFT

This process is mathematically identical both 2R-SAFT and 4R-SAFT being the system traffic data different in each case, $2N-1$ and $4N-3$ respectively. The main difference in 4R-SAFT method is relative to the number of firings which have been reduced from $2N$ to N considering N the number of array elements.

IV. Experimental results

We present a demonstration about the use of codes and synthetic aperture techniques, being possible to obtain images with higher penetration depth and suitable for medical imaging. The measurements were done on a tissue phantom (Model 040GSE - CIRS Inc.) with a 0.5dB/cm attenuation, where wires of 0.1mm diameter were examined using a 2.6MHz phased array transducer of $N = 64$ elements, 0.28mm of pitch (Vermon Inc.). We will use the Total Focusing Method (TFM) as a reference model to examine a line of wires in order to evaluate the effects of attenuation (covering an area starting at 50 to 122mm depth) compared to 2R-SAFT and 4R-SAFT techniques without and with encoding. The dynamic range is 70dB in all cases. Therefore, the experiments were intended to try two different combinations of encodings: Figure 3 shows the images with no encoding and figure 5 presents temporal encoding using Golay codes.

The TFM image was composed for the DAS algorithm using the complete set of signals N^2 = 4096. Next, 2R-SAFT and 4R-SAFT were calculated using $2N-1$ = 127 signals. The selected region is composed by a line of 8 wires situated in near field.

Corresponding images with no encoding and relative to TFM, 2R-SAFT and 4R-SAFT are represented in figures 3a,3b,3c respectively. As we can see, the TFM image shows the 8 wires and it has an SNR of 20dB in the nearest zone (50mm) which decreases to 5dB at maximum depth (120mm), where the background noise has decreased from the initial -20dB to -40dB. The test for 2R-SAFT and 4R-SAFT methods has achieved similar results. Only 6 of the 8 wires are clearly

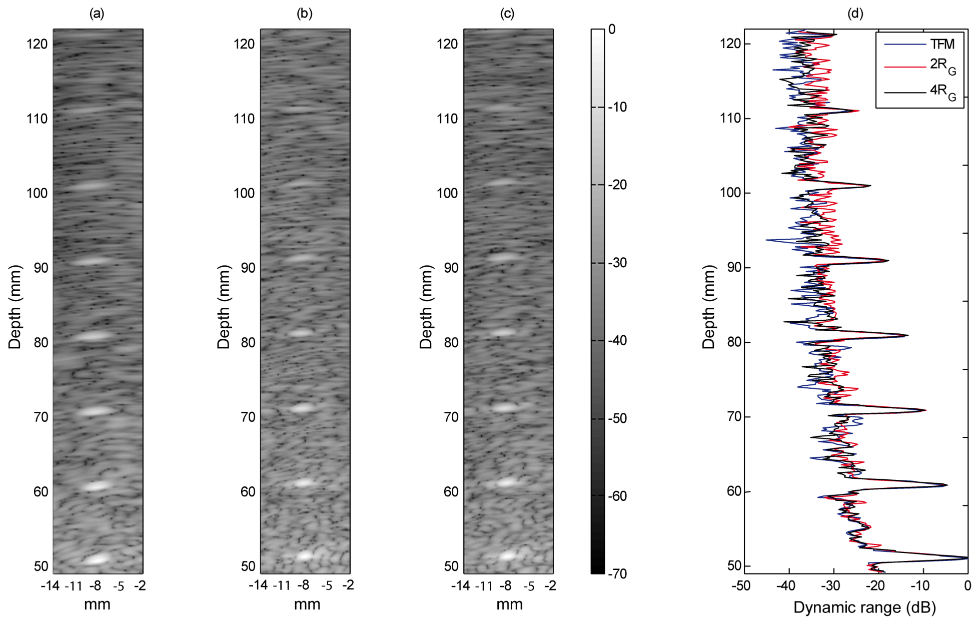


Fig. 5. Experimental results with Golay coded excitation for (a) TFM, (b) 2R-SAFT, (c) 4R-SAFT and (d) Lateral profiles in depth (50mm to 122mm). Dynamic range in all images is -70dB

distinguished and also SNR is worse compared to the previous case, with a background noise that is maintained in -30dB at 60mm depth.

Most interesting are the images using Golay coding excitation relative to 2R-SAFT and 4R-SAFT which are represented in figures 5b and 5c respectively. TFM image here presented has been calculated with no encoding as the previous experiment. Thus, if we analyse the results of the images generated using Golay codes, we can easily observe that the level of detection has been improved substantially. Now, both 2R-SAFT and 4R-SAFT techniques distinguish the complete line of wires. Moreover, the results obtained with 4R-SAFT are specially relevant because of two main factors: (1) the SNR achieved is comparable to TFM's method, which means a great improvement of the image quality in all the inspected region; and (2) the number of signals is drastically reduced from N^2 to $2N-1$, accelerating acquisition and processing velocities and therefore, producing an increment of the system's frame rate.

V. CONCLUSION

This work has presented two synthetic aperture techniques which in conjunction with the application of Golay encoding allow to optimize the relation between resources, velocity and image quality. Architecturally, we have described the integration of excitation codes into the acquisition system using 2R-SAFT and 4R-SAFT coarray-based techniques. In particular, with a limited increase of hardware complexity, 4R-SAFT technique makes possible to halve the number of firings, avoiding the double shot process of Golay codes. Therefore, this solution allows the development of high quality imaging systems with low requirements in a compact architecture. The experimental results have shown that the image quality obtained with this union is comparable to the Total Focusing Method.

ACKNOWLEDGMENT

This work has been supported by the Spanish Ministry of Science and Innovation under the project DPI2010-19376.

REFERENCES

[1] B.D. Steinberg, "Principles of aperture and array system design", New York: Willey and Song, (1976)
[2] J.A. Jensen et al, "Synthetic aperture ultrasound imaging", Ultrasonics, 44, pp. 515, (2006)
[3] Caroline Holmes, Bruce W. Drinkwater, Paul D. Wilcox, Post-processing of the full matrix of ultrasonic transmitreceive array data for non-destructive evaluation, NDT & International, Volume 38, Issue 8, December 2005, Pages 701-711
[4] Romero-Laorden, Martinez-Graullera, O., Martin, C.J., Higuti, R.T., Octavio, A., "Using GPUs for beamforming acceleration on SAFT imaging," IUS, 2009 IEEE International , pp.1334-1337
[5] Martin-Arguedas, C.J., Martinez, O., Ullate, L.G., Octavio, A., Godoy, G., "Reduction of grating lobes in SAFT images," IUS 2008. pp.721-724, 2-5
[6] Martin-Arguedas, C.J. , "Synthetic aperture techniques for ultrasonic image generation," Thesis, 2010. ISSN: 978-84-693-2364-9
[7] O'Donnell, M.: Coded Excitations Systems for Improving the Penetration of Real-Time Phased-Array Imaging Systems. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control, Vol.39, No.3, 1992.
[8] Golay, M.: Complementary Series. IRE Transactions on Information Theory, 1961.
[9] Nowicki A, Secomski W, Litniewski J, Trots I., "On the application of signal compression using Golays codes sequences in ultrasound diagnostic" Archives of Acoustics. 2003. vol. 28. 31324
[10] G.R. Lockwood, J.R. Talman, and S.S. Brunke, "Real-Time 3-D Ultrasound Imaging Using Sparse Synthetic Aperture Beamforming", IEEE Trans. Ultrason. Ferroelec. Freq. Contr. 45 (4), pp. 980-988, (1998)

4.4 Strategies for hardware reduction on the design of portable ultrasound imaging systems

Chapter 9

Strategies for Hardware Reduction on the Design of Portable Ultrasound Imaging Systems

D. Romero-Laorden, J. Villazón-Terrazas,
O. Martínez-Graullera and A. Ibáñez

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/55910

1. Introduction

In the last decade, ultrasonic imaging systems have been an essential tool for diagnosis in medical and industrial applications, especially in the Non Destructive Testing area (NDT). Conventional ultrasonic imaging devices produce high quality images with good resolution and contrast. However, these machines are usually associated to a high cost in hardware resources, as well as in the time required for the data acquisition and processing stages. This fact hinders the development of good quality, compact and low-power systems that can operate in a wide range of real-time applications.

In this sense, the Synthetic Aperture techniques (SAFT) have demonstrated to be an effective method to achieve these goals, minimizing the size of the systems and accelerating the image acquisition processes. Consequently, both power consumption and overall cost of the systems can be reduced making possible their miniaturization and portability. Conventional SAFT techniques are based on the sequential activation in emission and reception of every transducer element. Once all acoustic signals have been stored in memory, a beamforming process is applied in a post-processing stage in order to focus the image dynamically in emission and reception, obtaining the maximum quality at each image pixel. Despite of this, conventional SAFT techniques present some inconveniences which are summarized in the following points:

1. **Artifacts**. Conventional SAFT techniques produce grating lobes in the images due to the acquisition processes.
2. **Low contrast**. As a consequence of firing only one element at time the received signals have low signal-to-noise ratio, which results in low contrast images that are not feasible for regular imaging visualization (e.g. echography imaging needs very good images in order to reduce the fails in the diagnostic).

3. **Medium penetration**. And for the same reason, the penetration deep of ultrasound in the region of interest is smaller than the achieved using conventional imaging techniques (e.g. needed by cardiac imaging or industrial inspections).

In order to reduce some of these drawbacks, more sophisticated SAFT techniques have been proposed. Total Focusing Method (TFM) [1] is one of them, where each array element is sequentially used as a single emitter and all array elements are used as receivers. Thus, it is possible to obtain a set of $N \times N$ signals (Full Matrix Array capture, FMA) that is used to form the image. According to the description of professors Drinkwater and Wilcox [1–3], its name refers to the possibility of implementing dynamic focusing in emission and reception, which enables to obtain images perfectly focused at all points in the region of interest. However, the complexity of the acquisition process and the computational requirements of the beamforming make this method not appropriate for real-time purposes [1]. Other solutions that use an emission and reception sub-aperture have been also proposed [4–6], although they maintain a certain degree of hardware complexity (focussing is needed in emission and reception) and also require intensive computational capabilities to produce a real-time ultrasonic image.

To overcome the last inconveniences we propose a SAFT methodology based on a new paradigm, known as coarray [5, 6], which allows to use only one element in emission and a limited number of parallel channels in reception at each time. With the proposed solution, a strategy for a hardware reduction in ultrasonic imaging systems is possible, and it involves the following aspects:

- Optimization of the acquisition strategies to achieve the completeness of the coarray with a minimum number of hardware elements. In this sense, our objective is to establish a trade-off between the number of electronic channels, image quality and acquisition velocity [6].
- The use of pulse compression techniques to overcome the reduced capability of penetration when emission is limited to one element [5].
- The development of GPGPU[1] parallel beamforming techniques to achieve real time imaging [7].

This chapter is divided into two main sections. The first one is dedicated to analyse the use of the coarray paradigm as a tool for the design of ultrasonic imaging systems and to present several minimum redundancy coarray techniques. Moreover, Golay codes are presented and their integration within the presented SAFT methods is described. The second section presents the general ultrasonic imaging system's overview, its architecture and the parallel beamforming as a solution for ultrafast beamforming. Finally, we expose our conclusions and future research developments.

[1] General-purpose computing on Graphics Processing Units is the utilization of a graphics processing unit (GPU), which typically handles computation only for computer graphics, to perform computation in applications traditionally handled by the central processing unit (CPU). http://gpgpu.org

2. Coarray: New paradigm for the design of imaging systems

This section is focused on the development of ultrasonic imaging systems based on the pulse/echo aperture model which is known as coarray. In order to clarify this point, we are going to briefly review this mathematical concept and its principal implications.

The coarray is a mathematical tool that is often used by several authors as a way to quickly study the radiation properties of an imaging system [5, 6, 8, 9]. This concept is frequently referred to as *effective aperture* in ultrasound literature, and it basically is the virtual aperture which produces in one way the same beam pattern as the real aperture working in emission and reception as Figure 12 suggests.

Suppose a linear array with N elements. In far-field and assuming very narrow band signals, the radiation pattern could be written as:

$$f(u) = \sum_{n=0}^{N-1} a_n e^{jkx_n u} = \sum_{n=0}^{N-1} a_n e^{jkndu} = \sum_{n=0}^{N-1} a_n (e^{jkdu})^n \tag{1}$$

where a_n are the complex weights of the transducers and $u = sin(\theta)$ being θ the angle measured from the perpendicular to the array. Substituting e^{jkdu} by the complex variable z, the radiation pattern can be expressed as a polynomial, which corresponds with the Z-Transform of the sequence a_n. Thus, considering a pulse-echo system, the complex radiation pattern will be the product of two polynomials with degree $N-1$:

$$f_{total}(z) = Z\{c_n\} = \sum_{n=0}^{2N-2} c_n z^n = \sum_{n=0}^{N-1} a_n z^n \, . \, \sum_{n=0}^{N-1} b_n z^n \tag{2}$$

where a_n and b_n are the gains applied to the transducers in emission and reception, and c_n is the coarray ($Z\{c_n\}$ represents the Z-Transform of the sequence c_n). Returning to the unit circle ($|z| = 1$, $z = e^{jkdu}$) and considering equation 1 then the radiation pattern of the system in continuous wave is directly the DFT of the coarray [10].

In synthetic aperture systems, each scanned image is obtained after several firing sequences of the elements. According to this, the coarray can then be expressed as a sum of several sub-coarrays. Each of these sub-coarrays will be obtained as the convolution of two sub-apertures that represent the weights of the active elements used to emit and receive the signals each time.

Figure 1 illustrates the coarray generated by TFM method, which has been applied in ultrasound area since the late 60's and early 70's [11, 12]. As we briefly introduced in Section 1, it consists on the sequential emission with each one of the array elements in turn, and the reception in each shot with the full transducer aperture. As we can see, its coarray is fully populated what ensures a grating-lobe free radiation pattern.

The image quality achieved when TFM is employed is the highest possible, but it has, as its counterpart, the huge volume of data which is necessary to acquire. Thus, it requires more storage resources and processing capability than other techniques, which makes difficult its

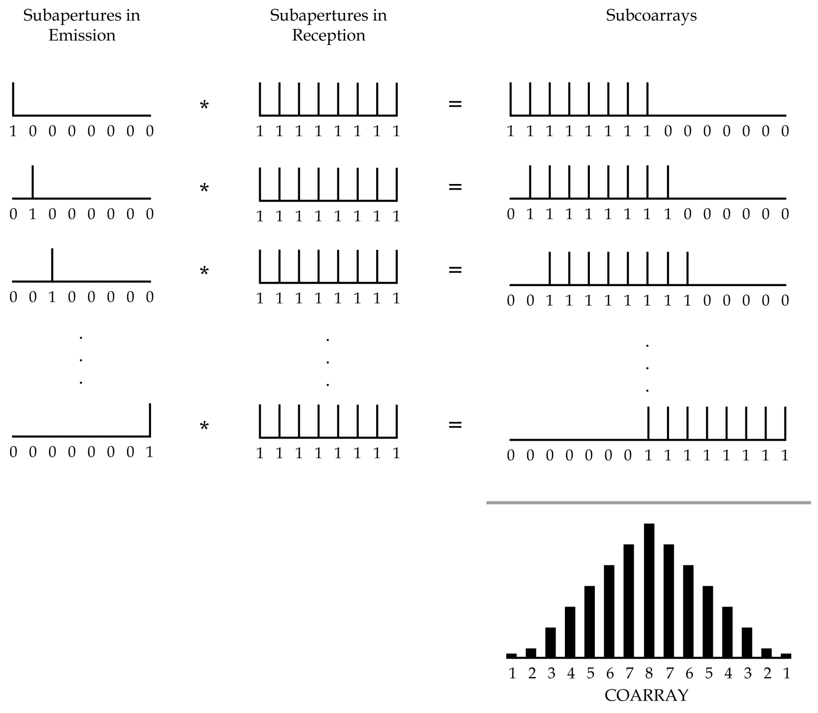


Figure 1. Firing sequences of the elements in TFM, and its corresponding generated coarray

practical implementation with todays' technology. To illustrate this, consider the following example: a 15 cm depth image, 40 MHz sampling rate, 64 channels, 1500 m/s medium velocity and 2 bytes per sample. Each firing generates approximately 1 MB of pulse-echo data, what supposes 64 MB of data to generate a single image frame when TFM is applied. For a frame rate of 20 images per second, it would be necessary to acquire and process 1.2 Gbytes of data per second.

The bandwidth of most I/O standards available today put in evidence that any of current data protocols can not deal with TFM requirements. Supposing a good efficiency and use of the resources (around 80)%, USB 2.0 port (released in April 2000) would be able to transfer less than one image per second (48 MB/s). A similar situation occurs if USB 3.0 (released in November 2008) is employed, being the maximum transmission speed up to 480 MB/s allowing to transfer around 7 images per second, even far respect to the maximum number of images which could be theoretically achieved. Finally, the most recent standard released in February 2011, known as Thunderbolt port and developed by Intel [13], combines PCI Express and DisplayPort into a new serial data interface that can be carried over longer and less costly cables. Thunderbolt has twice the transfer speed of USB 3.0 over copper wire (960 MB/s) giving us transferences of 14 images per second.

Therefore, it is clear that a reduction of data volume is desirable. In this sense, applying the coarray concept permits us to propose system designs that use less channels simultaneously working in emission and reception, but maintaining the same level of image quality. The key point for this is to use the coarray to search for solutions of minimum redundancy. This approach in conjunction with parallel computing techniques will offer an increment of acquisition velocity maintaining the highest quality and producing high frame rates with low power consumption. This topic will be the main focus of next two sections.

2.1. Minimum redundancy coarray solutions

Coarray analysis identifies which emitter-receiver combination completes each of its elements. In the TFM method seen before, we find that some of the elements are formed by a single signal (in concrete boundary elements) while the others increase progressively until reaching coarray centre with a value of N elements (Figure 1). Thus, we can consider as a minimum redundancy coarray that in which each element is composed of only one signal. Therefore, using the minimum possible number of signals the aperture's diffraction properties can be improved by manipulating the gain of the elements. With this goal in mind, it is possible to establish several strategies which maintain a balance between the number of parallel channels and the number of shots during acquisition processes.

2.1.1. 2R-SAFT acquisition strategy

2R-SAFT technique [14] has some particular advantages that make it very useful for ultrasonic imaging systems. 2R-SAFT uses only one element to transmit and two elements to receive. As it is shown in Figure 2, all elements are consecutively activated as single emitters, without the use of any beamformer in emission. At each shot, two consecutive channels are used as receivers requiring to store two signals per emission.

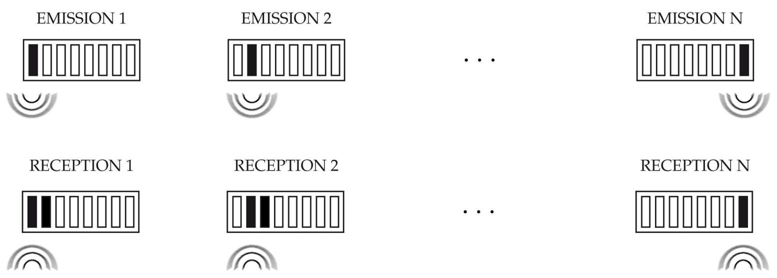


Figure 2. Firing sequences of the elements in 2R-SAFT

Thus, when the i^{th} element is used to emit a waveform, i and $i+1$ elements are used for receiving signals. For the last element of the array, only one signal is recorded. By employing an emitter in each shot all the received signals are completely uncorrelated, containing only information of a single transmitter-receiver pair.

Figure 3 shows the coarray generated when 2R-SAFT is employed. As we can observe, the coarray is fully populated ensuring the suppression of grating lobes in the radiation pattern which produces good quality images [14, 15].

2.1.2. Accelerated-SAFT acquisition strategy

Here we present a minimum-redundancy technique we have denominated Accelerated-SAFT or, in its short form, kA-SAFT. The k subscript refers to the acceleration factor carried out during the acquisition stage which can go from 2x to Nx depending on the number of

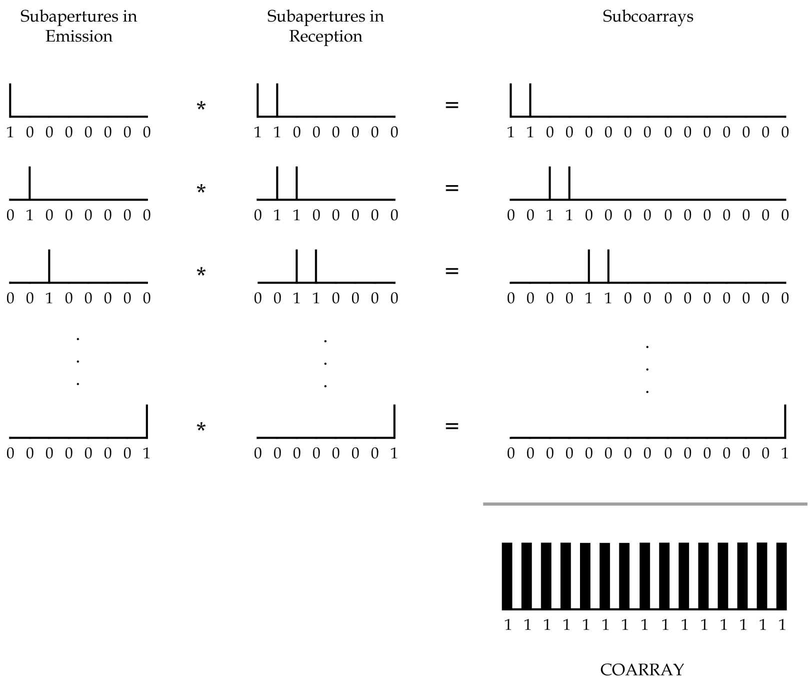


Figure 3. Coarray sequences for 2R-SAFT

channels used for the reception. This strategy increases a little bit the cost involved in the acquisition system respect to 2R-SAFT, but at the same time, reduces the number of shots by k times.

The kA-SAFT uses n_A consecutive elements to receive and a single element to emit which is centred in the active subaperture. As shown in Figure 4, the elements on emission are sequentially activated with a shift of $\frac{n_A}{2}$ elements. At each shot n_A consecutive channels are used as receivers, needing to store n_A signals per emission except for the first and the last array elements where half of the signals is acquired.

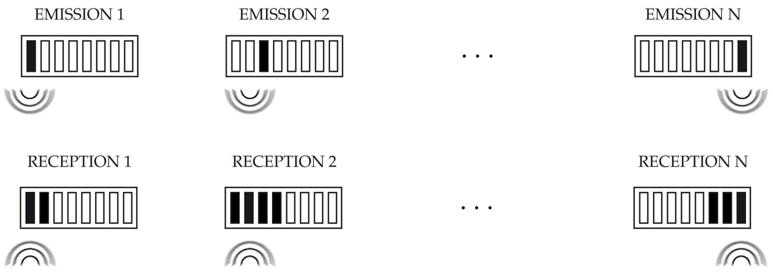


Figure 4. Firing sequences of the elements in kA-SAFT being k = 2x and $n_A = 4$

In this sense, when the i^{th} element is used to emit the elements that are going to use as receivers are given by:

$$Elements_{rx} = \left\{ i - \frac{n_A}{2} + j \right\} \qquad 0 \leq j \leq n_A \tag{3}$$

Figure 5 shows the coarray generated when kA-SAFT is employed for the case of $n_A = 4$. As we can observe, the coarray is identical to that obtained with 2R-SAFT (Figure 3) preserving all its advantages but multiplying by 4 the frame rate in acquisition.

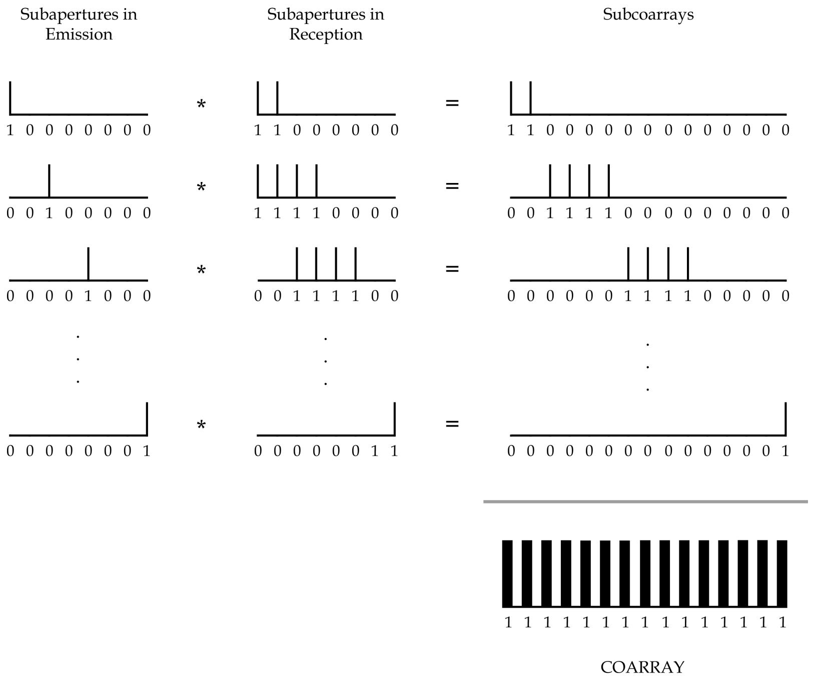


Figure 5. Coarray sequences for kA-SAFT being k = 2x and $n_A = 4$

2.1.3. Experimental results

We present some experimental results that have been done on a tissue phantom (Model 040GSE - CIRS Inc.) with 0.5dB/cm attenuation, where several cysts and wires of 0.1mm diameter are located at different depths (Figure 6). We have used a 2.6MHz phased array transducer with $N = 64$ elements, 0.28mm of pitch (Vermon Inc.) for the measurements. We will use the Total Focusing Method as a reference model to examine the cysts and wires in the tissue covering an area starting from 25mm to 80mm depth, and we will compare it to 2R-SAFT and kA-SAFT techniques. All images have been obtained by applying the DAS algorithm. TFM uses the complete set of signals N^2 = 4096 while 2R-SAFT and kA-SAFT images have been calculated using $2N - 1$ = 127 signals.

In Figure 7, images for all strategies are presented. It is easily observed how Figures 7(a,b,d,e,g) are very similar in terms of quality. Consequently, the strategy to be chosen relies fundamentally on the hardware requisites.

Nevertheless, at a depth greater than 60 mm none reaches the same contrast level as TFM (Figure 7(h)), highlighting the limited signal to noise ratio suffered by all minimum

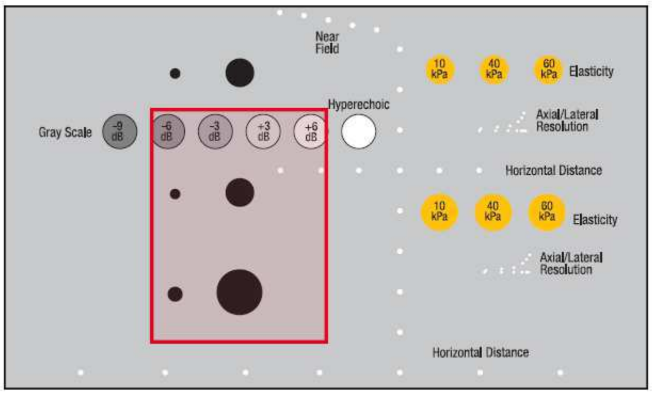


Figure 6. Region of interest analysed from tissue phantom model 040GSE by CIRS Inc.

redundancy techniques. In table 1, a comparison between the number of channels in emission and reception, number of firings, acquisition frame rates and memory buffers needed is performed for the different strategies presented. As we can see, TFM is the technique which more storage as well as more hardware channels needs. By contrast, minimum redundancy techniques requisites are more affordable and suitable for applications where size matters.

Strategy	Channels (tx,rx)	Firings	Framerate	Buffer
2R-SAFT	(1,2)	$N_{firings} = N$	$f_{frame} = \frac{f_{prf}}{N}$	$2N - 1 \times L$
2xA-SAFT ($n_A = 4$)	(1,4)	$N_{firings} = \frac{N}{2}$	$f_{frame} = 2\frac{f_{prf}}{N}$	$2N - 1 \times L$
4xA-SAFT ($n_A = 8$)	(1,8)	$N_{firings} = \frac{N}{4}$	$f_{frame} = 4\frac{f_{prf}}{N}$	
8xA-SAFT ($n_A = 16$)	(1,16)	$N_{firings} = \frac{N}{8}$	$f_{frame} = 8\frac{f_{prf}}{N}$	
16xA-SAFT ($n_A = 32$)	(1,32)	$N_{firings} = \frac{N}{16}$	$f_{frame} = 16\frac{f_{prf}}{N}$	
TFM	(1,N)	$N_{firings} = N$	$f_{frame} = \frac{f_{prf}}{N}$	$N^2 \times L$

Table 1. Comparison of the several acquisition strategies presented

2.2. Golay Codes

As we have seen, synthetic aperture images have low contrast due to the poor signal to noise ratio (SNR). Along this section, we will study how the use of pulse coding based on Golay codes [16, 17] can help to improve the dynamic range and SNR, in order to achieve an image quality comparable to that of Total Focusing Method.

2.2.1. Golay encoding for ultrasonic excitation

Golay complementary pairs have been widely used for transducer excitation because the sum of its auto-correlation function has a main peak and zero side-lobes [16]. A complementary pair is composed of two binary sequences, $A[n] = [a_0, a_1, \ldots, a_{N-1}]$ and $B[n] = [b_0, b_1, \ldots, b_{N-1}]$, of the same length N such that $a_i, b_i \in \{-1, +1\}$.

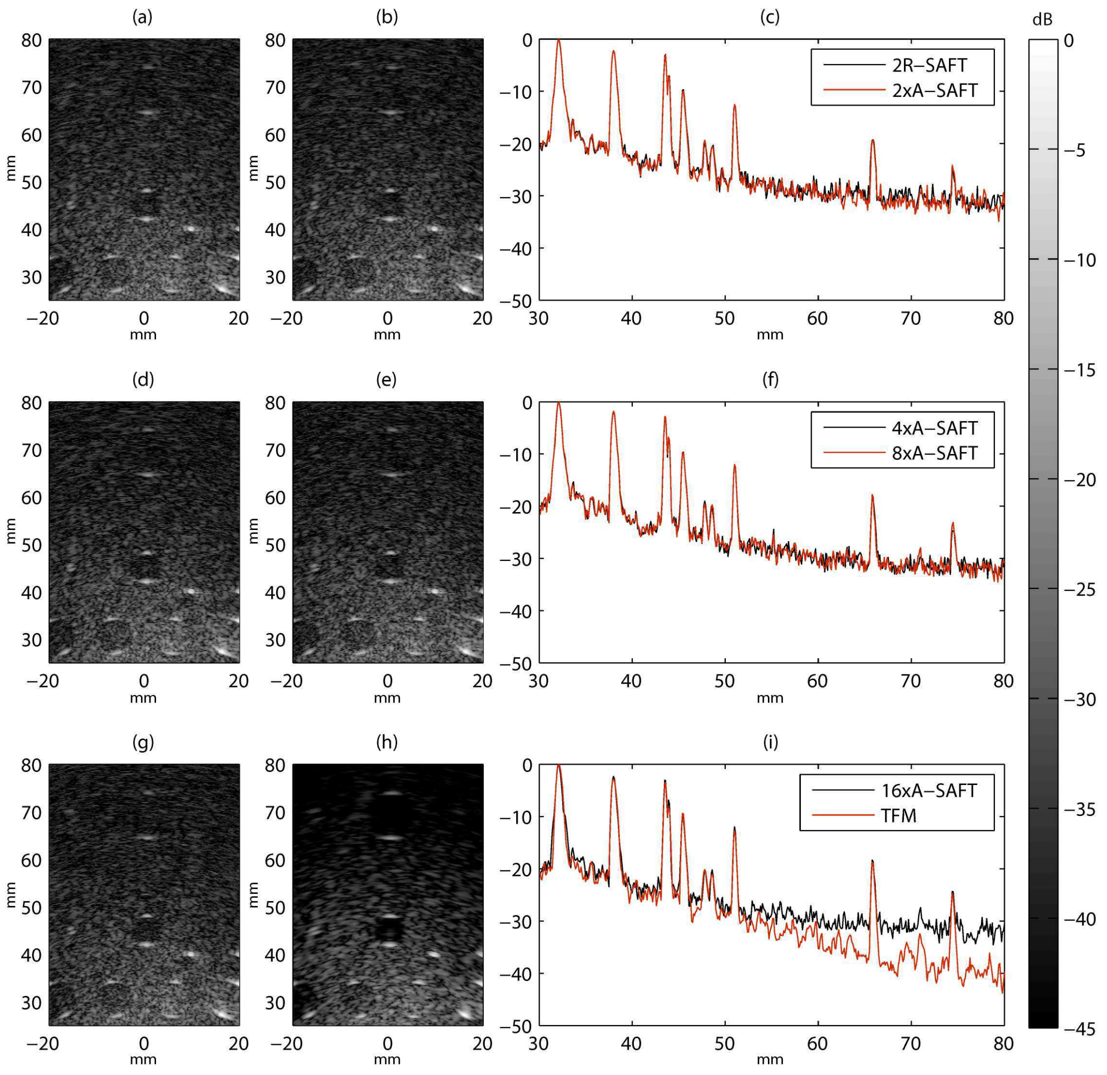


Figure 7. Experimental images from tissue phantom. (a) 2R-SAFT, (b) 2xA-SAFT, (c) Lateral profiles comparison between 2R-SAFT and 2xA-SAFT, (d) 4xA-SAFT, (e) 8xA-SAFT, (f) Lateral profiles comparison between 4xA-SAFT and 8xA-SAFT, (g) 16xA-SAFT, (h) TFM, (i) Lateral profiles comparison between 16xA-SAFT and TFM

The auto-correlation functions of $A[n]$ and $B[n]$ have side lobes with equal magnitude but opposite sign. The sum of these independent auto-correlation functions provides an ideal delta function according to:

$$C_A[n] + C_B[n] = \begin{cases} 0, \, n = 0 \\ 2N, \, otherwise \end{cases} \tag{4}$$

where $C_A[n]$ and $C_B[n]$ are the auto-correlation functions of $A[n]$ and $B[n]$, respectively, for any integer n satisfying the equation 4. The construction of Golay code pairs is done recursively with the *"negate and concatenate"* method, a technique used by Golay [16] to create longer pairs from shorter hand-constructed given pairs. Specifically, if $A[n]$ and $B[n]$ are the N-digit binary representations of a complementary pair of codes, then a new pair of complementary codes $A'[n]$ and $B'[n]$ of length $2N$ can be formed by concatenating $B[n]$ to $A[n]$ and concatenating $\sim B[n]$ to $A[n]$ where $\sim B[n]$ is the complement of $B[n]$. Thus, $A'[n] = A[n] \mid B[n]$, and $B'[n] = A[n] \mid\sim B[n]$.

One of the major drawbacks of Golay codes is that two shots are needed for each emitting element in order to complete both A and B codes respectively. In our work, Golay codes of length equal to 8 bits have been used, being $A[8] = [+1+1+1+1+1-1-1+1]$ and $B[8] = [+1-1+1-1+1+1-1-1]$, producing a gain of 24dB according to equation 4.

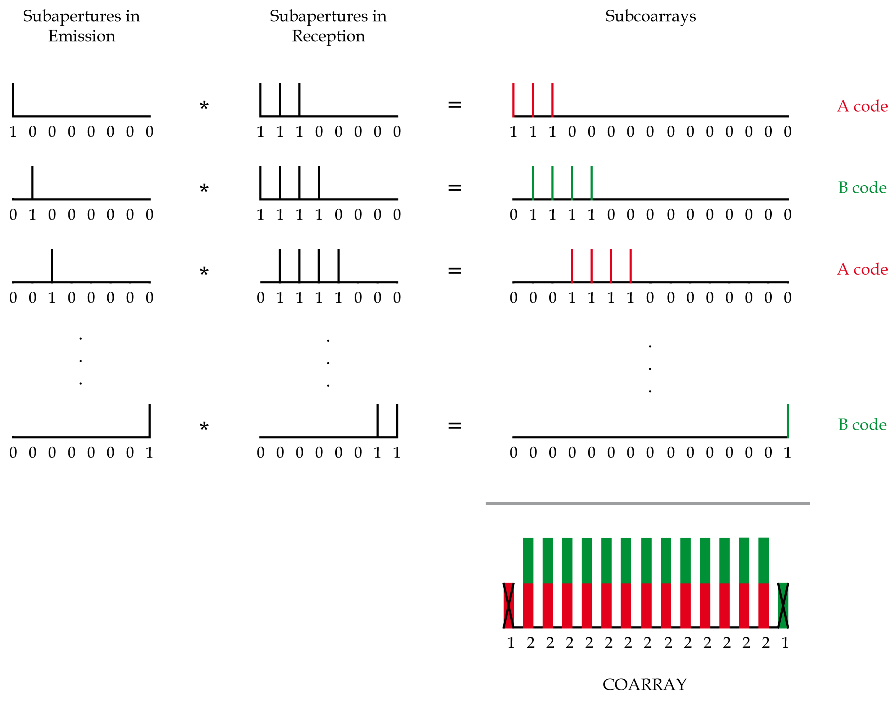


Figure 8. Golay encoding integration example

2.2.2. *Coarray for Golay encoding*

Golay codes, described previously, and minimum redundancy techniques can be combined. In order to illustrate how this can be done, Figure 8 shows an example using a 4R-SAFT (four receivers) [15] plus Golay codes. Here, two signals per coarray element are acquired and because Golay encoding needs to fire twice, A or B codes are alternated between shots.

This process is mathematically identical for the formerly presented strategies 2R-SAFT and kA-SAFT but with some particularities:

1. The number of channels in reception (sensors) must double the original number, in order to have two signals per coarray element for A and B codes.
2. The amount of acquired data signals also doubles the original, because of the first point.
3. The original firing rate is preserved, which means achieving identical performance at the expense of doubling the hardware involved in the reception process.

2.2.3. Experimental results

With the use of Golay codes to image the same area than in previous results, the panorama has changed. As before, TFM image has been composed from the complete set of signals N^2 = 4096, but now 2R-SAFT and kA-SAFT have been calculated using $4N - 2$ = 254 signals. From Figure 7 in section 2.1.3, where the corresponding images with no encoding were analysed, it can be seen how the reduction in the number of signals employed produces a loss of dynamic range respect to TFM method. Thus, with the use of Golay codes in Figure 9 we can observe how the contrast and level of detection have substantially increased. Now, both 2R-SAFT and kA-SAFT techniques distinguish the complete set of defects. Thus, in relation to TFM the number of signals is drastically reduced from N^2 to $4N - 2$, accelerating acquisition and processing velocities and the system's frame rate.

3. Ultrasonic imaging system

3.1. General system's overview

As we have said, our goal is centred in the design of ultrasonic imaging systems based on solutions which require fewer resources and storage capacity than conventional systems. Thus, in Figure 10 is schematically represented our vision of the system, which is composed by three parts:

1. **The array or probe**. It is usually composed by 64, 96, 128 or even more transducers depending on the type of application.
2. **Acquisition subsystem**. The hardware subsystem used for transducer excitation and data acquisition (represented by the box in the center). Nowadays, several electronic manufacturers have in their catalogues electronic boards and systems, which are small and can be easily used for our purposes. For example, National Instruments has 32-channel digitizer module capable of sampling on all channels at 50 MS/s with 12-bit resolution. The module is optimized for ultrasound applications [18]. Additionally, both multiplexer and bipolar programmable pulser are required. Specific architectures depending on the type of acquisition strategy will be studied in the next section.
3. **Image generation subsystem**. It is the software system which can take place in any computational device (PC, laptop, ...) shown on the right side of Figure 10. These processes include the digital signal pre-processing of the received signals and filtering; beamforming of the image, delaying and adding signals according to emission and reception lenses, post-processing the image and its representation to properly show data on the screen. To achieve these tasks, the use of GPU's great power for parallel computing will allow us to quickly and efficiently accelerate the algorithms.

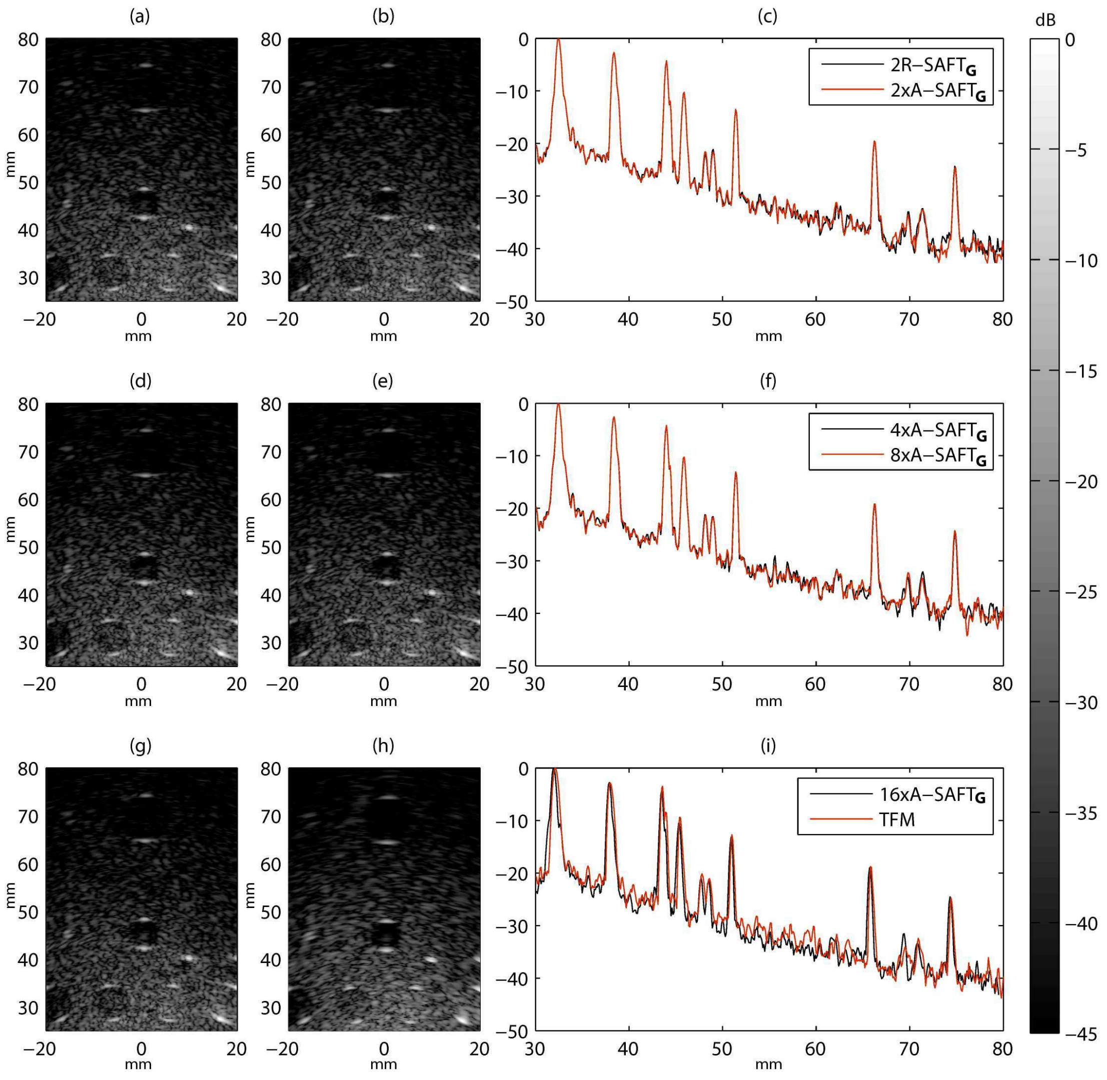


Figure 9. Experimental images from tissue phantom. (a) 2R-SAFT + Golay, (b) 2xA-SAFT + Golay , (c) Lateral profiles comparison between 2R-SAFT + Golay and 2xA-SAFT + Golay, (d) 4xA-SAFT + Golay, (e) 8xA-SAFT + Golay, (f) Lateral profiles comparison between 4xA-SAFT and 8xA-SAFT + Golay , (g) 16xA-SAFT + Golay, (h) TFM , (i) Lateral profiles comparison between 16xA-SAFT and TFM

3.2. Acquisition subsystem

In this section, two acquisition architectures are exposed. On one hand, a minimal system for 2R-SAFT strategy which allows a low-cost and small imaging system and, in the other hand, the architecture which implements 8xA-SAFT plus Golay encoding strategy and uses more hardware but yields better quality images. Which strategy to use depends on the concrete application. Any of these configurations can be carried out using boards systems available in the market.

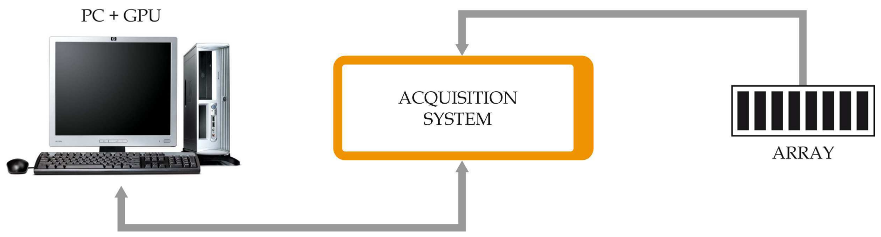


Figure 10. Hardware/Software system proposed

3.2.1. 2R-SAFT architecture

As we study in section 2.1.1, it is basically composed of one channel in emission and two channels in reception. Figure 11 shows the complete architecture for 2R-SAFT implementation. As we can see, a multiplexer is connected to the transmission channel for sequentially activate each element as an emitter, and a second multiplexer will be on charge of connecting the selected elements to both reception channels.

All the acquisition process is managed by a hardware control system which is located in a field-programmable gate array (FPGA). In addition, a local memory is also used to store every received signal. Finally, the signals are transferred to the imaging system using any communication interface (USB, Ethernet, PCI Express). In the imaging system, raw data is stored in a RAM memory of $2N - 1$ signals of capacity to be used for compose and beamform the ultrasonic images using a GPU.

3.2.2. 8xA-SAFT with Golay encoding architecture

As we see in sections 2.1.2 and 2.2, and in order to combine 8xA-SAFT with Golay codes, we will double the number of channels in reception to maintain the number of original firings. Thus, in this case the system is composed of one channel in emission and 32 channels in reception as Figure 12 suggests. A multiplexer connects the transmission channel to elements for sequentially activate one of them, in steps of 8 elements, to transmit an A or B code for odd or even shots respectively. A second multiplexer will be on charge of connecting the 32 reception channels to the receiving aperture ensuring that every coded signal is stored in a local memory. Therefore, two signals per coarray element are overlapped, each one belonging to an A or B code respectively. Additionally, an offset is added to the coarray structure in order to centre its elements, and the boundary coarray elements are removed from it as we illustrated in section 2.2.2.

Now the software imaging system requires a bigger memory and an additional decoding stage, where the complete set of signals is deconvolved, generating a $2N - 8$ data set. Later on, as usual, the data will be beamformed using the graphics processing unit.

3.3. Image generation subsystem: Parallel beamforming

In recent years, computing industry has been opened a way to parallel computing. Nowadays, all consumer computers ship with multi-core processors. Dual-core processors (CPUs) were introduced in personal systems at the beginning of 2006, and it is currently common to find them in laptops as well as 8 and 16-core workstation computers, which

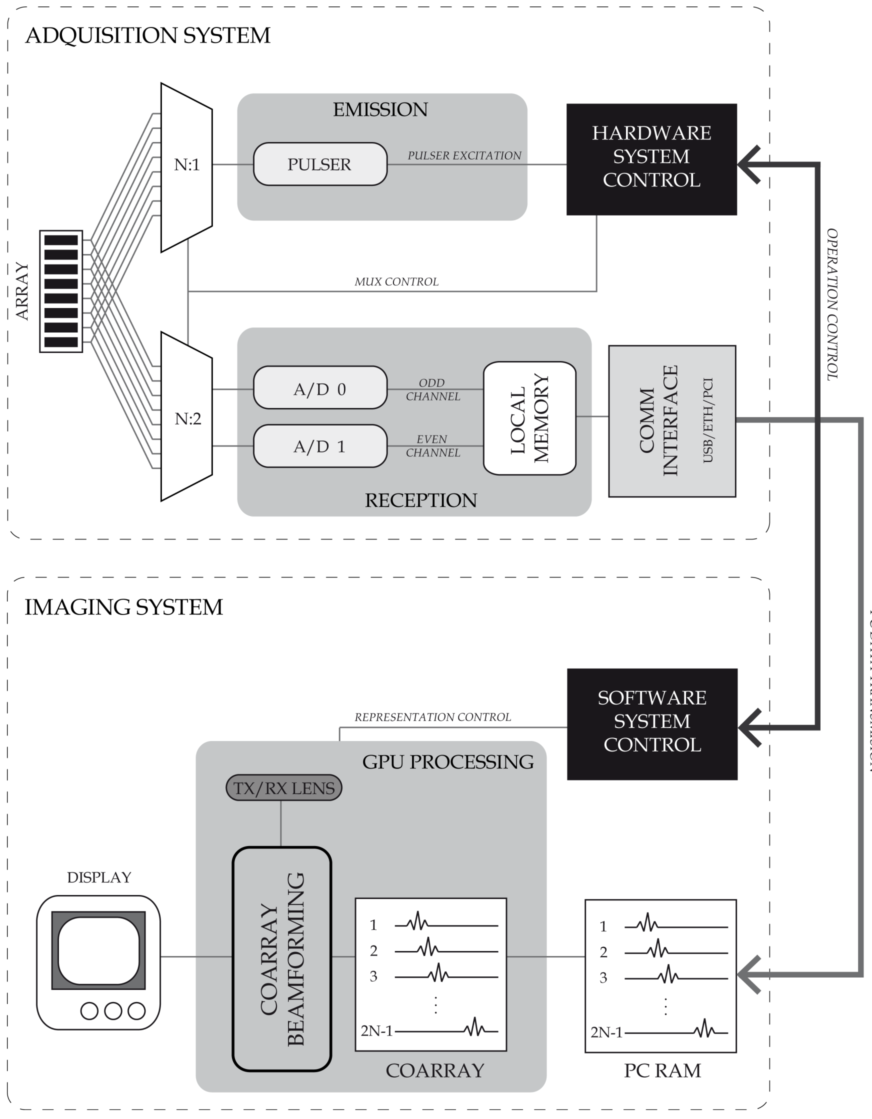


Figure 11. 2R-SAFT Minimal Architecture

means that parallel computing is not relegated to big supercomputers or mainframes computers. However, Graphics Processor Units (GPUs), as their name suggests, came about as accelerators for graphics applications, predominantly those using the OpenGL and DirectX programming interfaces. Although originally they were pure fixed-function devices, the demand for real time and 3D graphics made them evolve into increasingly flexible highly parallel, multithreaded processors with extremely high computational power and very high memory bandwidth converting them into massively parallel machines.

Unlike earlier GPU generations, where computing resources were partitioned into vertex and pixel shaders, nowadays they can be programmed directly in C using CUDA or OpenCL [19], APIs which include a unified shader pipeline, allowing each and every arithmetic logic

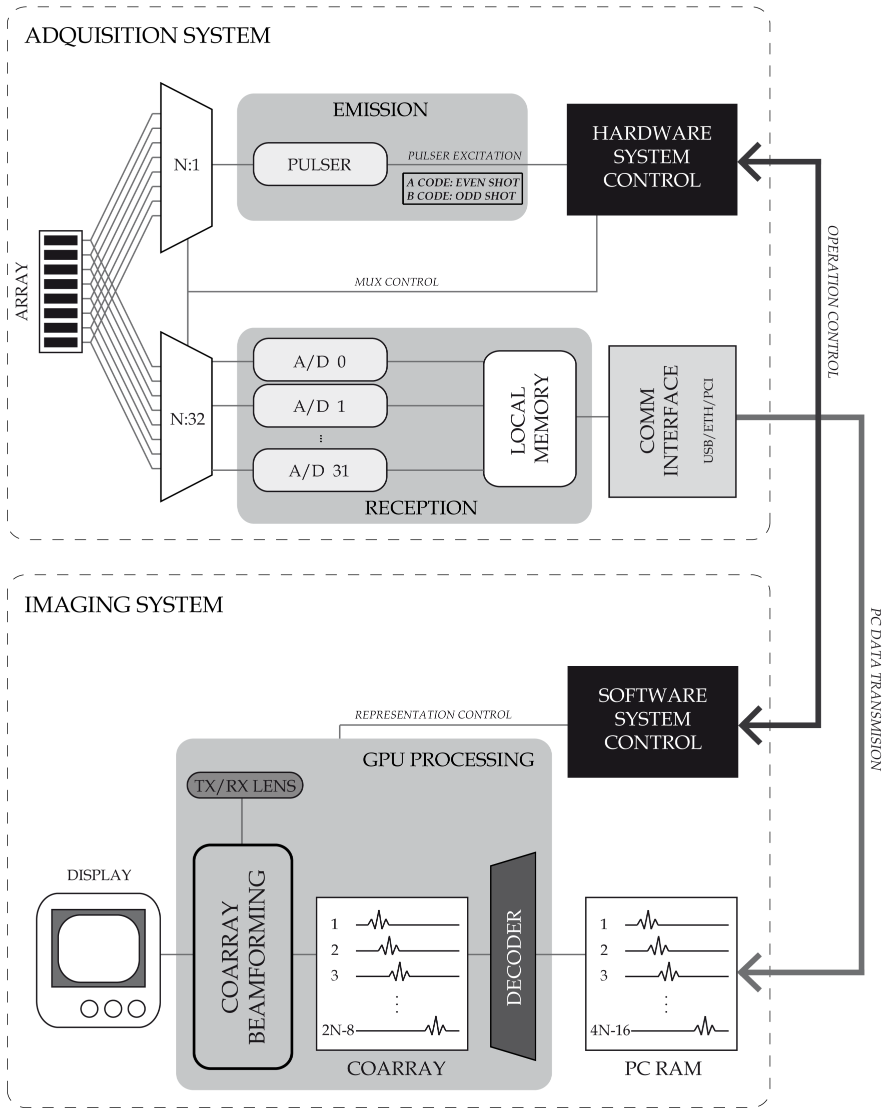


Figure 12. 8xA-SAFT Architecture with 32 channels in reception needed for Golay encoding

unit on the chip to be used by a program intending to perform general-purpose computations (GPGPU). Furthermore, the execution units on the GPU allow arbitrary read and write access to memory as well as access to a software-managed cache known as shared memory. A CUDA program consists of one or more phases that are executed on either the host (CPU) or a device such as a GPU. The phases that exhibit little or no data parallelism are implemented in CPU code. The phases that exhibit rich amount of data parallelism are implemented in the GPU code. The parallel functions (called kernels) typically generate a large number of threads to exploit data parallelism. It is worth noting that CUDA threads are of much lighter weight than the CPU threads. CUDA programmers can assume that these threads take very few cycles to generate and schedule due to efficient hardware support. This differs from

CPU threads which typically require thousands of clock cycles for their generation and their scheduling.

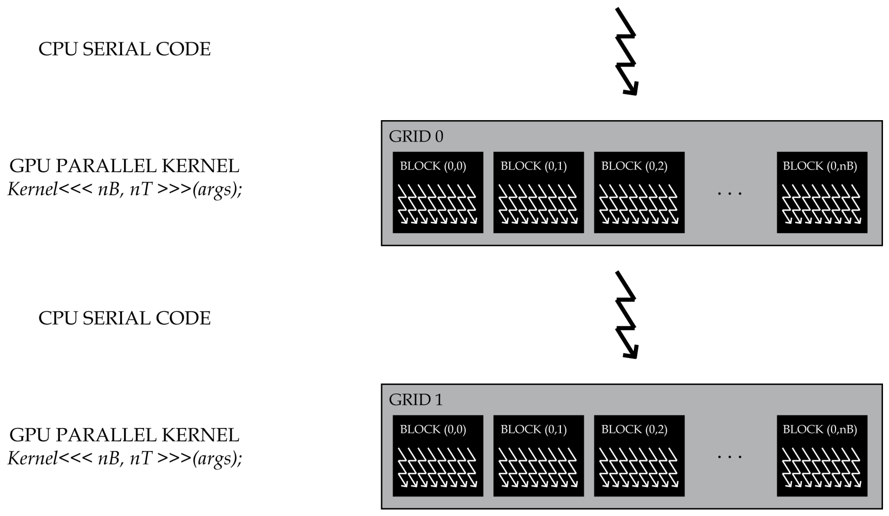


Figure 13. CUDA program execution diagram

The execution of a typical CUDA program is illustrated in Figure 13 where it is observed that the execution starts with host (CPU) execution. When a kernel function is invoked (or launched), the execution is moved to a device (GPU), where a large number of threads are generated to take advantage of huge data parallelism. All the threads generated by a kernel during an invocation are collectively called a grid. Figure 13 shows the execution of two grids of threads. A grid is a 1D, 2D or 3D structure of blocks, and a block is a 1D, 2D or 3D structure of threads. Thus, the program code is composed by classical functions, which run on CPU using only one thread of execution; and kernels, which run on GPU using multiple parallel threads. When all threads of a kernel complete their execution, the corresponding grid terminates, and the execution continues on the host until another kernel is invoked. It is not our purpose to fully cover all the aspects involved in CUDA Architecture. Thus, an extended discussion about the CUDA hardware and programming model is available in multiple sources in the literature [19–21].

Therefore, in this section we will examine different ways to implement the beamforming process on the GPU using the CUDA programming model. From the model, it is extracted that functions which are executed many times independently over different data are the ideal candidates for this kind of computing. In this sense, several algorithms have been implemented to cover the fundamental parts of a conventional Delay-and-Sum Beamformer (DAS) and they have been also evaluated for their performance. This analysis helps to give a better understanding of the GPU architecture and how to write applications for it.

Schematically, Figure 14 show the main stages of a general beamformer. As we can appreciate there are three main operations to be done: pre-processing of signals, beamforming and post-processing. In the software system we propose (Figure 10) all beamforming procedures take place in the GPU.

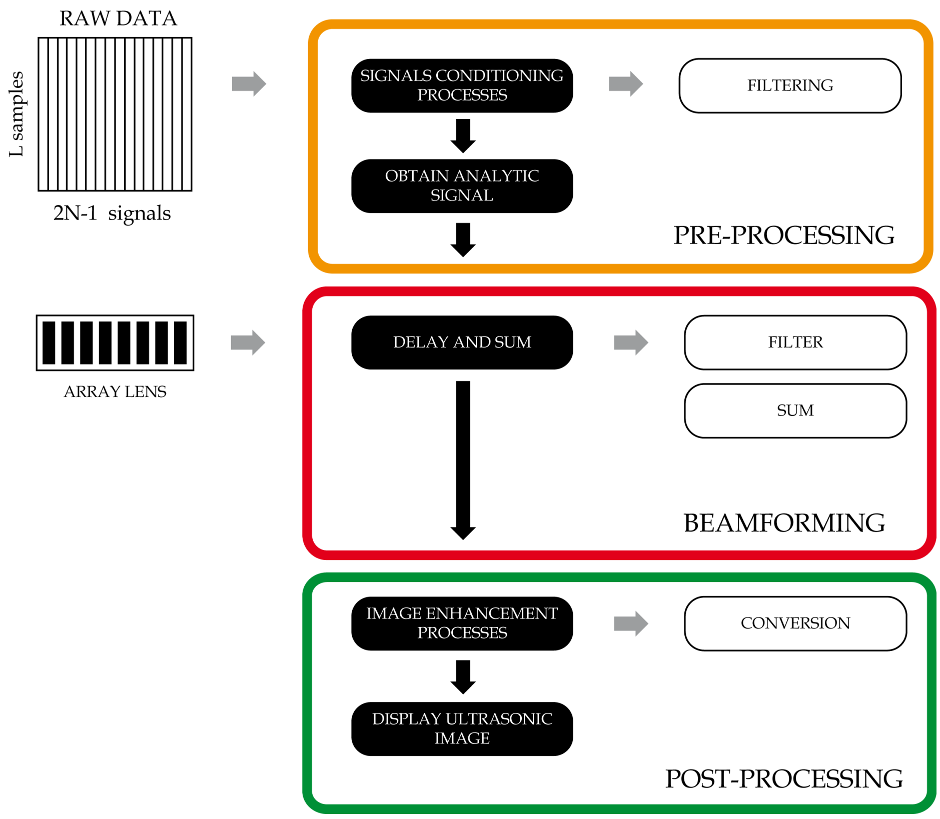


Figure 14. Schematic diagram main parts of a general SAFT beamformer

Implementing the imaging algorithm on GPU systems primarily involves the parallelization of the core algorithm into small independent threads which can be executed by the GPU in runtime. Thus, the imaging process occurs in multiple stages, which follows closely to that has been detailed in Figure 14. Thus, in order to maximize GPUs efficiency and reduce image generation time as much as possible, a specific solution for every different task have been designed. Figure 15 shows how these tasks have been parallelized on the GPU.

The first step consists on copying the complete set of acquired signals from CPU memory to GPU memory. We already know that this transaction is slow, and therefore it is recommended to copy all signals at the same time rather than doing it signal by signal.

3.4. Pre-processing

The pre-processing of the complete set of signals is a fundamental part of the image generation process. Supposing $X_{tx,rx}(t)$ the received signal from any emitter tx and receiver rx pair, a function $H(t)$ is applied to every signal as the following expression suggests:

$$Y_{tx,rx}(t) = X_{tx,rx}(t) \cdot H(t) \tag{5}$$

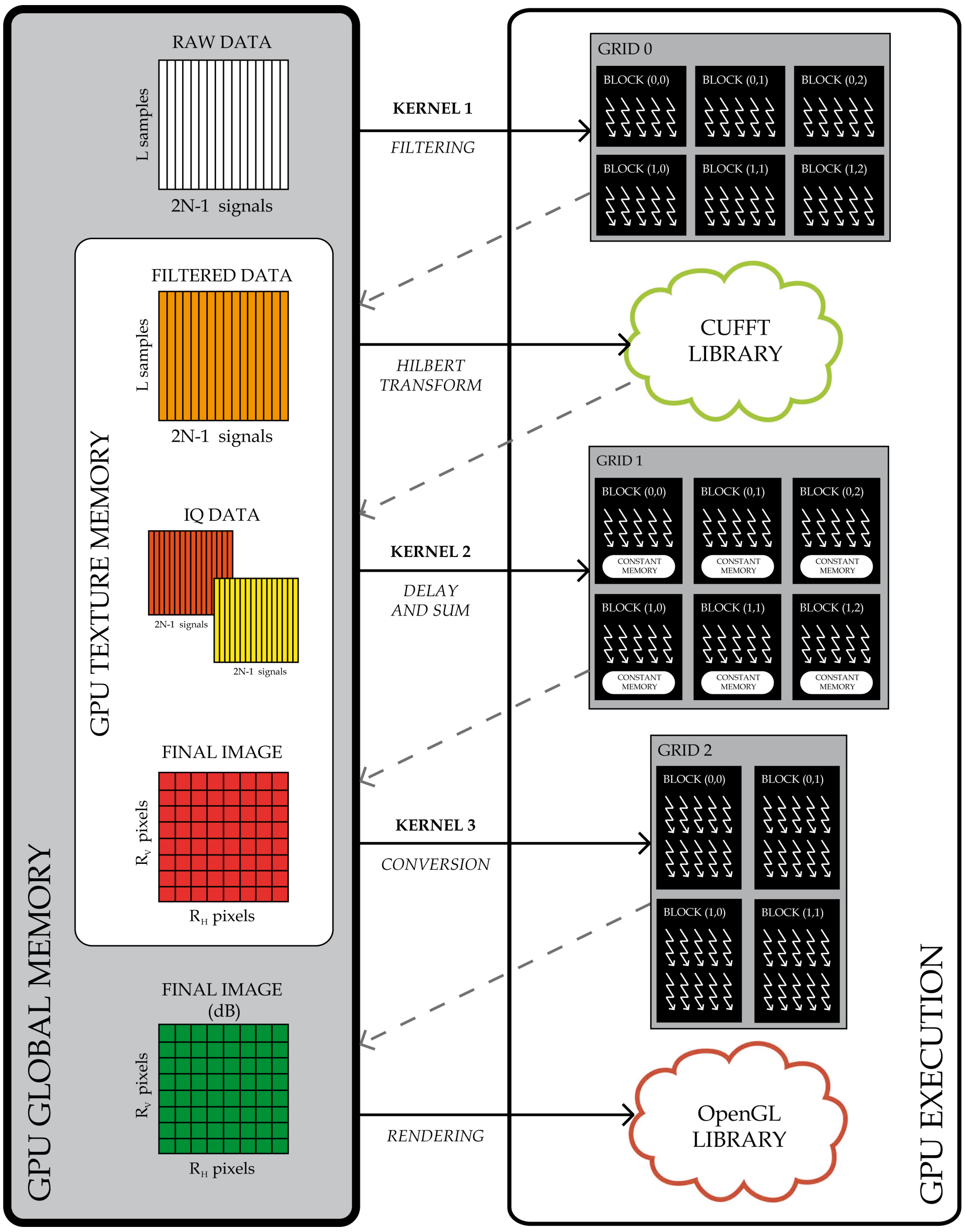


Figure 15. System beamforming loop parallelized on GPU for SAFT implementation

and

$$H(t) = H_F(t) \cdot H_{IQ}(t) \tag{6}$$

where $H_F(t)$ is a signal conditioning process where a filter is applied in order to remove the offset level introduced during the acquisition system and to reduce the noise.

Additionally and for convenience, the acquired signals can be decomposed into their analytic signals form [22] (in-phase I and quadrature components Q) . Thereby, the second function $H_{IQ}(t)$ is the Hilbert Transform in order to reduce errors and artefacts which appear at the envelope detection stage. Then, the signals $X_{tx,rx}(t)$ can now be expressed as:

$$X_{tx,rx}(t) = I_{tx,rx}(t) + jQ_{tx,rx}(t) = \mathbf{X}_{tx,rx}(t)e^{j\phi_{tx,rx}(t)} \tag{7}$$

where $\mathbf{X}_{tx,rx}(t)$ is the modulus and $\phi_{tx,rx}(t)$ its corresponding signal phase.

3.4.1. Parallel implementation

In order to carry out a parallel implementation of these operations, the proposed parallelism strategy lies in a signal-oriented parallelization. This means that a GPU computational thread will be associated to each stored signal sample. Thus, considering signals with L samples, the computational grid of the kernel will be formed as shown in Figure 15 being the number of blocks in x-dimension $BX = \lceil \frac{L}{T_{BX}} \rceil$ and the number of blocks in y-dimension $BY = 2N - 1$. As we know, the number of threads per block T_{BX} is an empirical value and the designer should evaluate what is the best according to the GPU resources. Typical values are 32, 64 or 128 threads per block, generally any power of two, and attending to our tests we have chosen 256 as the optimal value.

There is no limitation on filter length because its coefficients are stored in texture memory, which resides in the device memory and is cached in texture cache to optimize read accesses. Thus, each thread reads from memory the filter coefficients and L samples of a signal, convolving them to obtain a filtered sample.

Later on, the Hilbert Transform is applied to every filtered signal so we can obtain their analytic signals. In this case, FFT algorithms provided by CUDA (CUFFT libraries [20]) are used to compute the IFFT of the product of the corresponding signal and the Hilbert Transformer FFTs, as it is defined in [23]. With these libraries, there is no need to define a new kernel nor specify grid and block dimensions, since they are responsible for properly parallelizing and splitting the algorithm, computing the FFT of the data set directly on the GPU. In our particular case, a total of $2N - 1$ FFTs of L points are calculated in parallel. The whole resultant I/Q (in-phase/quadrature-phase) signals pairs (Figure 15) are stored in texture memory, and they are passed to the next stage via global device memory.

3.4.2. Optional stage: Decoding

When Golay encoding is used during acquisition, it is necessary to first merge and deconvolve the $4N - 16$ received signals, where 50% of signals belong to A and B codes respectively. This can be done very fast making a parallel implementation where the parallelism strategy is also signal-oriented. In this sense, the previous kernel can be modified in order to include the sum of both signals in parallel before the application of the filters coefficients to finally obtain $2N - 8$ signals.

3.5. Beamforming: Delay and Sum

All time-domain imaging algorithms are based on the principle of delay-and-sum beamforming. Typically, these algorithms emulate an acoustic lens by applying appropriate time delays to the array elements in order to focus or steer the beam as desired. SAFT beamformers focus the beam at every point in the image, giving better defect detectability as we mentioned [2, 4, 5, 7]. DAS beamforming is not difficult to implement and permits the use of arbitrary array geometries what makes suitable for a wide range of applications.

According to the Hilbert transformation of the first step, two processing streams have been created where two parallel images will be calculated following these equations:

$$A_I(x,z) = \sum_{i=1}^{N}\sum_{k=1}^{N} I_{tx_i,rx_k}(D(x,z)) \tag{8}$$

$$A_Q(x,z) = \sum_{i=1}^{N}\sum_{k=1}^{N} Q_{tx_i,rx_k}(D(x,z)) \tag{9}$$

where $A_I(x,z)$ and $A_Q(x,z)$ are the in-phase and quadrature images respectively, and $D(x,z)$ is the focussing delay for the spatial point (x_p, z_p) in the grid which is calculated as follows:

$$D(x,z) = \frac{\sqrt{(x_p - x_{tx})^2 + z_p^2} + \sqrt{(x_p - x_{rx})^2 + z_p^2}}{c} \tag{10}$$

being x_{tx} and x_{rx} the coordinates of the transducer elements tx and rx, respectively.

Henceforth, we will focus on the all the operations involved in Delay-and-Sum algorithm, studying the diverse alternatives and their parallel implementation as well as the best way of their optimization.

- **Lens calculation**. A fundamental part of beamforming is calculating the differences in wave arrival time between array elements. Therefore, each signal sample has to be properly delayed according to the distance from the spatial point to the emitter or receiver array elements. The calculation of delays is achieved using equation 10. Although in a conceptual form is a delay, what is actually done is a mapping to the memory buffer (at the sampling frequency) where the corresponding sample value of the signal is retrieved. Therefore, the number of delays to be calculated is usually large and it is given by:

$$Memory|_{lens} = R_H \times R_V \times 2N - 1 \tag{11}$$

where $R_H \times R_V$ are the dimensions of the desired ultrasonic image. Thus, the lens calculation can be afforded using two different approaches:

- *Load pre-calculated delays*. The delays are pre-calculated before beamforming and they are recovered from a look-up table inside the image generation process. The necessary memory to store all the delays is not a significant problem, but the main drawback is the requirement of high bandwidth to make the process faster as well as the fact of updating the table each time. Thereby, this would be a good solution for no in-vivo inspections, where the scenario is known and the delays are calculated only once for the complete acquisition.
- *Calculate delays on-the-fly*. The delays are dynamically calculated inside the beamforming process. This task, which can be at first computationally more expensive than the first alternative, is however not a heavy computational problem because of the great power of actual systems. In this regard, dynamic calculation of the lenses inside the threads will simplify other operations on images, such as scrolling and zooming.

Which approach to choose relies on the rest of the beamformer implementation. Thus, in order to take full advantage of the GPU it is needed to have a balance between bandwidth use and arithmetic operations. In this regard, it has been proved that it is faster to obtain the values for the lenses inside the kernels instead of having them stored in the device memory. Therefore in our proposal, it makes sense to calculate the delays on-the-fly.

- **Filtering**. In a real implementation, we sample the elements at a rate just above the Nyquist criteria. Although this preserves the frequency content of the signal, this does not give enough steering delay resolution. The solution is to perform a digital interpolation, increasing the steering-delay resolution. In this particular case, linear interpolation and polynomial interpolation can be easily implemented. The results obtained are practically identical, although the cost associated to each solution differs being the polynomial interpolation time the double of linear interpolation. For this reason, we decided to simply interpolate across two consecutive samples. The penultimate operation is the application of a window function which is multiplied with the data from each channel in order to reduce mainly the level of sidelobes.
- **Sum**. The final step in the ultrasonic generation process is to obtain the accumulated sum of all the signals samples which contribute to a given spatial point.

3.5.1. Parallel implementation

The delay-and-sum process is applied to the complex signals obtained in the previous stage. We have identified different strategies to implement the ultrasonic image generation process in a GPU depending on how the algorithm is parallelized with respect to threads and blocks and relative to the use of GPU resources.

As Figure 16 shows, the parallelization is carried out by launching a thread per image pixel. To this end, a computation grid ($GRID_1$) with $BX = \lceil \frac{R_H}{T_{BX}} \rceil$ and $BY = \lceil \frac{R_V}{T_{BY}} \rceil$ blocks of $T_{BX} \times T_{BY}$ threads is defined on the kernel, where R_H and R_V are the desired image resolution in horizontal and vertical directions, respectively. Each thread is responsible then for calculating the coordinates for the spatial point (x_p, z_p) of a specific image pixel and calculating the lens to focus at this point.

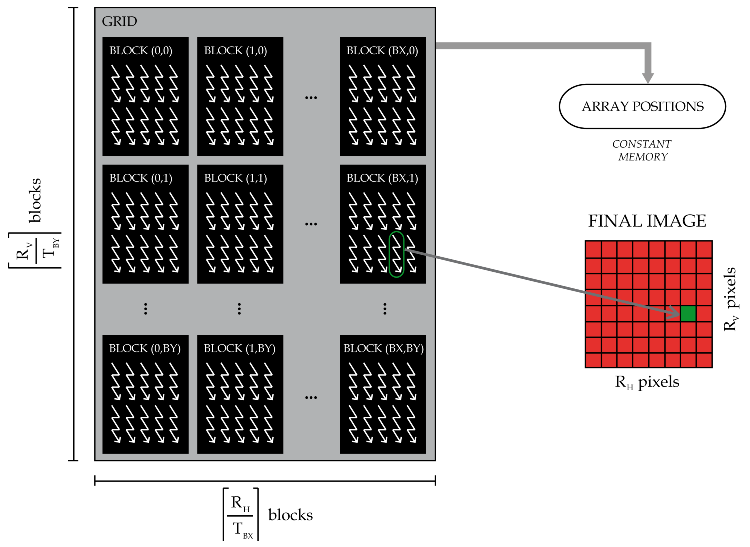


Figure 16. One thread is responsible for a image pixel

In this case, the lens is formed by the $2N - 1$ times of flight of each emission-reception pair combination. Thus, in order to accelerate all these calculations, the transducer elements coordinates are stored in constant memory in each GPU multiprocessor. In addition, the computed distances from an array element to an image pixel are reused to save time avoiding duplicate calculations. The lens obtained allow us to index in the complex signals stored in texture memory, and real and imaginary parts are interpolated when needed. To this respect, lineal interpolation was implemented obtaining good performance. Then, the $2N - 1$ resultant complex samples are multiplied by the corresponding apodization gains and added together. Finally, the resultant image (final image in Figure 15) is also stored in texture memory, for a quick data access to the post-processing stage.

3.6. Post-processing

The post-processing stage involves firstly calculate the envelope (in essence the modulus) of the beamformed images, according to the following expression:

$$A = \sqrt{{A_I}^2 + {A_Q}^2} \tag{12}$$

where A_I and A_Q are the *In-phase* and *Quadrature* images derived from the beamforming process. This operation prevents the appearance of diverse artefacts associated with the Hilbert Transform.

Likewise, (an optional) stage in the process is in charge of normalizing and converting the image to decibels scale. Although this is not a complex task, it cannot be carried out in the previous stage because we need to know what the maximum value for the image:

$$A|_{decibels} = 20\log_{10}\left(\frac{A}{max(A)}\right) \tag{13}$$

Finally, the generated ultrasonic image (Final image (dB) referenced in Figure 15) is directly displayed on the screen using the OpenGL libraries, which provide specific functionality for graphics representation.

3.6.1. Parallel implementation

The parallel implementation of the envelope calculation is carried out inside the beamforming kernel. This is because at the end of the pixel calculation, we have the final output values for both I and Q components. Thus, we avoid writing twice and we only obtain a single image. For the optional conversion to decibels scale, a new kernel (Kernel 3 in Figure 15) is defined which uses a grid with $\lceil \frac{R_H}{B_x} \rceil$ and $\lceil \frac{R_V}{B_y} \rceil$ blocks of T_{BX} threads having a thread per image pixel as before.

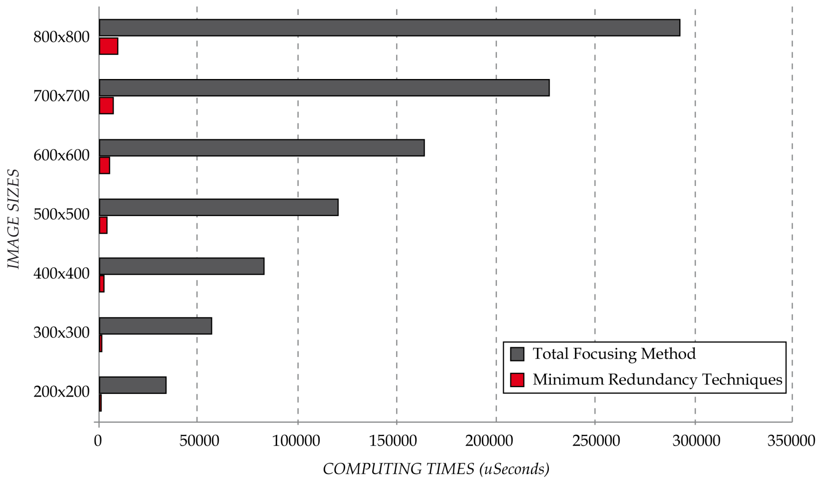


Figure 17. Computing times for TFM and MR solutions using GPU in μseconds

3.7. Performance

A NVIDIA Quadro 4000 graphics card was used to test the beamforming time achieved with the system proposed here. This card has 256 cores and 1GB global memory. It was installed in a computer with a four-core 2.66GHz Intel Q9450 processor and 4GB RAM. GPU-based implementation of the beamformer was done and tested for all acquisition strategies exposed along this chapter. In Figure 17 computing times considering image sizes starting from 200×200 to big size 800×800 for both TFM and minimum redundancy solutions are presented where it is evident than despite using the great power of GPU's the TFM solution is a very intensive procedure.

In Figure 18, the frame rate obtained for different image sizes when 2R-SAFT and kA-SAFT are employed is presented. In particular, attending to the case of an image with 500×500

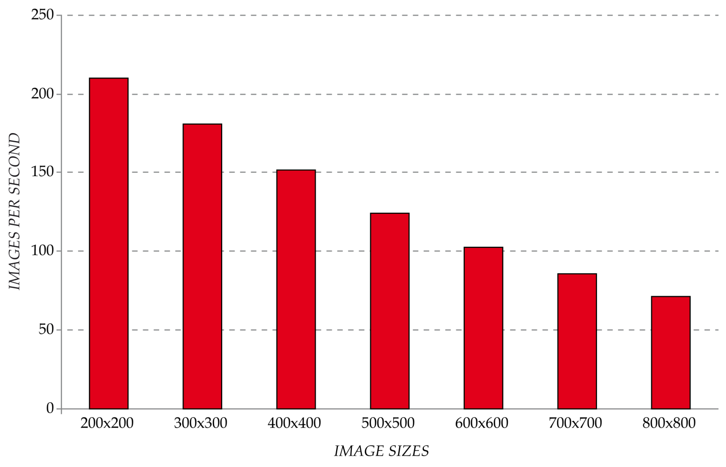


Figure 18. Images per second achieved using GPU for different image sizes for 2R-SAFT and kA-SAFT

pixels, the GPU is able to get 135 images per second, which is in nearly to the acquisition system's rate. The evidence here is that we are using a smaller dataset than that obtained with TFM method but preserving the image quality with all GPU cores completely dedicated to fast computation.

4. Conclusion and future developments

This work has presented how the use of coarray paradigm makes possible the design of ultrasonic imaging systems with reduced hardware requirements. The system is divided into two subsystems, hardware and software respectively. The first one is focused on the development of the data acquisition system, whose design is done analysing the compromise between parallel electronic resources and acquisition time. The second one exploits GPU technology to implement the beamformer via software, compensating the emission and reception distances to each image point, providing the maximum possible quality at each image pixel.

Two solutions, based on the availability of instrumentation in the market, are presented attending to this design following the minimum redundancy coarray model. In one case, it has been emphasized the miniaturization of the hardware (with only two channels in reception), and in the second case the focus has been the reduction of the acquisition time at the expense of increasing and parallelize reception channels (up to a maximum of 32). From the point of view of image quality, both beamforming techniques present similar results. Consequently it is possible to adapt the design of our system to several implementation models depending on the final application requirements.

The problems associated to the low level of the transmitted signals and the signal losses through the material have been analysed. As a solution, we have introduced pulse compression techniques in order to increase the signal to noise ratio. In addition, we have studied the implementation cost of this technique and it has been compared with the TFM technique (based on the FMA capture), verifying that the results are very similar.

Finally, we have made a detailed description of the beamforming process in GPU and it has been quantified the advantage of using the GPU as a processing tool from the image frame rate point of view. So by using a simple graphics card equipped with NVIDIA CUDA technology, rates that go up to 200 images per second were obtained depending on the image size chosen. Therefore, this solution allows the development of high quality imaging systems with low requirements and excellent capabilities in a compact architecture.

Acknowledgments

This work has been supported by the Spanish Ministry of Science and Competitiveness under the project DPI2010-19376.

Author details

D. Romero-Laorden, J. Villazón-Terrazas,
O. Martínez-Graullera and A. Ibáñez

Centro de Acústica Aplicada y Evaluación No Destructiva (CSIC), Madrid, Spain

References

[1] Caroline Holmes, Bruce W. Drinkwater, and Paul D. Wilcox. Post-processing of the full matrix of ultrasonic transmit–receive array data for non-destructive evaluation. *NDT & E International*, 38(8):701–711, December 2005.

[2] Caroline Holmes, Bruce W. Drinkwater, and Paul D. Wilcox. Advanced post-processing for scanned ultrasonic arrays: application to defect detection and classification in non-destructive evaluation. *Ultrasonics*, 48(6-7):636–42, November 2008.

[3] Alan J. Hunter, Bruce W. Drinkwater, and Paul D. Wilcox. The wavenumber algorithm for full-matrix imaging using an ultrasonic array. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 55(11):2450–62, November 2008.

[4] Mustafa Karaman, Pai-Chi Li, and Matthew O'Donnell. Synthetic aperture imaging for small scale systems. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 42(3):429–442, May 1995.

[5] Jørgen Arendt Jensen, Svetoslav Ivanov Nikolov, Kim Lø kke Gammelmark, and Morten Hø gholm Pedersen. Synthetic aperture ultrasound imaging. *Ultrasonics*, 44:5–15, December 2006.

[6] Carlos J. Martín-Arguedas, O. Martínez-Graullera, G. Godoy, and L. Gómez-Ullate. Coarray synthesis based on polynomial decomposition. *IEEE transactions on image processing*, 19(4):1102–1107, 2010.

[7] Carlos J. Martín-Arguedas, D. Romero-Laorden, O. Martínez-Graullera, M. Pérez-Lopez, and L. Gómez-Ullate. An Ultrasonic Imaging System Based on a New SAFT Approach and a GPU Beamformer. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 59(7):1402–1412, 2012.

[8] Geoffrey R Lockwood, Pai-chi Li, Matthew O'Donnell, and F. Stuart Foster. Optimizing the Radiation Pattern of Sparse Periodic Linear Arrays. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 43(1):7–14, 1996.

[9] Svetoslav Ivanov Nikolov. *Synthetic aperture tissue and flow ultrasound imaging*. PhD thesis, Technical University of Denmark, 2001.

[10] Bernard D. Steinberg. *Principles of Aperture and Array System Design*. Wiley, New York, 1976.

[11] John J. Flaherty, Kenneth R. Erikson, and Van Metre Lund. Synthetic aperture ultrasonic imaging systems, Patent number 3548642, 1967.

[12] Christoph B. Burckhardt, Pierre-André Grandchamp, and Heinz Hoffman. An Experimental 2 MHz Synthetic Aperture Sonar System Intended for Medical Use. *IEEE Transactions on Sonics and Ultrasonics*, 21(1):1–6, 1974.

[13] Intel Corporation. Thunderbolt™ technology. http://www.intel.com/thunderbolt, (accessed 6 July 2012).

[14] Carlos J. Martín-Arguedas, O. Martínez-Graullera, and L. G. Reduction of grating lobes in SAFT Images. In *IEEE International Ultrasonics Symposium*, number 1, pages 721–724, Beijing, China, 2008.

[15] Carlos J. Martín-Arguedas. *Técnicas de apertura sintética para la generación de imagen ultrasónica*. PhD thesis, Universidad de Alcalá, 2010.

[16] M. Golay. Golay's complementary series. *IRE Transactions on Information Theory*, pages 273–276, 1961.

[17] Andrzej Nowicki, Igor Trots, Wojciech Secomski, and Jerzy Litniewski. Golay's codes sequences in ultrasonography. *Archives of Acoustics*, 28:313–324, 2003.

[18] National Instruments. 32-channel digitizer module for ultrasound applications. http://sine.ni.com/nips/cds/view/p/lang/en/nid/208657, (accessed 1 October 2012).

[19] Wen-Mei W. Hwu and David B. Kirk. *Programming Massively Parallel Processors : A Hands-on Approach*. Morgan Kaufmann, 2010.

[20] NVIDIA Developer Zone. Software development kit 4.2 version, https://developer.nvidia.com/cuda-education-training, (accessed 1 October 2012).

[21] Jason Sanders and Edward Kandrot. *CUDA by Example*. Addison-Wesley, 2010.

[22] A. V. Oppenheim and W. R. Schafer. *Discrete-Time Signal Processing*. Prentice-Hall, Englewoods Cliffs (NJ), 1989.

[23] S. Lawrence Marple. Computing the Discrete-Time "Analytic" Signal via FFT. *IEEE Transactions on Signal Processing*, 47(9):2600–2603, 1999.