A Unified Approach to Improve the Computation Time and Image Quality of Ultrasound Strain Imaging

A dissertation submitted in partial fulfillment of requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering

Islamic University of Technology

Organization of Islamic Cooperation

Submitted by

Rishad Arfin

Rifat Ahmed

Monir Hossan

Under the supervision of

Dr. Kazi Khairul Islam

Professor

Department of Electrical and Electronic Engineering

Islamic University of Technology, Bangladesh

A Unified Approach to Improve the Computation Time and Image Quality of Ultrasound Strain Imaging

Submitted by

Rishad Arfin (Student ID 082402)

Rifat Ahmed (Student ID 082414)

Monir Hossan (Student ID 082436)

A dissertation submitted in partial fulfillment of requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering

Islamic University of Technology

October, 2012

A Unified Approach to Improve the Computation Time and Image Quality of Ultrasound Strain Imaging

Approved by

Dr. Kazi Khairul Islam

Professor

Department of Electrical and Electronic Engineering

Islamic University of Technology, Bangladesh

Project Members

1. Rishad Arfin
   Student ID 082402

   ……..
2. Rifat Ahmed
   Student ID 082414

   ……..
3. Monir Hossan
   Student ID 082436

   ……..

Summary

Medical science relies on technology now more than ever. For both diagnosis and treatment blessing of technology is exploited. Diagnosis of disease using technology comes in various forms. Imaging is a particularly attractive diagnosis methodology. Among the existing diagnostic imaging techniques ultrasound has gained much popularity over the years.

Apart from the regular ultrasound imaging techniques ultrasound has some special use. Elastography is such an application of ultrasound. It is actually the imaging of the stiffness of the tissue. Elastography is particularly efficient in diagnosing early cancer particularly breast and prostate cancer. For example, breast cancer is diagnosed by invasive biopsy. But elastography can reduce the rate of biopsy since a large fraction of the tumors of breast are found to be benign. Unlike biopsy elastography provides instant result. It relieves the patient from psychological stress to wait for the result. But elastography is much of a qualitative technique. It can be unreliable at some cases. That's why extensive research is being performed to improve its quality.

In our thesis we have worked on two different aspects of the research. We have presented a software tool that can be used for improving the computation time in elastography. Also, we have proposed a concept of using smooth window functions to improve the image quality and presented some experimental results.

Keywords: elastography, computation time, image quality, smooth window, experimental results.

Acknowledgments:

We are grateful to Dr. Kazi Khairul Islam for his direction and support for this thesis.

We are also grateful to Dr. Sheikh Kaiser Alam for his wonderful support with his ideas, vast experience and technical resources. As a part of our undergraduate thesis we started this work with no prior experience of ultrasound and elastography. It would never have been possible to learn about the ultrasound, elastography and then perform an effective research on elastography without Dr. Alam's guidance.

Also, we would like to thank Dr. Kamrul Hasan from BUET for his advice. We also thank Arafat Hussain for helping us when we were facing problems with our thesis.

Finally, we are grateful to our friends at IUT and our families for their support.

Contents

List of Figures:

List of Tables:

Glossary

FFT	Fast Fourier Transform
SNRe	elastographic signal to noise ratio
CNRe	elastographic signal to noise ratio
In-vivo	tissue in a living body
DASE	direct average strain estimation
GBASE	gradient based average strain estimation
CPU	central processing unit
GPU	graphics processing unit
DCS	distributed computing server
parfor	MATLAB parallel for loop
spmd	single program multiple data
PSNR	peak signal to noise ratio
MSSIM	mean structural similarity
FEM	Finite Element Modelling

Chapter 1

Introduction

1.1 Background

Ultrasound strain imaging [1] is becoming more and more popular for qualitative analysis of mechanical properties of tissues. A lot of algorithms have been developed to obtain the strain profile throughout a tissue cross-section when a small deformation is being applied. The image quality has become reasonable over the years. Also, with the development of sophisticated hardware and algorithm it is possible to obtain such images at real time frame rate. This work is intended for further improving the image quality of strain images and improving the computation time of the existing algorithms to facilitate real time imaging further.

Before going into the details, this chapter is presented with the basic introduction of ultrasound, elastography, basic elastographic procedure and different elastographic algorithms.

1.2 Ultrasound imaging

The term 'Ultrasound' relates to frequency or wavelength. There are three region of frequency range-'subsonic', 'sonic' and 'ultrasonic'. The acoustic wave that fall in the frequency range of human ear (20Hz to 20 kHz) is called sonic. the waves having a frequency range less than 20Hz are subsonic and importantly the waves having frequency range greater than 20kHz are regarded as 'ultrasonic' or 'ultrasound'. Here we will use ultrasound and ultrasonic interchangeably.

Ultrasound is used in the clinical applications extensively these days. It is noninvasive, portable and more over the cost of clinical treatments with ultrasound technology is less expensive and affordable. It is not only possible to visualize the anatomy or morphology with ultrasound imaging but can also measure or predict the almost all kind of function by means of blood. Nowadays it is extensively used in fetal imaging, carding imaging, breast cancer detection, and detection of benign and malignant tissue in the human body. Though the ultrasound frequency range starts from 20 kHz, in clinical applications we use typically a range from 1MHz to 15MHz. The typical velocity of ultrasound in the human tissue is 1540 m/s.

The ultrasound is generated by a piezoelectric crystal. This piezoelectric crystal is embedded in the transducer which acts both as transmitter and receiver in the ultrasound imaging process. The crystal deform under the influence of an electric field and vice versa.

Figure 1.1: Ultrasound beams are coming out sequentially from the Ultrasound transducer. Image is reproduced from [2].

Ultrasound is longitudinal mechanical wave. In case of longitudinal wave the displacement of the particles in the medium is parallel to the motion of the wave. As the ultrasound is generated from the piezoelectric crystal, a mechanical wave is generated from the application of an electrical signal (electric field). The ultrasound that is produced from the transducer passes through the tissues and tissue boundaries. As a result reflection, refraction, diffraction of the acoustic wave occurs in the tissue boundaries. At the boundaries of the tissues, the energy of the wave is slightly reflected back to the transducer and the rest of the energy is transmitted through the tissue boundary. The reflected waves are detected by the transducer and the time between the signal transmission and reception in the transducer is calculated to reveal the position and the depth of the tissue boundaries. This process is repeated again and again until the entire image is found. In this way the entire image can be formed within a fraction of second leading to real time imaging [2].

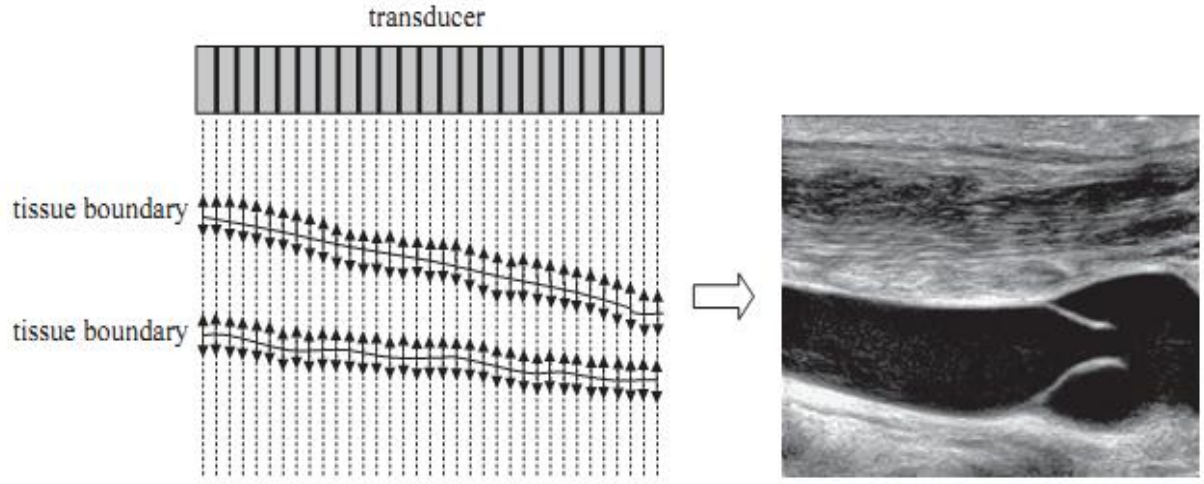


Figure 1.2: In the first figure one on the left, the ultrasound wave is going through the tissue, while some portion of the wave is reflected from the tissue boundaries and the rest are transmitted again through the tissue. Second figure on the right shows an image that can be obtained using ultrasound (typically a B-Mode scan). Image is reproduced from [2].

1.3 Elastography

Strain imaging or elastography is becoming very popular now-a-days. Various diagnostic properties of tissues are not visualized form regular ultrasound images. Regular ultrasound images are related to the acoustic properties of the tissues. But various cancers are related to the changes in the stiffness of the tissue. Cancers, like scirrhous carcinoma of breast appear as extremely hard nodules [3]. In many cases the lesion may or may not possess echogenic properties which would make it detectable. For example, tumors of the breast or the prostrate could be invisible or barely visible in standard ultrasound examinations [4]. This is the primary motivation behind elastography. Elastography incorporates imaging the strain profile which relates qualitatively to the mechanical properties of the tissue or stiffness of the tissue.

1.4 Elastography: quantitative or qualitative?

Though the target of elastography is to image the tissue stiffness, it actually images the tissue strain under the application of a small stress on the tissue. Tissue strain relates with the stiffness qualitatively. There are practical difficulties in estimating the stiffness quantitatively. Tissue is highly anisotropic and non-linear. So, the linear theory of elasticity [5] cannot be applied here. Quantitative imaging possesses three fundamental problems as discussed in [6].

1. **Complexity:** this is one of the major obstacles that is faced in any attempt to make the strain imaging quantitative. The mechanical behavior of the tissue cannot be accurately described by simple models [7].
2. **Computational stability:** Quantitative elasticity imaging requires solving an inverse problem. The uniqueness of the solution depends on the number of the parameter. As the number of the parameter increases, the probability of getting a unique solution decreases. As well as the quantity of data required for achieving acceptably low error dramatically increases [8].
3. **Unknown boundary condition:** elasticity imaging is usually based on scanning a limited region of tissue, spanning a small volume which causes problem because some approaches to quantitative analysis only produce correct solutions if the mechanical properties are known all over the boundary [9].

1.5 Basic Imaging Procedures of Elastography

Elastography was first demonstrated by [1]. The procedure discussed in that introductory paper is discussed here in brief.

The first signal processing task for generating a strain image is deformation estimation throughout the scan region. A single RF ultrasound frame spanning a square region of tissue typically consists of around 128 columns of RF ultrasound data, called A-lines with several thousand RF sample along each A-line. The axial direction in RF ultrasound frames is far more densely sampled than the lateral dimension.

The inputs are taken in two phases or segments-

1. Pre compression
2. Post compression

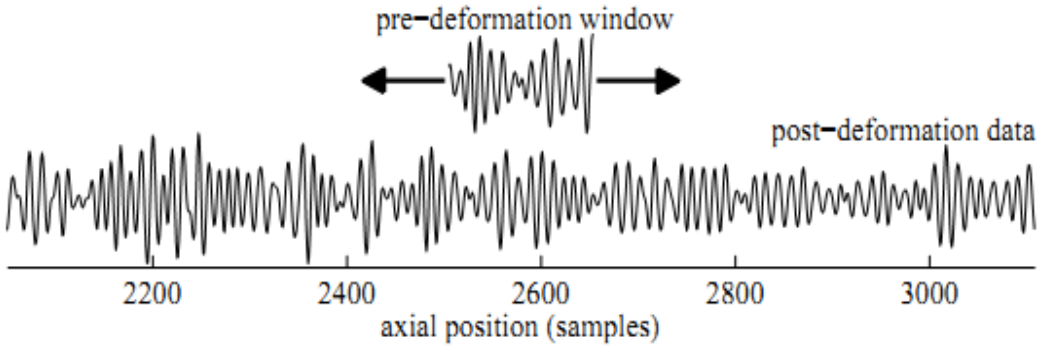


Figure 1.3: it shows a typical pre- and post-deformation/compression window. These two can also be considered as RF frame because of containing RF data. There will be a relative compression (one will be compressed with respect to other) between these two frames of data. Image is reproduced from [7].

In the first phase, ultrasounds along the A lines (parallel to each other) are sent to the tissue using ultrasound probe or transducer (acts as a transmitter). The ultrasound gets reflected from the tissue boundaries with sufficient amount of information or data which is received by the ultrasound probe or transducer (this time it acts as receiver).

In the second phase, ultrasound along the A lines are sent again to the tissue using the probe but this time the probe or the transducer is pushed against the surface of the tissue. This causes a uniform change in the axial and lateral component of the image data or information. Typically the information or data of the second phase will be a compressed version of the first phase.

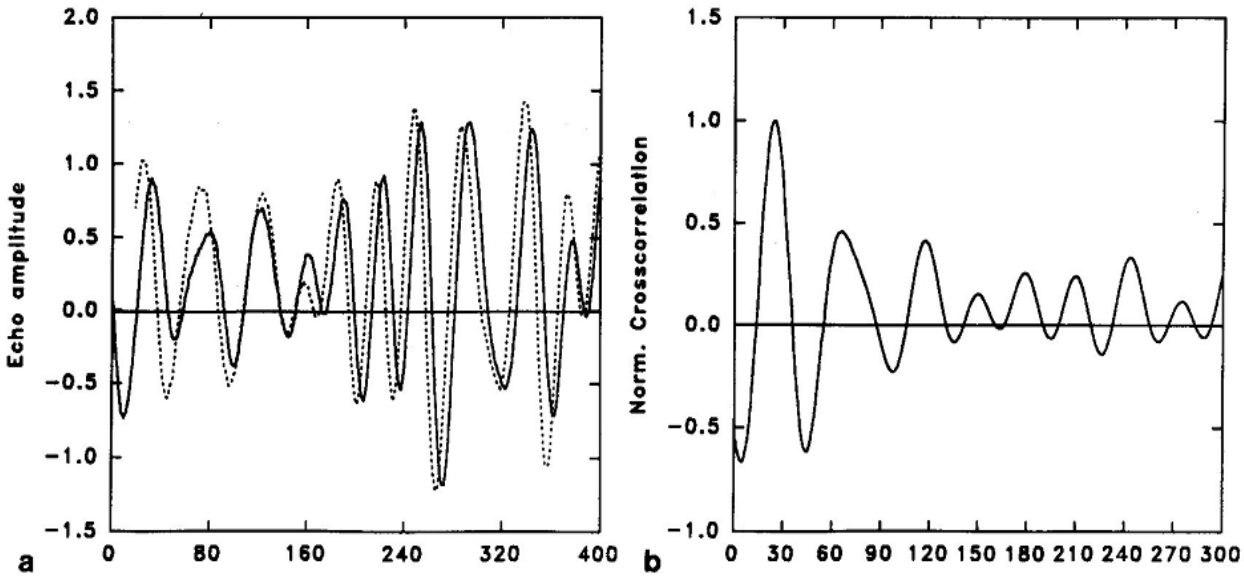


Figure 1.4: (a) shows typical pre- and post- segment pair. (b) shows the correlation between the two segment pair. Image is reproduced from [1].

To explain in detail, a strain image is created from 40-60 A- lines pairs obtained with a 1-2 mm lateral translation of the transducer between pairs. The axial compression for the second phase is an additional dz. The compressed A-lines (second phase) is shorter than the original A-lines (first phase) by 2dz/c, where c is the speed of the sound in the tissue (medium of interest) [1].

A strain profile is developed from the cross-correlation analysis of the pre- and post-compression A-line pairs. The strain is estimated in the axial direction only. These strain profile may be then converted in to elastic modulus profile by measuring the stresses applied. The resultant image is called Elastogram.

Elastogram are based on the time shift differences among segment of the ultrasound A-lines and rely heavily on the cross-correlation computations. Cross correlation analysis is well established in the literature and recently it is being widely used in the medical applications. In the medical field a number of publications describe the measurement of the blood velocity profiles using one-dimensional and two-dimensional correlators.

A pairwise evaluation of the time shift between the subsequent segments in an A –lines pair by a cross correlation technique generates the Elastogram. The linear cross correlation of the segment pairs is computed using FFTs.

The location of the maximum peak of the cross correlation function is time shift estimation between the data in the two segments as shown in the figure above.

From the above figure it is apparent that the relative time shifting between the two subsequent A lines is very small at the beginning but later it becomes dominating. In general the time shift of the second phase (compressed) A-lines relative to the first phase (uncompressed) A-lines increases from 0 to maximum of 2dz/c, where dz is the axial compression against the tissue surface and c is the speed of the sound in the tissue (1540m/s).

The resolution of the time shift measurement is determined by the sampling period at which the data is digitized. Some interpolation algorithms have been proposed to improve the resolution [10], [11]. A quadratic interpolation algorithm has been shown to be effective [11], [12] and it is simple to implement. The interpolation first fits a second –order polynomial which passes through the peak sample value of the cross correlation and its neighbor using the Lagrange polynomial interpolation. Then it analytically locates the peak of the fitted polynomial, assigning that temporal value to the improved time shift estimate. A similar technique has been reported by Boucher and Hassab [13].

After processing two subsequent A-lines a set of time shifts is obtained from t1 to t60. The corresponding strain profile is defined by-

$S_i = \frac{t_{i+1} - t_i}{2dz/c}$, i=1 to 59

Where S_i is the strain estimate for the segment pair i.

This process is repeated for all the A-line pairs resulting in a matrix of strain data. These values are then scaled and assigned as the intensity of the pixels in a 256 grey level image.

1.6 Algorithms available in the literature

Basically strain imaging involves obtaining the displacement estimate and then the strain profile form the displacement estimate. Some algorithms involve estimation of strain directly [14]. We present here a brief introduction to the strain imaging algorithms. We have discussed some recent algorithms (e.g. spline based estimate and direct and gradient based average strain estimator using nearest neighbor cross-correlation peaks) briefly to show the recent research trend in this field.

1.6.1 Correlation based estimate

In this method the local displacement of tissue due to the application of a small stress by transducer surface is obtained by using the cross-correlation among small temporal windows of the pre- and post-compression RF frames. This is basically the process discussed in [1]. From the displacement the strain is obtained by taking the gradient of the displacement field. But variations in the strain estimation from displacement (other than taking gradient) are found in the literature which will be discussed in this chapter.

1.6.2 Displacement estimate using prior estimate

There might be two types error occurring in tissue displacement estimate – bulk error and fine error [6]. Bulk error can be visualized as choosing a wrong cross-correlation peak. On the other hand, fine error arises from the discrete nature of the data. It is basically the sub-sample displacement estimation error. Bulk error is the most severe.

From our discussion it is clear that displacement estimate in one temporal window does not depend on the estimate of the displacement on other windows. But this might lead to bulk displacement estimation error. To reduce this error the displacement search is confined within a range. The range is limited by the estimate in the neighboring windows. Zahiri-Azar [15] presented some algorithms that use prior estimates to obtain the local displacement estimates. We present here such algorithms discussed in [6].

Basic tracking:

Basic tracking is used by several authors in their strain imaging previously [16, 17]. In this process the axial displacement is estimated in each of the column independently. As the relative movement of the tissue surface close to the probe or transducer is assumed to be too small, the first window is seeded with zero (zero displacement). As well as the relative movement between the windows is also assumed to be small, each lower window is seeded with the estimate with the previous window. That means the second window is seeded with the first estimate, the third window is with the second one and so on [6].

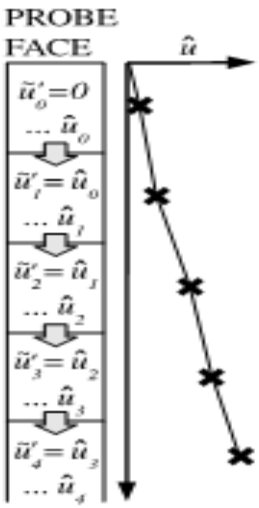


Figure 1.5: Basic tracking. Image is reproduced from [6].

Cross seeding:

In basic tracking each column are processed independently, where else in cross seeding each of the row are proceed at a time which means all the column are proceed in parallel. This is sometimes called 'single pass'. The significance of this name will be clear in the next process (multi pass analysis). In this case each window is seeded with a high-correlation estimate from the previous row on L column either side. A high coefficient of the previous row does not necessarily indicate that the estimate has no flaws rather it reduce the probability of bulk errors. Choosing a suitable value of L can reduce the rate of bulk error to a great extent [6].

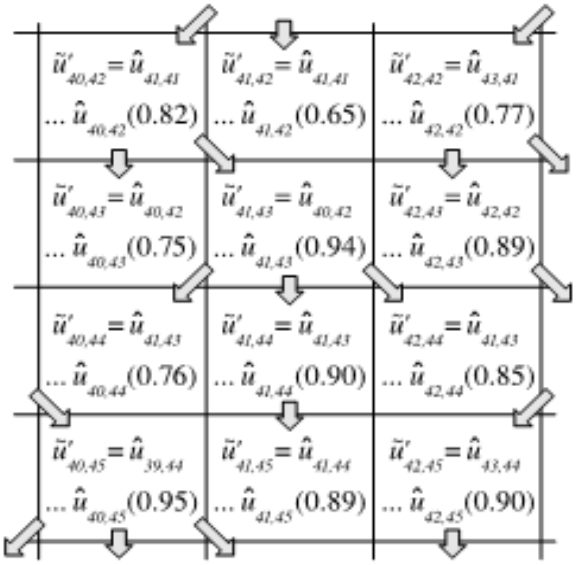


Figure 1.6: Cross seeding. Image is reproduced from [6].

Multi-pass analysis:

In the previous tracking the seeding is performed from the top to bottom only once, this means it is 'single pass' (top to bottom). From the name of the current process it is evident that there is multiple 'pass' in this process. More precisely, there is a cross seeding from the top to bottom at first pass, and then from bottom to top in the second pass. The second pass direction is reversed of that first one. In the first pass the displacement estimate and the correlation coefficient are calculated at every window and stored in a display buffer.

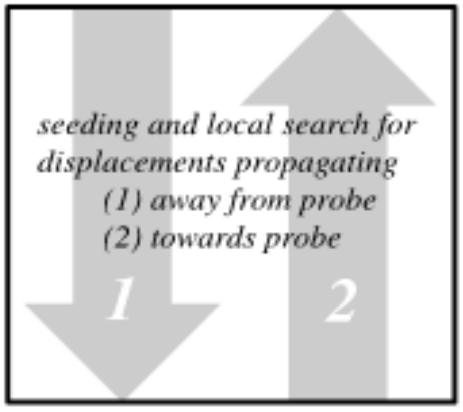


Figure 1.7: Multi-pass analysis. Image is reproduced from [6].

During the second pass a whole new estimates and correlation coefficient are calculated. When the new correlation coefficient is higher than that of the first pass it enters in the display buffer and the lower correlation coefficient are stored in the reserve buffer [6].

Coarse lateral tracking:

For significant amount of multi-dimensional motion, researchers have to deal with the multi-dimensional tracking otherwise data misalignment causes axial displacement estimation to fail. When the non-axial motion is not small enough and cannot be ignored, unfortunately there is no way to track elevational movement using ultrasound data required in 2D format ,but coarse lateral tracking creates a scope to avoid failure due to lateral movement (non-axial movements).

In the coarse lateral tracking the initial row begins with zero lateral displacement. After a short range axial search ,two extra correlation coefficient is calculated for post-compression windows in the neighboring A-lines with the same axial displacement .the lateral displacement estimate of the current window is incremented if a higher correlation coefficient is found in either neighbor. Though type of tracking turns to be useful for alignment purpose, they are not suitable for calculating lateral strain [6].

1.6.3 Adaptive stretching

A problem of the correlation based method is the tissue de-correlation due to the spatial compression. Also, the procedure of the obtaining strain from the gradient of the displacement is noisy, because differentiation is a noisy process. These problems are solved in adaptive stretching technique [14]. In this process the pre and post compression windows are divided into small overlapping temporal windows as usual. Now, due to tissue compression a post-compression window will be a compressed version of its corresponding pre-compression window. So, a stretch factor for post-compression window is searched that would make the correlation between the pre-compression window and stretched post-compression window maximum. The stretch factor gives a measure of the strain directly.

In practical, the post- compression window is stretched using a different factors and each time a cross-correlation is measured. The factor that gives the maximum cross-correlation is taken to measure the strain. Now, the search for stretch factor is done by different

techniques. Alam [9] demonstrated a binary search algorithm. The search range for binary search can also be limited by using uniformly spaced strains.

The problem of this method is it is computation intensive.

1.6.4 Spline based method

This method improves the image quality of the gradient based estimate which is quantitatively measured in terms of elastographic signal to noise ratio (SNRe) and elastographic contrast to noise ratio (CNRe). This method fits a smoothing spline on the tissue displacement estimate. For a noisy displacement y, the piecewise smoothing-spline minimizes the following penalized least squares score computed over all (x_i,y_i):

$$p\sum_i w_i\left[y(x_i)-\hat{y}(x_i)\right]^2+(1-p)\int\left(\frac{d^2\hat{y}}{dx^2}\right)^2 dx$$

Here p is the smoothing parameter. This equation optimizes between fitting error and roughness penalty.

Then strain is estimated from the gradient of the spline. Improvement in SNRe and CNRe has been reported in [18].

1.6.5 DASE and GBASE

Direct and gradient based average strain estimator using nearest neighbor cross-correlation peaks are two methods proposed very recently [19]. In this method the average strain is estimated by incorporating a cost function maximization using the nearest neighbor normalized cross-correlation peaks. Much improvement has been achieved in terms of SNRe and CNRe using these two methods. These methods have been reported to work better than adaptive stretching in case of in-vivo data.

1.7 our contribution

Improvement of computation time and image quality of strain imaging has been a target of research for the last two decades. We have presented some strain imaging algorithms. Some of these produce good quality images but are computation intensive while others produce images at sufficient frame rate. Both of the attributes are important. Producing good quality images are important for obtaining sufficient diagnostic information while imaging at a real time frame rate is important because it enables the clinicians to make proper probe adjustment and make effective investigation over a region of interest.

We have found that parallel computation is an effective solution for improving the run-time. But the programming task is cumbersome if one has to re-program all the existing algorithms in his strain imaging system. So we have developed a software tool that can automatically convert any strain imaging code into a parallel code and thus provide an improvement in run-time. Because this is a general structure, this can be applied to the codes which are inherently slow but provide good quality images.

Also, from a study on the signal processing tasks performed in elastography, we suggest that using windows of better spectral behavior other than rectangular window should provide better images. We have performed this operation on an experimental phantom data and demonstrated the results.

Since we have worked on the improvement of two aspects (image quality and computation time) of ultrasound strain imaging and these two improvements can be achieved simultaneously, we have titled our thesis as, '**a unified approach to improve the computation time and image quality of ultrasound strain imaging**'.

CHAPTER 2

Improving the computation time

2.1 Background

Ultrasound elastography is becoming a significant tool for early cancer detection. This imaging system typically uses serial algorithms and sometimes they are computation intensive. It is important to improve the computation time for achieving real-time imaging. This improvement can be achieved through parallel computing and using hardware that have parallel architectures. Recently some efforts to reduce the computation time by parallel programming are found in the literature. These works were focused on programming some of the existing ultrasound strain imaging algorithms in a parallel manner. Instead of manually converting a wide variety of algorithms to parallel form, we have developed a software that checks the loops of a code for dependency and parallelizes the code irrespective of the algorithm used. It was developed for ultrasound strain imaging MATLAB codes and uses multi-core CPU as a parallel hardware. We have given here the results of parallelization of strain imaging when performed in a multi-core CPU. As the graphics processing units(GPUs) provide more speed-up, feasibility of a similar system using GPU is also discussed here.

2.2 Parallel computation

Parallel computation is intended for accelerating the computation time of a program. Typically the programs are written in a serial fashion such that the program can be broken into discrete instructions and these instructions are run in a CPU one-by-one serially. For parallel computation we need multiple CPU cores. In that case one can write a program such that it can be broken into independent segments and each segment can be run simultaneously on different CPU cores. As tasks are performed simultaneously, it should take lesser time to perform the computation.

As already stated, for parallel computation a parallel architecture is necessary. Modern day processors are equipped with multiple cores e.g. Intel core-i7, core-i5 etc. But the most efficient hardware for performing scientific computation is the graphics processing unit (GPU). GPUs presented by NVIDIA are particularly suitable for this and they have their own parallel computing platform and programming model CUDA.

Programming in parallel environment using a high level language like MATLAB is easier because MATLAB is popular and MATLAB is user friendly [20]. MATLAB provides facility to solve computationally and data-intensive problems using multicore processors, GPUs and computer clusters. MATLAB parallel computing syntaxes also appear to be similar to their serial counterparts. Also, there are other toolboxes like Jacket^{TM} and GPUmat that can be used in conjunction with MATLAB to use the GPU resources. A review among these toolboxes is available in the literature [21].

From our study, it is apparent that MATLAB GPU toolboxes are at their infancy. A lot of serial syntaxes cannot be used in these toolboxes. On the other hand, MATLAB's resources for programming in a multi-core CPU environment are quite sophisticated though the acceleration is not comparable to the GPU one. GPU routines are way faster than multi-core CPU routines. But we confined our study on multi-core CPU.

2.3 Programming in a multi-core CPU using MATLAB

MATLAB has vast resources for programming in a multi-core CPU environment. These resources come as a product called MATLAB parallel computing toolbox. There are several useful features in this toolbox that helps exploiting the advantage of a parallel architecture:

a. Parallel for loop
b. SPMD (Single program multiple data) construct
c. Distributed array
d. Interactive and batch execution etc.

But it is parallel for loop or 'parfor 'that suits our purpose most.

2.4 Application to elastography

Ultrasound elastography [1] is intended for imaging the mechanical property of tissue by estimating tissue displacement and gradient of the displacement estimation. Recently real-time elastography imaging has found some important applications [22]-[25]. The requirement for

real-time imaging system is high frame rate which is primarily limited by data acquisition and processing time for each frame. The processing time can be reduced by using parallel codes in a parallel architecture like graphics processing units (GPUs), multi-core CPUs, distributed computing servers (DCS) etc.

Significant improvement in computation time has been reported in the literature for elastographic systems [26],[27] and other similar imaging modalities [32] by using graphics processing units(GPUs). CUDA C was used as the programming language in these cases. It is also possible to use a multi-core CPU to achieve acceleration in the run-time though the improvement will be much less than the GPU case. In this case we can use MATLAB which is a more user friendly language than CUDA C. We have demonstrated the use of multi-core CPU and MATLAB for elastographic imaging system.

2.5 Our novel concept: automatic parallel conversion

Though parallel programming is a solution for improving computation time, it requires the knowledge and time to reconfigure the existing codes for parallel environment. Also, there are numerous strain estimation techniques, each of them having some definite advantages over others. For example lest-square fit strain estimator shows significantly better SNR and CNR performance than gradient based estimator. However this improvement is achieved at a cost of worsened resolution. Adaptive stretching provides robust strain estimation without worsening the resolution. But it is extremely computation intensive [18]. So a trade-off between imaging performance parameters can be achieved by using different methods and the choice of a method might depend on a particular application. It would help the clinicians if all these methods are available and programmed in a parallel manner. It is a tedious job to rewrite the existing serial codes for all these techniques in a parallel form. **So, we developed an automated system that can generate a parallel code when a serial code is given as input**. This work was actually intended for checking the feasibility of such a system. So, primarily we developed the system for a multi-core CPU system. As an obvious extension of this work, we have discussed the possibility of a similar system that uses GPU.

2.6 Approach to parallel conversion

Before going into the details of parallel coding we need to identify the extent to which we wish our codes to be parallel. A same code can be parallelized in numerous ways depending on respective programmer. In our study, we converted a strain imaging code into parallel using two different ways which will be discussed in the results section. But, as we are aiming at an

automatic converter we need a generalized platform. As we will demonstrate later, ultrasound strain imaging systems, which is our field of interest, often use MATLAB codes characterized by large for loops for strain or displacement calculation. In most of the cases these loops are the bottle-neck of the code's run-time. So, we infer that these large loops can be partitioned and run in parallel to save time. Our inference was supported by results obtained from the study of strain imaging codes. We set this loop (only for loops) splitting job as our primary definition of parallelization for ultrasound strain imaging systems.

We have designed a software which takes any strain imaging MATLAB code, splits the for loops inside the code in a defined manner and returns a new code. This new code runs faster when it is run on a multi-core processor.

Using MATLAB parallel computing toolbox it is fairly easy to do such loop splitting. If we replace the 'for' loops with 'parfor' (parallel for loop) by following some well-defined criterion then the new code will run in parallel manner in a multi-core processor irrespective of the number of processor cores. But this apparently simple procedure has two problems associated:

PROBLEM 1. All for loops do not maintain the required conditions to be replaced by a parfor.

PROBLEM 2. A parfor loop cannot be nested inside another parfor loop.

To understand the 'problem 1' we need to take a look at how MATLAB deals with a parallel for loop.

In the context of MATLAB parallel for loop is issued as parfor. Parallel for loop (parfor) is conceptually almost the same as ordinary MATLAB for loop with an important exception that different iterations of the parfor loop are performed on different MATLAB workers [28]. Thus a speed-up in computation time is expected. MATLAB successfully runs a parfor loop if it can classify the variables inside that loop into one of the five defined categories. These categories are

1. Loop variable 2. Sliced variable 3. Broadcast variable 4. Reduction variable 5. Temporary variable

This categorization is mostly based on variable's indexing scheme. Basically it makes sure that the issued parfor loop's iterations are independent of one another. We have used this categorization of variables in our software. So it is important that we discuss in brief about these variables. Those who are familiar with parallel computing toolbox can skip the following section of variable’s definition without a loss of continuity. This section was partially reproduced from Mathworks’ technical documents [28].

***Loop variable**:* Used as loop index and never assigned inside a loop.

Sliced variable: An array whose segments are operated on different iterations of the loop. These variables should have some definite indexing scheme and all the occurrences of the variable inside a loop should have same indexing scheme.

***Broadcast variable**:* Defined before the loop and never assigned inside the loop.

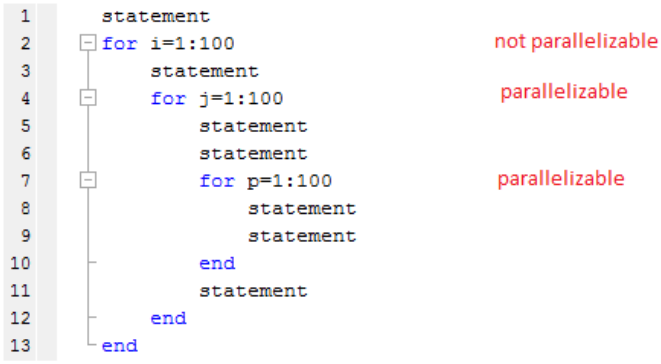
Reduction variable: Variables which are used for data accumulation across iteration of the loop. For example, X=X+a, X=[X a] etc., where X is a reduction variable.

***Temporary variable**:* Variables which are subject to a simple non-indexed assignment inside a loop. These variables are not available outside the respective parallel loops where they are created. So any reference to such variable outside the loop would generate an error.

If each of the variables of a particular parfor loop falls under one of these categories then it is a valid parfor loop. So, if a loop's variables have fulfilled these definitions then we can say, that particular loop is free of 'Problem 1' stated earlier.

Now, let us focus on 'Problem 2'. There might be situations when we have nested for loops and all of them (or some of them) fulfill the conditions to become a parfor loop. We cannot simply convert all nested convertible loops to parallel form because MATLAB does not support it. In such a case we would parallelize the outermost parallelizable for loop.

Code segment 1



the above code block is parallelized as follows:

code segment 2

```
statement
for i=1:100
    statement
    parfor j=1:100
        statement
        statement
        for p=1:100
            statement
            statement
        end
        statement
    end
end
```

Parallelizing the outermost loop was found to be most efficient than parallelizing the inner loops. We implemented the displacement estimation method described in [1] using MATLAB code. This code had two for loops. When the outermost loop was parallelized it resulted in an improvement in runtime. We have given the numerical data in the results section. Parallelizing the inner loop resulted in a code that took more time than serial one.

We can summarize the above discussions as follows. The ‘for’ loops inside a code will be checked for dependency information. It will be done by checking the indexing of the variables and following the dependency definitions given by MATLAB. Loops which do not have any dependency will be replaced by a MATLAB parallel for loop (parfor). If there are nested independent loops then only the outermost independent loop will be replaced by parfor. The system that we developed follows this procedure to parallelize the loops of the given code.

2.7 Scope for automatic parallelization of ultrasound strain imaging codes:

Code parallelization is a more complicated job than the one we are aiming at. Here, by the word 'parallelization' we refer only to the job of launching several iteration of a for loop on

different MATLAB workers. This is, in the context of MATLAB, is tantamount to just replacing the for loop with parfor. But why is this simple manipulation expected to work every time? After all we might need to modify our serial codes adaptively rather than changing 'for' to 'parfor', even when algorithms behind the codes are parallel in nature. The answer to this question is, this should not work in all cases. But under some simplicity this 'for' to 'parfor' replacement is often a viable solution. Ultrasonic strain estimation codes offer us such simplicity and this is the reason we are mentioning that this work is intended for strain imaging. Otherwise it could have been a universal automatic parallel converter

.

What simplicity do we need for such implementation and how simple are the strain estimation codes? Let us explore this from elastographic point of view.

1. Strain estimation MATLAB codes are marked by their large for loops, which are often the bottlenecks of the entire code. Partitioning these for loops is often the most efficient solution. As a demonstration of a typical large loop, we implemented the method discussed in Ophir [1] using MATLAB (this is regular implementation, not a parallel code). The code took the form of the code segment 3 (simplistic form, not the actual one):

Code segment 3

```
statements
for i=x:y
    statement
    for j=w:z
        displacement=xcorr(data1,data2);
        statement
    end
end
```

The entire code required 1.0427 seconds to run, whereas the segment from line 2 to line 8 took 1.0426 seconds to run for a particular set of data which is given in results section. That means almost all of the total run-time was spent on the loops. These calculations are average values from 50 runs of the code.

2. Typical displacement estimation routines (e.g. Basic elastography [1], adaptive stretching [14] etc.) operate on pixels one at a time. Often these operations on a pixel are independent of the operations on other pixels. Of course, there are exceptions. Zahiri-Azar [15] proposed a method of obtaining displacement estimate using prior estimates. These methods do not offer a high degree of parallelism. But still a lot of displacement estimation methods are inherently parallel including the primitive and popular ones.

3. Most of the cases the for loops are intended to sweep through the pixels as in the case of implementation of [1]. Here, two loops sweep through row and columns. Also, in many methods operations on pixels are identical and independent of one another. So, it much likely that indexing schemes inside these loops would contain simple linear expressions involving loop variables. This makes a variable inside a loop a good candidate for fulfilling the conditions of variable categorization stated earlier.

4. For 2D elastography data are 2 dimensional. Two sets of data are obtained: pre-compression data and post-compression data. Each pixel intensity is obtained by using only a portion of the data from both sets. This makes the data variables easier to be classified as sliced variables.

5. Data types are double in almost of the cases. We do not expect cell, structure or string data types in strain estimating algorithms quite often.

2.8 Detailed algorithm behind automatic parallel conversion

Following flow chart explains much of the basic principles of our system. This flow chart is elaborated here with details.

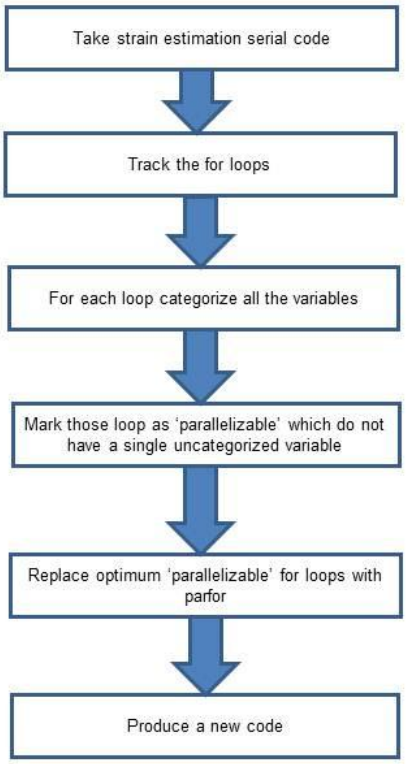


Figure 2.1: A simplistic flow chart showing the operation of the automatic parallel converter

The system is given a file that contains a serial code. The file is opened to read. It is scanned throughout and all its for loops are tracked and listed down along with their corresponding 'end's. This list may be called 'loop list'.

Code segment 4

```
1   for i=1:30
2       a(i,:)=3*i;
3       [b c]=d(i+1:end);
4   end
```

For the code segment 4 given above, the 'loop list' should look like Table 1.

TABLE 1

ILLUSTRATION OF THE LOOP LIST

2	8	0
4	7	1
Loop start	Loop end	Degree of nested loop

Next, the variables of the code are scanned. The system extracts all the variables, their line of appearances and their indexing schemes. These variables are grouped into two. Group 1 contains those variables who are which appear on the left side of an assignment. Rests are placed on Group 2. For example, for a line a=b(2:end)+c(1:end-1), variable a will be placed in Group 1 and b and c will be placed in Group 2. For a statement like userDefinedFunction (a,b), variables a and b will be placed in Group 2.

Next, the Group 1 variables are listed in a cell array, where each cell corresponds to a line. For

example, i^{th} cell of the cell array will contain those variables of i^{th} line which fall under Group 1. Also the indexing scheme of a particular variable's particular occurrence will be stored side by side.

We call this cell array 'list 1'. Group 2 variables are also listed in a similar cell array and we call this cell array 'list 2'.

TABLE 2

LIST (LIST 1) OF ASSIGNED VARIABLES FOR THE GIVEN CODE SEGMENT

I	none	None	A	i	:	B	none	none			
						C	none	none			
	X	Y		x	y		x	y		x	y
variable	Indexing		variable	indexing		variable	indexing		variable	indexing	
Line 1			Line 2			Line 3			Line 4		

TABLE 3

LIST (LIST 2) OF VARIABLES WHICH ARE NOT TARGET TO AN ASSIGNMENT

			I	none	none	d	i+1	end			
	X	Y		x	y		x	y		x	y
variable	Indexing		variable	indexing		variable	indexing		variable	indexing	
Line 1			Line 2			Line 3			Line 4		

A variable might appear in both lists 1 and list 2 depending on whether it was assigned or called on different lines.

Notice that, the system has listed all the information about the code that is necessary for further steps. As stated earlier, to arrive at the decision that a particular for loop can be replaced by parfor, we should be able to categorize all the variables inside the loop. This categorization depends on how the variables were assigned, their occurrences and their

indexing schemes. All these information are contained in loop list, list 1 and list 2.

Next, the system takes a particular for loop from 'loop list' and the variables inside it from list 1 and list 2 and tries to categorize these variables.

Loop variables are easily tracked as the first variable of i^{th} cell of list 1, where i is the line number where the particular for loop began. i is collected from loop list. System has to make sure that it is never assigned inside that particular loop. This is made sure by checking that loop variable is not present further in list 1 within the context of the particular loop.

Temporary variables are tracked by finding variables from list 1 which are target of a non-indexed assignment, this require checking the indexing scheme. We need to make sure that these variables are never called outside the loop. This is made sure by noting that such a variable 'var' did not appear in list 2 outside the loop or it appeared outside the loop in i^{th} cell of list 1 and j^{th} cell of list 2, where i<j. But there is one problem associated in determining temporary variables. Temporary variables are also non-indexed, but they are allowed to appear outside the loop. So, in the case of tracking temporary variables, we first determine the reduction variables and make sure they are not considered when temporary variables are taken into account. Reduction variables are tracked by finding those non-indexed variables which appear in a single line on both lists 1 and list 2.

Next the system has to categorize the sliced variables. First the variables inside the loop which are subject to indexed assignments are extracted from list 1 and list 2. Each of these variables must have same indexing scheme for all their occurrences. After that the system takes each of these variables and checks their indexing scheme to make sure that the indexing is of the form allowed by sliced variable definition (Appendix A shows allowed sliced variable indexing scheme).

Notice that we did not discuss about those variables those which are not assigned inside the loop. In fact, those variables which do not appear in list 1, but do appear in list 2 are assumed to be categorized. This is consistent with the fact that such variables are actually MATLAB broadcast variables. Thus we are implicitly categorizing the broadcast variables.

If a single uncategorized variable exist in a loop that loop is not parfor replaceable. We obtain information about parallelizability for all the loops in a similar manner. This information is appended as a column in 'loop list' to form a new list (list 3). A typical list 3 should look like Table 4.

TABLE 4

FINAL LIST OBTAINED AFTER ANALYZING THE LOOPS

2	8	0	Parallelizable
4	7	1	Parallelizable
Loop start	Loop end	Degree of nested loop	Parallelizable

Finally, the given code is brought in again, read each line and written on a new file with its allowed for loops replaced by parfor. This allowed for loop is obtained from list 3 (Table 4), maintaining the conditions regarding nested parallel for loop.

2.9 Results

A. Implementation of displacement estimation on a multi-core CPU

Displacement estimation technique described in [1] was implemented in a serial or traditional manner using MATLAB. The routine was also reconfigured for a multi-core CPU environment using MATLAB parallel computing toolbox in two different ways. Data were collected from a Phantom experiment. For a window size of 128 samples and window shift of 64 samples, we obtained an image of 124x96 pixels. First the serial routine was run on a Intel core i5-2310 CPU with 32-bit windows 7 operating system. The code took 1.0427 seconds to run.

For using the multi-core CPU resources the code was parallelized in two ways. First, the loops of the code were parallelized by using 'parfor'. For same data, displacement estimation parameters (i.e. window shift etc.) and using the same system the parallel code took 0.4368 seconds. Next, the code was parallelized using MATLAB spmd construct without using any parfor. This time the code took 0.5322 seconds. All the run-time calculations are obtained from averaging multiple run-times of the respective code.

B. Results obtained using automatic parallel converter

Parallelization discussed above was actually achieved by manually converting or reprogramming the routine for using multi-core resources. Next, we use the serial code of the implementation of [1] as an input to the automatic parallel converter. A parallel code is retuned which looks almost same as the first manual conversion discussed above. The resulting code

took 0.4395 seconds under same operating conditions same as the previous cases. This is practically almost equal to the manual conversion result.

The system was also tested by using simulated codes as input to check its ability to identify different variable types and looping conditions. Multiple nested loops were simulated with a mixture of parallelizable loops and unparallelizable loops. The system was able to identify the parallelizable loops.

2.10 Automatic parallel converter for GPU

Using GPU gives much more acceleration than a core-i7 processor. So, efforts should be focused on developing a similar converter for GPU environment. But using MATLAB it is much more complicated than the CPU case. In our case the advancement of parallel computing toolbox assured minimum modifications for a serial code to be converted to a parallel code. The modification was generalized for strain imaging system and declared as a replacement of 'for' loop with 'parfor'. But our study on MATLAB GPU programming toolboxes (parallel computing toolbox, GPUmat, Jacket) shows that programming with these tools is not so general. It requires careful modifications of the serial code. So, making a similar automatic converter is not so easy for GPU environment using MATLAB. But, with future versions of these toolboxes or using languages other than MATLAB it might be possible.

2.11 Conclusion

We have presented our approach to aid the effort to accelerate the computation time of strain imaging systems. Our approach is to generalize the parallelization effort and develop system which can be used in more than one scenario. We developed a software tool using MATLAB that can convert any elastographic MATLAB code to a parallel form. We have demonstrated the characteristics of typical elastographic MATLAB codes and found out that they give us some general ways to parallelize them. We were able to do such parallelization in an automated manner by scanning the indexing schemes of the variables and replaced the potential loops with parallel for loops. The automatically generated codes were found to perform in a similar manner to those parallelized manually in a multi-core CPU environment.

Chapter 3

Improving the image quality

3.1 Background

For practical processing purpose, the data that we use must be of finite and reasonable extent. This requires us to truncate the data in time domain. In that case numerical calculation is performed by performing small calculation in several segments or truncated data. Truncation more precisely means taking off a portion or a segment of data from the large data in order to make numerical calculations more reliable and easy. This kind of truncation can be extensively used in strain image forming. While forming an image, there are large amount of data for which large numerical calculation is required, this calculation of forming an image can be more easy and reliable by truncating data or information in to small segments or portions.

This truncation of data or information can be done either in frequency domain or time domain. But for forming an image the latter one is discussed here. Truncation of data or information in time domain means picking up a portion of data or information in time domain for calculation. For example we have a large signal from which an image is to be formed. We cannot actually deal with all the data or information of that large signal at the same time. So we need data truncation in time domain. Data truncation in time domain typically means multiplying a rectangular signal to the large signal. To define rectangular signal, in which equal weight is assigned to all the data within a certain range. It can also be called a window. Multiplying a rectangular window with a large signal in time domain will assign equal weight to a range of data in the large signal and rest data will be assigned to be zero [29].

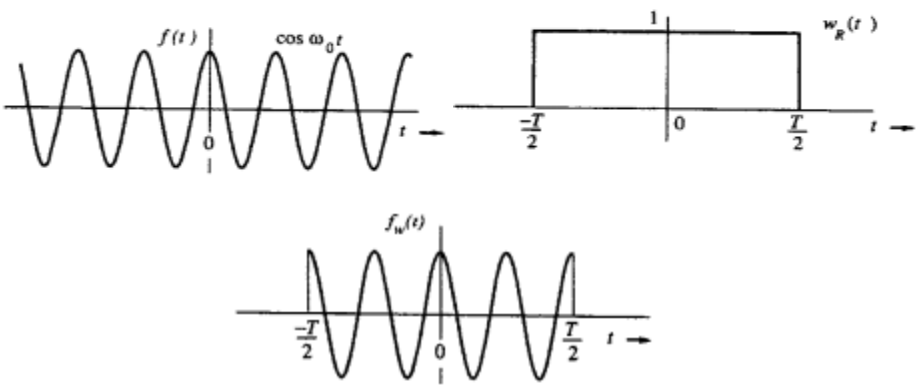


Figure 3.1: first one on the top-left is a typical everlasting cosine signal which can be considered as large signal. Second one on the top-right is a rectangular signal or window which is used for data truncation and the last one on the bottom middle is the truncated signal which is to be computed. Remember all these signals in the figure are in time domain. Reproduced from [29]

3.2 THEORY

From the typical relationship between the time and frequency domain it is apparent that multiplication in the time domain actually means convolution in the frequency domain. So data truncation in the time domain always results in convolution in the frequency domain. To understand more conveniently we can assume that the large signal is sinusoidal, when a truncation is performed in the time domain, the rectangular window is multiplied with the sinusoidal signal. To find out the clue what is actually happening in the frequency domain convolution we have to analyze the frequency domain of rectangular window and sinusoidal signal individually.

For the case of rectangular signal, if we analyze the frequency domain it is apparent that we will find a sinc function. This sinc function occupies typically a large bandwidth. The amplitude is more in the center and gradual slow decay in both sides.

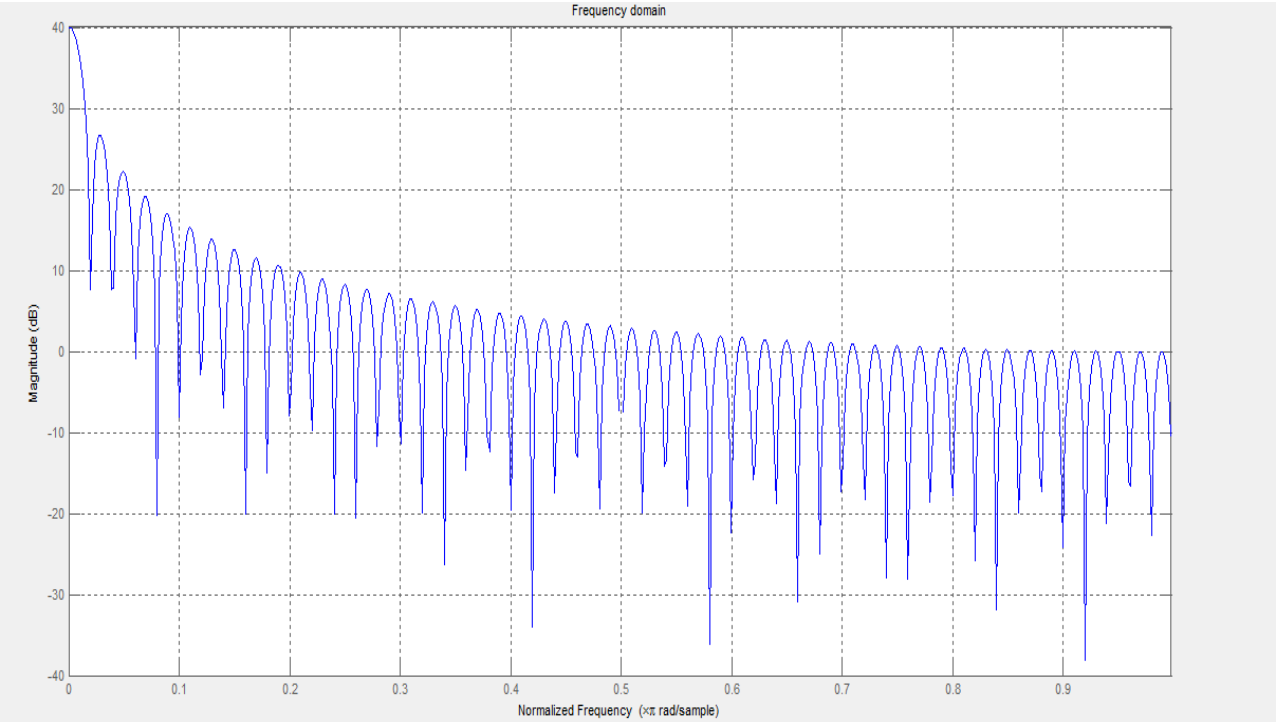
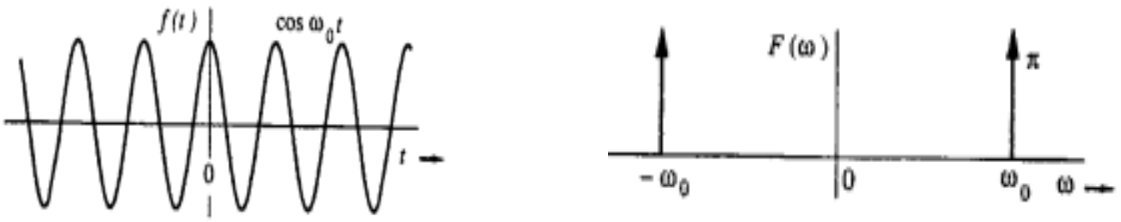


Figure 3.2: Frequency domain of rectangular window. It is shown one sided (only positive frequency) on the above figure. There is a gradual decay in the amplitude or magnitude as it is going to left to right.

On the other side for the case of large signal in time domain, it is apparent that the bandwidth of a large signal in time domain will result in small bandwidth. Again this is from the typical time frequency inverse relationship.



(a) (b)

Figure 3.3: In (a) a typical large signal possibly an everlasting cosine. In (b) the frequency domain of the large signal showing that it has a small (almost zero) bandwidth. Reproduced from [29]

Convolution of two signals in frequency domain means the bandwidth increases. This can be realized because a large signal has a narrow bandwidth in the frequency domain and after just truncating it, it becomes narrow (selective portion) in time domain, which can be derived from the time frequency relationship that the bandwidth of the truncated signal will definitely increase.

In this particular example when the truncation is performed by multiplying a rectangular window which has a SINC function (large bandwidth) in the frequency domain, the resulting bandwidth of the truncated signal also becomes large because of the convolution though before truncation the signal has a narrow bandwidth. So it can be considered that the bandwidth of the truncated signal or data depends on the bandwidth of the truncating window. Here the truncating window is the rectangular signal [29].

3.3 Application to elastography

In the correlation method of strain estimation, it is necessary to take the pre and post RF data. To form the strain image, it is required to find out the correlation between the pre and post RF data. To perform the correlation between the pre and post RF frame, at first the desired portions or segments from both pre and post RF data or signals are truncated. Actually two things are occurring in this case- the former one is truncation in time domain and the latter one is correlation in time domain.

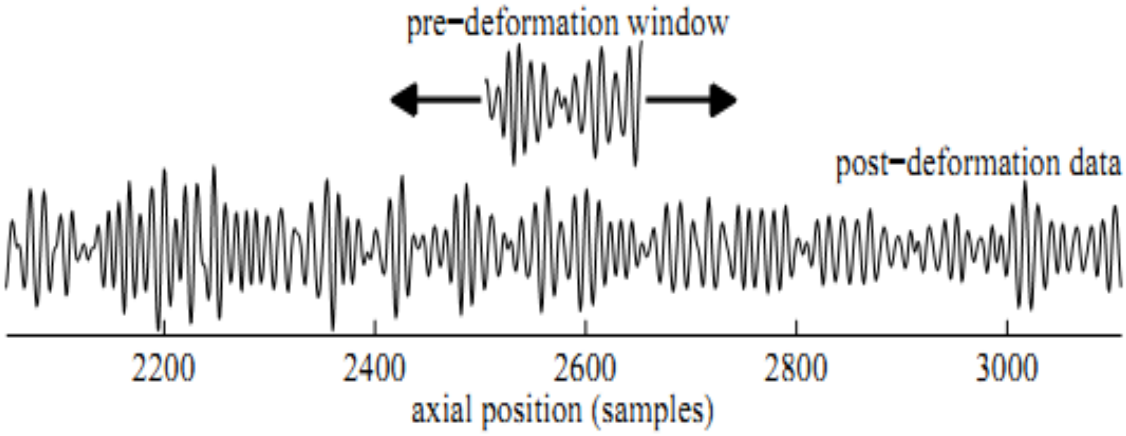


Figure 3.4: Typical pre- and post-compression data are shown. These two frames are lagging or leading each other by some samples.

The former one represents convolution is frequency domain as it was discussed before. Correlation in time domain is analogous to convolution in the time domain which in turns means some sort of multiplication in the frequency domain.

In the continuous time domain, convolution can be defined as-

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau = x(t) * h(t)$$

Also for the discrete time domain, it can be defined as-

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]$$

So putting it all together in the elastography process at first the bandwidth of the signal increases in frequency domain after data truncation in time domain and then secondly the magnitude of the signal in the frequency domain increases after correlation in time domain.

Forming of a strain image using rectangular window is given in the figure below-

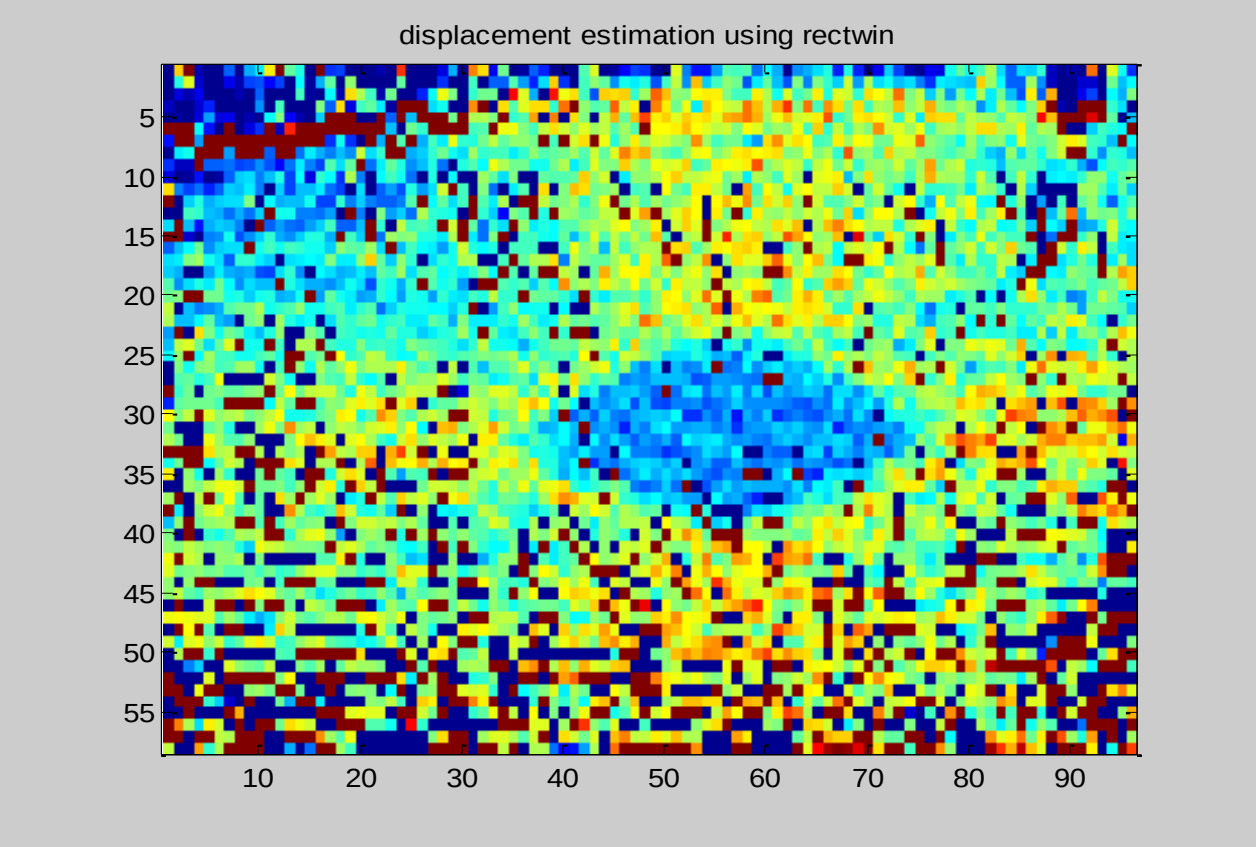


Figure 3.5: strain image using the rectangular window

3.4 Spectral characteristics of the rectangular window

In the previous discussion, we emphasize on the phenomena that occurs in the frequency domain after performing data truncation and correlation between the pre- and post- RF frame. By this time it is more or less realizable that the frequency domain of the truncated data depends on the bandwidth of the rectangular window.

Basically, the bandwidth in the frequency domain for the rectangular window is extensively large. We can divide this large bandwidth in to two segments

(i) Main lobe
(ii) Side lobes

In the figure below, the mainlobe can be defined as the center lobe of the bandwidth which has sufficiently high magnitude. The lobes on the both side with a decaying magnitude can be called as side lobes. The side lobes have always less magnitude than the mainlobe, which means most of the information of the signal is confined in the mainlobe and some of information leak in to the side lobes.

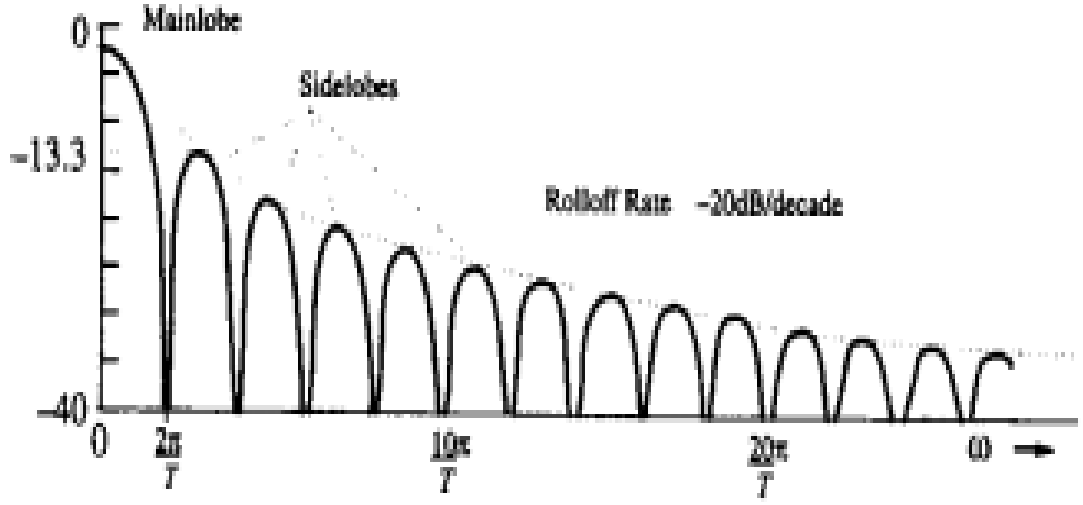
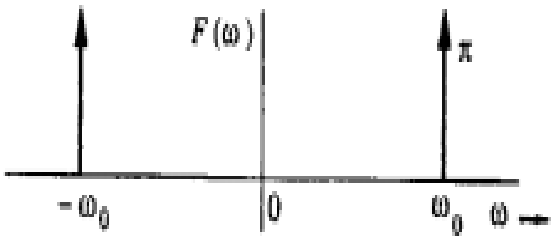


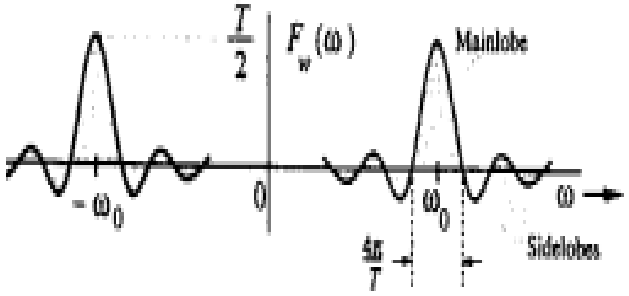
Figure 3.6: Main lobe and the side lobes in the frequency domain of a truncating window is shown. Reproduced from [29]

In case of rectangular window, in the frequency domain it has a mainlobe with high magnitude and decaying sidelobes on both sides which extends to almost infinite.

It can be explained more elaborately, for example we have a large signal (precisely an everlasting cosine), if the frequency domain of this cosine signal is analyzed, we will see there are two impulses in two frequencies, when the truncation and correlation are performed on the everlasting cosine signal, it can be seen in the figure that these two impulses spreads and the bandwidth increases for both of them .Apparently these are the main lobes, which can also be termed as Spectral Spreading.



(a)



(b)

Figure 3.7: In (a) the frequency domain of a large signal (everlasting cosine) is shown. In (b) the frequency domain of the signal which is truncated from the large signal is shown. The latter one is specifically showing the spectral spreading of the former one. Reproduced from [29]

From the figure above, it is clear that at first there were impulses before truncation and correlation and after truncation the impulse bandwidth increases from zero to a finite value and performing correlation increases the magnitude of the mainlobe as well.

It also seems that some information of the main lobes leak in the side lobes. In here the side lobes arises because of the data truncation but the correlation in time domain magnifies the side lobes which is undesirable.

3.5 Problem with the rectangular window:

The main problem of the data truncation is the sidelobes in the frequency domain. It can also be termed as spectral leakage. More over for the cosine signal on the above figure the sidelobes becomes overlapping in the tail end and might result in aliasing. It becomes more vulnerable when performing correlation in the time domain magnifies the amplitude of the side lobes. A typical rectangular window has side lobes with a very slow decaying nature. The decay rate of the sidelobes is proportional to 1/w. It is called the roll off rate.

This can also be explained physically, as the rectangular signal is discontinuous in the time domain, from the Fourier series , it is apparent that it needs very large (infinite) number of sinusoidal waves of different frequencies for closer approximation. In other way, it requires a large number (infinite) of sinusoidal signals of different frequencies to represent a rectangular signal in time domain .This in terms means that the bandwidth of the rectangular signal will be very large and with a slow decay on the side lobes.

3.6 Smooth Windows:

So the Fourier representation of a signal, having a sharp edge (discontinuous) requires large number of sinusoidal of different frequencies, which results in large bandwidth with a slow decay in the side lobes. This large bandwidth results in frequency interference or overlapping in the tail. This kind of interference or aliasing in the tail deteriorates the image quality. So to get good quality of strain image we are concentrating on the window which is not discontinuous in nature and not having sharp change or edge. This kind of window can be termed as Smooth window or tapered window.

Blackman, Chebyshev, Hanning, Hamming, Kaiser, Gaussian are such smooth window. For performing the data truncation these smooth windows can be used to get a good image quality.

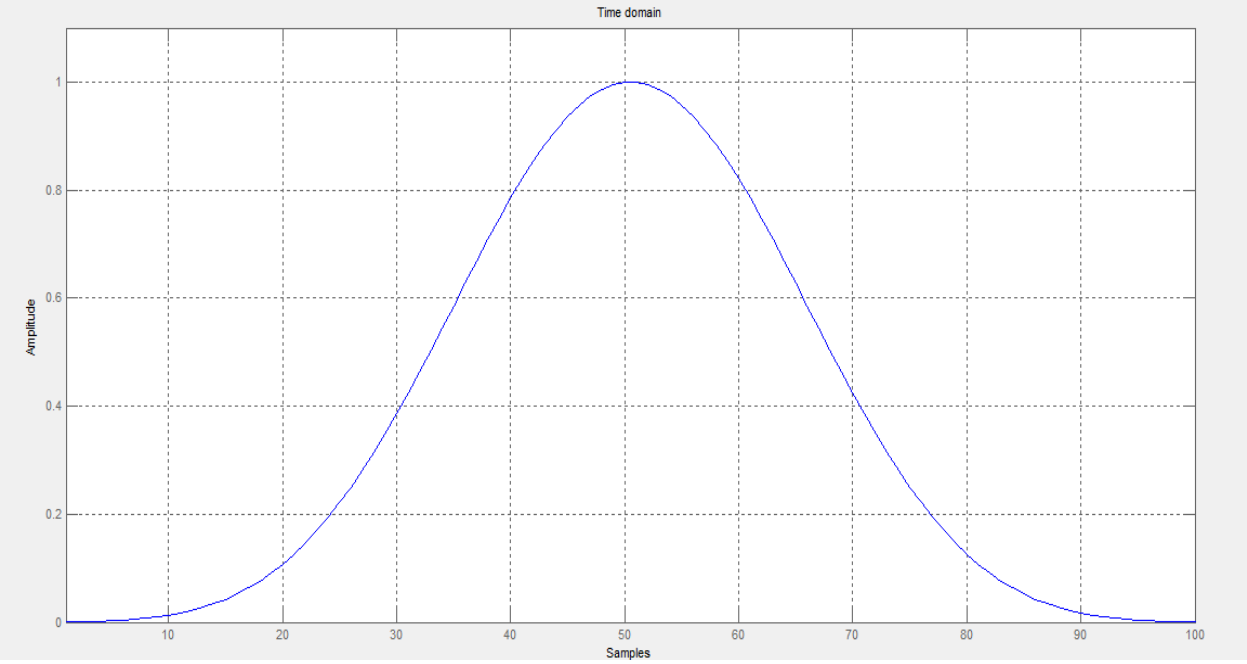


Figure 3.8: Chebyshev window in time domain

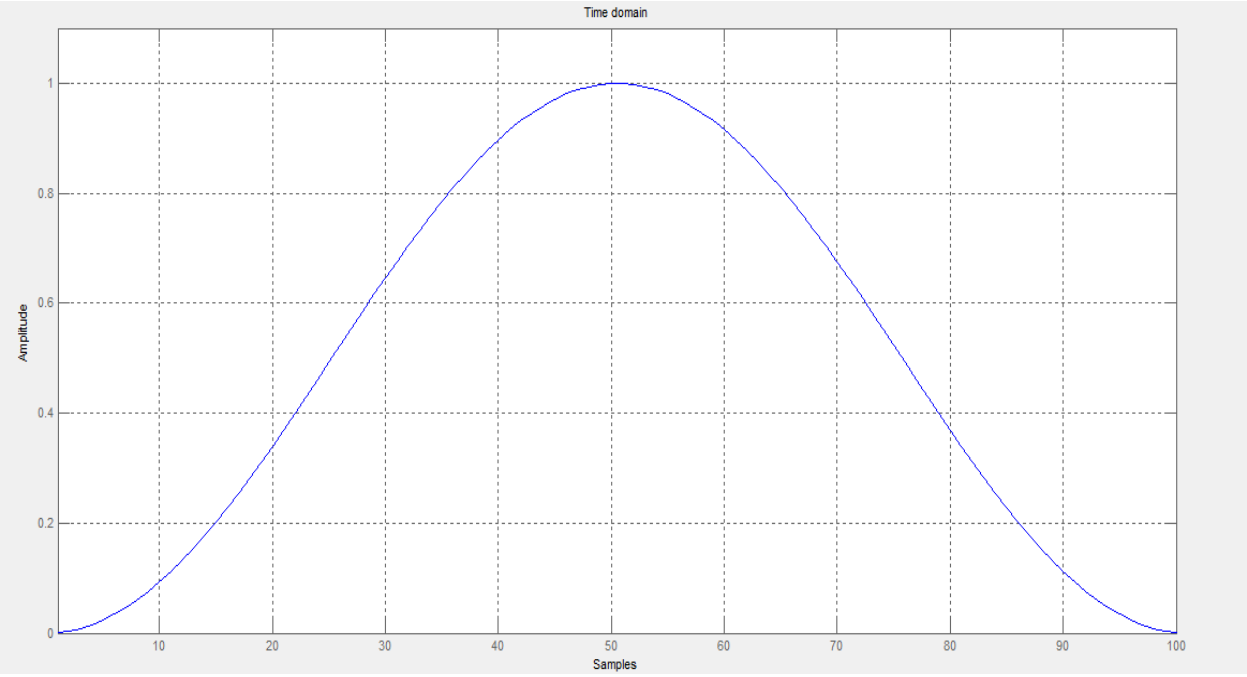


Figure 3.9: Hanning window in time domain

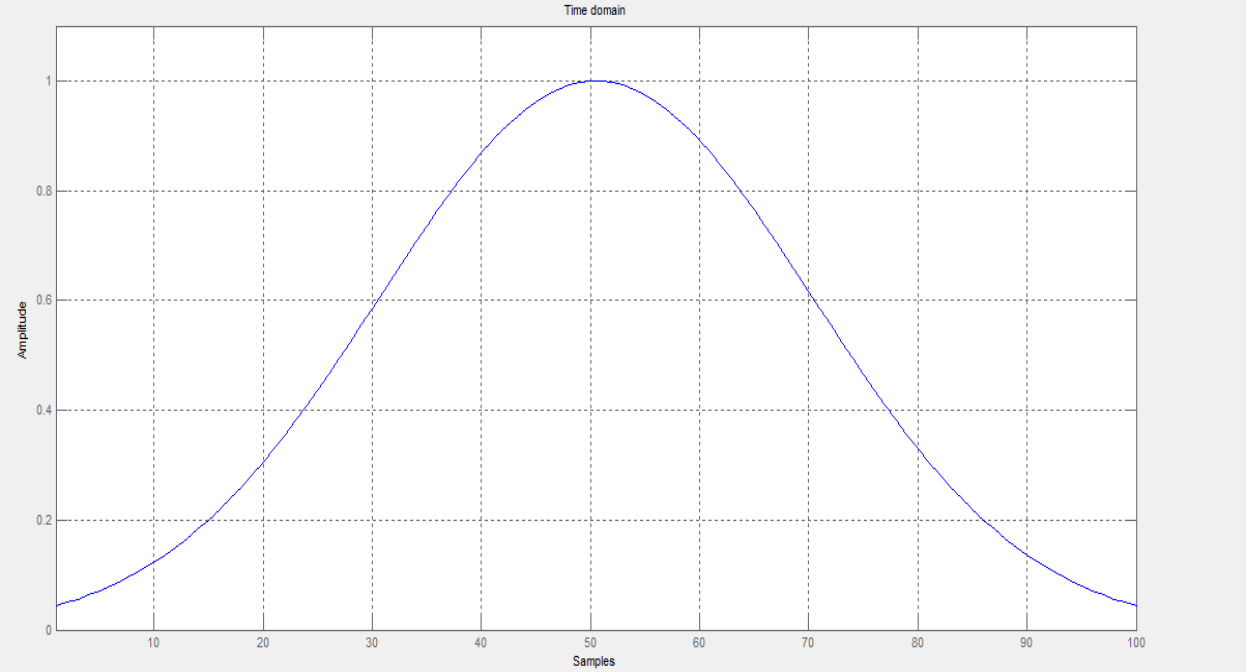


Figure 3.10: Gaussian window in the time domain

3.7 Spectral characteristics of the Smooth window:

let us think in a simple way first, the smooth window means there will not be any sharp change or edge in the window, that means in the Fourier representation the smooth window requires less amount of sinusoidal of different frequencies than the rectangular window (discontinuous).it implies that the bandwidth of the smooth window will be less than the rectangular window and the so the decay rate or the roll of rate of the side lobes will be much higher. The roll of rate for smooth window is 1/w2, which means as the frequency increase the decay rate will be much higher.

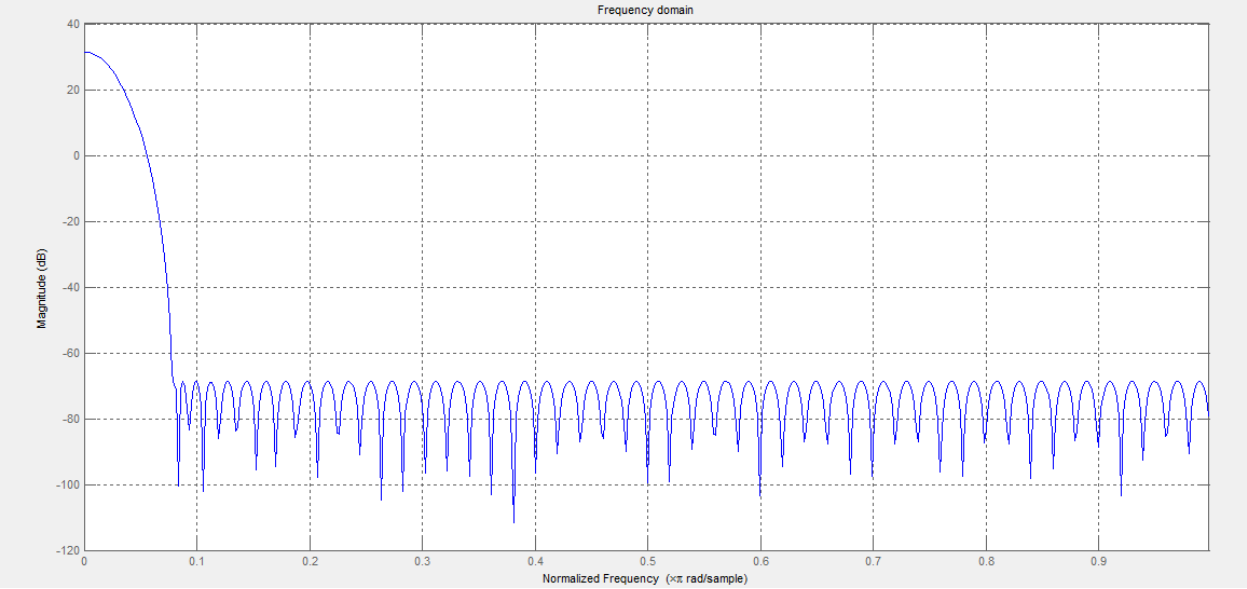


Figure 3.11: frequency domain of chebyshev window

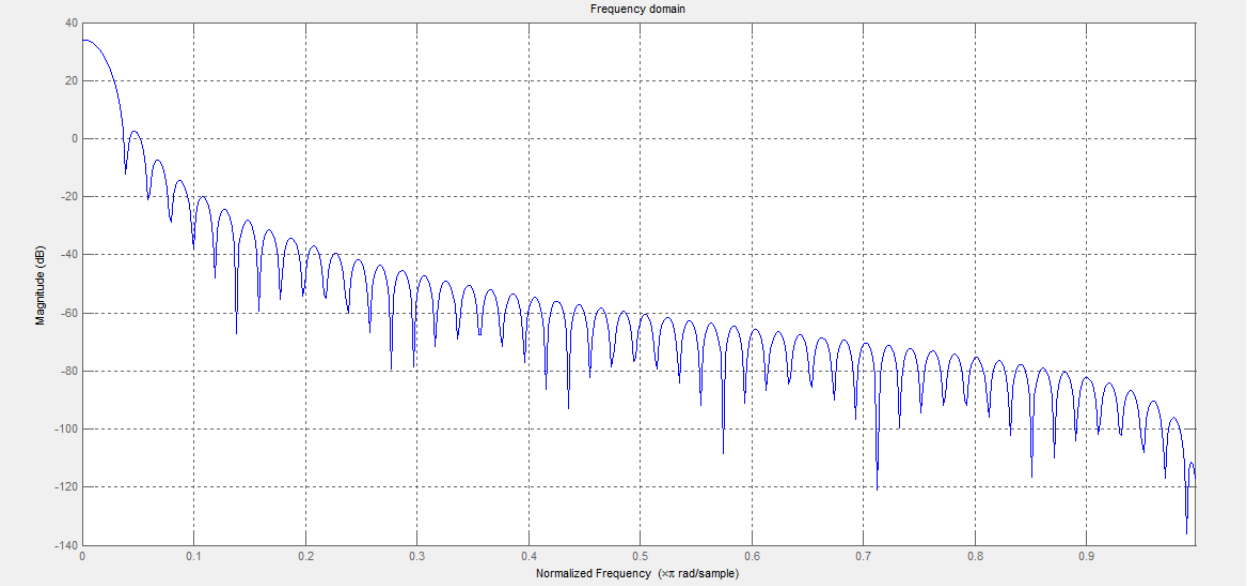


Figure 3.12: frequency domain of hanning window

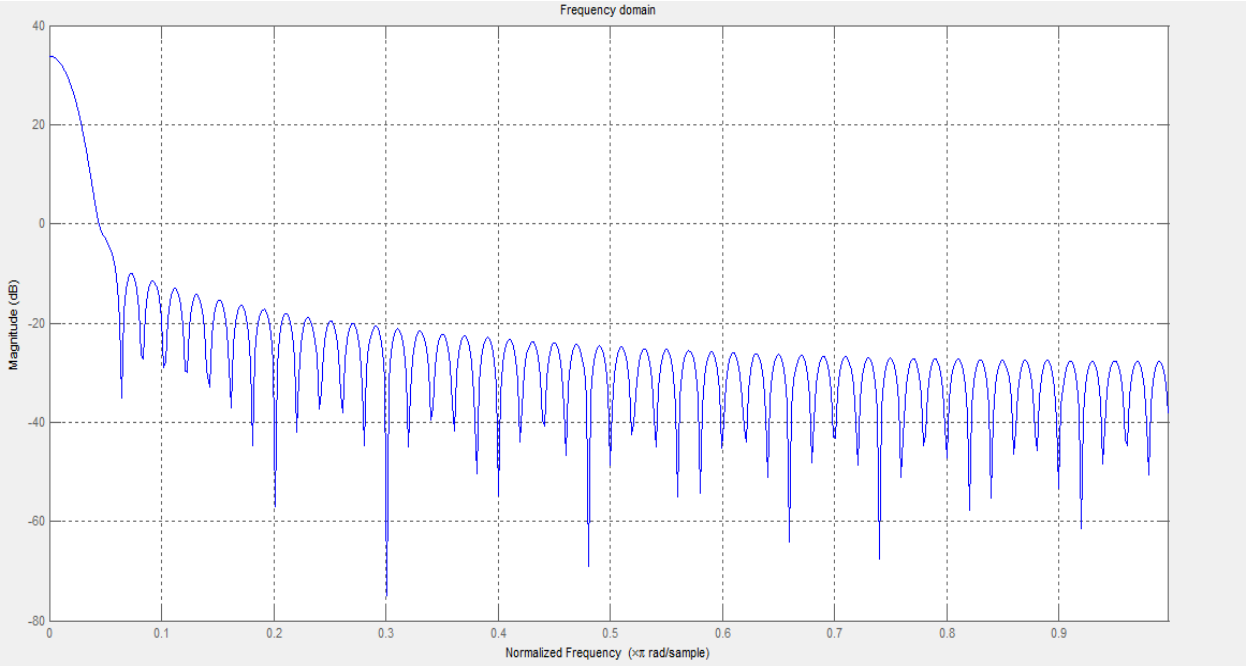


Figure 3.13: frequency domain of Gaussian window

As the decay rate is higher and the bandwidth in less than the rectangular window, the frequency interference or aliasing in the tail of the bandwidth will be less compared to the rectangular window. So a more good quality of strain image is expected in the output.

Some good windows are listed in a popular paper of Harris [30].

Formation of strain image using some smooth windows such as Chebbychev, Hanning, Gaussian are given in the figure below-

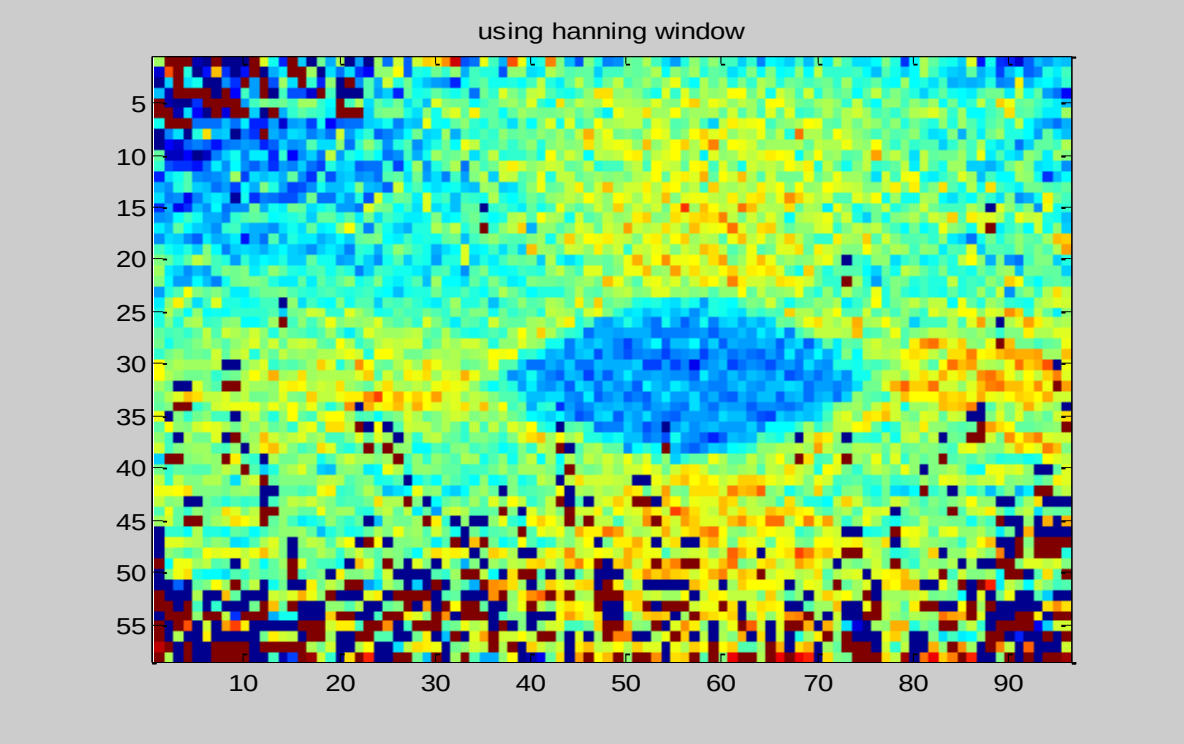


Figure 3.14: strain image using hanning window

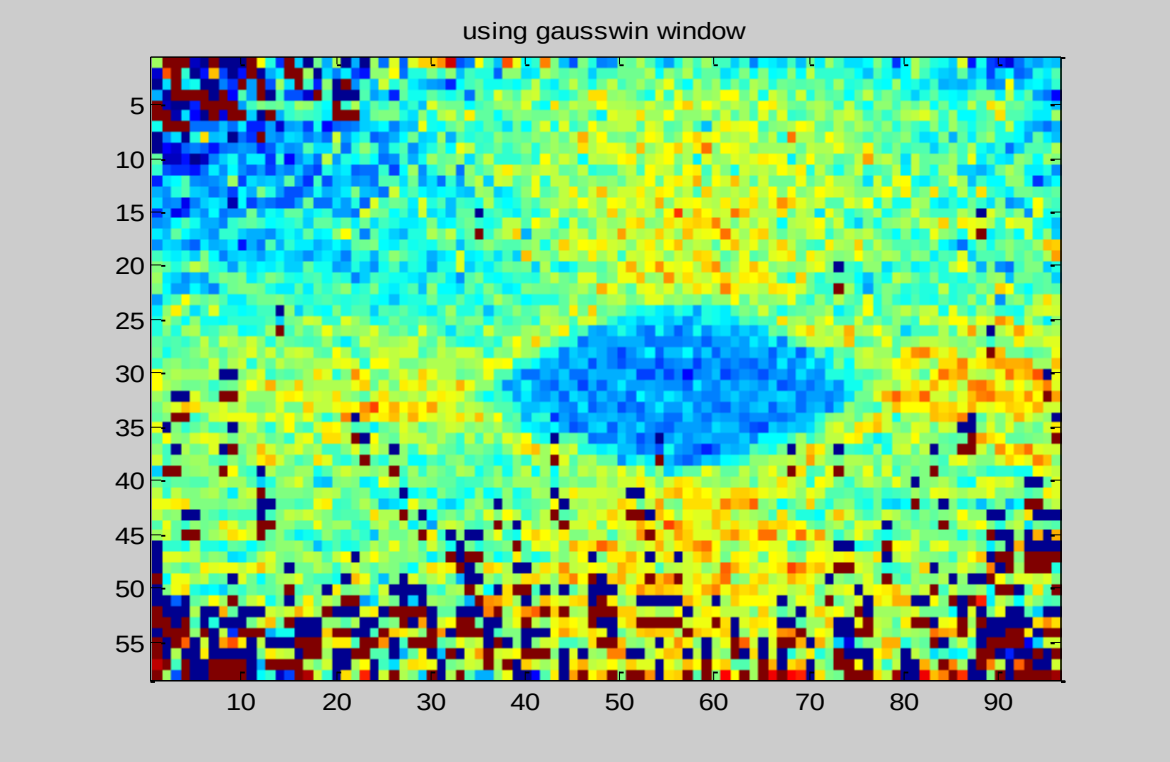


Figure 3.15: strain image using Gaussian window

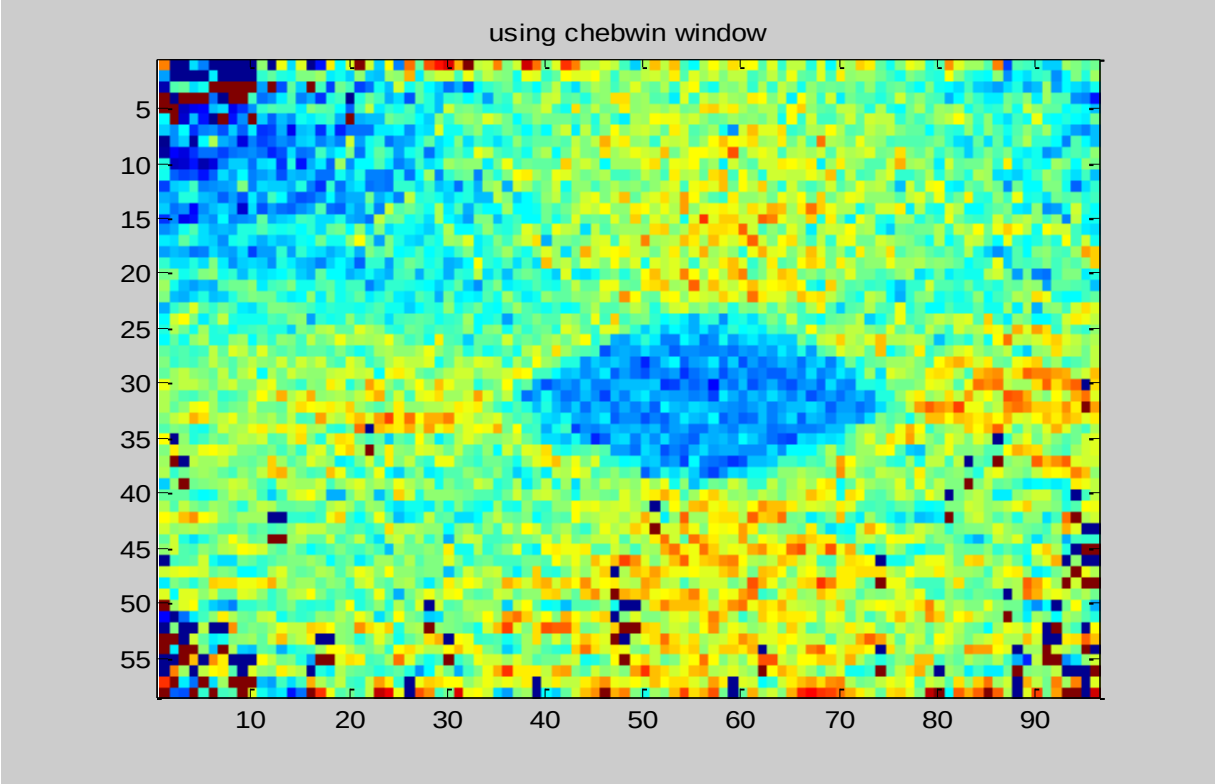


Figure 3.16: strain image using Chebyshev window

3.8 Comparison issues:

We want to compare different types of images that have been obtained using different methods. But there are some issues associated with it.

(i) Effective window size:

The rectangular window applies same weight to all the data samples. But smooth windows apply smaller weight near the edges. So, window size has to be increased for a smooth window for including same amount of data in the cross-correlation estimate. We take effective size of a smooth window such that it equals the area under the smooth window to the area under the rectangular window. For a 128-sample rectangular window the effective window size for different smooth windows are given below.

Table 5: Effective window size for different smooth windows corresponding to a rectangular window of 128 samples

Hamming window	239
Hanning window	256
Chebyshev window	347
Gaussian window	260

Area has been found by numerical integration using MATLAB.

(ii) Image registration:

In elastography image is obtained by using small temporal windows. For each window a strain value is obtained. Now this strain value is assigned to the middle point of the window. But as we have changed the effective size of the smooth windows, the strain values will be now be registered at different points for different windows. So comparison will not be fare. We have to make sure that all the images obtained using different methods corresponds to same region in space pixel by pixel. For that we have done the following.

1. Select which smooth windows are to be compared
2. Take rectangular window size to be N samples. In our case N=128.
3. Find which smooth window of interest has the largest effective window size. Say its size is n_1.
4. Take small temporal windows for other windows (including rectangular window) for correlation measurement as usual with an exception that each window has to be shifted by an amount n from its regular position. If n_2 is the effective window size of a particular window then shift n for that particular window is given by $n=(n_1-n_2)/2$. Doing this makes sure that for each pixel centers of the windows of all types of interest are located at the same position.

3.9 Summary

Data truncation is often desirable in diverse situation. In elastrography, image formation deals with huge amount of numerical calculation. To perform this huge numerical calculation and correlation between the RF frames data truncation is required. Using rectangular window for

data truncation does not results in good quality image because of large bandwidth with having side lobes which are slow decaying in nature. To avoid frequency interference or aliasing in the tail Smooth window such as Hanning, Gaussian, and Chebbychev can be used. Use of these windows will not totally eliminate the problem of sidelobes overlapping or aliasing but it can surely minimize the problem to some extent resulting in a better quality image than that is obtained from the typical rectangular window.

3.10 Conclusion

We have used en experimental Phantom for obtaining the images of 3.5, 3.14, 3.15, 3.16. The data was obtained at 20MHz and then up-sampled at 50MHz. The window shift was 64 samples and the rectangular window size was 128 samples. The images are obtained are obtained for 1% strain only.

For a rigorous study on our proposed concept we have to use higher strain and plot SNRe and CNRe. But shortage of time due to the time-consuming work of automatic parallel converter did not allow us to do this. We expect that this method should work fine at higher strains as well. Of course, this has to be verified in future.

Chapter 4

Conclusion and Possible extension of the work

We have presented a general framework for using parallel computation resources. This framework can be extended as follows.

1. The extension of this work can be the development of similar system for GPU environment. Though MATLAB GPU toolboxes are not suitable for this work, with the future versions of these toolboxes this might become a possible research work.

2. There could be other ways to achieve generalized parallelization. For example, commonly used MATLAB functions in elastography can be parallelized for GPU like the one done for MATLAB image processing toolbox [31]. Such a toolbox may look for the instructions which are the bottlenecks of the code and run the GPU version of those instructions itself.

3. In our system we parallelized the outermost loops. A system can be developed that checks all possible combination of parallel loops for minimum runtime when there are nested parallelizable for loops.

We also worked on application of smooth window functions briefly. This work can be extended by finding the optimum window functions at different strain conditions. Also, the image quality can be expressed in terms of the side-lobe attenuation factor using an adjustable window function like Kaiser Window. We have not shown the SNRe, CNRe, PSNR, MSSIM and other image performance parameters. This should be done to verify our qualitative proposition. Also, to complete our work images should be obtained for higher strains and also images should be obtained using FEM simulation data and in-vivo data.

This research work was motivated from the extensive research in elastography for early cancer detection. We have presented our effort to develop a generalized software for parallel computation and our study on applying window functions. Also we have presented the future extension of the work to make it more realistic and rigorous.

Appendix A

ALLOWED INDEXING SCHEME FOR SLICED VARIABLES

Sliced variables are one of the most important variables in the context of MATLAB parallel programming. As we are dealing with scanning the indexing schemes of variables, it is important to elaborate the allowed indexing schemes of sliced variables. These are given below and have been reproduced from [28].

Type of First-Level Indexing — the first level of indexing is either parentheses, (), or braces, {}.

Fixed Index Listing — within the first-level parenthesis or braces, the list of indices is the same for all occurrences of a given variable.

Form of Indexing — within the list of indices for the variable, exactly one index involves the loop variable.

Shape of Array — in assigning to a sliced variable, the right-hand side of the assignment is not [] or '' (these operators indicate deletion of elements).

Form of Indexing: Within the list of indices for a sliced variable, one of these indices is of the form i, i+k, i-k, k+i, or k-i, where i is the loop variable and k is a constant or a simple (non-indexed) variable; and every other index is a constant, a simple variable, colon, or end.

Bibliography

[1] J. Ophir, I. Céspedes, H. Ponnekanti, Y. Yazdi, X. Li, Elastography: A quantitative method for imaging the elasticity of biological tissues, Ultrasonic Imaging, Volume 13, Issue 2, April 1991, Pages 111-134, ISSN 0161-7346, 10.1016/0161-7346(91)90079-W.

[2] A. Webb, Introduction to Biomedical Imaging, John Wiley & Sons, Inc, 2003

[3] Anderson WAD, Pathology, CW Mosby Co., St Louis, 1953.

[4] J. Ophir, I. Cespedes, B. Garra, H. Ponnekanti, Y. Huang, N. Maklad, Elastography: Ultrasonic imaging of tissue strain and elastic modulus in vivo, European Journal of Ultrasound, Volume 3, Issue 1, January 1996, Pages 49-70, ISSN 0929-8266, 10.1016/0929-8266(95)00134-4.

[5] E. D. Landau and E. M. Lifshitz. Theory of elasticity. Pergamon Press, Oxford, third edition, 1986.

[6] J. E. Lindop, 2D and 3D elasticity imaging using freehand ultrasound, Ph.D. dissertation, University of Cambridge

[7] Y. C. Fung. Biomechanics: Mechanical properties of living tissues. Springer, 1993.

[8] A. Tarantola. Inverse Problem Theory and Methods for Model Parameter Estimation. Society for Industrial and Applied Mathematics, 1988.

[9] P. E. Barbone and J. C. Bamber. Quantitative elasticity imaging: what can and cannot eb inferred from strain images. Physics in Medicine and Biology, 47(12):2147-2164, June 2002.

[10] De Jong, P.G.M., Arts, T., Hoeks, A.P.G., and Reneman, R.S., Determination of tissue motion velocity by correlation interpolation of pulsed ultrasound echo signals, Ultrasonic Imaging 12, 84-98 (1990).

[11] Foster. S.G., Embree, M.P., and O'Brien, W.D., Flow velocity profile via time-domain correlation: error analysis and computer simulation, IEEE Trans. Ultrason. Ferroel. Freq. Control, Vol.37, No,2, 164-174,1990.

[12] Stark, P.A., Introductionto numerical analysis, Ch. 8 (Macmillan Publishing Co., New York, 1970).

[13] Boucher, R.E. and Hassab, J.C., Analysis of discrete implementation of generalized cross correlator, IEEE Trans. Acoust., Speech and Sig. Proc. ASSP-29, 609-611 (1981) .

[14] Alam, S.K.; Ophir, J.; Konofagou, E.E.; , "An adaptive strain estimator for elastography," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on* , vol.45, no.2, pp.461-472, March 1998.

[15] Zahiri-Azar, R.; Salcudean, S.E.; , "Motion Estimation in Ultrasound Images Using Time Domain Cross Correlation With Prior Estimates," *Biomedical Engineering, IEEE Transactions on* , vol.53, no.10, pp.1990-2000, Oct. 2006.

[16] Pesavento, A.; Perrey, C.; Krueger, M.; Ermert, H.; , "A time-efficient and accurate strain

estimation concept for ultrasonic elastography using iterative phase zero estimation," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on* , vol.46, no.5, pp.1057-1067, Sept. 1999.

[17] O'Donnell, M.; Skovoroda, A.R.; Shapo, B.M.; Emelianov, S.Y.; , "Internal displacement and strain imaging using ultrasonic speckle tracking," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on* , vol.41, no.3, pp.314-325, May 1994.

[18] S.K. Alam, Novel spline-based approach for robust strain estimation in elastography, Ultrason. Imaging, vol. 32, no.2 pp. 91-102,2010.

[19] Hussain, M.A.; Abu Anas, E.M.; Alam, S.K.; Lee, S.Y.; Hasan, Md.K.; , "Direct and gradient-based average strain estimation by using weighted nearest neighbor cross-correlation peaks," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on* , vol.59, no.8, pp.1713-1728, August 2012

[20] Parallel MATLAB doing it right

[21] Baida Zhang; Shuai Xu; Feng Zhang; Yuan Bi; Linqi Huang; , "Accelerating MatLab code using GPU: A review of tools and strategies," *Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on* , vol., no., pp.1875-1878, 8-10 Aug. 2011

[22] Pallwein, L., Mitterberger, M., Struve, P., Pinggera, G., Horninger, W., Bartsch, G., Aigner, F., Lorenz, A., Pedross, F. and Frauscher, F. (2007), Real-time elastography for detecting prostate cancer: preliminary experience. BJU International, 100: 42–46

[23] K. König, U. Scheipers, A. Pesavento, A. Lorenz, H. Ermert, T. Senge, Initial experience with real-time elastography guided biopsies of the prostate, The Journal of Urology, Volume 174, Issue 1, July 2005, Pages 115-117, ISSN 0022-5347, 10.1097/01.ju.0000162043.72294.4a.

[24] Thomas, A., Fischer, T., Frey, H., Ohlinger, R., Grunwald, S., Blohmer, J.-U., Winzer, K.-J., Weber, S., Kristiansen, G., Ebert, B. and Kümmel, S. (2006), Real-time elastography — an advanced method of ultrasound: first results in 108 patients with breast lesions. Ultrasound Obstet Gynecol, 28: 335–340. doi: 10.1002/uog.2823

[25] M. Friedrich-Rust, M.F. Ong, E. Herrmann, V. Dries, P. Samaras, S. Zeuzem, C. Sarrazin, Real-Time Elastography for Noninvasive Assessment of Liver Fibrosis in Chronic Viral Hepatitis, American Journal of Roentgenology, *AJR March 2007 188:758-764.*

[26] Pin Lu; Dan Shi; Liu, D.C.; , "Optimized GPU Framework for Ultrasound Strain Imaging," *Bioinformatics and Biomedical Engineering, (iCBBE) 2011 5th International Conference on* , vol., no., pp.1-4, 10-12 May 2011.

[27] N. Deshmukh, H. Rivaz, and E. Boctor, “GPU-based elasticity imaging algorithms”,Proc. Int. Conf. Imag. Comp. & Comp. Assist. Interven., 2009.

[28] MATLAB user Guide, http://www.mathworks.com/help/pdf_doc/distcomp/distcomp.pdf

[29] B.P. Lathi, Signal processing and linear systems, Oxford University Press

[30] Harris, F.J.; , "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE* , vol.66, no.1, pp. 51- 83, Jan. 1978

[31] J. Kong, M. Dimitrov, Y. Yang, J. Liyanage, L. Cao, J. Staple, M. Mantor, H. Zhou, Accelerating MATLAB Image Processing Toolbox functions on GPUs, Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units Pages 75-85.

[32] Rosenzweig, S.; Palmeri, M.; Nightingale, K.; , "GPU-based real-time small displacement estimation with ultrasound," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on* , vol.58, no.2, pp.399-405, February 2011.