Medical Ultrasound: A Study of Real-Time Three Dimensional Ultrasound Imaging

by

ALI SALEH KH. AL-MEJRAD

B.Sc.(Honors), M.Sc.

Thesis submitted for the degree of Doctor of Philosophy in the Medical Faculty, University of Edinburgh, March 1996.

To my Mother & my late Father Saleh

To my Wife & my Son Saleh

To my Brothers & Sisters

DECLARATION

I declare that the work contained in this thesis is my own and that the thesis has been composed by my self.

ALI S. KH. AL-MEJRAD

10 March 1996

ABSTRACT

Ultrasonic techniques are very widely applied in medicine. Real-time two dimensional imaging is a technology which is extremely well-suited to medical applications since it enables moving structures to be observed and rapid searching through tissue structures to be performed. Three-dimensional (3D) ultrasonic imaging techniques have been developed but to date there has been very limited success in the development of real-time versions.

The aim of this thesis is to study the feasibility of real-time 3D ultrasonic imaging to see if ways can be found to overcome the fundamental problem of sparsity of echo line data when a volume is scanned in real-time. The fundamental problem arises because conventional ultrasonic scanners have an upper limit of rate of generation of scan lines of around 10 kHz. The number of scan lines in each scanned volume is therefore low e.g. 2000 for a volume scan rate of 5 volumes per second. The aim of this thesis is to investigate whether or not modern electronic and image processing techniques can overcome this fundamental problem.

During the first phase of our study, a microcomputer based C-scan test-rig system including hardware and software has been constructed to investigate the effectiveness of real-time image processing in compensating for the fundamental sparsity of echo data. This was investigated initially since C-scans suffer from the same sparsity of echo data as 3D scans. After the promising results obtained from this system using a number of image processing techniques, a hand-held 3D ultrasound system including hardware and software based on one of the commercial scanners (Dynamic Imaging C2000) has been constructed to extend our study to 3D. A number of test objects in addition to volunteers were scanned to investigate the feasibility of real-time 3D ultrasound imaging.

Finally, a specification for real-time ultrasound imaging is discussed.

ACKNOWLEDGMENTS

Upon completion of this work, I would like to express my grateful thank to my supervisor Professor W. N. McDicken for his interest, advice, guidance, and encouragement during the course of this work . I also thank him again as Head of the Department of Medical Physics and Medical Engineering for making the facilities available to me. I also thank my second supervisor Dr. H. Brash for advice and performing the proof-reading of this thesis. I am particularly indebted to Mr. T. Anderson for his excellent suggestions and electronics expertise during the project. I am grateful to Mr. D. Cameron from Dynamic Imaging Ltd. who suggested that the company's scanner be used in my project and for his suggestion regarding the scanner's design. I am grateful to Mr. R. Borthwick for his skilled assistance during mechanical construction of the different types of test scanner.

My special thanks which is not enough goes to my friend Mr. A. Soares for his friendship and for giving me the chance to share with him his excellent wide experience in electronics and computing. Also I thank him for his support and kindness in the crisis which I faced during the project.

I wish to thank Mr. Martin Connell who has long experience in medical imaging for his generous help in computing and especially in 3D scan conversion and image processing.

I would like to thank in particular Dr. Shaofeng Li with whom I shared the movable student lab for most of the duration of this project, for his good friendship and help, and also Dr. D. Potter for his friendship and help. I thank Dr. P. Hoskins for his help and support. I thank also Dr. S. Pye for his help. I thank also Dr. C. Moran for her help and kindness. I also thank Dr. T. Spencer for his helpful discussions. I also thank my good friend Miss M. Aspradakis for her friendship, kindness and encouragement.

I would like to express my grateful thanks to the Department's secretary Mrs I. Craig for her great help in making everything easy and recently for typing my thesis draft.

I would like to thank the staff of the Department of Medical Physics and Medical Engineering for their help and support, in particular Dr. M. Glabus, Mr. I. Macleod, Mr. W. Ellis, Mr. J. Campbell, Mr. I. Peaston, Mr J. Fortune.

I would to express my gratitude and thanks to my government represented by King Saud University that gave me the financial support. Thanks are also due to Dr. M. H. Al-Turaiki for his constant help and advice. I also I wish to thank Mr. A. Al-Nasser in the Culture Office in London for his support and help.

I would like to thank my wife who fills a big gap in my life and makes it happy and also for my new baby who changed my life to be happy after the crisis I faced four months ago when my father passed away.

Finally I would like to thank all my friends in Edinburgh with whom I shared a good time and especially my good friend Mr. M. Al-Saddiq.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1. Basics

1.1.1. General

The ability to see with sound has long been an intriguing concept (Ahmed et al., 1980). In the natural world certain animals, such as bats and dolphins can do this readily by converting sound to images and use them as an aid to navigation. The human species is not so endowed by nature. However, this lack of natural ability has been overcome by developing an appropriate technology. Many applications for imaging with sound are found after developing the appropriate technology like non-destructive testing, sonar and geographical exploration. By now a wide variety of system concepts for acoustic imaging exist and are applied for medical diagnosis to image the internal structure of the body and motion of the internal organs after much development of pulse-echo techniques first used in sonar systems of world war 1.

1.1.2. Medical Diagnostic Ultrasound

In health care nowadays many modalities in medical imaging like traditional x-ray, radioisotope, ultrasonic imaging, themography, computed tomography (CT), magnetic resonance imaging (MRI) and digital radiography are found. Ultrasound is one of the most important modalities for medical diagnosis. Since the late 1950's (Wild and Reid , 1952; Baum and Greenwood, 1958) when ultrasound was used to image internal organs, diagnostic ultrasound has expanded quickly. Diagnostic ultrasonic instruments have developed from basic instruments to the present complex medical instruments. A number of techniques in ultrasound are established and other new ones are being introduced in many clinical applications: abdominal scanning, obstetric scanning, cardiac scanning, small parts scanning, endosonography and intra-operative scanning. Many clinical applications of ultrasound are described in an excellent series of texts "Clinics in Diagnostic

Ultrasound" published by Churchill Livingstone. Ultrasound is considered as the favourite diagnostic approach with comparison to the other approaches (Margulis and Shea, 1986). The reason is because of the following :

1. Soft-tissue imaging: Ultrasound achieves this more readily than X-ray imaging.
2. Non-invasive: Most examinations in ultrasound are non-invasive so repeated examinations can be made.
3. Safety: Until now, no confirmed harmful effects have been found with diagnostic instruments. Therefore is it the only technique used to scan the pregnant woman.
4. Cheap cost: Ultrasound is less expensive compared to other modalities like computerised tomography (CT) and magnetic resonance imaging (MRI) by a factor of 20 concerning the price of equipment and running cost.
5. Patient comfort: This is because of the fast speed of examination using real-time scanning. Also the fast speed enables many examinations in less time.
6. Real-Time: This feature is useful because it enables the examination of moving structures like heart and foetus and also permits the investigator to search through tissue structures.

The above are good characteristics but there are still other features to be added like the three-dimensional mode. McDicken (1991) describes modern ultrasonic instruments from the user point of view and lists its advantages and attractions in a text on "Diagnostic Ultrasonics: Principles and use of instruments". Other texts on the applications of diagnostic ultrasound are by Hussey(1985), Lerski (1988), Fish(1990). The physics of ultrasound is covered by many texts e.g. Wells (1977) and Hill (1986).

Most ultrasonic images have been obtained by making use of echoes produced by reflection or scattering of ultrasound within the body. Recently there has been a large growth in the use of the Doppler effect to acquire information and images from moving blood and tissue (Evans et al., 1989; McDonald, 1974; Barber et al.,1974; Fish, 1972; Omoto, 1987; Reid and Spencer, 1972; Taylor and Strandness, 1990; Hoskins, 1990; Taylor et al., 1988). Doppler techniques are only considered briefly in this thesis.

1.1.3. Ultrasonic Imaging

An ultrasound image is built using a pulse-echo technique. In this technique sound pulses transmitted into the body from a hand held probe are scattered and reflected by tissue and two dimensional images are formed using the echoes returning to the probe. Ultrasound is sound whose frequencies extend beyond the range of human hearing i.e. frequencies above 20 kHz and specifically between 1 MHz and 15 MHz in medical applications. In this frequency range, the wavelength of sound in soft tissue is between 1.5 mm and 0.1mm. Various techniques have been used to display echo-signal data in graphic and image formats; called A-mode, B-mode and M-mode displays. These basic types of instrumentation are as follows:

- Amplitude A-mode. Information is shown on an oscilloscope in the form of echo amplitude plotted against time (depth). Figure 1.1 shows a block diagram of an A-scan instrument.

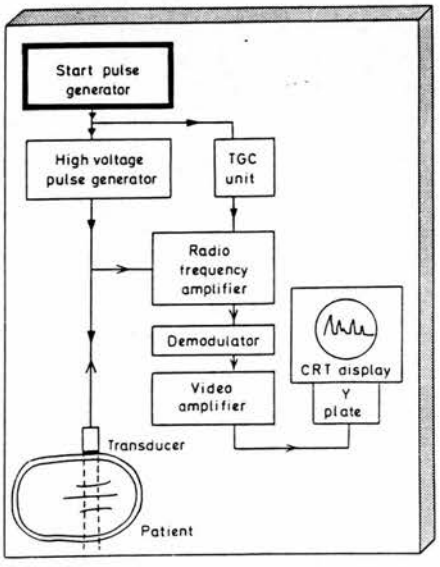


Figure 1.1 : Block diagram of an A-scan instrument .

- Brightness B-mode. A large proportion of ultrasonic examinations are performed using the B-scan, which was first described by Wild and Reid (1952). The brightness is related to the echo amplitude. The position and direction of the time base across the display are linked to the position and direction of the ultrasonic beam across the patient. If many B-scans from the same patient are stored, then a two dimensional display of echo producing interfaces in the plane of the scans are produced. Figure 1.2 shows a block diagram of the basic units in a real-time B-scanner.

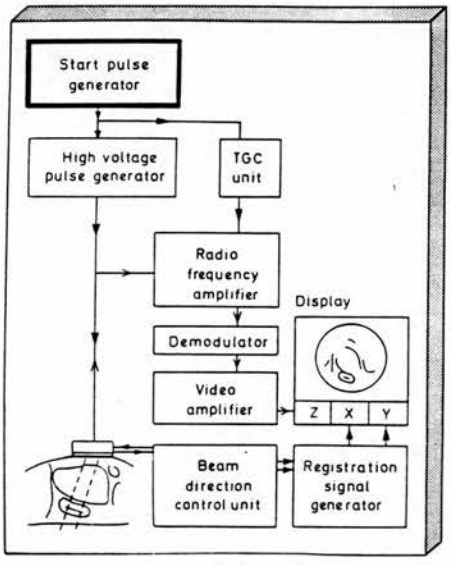


Figure 1.2 : A block diagram of a basic real-time B-scanner.

- M-mode (Motion mode). An echo range displacement versus time display is added to the basic instruments. An example of its use is in the recording and measuring of heart valve displacement and velocities. Figure 1.3 shows a block diagram of the functional units of an M-scan system.

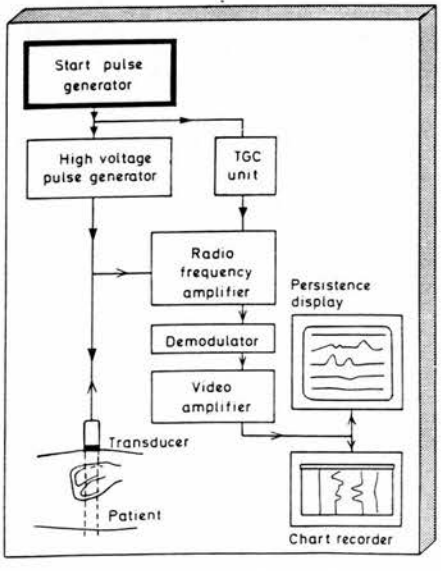


Figure 1.3 : Block diagram of the functional units of an M-scan system .

1.1.3.1. Real-Time Imaging

Early imaging techniques required the transducer to be scanned manually across the surface of the body. The resulting image was static. Rapidly moving tissue within the image area introduced artefacts. In order to adequately observe tissue motion, for example in the heart, an image must be produced approximately 30 or more times per second. An equally important aspect of real-time imaging is the ability to rapidly search through large volumes of tissue to identify the optimum image plane. There are three main methods of achieving real-time images.

1. Rapid motion of the transducer previously used in static image production by a suitably driven mechanical device - mechanical real-time scanner;

2. An array of small transducers, groups of which create lines in the image - linear array scanner;

3. An array of small transducers, the signals from which are variably phased relative to one another to steer the beam through a sector of tissue - phased array scanner.

The development of real-time scanners has been described by McDicken (1991) and others. Linear arrays were pioneered by Bom et al. (1971), and Somer (1968) is credited with much of the early work on phased array transducers.

The last important point related to real-time is the consequences of limited image forming time. A preferred upper limit of 40 ms for the image forming time, 25 frames per second (f/s), combined with the need to wait about 200 μs for the echoes to return from each transmission (i.e. scan line) means that the number of lines in each image is limited to about 200. This limits, in turn, the width of the scanned plane, since scan lines must be reasonably close together, (about 4 lines per beamwidth is about optimum). The width of the 'field of view' for an abdominal scanner, will therefore be about 50 beamwidths if it is to be adequately sampled. In principle, scanners for more superficial organs could use higher PRF and thus more beamwidths across the field of view. In practice, PRF is not often changed when the transducer frequency is changed and thus field of view generally decreases when a transducer of high frequency is used with a given machine. In general,

therefore, there is competition between the image qualities (image rate, field of view width and line density). This is what is called the 3-way compromise when considering real-time ultrasound imaging.

1.1.3.2. Image Construction

Image construction is started from a digital memory, which stores the amplitude of each echo and the co-ordinates of its point of origin (calculated from the measured probe co-ordinates x, y and θ, and an assumed speed of sound - the system velocity). For storage purposes the image is divided into a matrix (e.g. 512 x 512) of possible echo sites, or pixels. Each pixel has its own location on the memory, with a specific address. Each location stores a number that is proportional to the amplitude of the echo in that pixel. Then this pixel is displayed on a monitor's screen. To store each echo there are different 'sampling algorithms' by which a value for the echo amplitude is chosen for each pixel. These algorithms are point sampling, maximum sampling and maximum deviation. The last algorithm gives better axial resolution and edge-enhancement at the cost of noise. The other algorithms are 'update algorithms' in which the previously stored value for a pixel is modified by the new value. Common examples of these algorithms are last value or survey mode and compound mode.

1.1.3.3. Image Quality

One of the very important features in the ultrasound scanner is the image quality because a good quality image helps the success of an ultrasonic examination. A lot of work was done related to the improvement of image quality during the last 20 years, benefiting from the advanced technology in electronics and computing. Bistable displays have been replaced by grey scale which allow the visualisation of low level signals from parenchyma to be seen as well as large signals from strongly reflecting surfaces. This feature is very important from the diagnostic point of view (Hussey, 1985). Another development is the replacement of static scanning by real-time which allows the movement of internal structures to be studied instead of the image being distorted by this movement. Digital scan converters have been introduced which has improved the accuracy, reliability and processing of the displayed information (Ophir and Maklad, 1979; Robinson and Knight, 1981). Ultrasound beams can be manipulated to achieve dynamic focusing which has

improved spatial resolution (Halliwell, 1987). Adaptive time gain compensation was introduced which enables the scanner to use echo information to set up the gain compensation (Pye et al, 1988). This technique, though not widely used, when compared to manual time gain compensation resulted in improved images. Finally, a solution to reduce noise in ultrasonic images which is called speckle and represents a major source of image quality degradation was introduced (Bamber and Daft, 1986 ; Kerr et al, 1986; Loupas et al, 1994). This development improved image quality due to increased contrast and spatial resolution, enhanced viewer performance, reduced image variability and easier image interpretation. The next development to be introduced is development of other modes of scanning like 3D imaging. This development can benefit from the advanced technology in electronics, computing and the common field in all medical imaging systems namely 'image processing'.

1.2. Image Processing

Image processing is the science, though some call it a black art, of manipulating and using images to fulfil some purpose or benefit, such as extracting data automatically or assisting in the human interpretation of the information on view. Image processing can be regarded as a blend of statistics, applied and pure mathematics, and an increasing dose of psychology both in terms of human computer interaction and in attempts to emulate the recognition and analysis capabilities of the human brain. Initially, work was stimulated by the space research programme in the 1960's (Frieden, 1979). Since then, this area has become an interdisciplinary subject in such fields as engineering, robotics, computer science, geography physics, biology and medicine. Contributions in this field involve three areas: development of new algorithms, development of dedicated hardware for image processing and establishment of new applications.

Digital image processing is an essential part of many medical imaging techniques. In medical ultrasound imaging, digital image processing is becoming more suited to improving the quality of ultrasonic images. The general trend nowadays is towards digital techniques because of their flexibility, accuracy, reliability and good performance. Also the decreasing cost and increasing speed of digital signal processing devices makes them compatible with the low-cost and real-time nature of ultrasound. Hence, digital image processing will play an important role in the improvement of ultrasonic image quality.

Finally, the digital image is regarded as a two-dimensional function, where the value of the function f(x,y) at spatial co-ordinates (x,y) in the x-y plane defines a measure of light intensity or brightness or grey level at that point. The digital image is considered as an array of N x N pixels (e.g. 512 x 512 with 256 grey levels).

1.3. Overall Object and Structure of Thesis

The aim of this thesis is to study the feasibility of real-time 3D ultrasonic imaging and to see if ways can be found to overcome the fundamental problem of sparsity of echo line data when a volume is scanned in real-time. The fundamental problem arises because conventional ultrasonic scanners have an upper limit of rate of generation of scan lines of around 10 kHz. The number of scan lines in each scanned volume is therefore low e.g. 2000 for a volume scan rate of 5 volumes per second (vol/s). The aim of this thesis is to investigate whether or not modern electronic and image processing techniques can overcome this fundamental problem.

This thesis is divided into ten chapters. Chapter 1 introduces the basics of medical imaging and ultrasonic imaging and how it can be used in medical diagnosis. Image construction and quality of images are the next subjects to be discussed, followed by image processing which is very important to improve the quality of images.

Constant-depth ultrasound scanning(C-scanning) is the subject in Chapter 2. This chapter introduces the principles of this technique and compares it to the well-known B-scanning. A review of C-scanning systems is covered here to identify the advantages and disadvantages of the various systems. Hence, constant-depth system design specifications were derived. A simple C-scanning system was investigated initially since C-scans suffer from the same sparsity of echo data as 3D scans. Indeed the C-scan is an example of an oblique scan image extracted from a 3D volume scan.

In Chapter 3 the details of constant-depth system design and construction are discussed. This involves the computer hardware, mechanical design and electronic design (analogue and digital). Also in this chapter a study of transducer characteristics is discussed.

Chapter 4 discusses the design and specification of the constant-depth system's software. This includes control and data acquisition programs. Also, image processing techniques and software implementation are included in this chapter. This chapter considers how to display images using the microcomputer and how to get a good hardcopy of these images. Finally, this chapter discusses the diagnostic programs which are used during and after the construction of the system.

Experimental results obtained using the constant-depth scanning system are given in Chapter 5. This chapter presents the study of the system's resolution using a resolution phantom. Comparisons between acquired images before and after processing and also between constant-depth scans and B-scans are presented in this chapter.

The previous chapters represent the first and second phases of the project. The chapters 6,7,8 and 9 represent the third and last phase of the project.

In Chapter 6, overall design considerations of three-dimensional ultrasound imaging are discussed. Then a review of reported and commercial 3D ultrasound imaging systems is presented to study their advantages and disadvantages.

The design and construction of a 3D ultrasound test system's hardware are presented in Chapter 7. This chapter is divided into five main sections. The first section discusses the 3D scanner design in detail, mechanical and electronic design. The second section presents an overview of the commercial B-scanner incorporated into a 3D system. The image processor board is presented in the third section followed by the parallel input/output (PIO) digital board in the fourth section. The last section presents the fast data capture unit which contains four boards.

The system described in Chapter 7 is a computer-based system. Therefore, software is required to control the operation of the system. In Chapter 8 the system software is discussed in detail. This includes the data acquisition program; data and image display program; reconstruction and image processing program. Investigation of real-time image processing is covered in this chapter. Programs to test the different parts like the programmable I/O digital card and fast data capture board are presented in this chapter. Also, assembly programs to test the image processor board are described.

In Chapter 9 experimental results, image processing and discussion of the three-dimensional ultrasound imaging system are presented.

Finally, Chapter 10 discusses further future developments and possible improvement of real-time ultrasonic 3D scanners.

Circuit diagrams and program listings are presented in appendices A and B.

CHAPTER 2

OVERALL DESIGN CONSIDERATIONS OF CONSTANT DEPTH ULTRASOUND IMAGING

CHAPTER 2

OVERALL DESIGN CONSIDERATIONS OF CONSTANT DEPTH ULTRASOUND IMAGING

2.1. Introduction

A lot of research has already been published in the field of real-time two dimensional ultrasound imaging or what is called Real-Time B-scanning. However, up to the present only a limited amount of research work has been published on Constant Depth Scanning or what is called C-scanning. C-scanning is studied in this chapter since a C-scan image is an example of one which can be extracted from a block of 3D echo data and which is in a plane not accessible to B-scanning. If a C-scan is performed in real-time, due to the limitation on PRF it suffers from the same sparsity of echo data as does real-time 3D scanning. It is one of the aims of this thesis to investigate the effectiveness of real-time image processing in compensating for this fundamental sparsity of echo data. As a first step in such an investigation a C-scan test-rig has been constructed since it is a relatively simple system.

This chapter presents the principles of C-scanning and compares it to real-time B-scanning. Then a summary of the research work that has been done in the field of C-scan systems is presented. Finally, C-scan system design specifications are presented.

2.2. C-Scan Principles and its Value

C-scanning is the technique by which a cross-sectional image at a specific depth within the object is obtained. This is obtained by an interrogating ultrasound beam that is perpendicular to the image plane as shown in Figure 2.1. A range gate is used to select and display only echoes originating from a specified narrow depth interval.

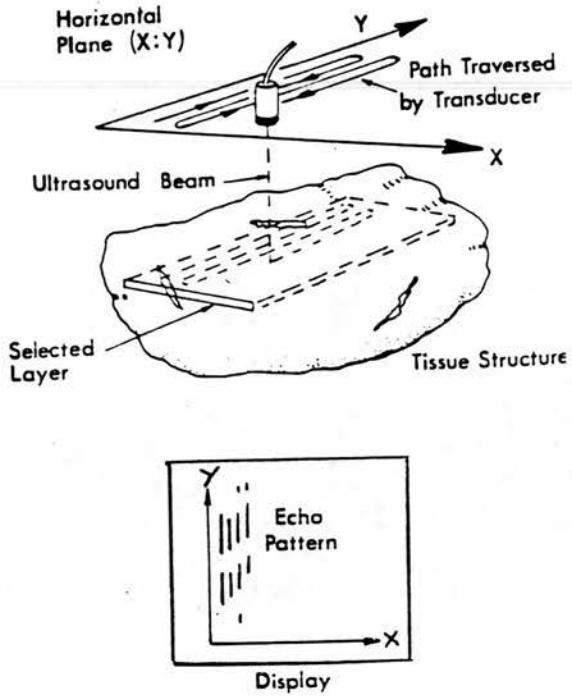


Figure 2.1 : The C-scanning principles.

The principal advantage of a C-Scan is that a strongly focused beam may be used to improve lateral resolution. The large beam width proximal and distal to the focus of a strongly focused beam is not a problem since the range gate is set to the focus. There may also be interpretative advantages in imaging a plane that lies parallel to the body surface as do most C-scans in practice as will be clear in the following sections. It should also be noted that there is no need for swept-gain (T.G.C.) when performing a C-scan.

The principal disadvantage is the long image forming time of all C-scan systems built to date as will be discussed in section 2.4. The reason for this is that the transducer must be moved in a scanning raster, stopping at each sampling point long enough to transmit a pulse and receive an echo from the depth of interest. For example, if say 200 μs is allowed at each of 200 by 200 scanning points, the total scan time would be 8 seconds. A further disadvantage is that a water-bath or immersion technique is usually needed firstly to accommodate the large aperture transducer (strong focusing) and secondly to permit mechanical raster scanning. These disadvantages can be overcome by manipulating the transducer in a way that performs scanning faster and in contact with the patient. "Judgement of the value of C-scanning , at this stage is perhaps premature as the technique has not been applied extensively" (McDicken, 1991).

2.3. Comparison between B-Scan and C-Scan

At present, most ultrasonic scanners employ what is called "Real-Time B-scanning" to obtain images. This method involves sampling at several points along an ultrasonic beam and converting these echo amplitude signals into a grey scale which can then be visualised on a monitor as a line. The beam is then displaced laterally and further sampling occurs, building up another line. Continuing this procedure results in an ultrasonic image built up in a plane described by the axis of the beam, as shown in Figure 2.2.

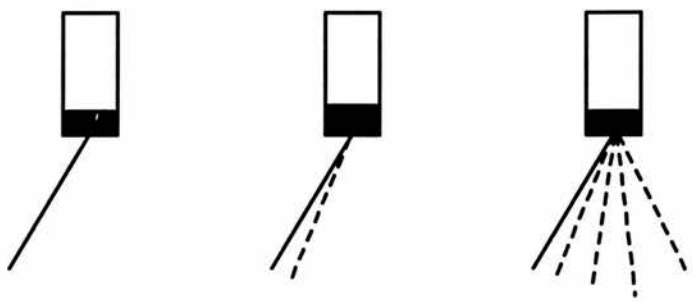


Figure 2.2 : Steps of image construction using B-scan.

Sampling occurs by transmitting the pulsed beam in the required direction, and then examining the echoes after certain time intervals corresponding to the time required for the pulse to reach the sampling points and be reflected back to the transducer. The time is calculated using the formula:

$$t = 2\,d/v \qquad (2.1)$$

where :

t = the total time

d = the distance from the face of transducer to the sampling point

v = the velocity of sound in the medium

If the maximum depth required is around 20 cm, then the time delay for one line using Equation 2.1 gives 267 micro-seconds (μ sec).

If there are around 150 lines per sector scan, the minimum time required for a sector is 40 milli-seconds (msec) after using Equation 2.1. This would allow a frame rate of 25 f/s. However, delays

caused by mechanical movement of the transducer reduces this, and there are sometimes variations on depth and number of lines per frame.

The problem with this type of imaging technique is that the method examines samples at variable depth, thus many points will occur out of the focus of a transducer. The transducers used try to produce a uniform focus along the length of the beam which in turn improves the lateral resolution. In sector scan, because the beam is swept out from the transducer source, at the points of greatest depth the distances between the scan lines become relatively large, reducing the lateral resolution of the image.

The farther the beam travels into a medium then the greater the amount of attenuation it encounters. This makes it necessary to employ what is called 'Time Gain Compensation', whereby allowance is made for the decrease in amplitude of the reflected pulses in proportion to their greater depth of origin.

Another restriction of real-time B-scanning is the fact it restricts the image plane to that described by the axis of the beam, which can be quite restrictive in clinical use. One way around this, is to construct an image via what is called 'C-scanning'.

This involves imaging on a plane of constant depth below the transducer and perpendicular to the axis of the beam (see section 2.2 for more details). Since with C-scanning we are dealing with a constant depth, then a fixed focus transducer can be used giving good lateral resolution. However, it is usually required to be able to image at a variable plane depth. This could be done using an annular array with a variable focus, preserving a relatively good lateral resolution. As stated before, taking samples from a greater depth increases the effect of attenuation. However, this effect is more or less uniform throughout the image plane and can therefore be more easily compensated for, than in B-scanning.

The thickness of the image plane in C-scanning is dependent upon the axial resolution of the system. This is determined by the effective duration of the echo from a point target and thus, in turn, by the effective duration of the transmitted pulse.

C-scans would also be more uniform that B-scans, due to their constant depth and separation between sample points. This makes it less complex to process images using for example filtering or smoothing.

Other advantages in C-scanning with comparison to B-scanning are that: first, the image is obtained in a plane similar to that of conventional x-ray, this perhaps making it easier to interpret; second, the shape and angle of the plane can easily be varied by simply altering the time delay used in analysing the echoes for each raster line. If a variable focus transducer is used then the focus can be altered in accordance with the time delay to keep good resolution.

One of the main disadvantages with C-scanning with comparison to B-scanning is that it takes substantially longer to construct an image than it does with a real-time B-scanner. For the B-scanner above, the time delay required for a pulse to travel 20 cm and its echoes to reach the transducer is about 270 μsec. If the C-scan image is made up of 100 x 100 points then the minimum time necessary for the image area to be scanned is 3 sec (calculated using Equation 2.2). However, mechanical movement of the transducer in a raster pattern over the area substantially increases the time taken to construct one frame.

2.4. Review of C-scan Systems

Since the early work of Wild and Reid (1957) and Donald et al (1958), the ultrasound techniques started to develop and other scanning arrangements were seen to be possible. In particular, the scanning arrangement to define a scan plane at a constant distance from the face of the transducer which is made to move over the area of interest.

This C-scanning arrangement or what is sometimes called a 'constant depth scanner' is achieved by many approaches. The simplest approach is to move the transducer in an x-y raster scan in a plane perpendicular to the ultrasound beam. This approach has been discussed early by several authors (Thurstone et al., 1965; Fry et al., 1968) but has not been applied clinically. Also Ito et al. (1979) used this approach, but with 128x128 sample points in a 10 cm x 10 cm area, imaging at 15 cm depth. The frame rate was 1 per 480 seconds, mainly due to the mechanical design of the system. Restori (1977), used a similar drive mechanism, but with 80 x 80 sample points in a 4 cm x 4 cm

area, imaging at a depth of 4.6 cm. This time, the frame took 11 seconds to complete. A high resolution ultrasonic Cartesian co-ordinate scanning technique which would lend itself to routine clinical application has been described by McCready and Hill in 1971. In this constant depth scanner a transducer having a beam width of 2 mm at the focus has been connected it to a small digital computer which enabled scans to be recorded rapidly and displayed later (Millan, 1972). Because this approach has serious limitations in clinical applications, particularly in relation to scanning speed and acoustic contact, another approach was suggested by Brown (1967) to solve both problems which employs a spiral scanning pattern. Mezrich in 1977 tried two different systems by which he reduced the time required to construct a frame. His first method is based on a simple mechanical system. His second method is based on Risley prisms. Both methods used 1.6 MHz ultrasonic frequency and the time to form the image was about 6 seconds.

A new scanning mode, called a flexible or F mode, has been devised which produces real-time ultrasonic diagnostic images of free surface planes in the human body (Ito et al., 1980). C-mode images consisting of 128x114 picture elements can also be obtained in only 4 seconds by using the electro-scanning probe. Another possibility for real-time C-scanning by electronically controlled arrays and CW beams has been introduced and theoretically explored by Whittingham (1982). Using this arrangement a good lateral resolution was obtained but with relatively poor axial resolution. Oates and Whittingham (1984) described an in-vitro experimental study of the above C-scanning technique proposed by Whittingham (1982). It has been shown that C-scan images can be obtained at real-time rate but with limited resolution and poor contrast. High resolution ultrasonic transmission imaging with a 2D receiver array has been developed which allows C-mode images to be shown with a frame rate of 25 Hz.

After this work, the researchers start to concentrate on 3D ultrasound from which C-scanning is a special case, Ylitalo and co-workers (1986). In this work, a 3D ultrasound C-scan imaging method based on holographic reconstruction has been developed. A full review of 3D ultrasound imaging will be presented later in the second part of this thesis in chapter 6. Before leaving the discussion of the C-scanning system, it is worth mentioning that C-scanning systems have been developed for non-destructive testing (Jonas, 1972; Warren, 1981; 1984; Toda et al., 1982; Djordjevic, 1983; Holton and Djordjevic, 1985; Bailey and Paglione, 1986).

Finally, the C-scanning technique can offer some advantages in addition to the advantages of conventional real-time B-scanning, but one major difficulty is its very low frame rate. However, with a frame rate of 2-4 f/s it can still be a useful diagnostic tool. Some clinical applications are in ophthalmology (Restori and Wright, 1977; Restori, 1979; 1985), breast scanning (Pluygers et al., 1984) and echoenchephalography (White et al., 1966; 1968). Also the C-scanning technique is very useful tool in industrial applications (Gordon et al., 1993).

2.5. C-Scan System Design Specification

The idea from the previous section will now be developed towards a practical system which can be used to study the feasibility of real-time C-scanning.

The specification can be split into two parts. Firstly the user specification is considered, this is essentially non-technical and lists the requirements of the physicians. The main requirements are that the system provides the user with a user-friendly interface i.e. allowing for ergonomic considerations and the system shall guarantee stability and accuracy under all defined operational conditions.

Secondly, a technical specification is required. This shall include definition of reliability requirements, fail-safes and override modes, system performance, alarm processing, data storage and electrical characteristics.

An outline of some of the hardware design factors and considerations are listed below:-

i. User interface
ii. Power supply requirements
iii. Power and signal earthing
iv. Electronic components and construction
v. Reliability
vi. Override facilities and fail-safe modes
vii. Cost

viii. Maintainability

ix. Manufacturing aspects

x. Ease of modification

xi. Quality

xii. Signal segregation and isolation

In consideration of the above the system requires a mechanical scanner with two motors in x and y directions and driving facility, data acquisition system and A-scan ultrasound system. Finally a computer is required to control the above.

CHAPTER 3

DESIGN AND CONSTRUCTION OF C-SCAN SYSTEM: HARDWARE

CHAPTER 3

DESIGN AND CONSTRUCTION OF C-SCAN SYSTEM: HARDWARE

3.1. Introduction

This chapter describes the design and construction of the C-scan system. The system has been used with a Transmitter/Receiver Ultrasound unit which works on the same principles of ultrasound scanners which are described in many texts. (Fish, 1990; McDicken, 1991).

It was decided to base the C-scan system around a commercially available microcomputer and to design and build the interface electronics required to link the microcomputer to the ultrasound system. This approach has several advantages. A microcomputer can be used to load the different parameters, to accomplish the control algorithms, to acquire the ultrasound data, to process acquired data and finally to display the ultrasound images.

A microcomputer offers great flexibility, but at the cost of reduced speed. The control centre of a microcomputer is a microprocessor, and microprocessors carry out tasks more slowly than dedicated hardware. The interface electronics was thus designed to accommodate this reduced processing speed.

Apart from the ultrasound system which is mostly analogue, the hardware of the C-scan system is digital. This approach has many advantages. The resulting instrument is accurate, reliable , smaller and output data can be processed by the computer.

This chapter describes the different components of the system under the following headings:

- A microcomputer and programmable Input/Output Card (section 3.2 to 3.3).
- A mechanical design and driving unit (section 3.4).

- Ultrasound system and transducer (sections 3.5 to 3.6).
- A detailed description of drive control and data sampling unit (section 3.7).

A block diagram of a C-scanning imaging system is shown in Figure 3.1.

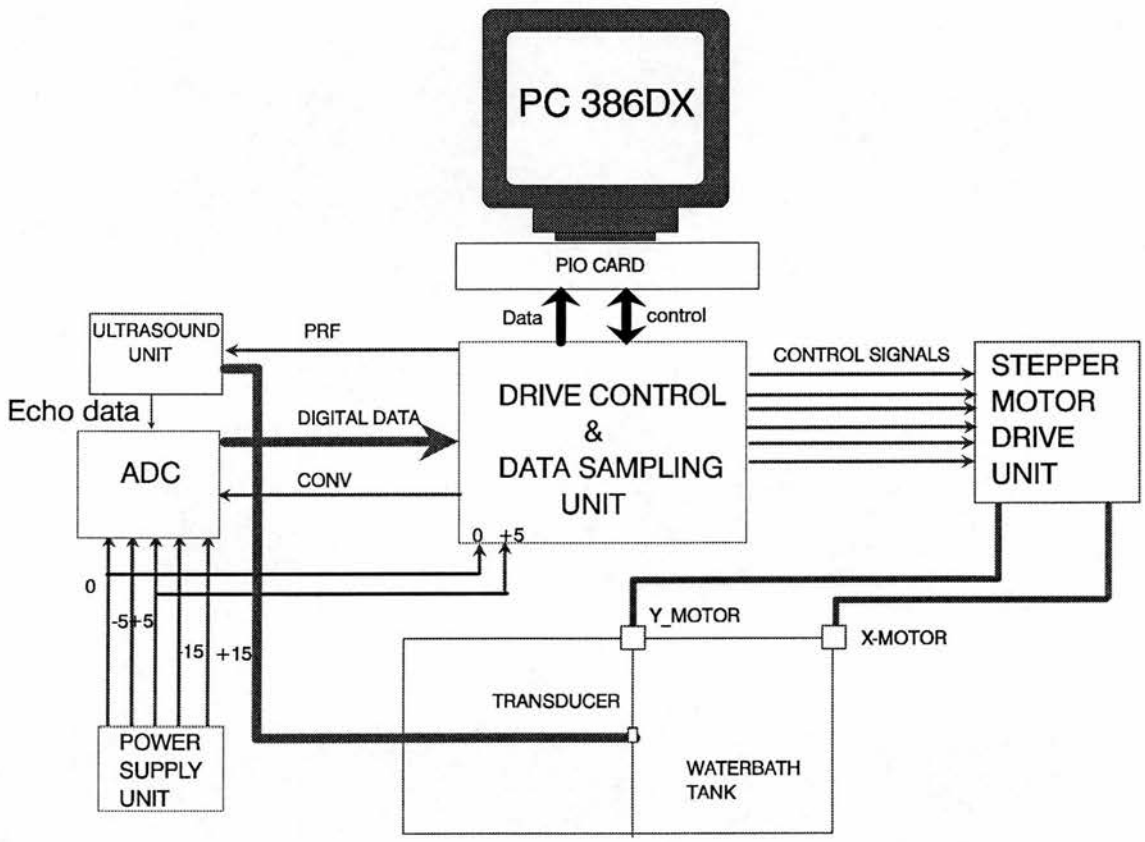


Figure 3.1 : C-scanning ultrasound imaging system.

3.2. Microcomputer

The microcomputer chosen for this project was a Packard Bell PC386DX. This was one of the most powerful machines available at the time. It has the following specifications: 33 MHz speed 8 Mbytes RAM, 320 Mbytes hard disk, 3.5 inch and 5.25 inch floppy disc drives and 5 slots for specialised cards. The system has a Mitsubishi colour monitor, with super VGA card.

3.3. Programmable I/O Card

The C-scan system consists of two main parts: the microcomputer and the external hardware. The digital data flows between the above two main parts via a parallel I/O board.

The main component of this board, used to achieve the data transfer, is the 8255 chip as shown in Figure 3.2.

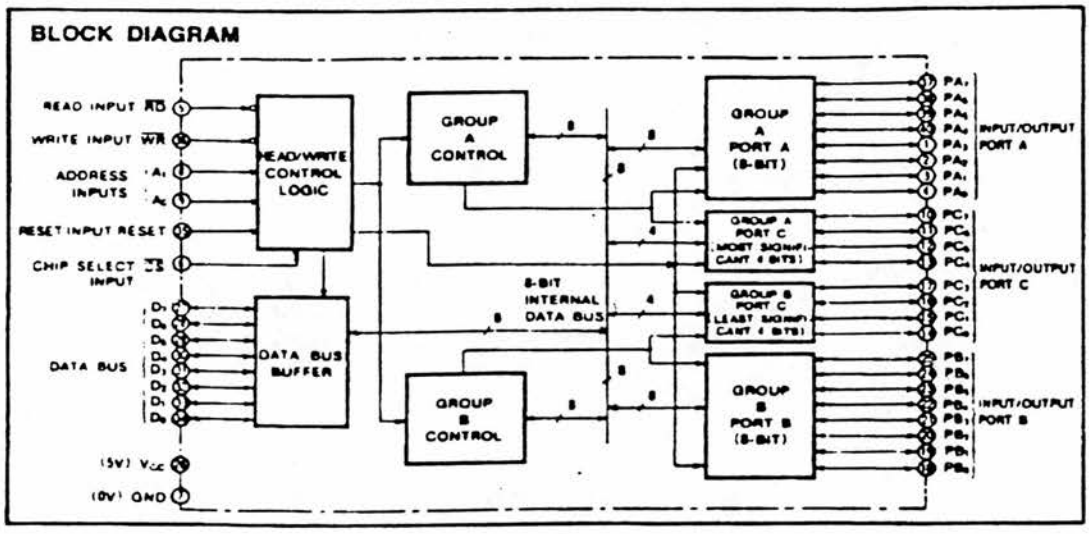


Figure 3.2 : Block diagram of Programmable Peripheral Interface (PPI 8255) (courtesy of Mitsubishi Electric).

This chip has 24 input/output pins grouped as three programmable I/O ports which can be programmed by a Central Processing Unit (CPU) through two 12-bit groups A and B. They are used in three operation modes, mode 0, mode 1 and mode 2. Operating in mode 0, each group of 12 pins may be programmed in sets of 4 to the inputs or outputs. In mode 1, the 24 I/O terminals may be programmed in two 12-bit groups, group A and B. Each group contains one 8-bit data port, which may be programmed to serve as input or output, and one 4-bit control port used for handshaking and interrupt control signals. Mode 2 is used with group A only, as one 8-bit bi-directional bus port and one 5-bit control port. Bit set/reset is controlled by CPU.

In C-scan system a board (PIO 24) based on the 8255 chip is used. It has three ports A, B, C High and C Low which can be independently programmed for input or output. Mode 0, input/output mode, is used. The codes for programming the 8255 in this mode are shown in Table 3.1.

Control words									Group A		Group B	
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Hexadecimal	Port A	Port C (high order 4 bits)	Port C (low order 4 bits)	Port B
1	0	0	0	0	0	0	0	80	OUT	OUT	OUT	OUT
1	0	0	0	0	0	0	1	81	OUT	OUT	IN	OUT
1	0	0	0	0	0	1	0	82	OUT	OUT	OUT	IN
1	0	0	0	0	0	1	1	83	OUT	OUT	IN	IN
1	0	0	0	1	0	0	0	88	OUT	IN	OUT	OUT
1	0	0	0	1	0	0	1	89	OUT	IN	IN	OUT
1	0	0	0	1	0	1	0	8A	OUT	IN	OUT	IN
1	0	0	0	1	0	1	1	8B	OUT	IN	IN	IN
1	0	0	1	0	0	0	0	90	IN	OUT	OUT	OUT
1	0	0	1	0	0	0	1	91	IN	OUT	IN	OUT
1	0	0	1	0	0	1	0	92	IN	OUT	OUT	IN
1	0	0	1	0	0	1	1	93	IN	OUT	IN	IN
1	0	0	1	1	0	0	0	98	IN	IN	OUT	OUT
1	0	0	1	1	0	0	1	99	IN	IN	IN	OUT
1	0	0	1	1	0	1	0	9A	IN	IN	OUT	IN
1	0	0	1	1	0	1	1	9B	IN	IN	IN	IN

Table 3.1: Mode 0 programming code.

3.4. Mechanical Design and Driving Unit

In Chapter 2, the C-scan system design specifications were discussed. One of its specifications is that the system uses a simple transducer which is driven by a mechanical system based on stepper motors. The test-tank arrangement, stepper motor characteristics and driving unit to these motors are discussed in the following sub-sections.

3.4.1. Design of Test Rig

The design of the test-tank is very important because it determines the accuracy of the C-scan system. Many precautions were taken to achieve the highest possible accuracy. The two shafts are

controlled by two stepper motors (discussed in the following sub-section). One motor gives an intermitent displacement in the y-direction while the other motor for the x-direction giving moves almost continuously. This arrangement accomplished the raster scan giving a two-dimensional ultrasonic image of structures lying in a plane at a specific depth within the object. Care was taken to ensure that water does not penetrate the housing of the transducer to comply with manufacturer's advice related to immersion of probes in water. This immersion was about 1-2 cm from the front face of the transducer. The transducer was mounted securely in the test rig with sufficient adjustment that it could be aligned parallel to the centre axis of the object or phantom. Figure 3.3 shows the test rig.

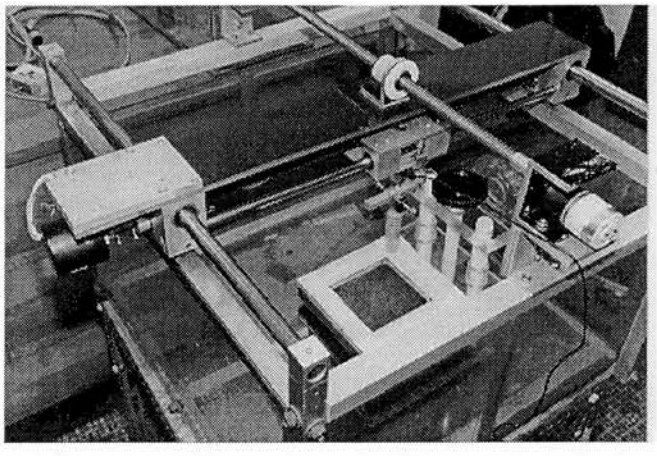


Figure 3.3 : The test rig.

3.4.2. Stepper Motors

In the introduction of this chapter, it was mentioned that the digital approach is the one used in the C-scan system. Hence an obvious choice when position is to be accurately controlled by digital circuits is a stepper motor. Stepper motors have many advantages.

- High resolution without gearing: the resolution can be doubled (i.e. steps per revolution are doubled) simply by changing the sequencing of drive signals to the stepper.

- Fast positioning: stepper motor at rate 250 steps/second, within a second can move to any position, stopping with a positional accuracy of 0.05 mm.

- Position error does not accumulate : with no external torque (i.e. a torque applied to the shaft by some source such as a weight or spring loading) the shaft of the stepper motor used will be within $\pm 1^0$ of its nominal angular position. Furthermore, stepping it 243 steps clockwise and then 243 steps counterclockwise will return to its exact starting position, within $\pm 1^0$.

Drive circuitry is simple and efficient: because a stepping actuator is driven by turning the current in four motor windings on or off, the drive circuitry requires only transistor switches.

After referring to many manufacturers of stepper motors, Sigma's 1.8^0 was chosen because it features with high speed performance, minimum settling time and availability of driving circuitry. Sigma's 1.8 motors are two-phase motors with permanent magnet rotors. Each pulse to a driver is translated to a change of current in the motor windings and produces a 1.8^0 movement of the rotor (0.9^0 half-step). Based on the C-scan system requirements - high resolution , smooth operation, and quick settling time - a stepper motor of type 20-2215D200-F15 was recommended (Figure 3.4) .

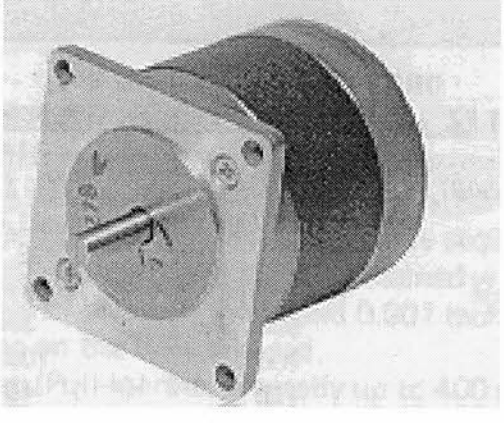


Figure 3.4 : The chosen stepper motor (courtesy of Unimatic Engineers Ltd.).

The specifications of these motors are shown in Table 3.2.

Steps per revolution	200
Step angle	1.8^0
Step angle tolerance	± 3%
Phase resistance	1.4 Ω
Phase inductance	1.25 mH
Phase current	1.5 A
Holding torque	0.19 Nm
Deterrent torque	0.01 Nm
Rotor inertia	0.006 x 10^{-3} Kg m^2
Weight	0.36 Kg

Table 3.2 : The stepper motor specifications.

Figure 3.5 shows its mechanical drawing .

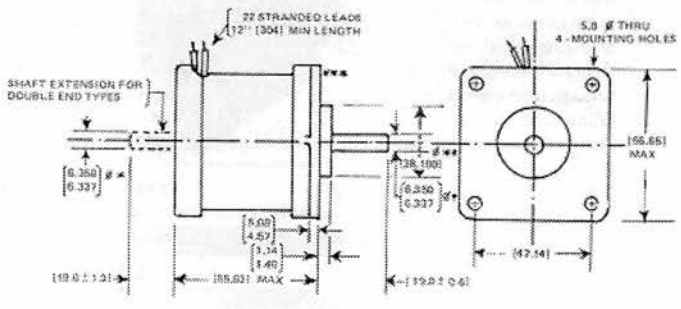


Figure 3.5 : A Mechanical drawing of the stepper motor.

3.4.3. Stepper Motor Drive Unit

The unit consists of a EM172 power supply, two TM162C stepper motor drive boards and one EM210 stepper motor voltage controlled oscillator. The EM210 allows one stepper motor to be controlled manually on the unit, or both can be controlled externally using gating signals. However, the board with the EM210 unit only needs to be told to start or stop, whether the velocity is high or low, and in which direction to drive the motor, since the actual pulses are produced by the EM210 unit. The other stepper motor drive board has no oscillator, so the actual oscillating drive pulse has to be provided externally. Figure 3.6 shows the stepper motor drive unit.

3.4.3.1. Power Supply

Power supply (EM172) is designed for use with the stepper motor control card. It is a self-contained unit that requires a mains input which is converted to an unregulated DC voltage . The specifications of the power supply is shown in Table 3.3.

Inputs	240 Vac @ 50 or 60 Hz
Outputs	24-28 Vdc
Current	6 A max

Table 3.3 : Power supply specifications.

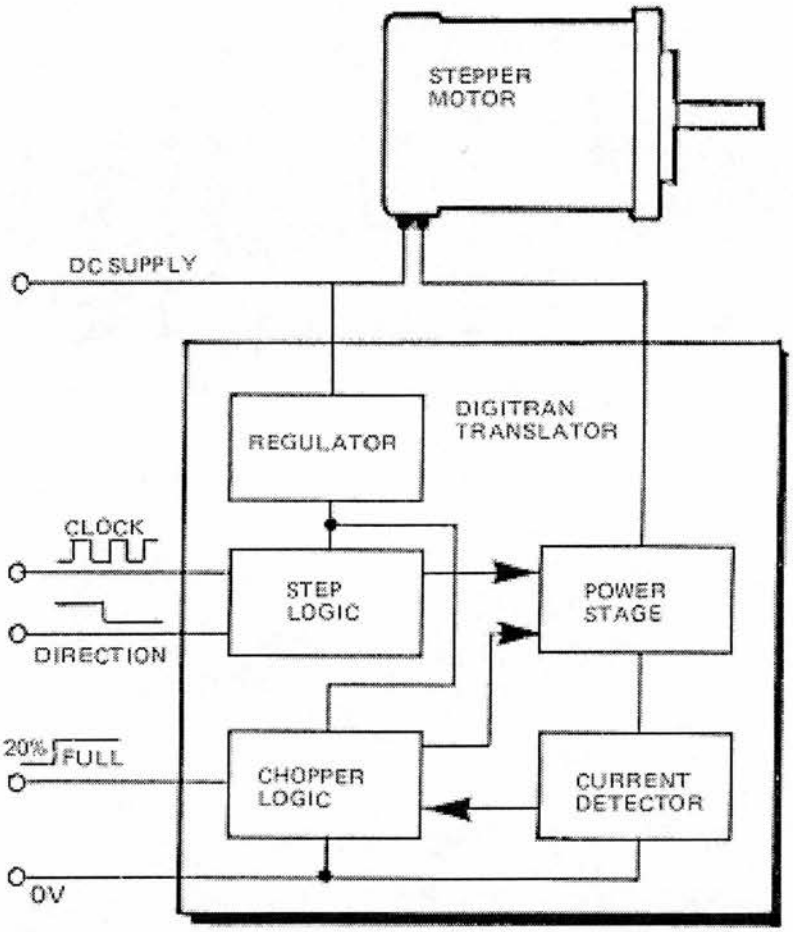


Figure 3.6 : Stepper motor drive unit.

3.4.3.2. Stepper motor drive

The stepper motor drive board (TM162C) was built on a printed circuit board. It operates from externally derived control pulses, direction of motor rotation being determined by a separate logic signal. Control facilities may be readily expanded to provide voltage by using Voltage Controlled Oscillators (VCO). Figure 3.7 shows the stepper motor drive block diagram and its connections to the stepper motor and VCO.

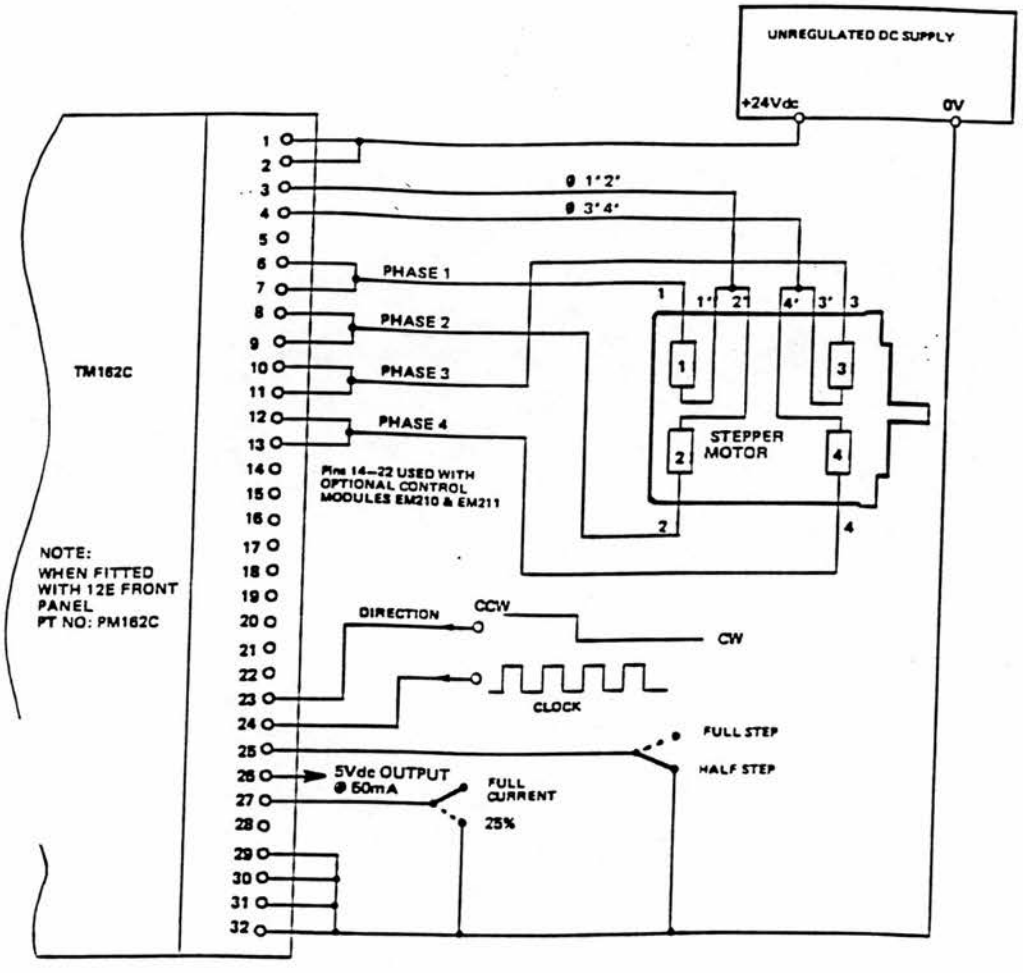


Figure 3.7 : Connection for TM 162C using external clock.

3.4.3.3. Voltage Controlled Oscillator

The voltage controlled oscillator (EM210) was designed for mounting on the stepper motor drive (TM162C) as shown in Figure 3.8 . It provides clock pulses to drive stepper motors in applications requiring voltage to frequency conversion, manual control of motor movements or in conjunction with process controlled logic signals. It was provided with two speed ranges, base and high speed. When gated 'ON' the oscillator runs at base speed which should be selected to be within the motor's "start/stop without error" range. A separate terminal can be used to gate the unit to run at the high speed setting which has a maximum rate of 12,500 steps/sec and which can be varied by a frequency control voltage between 0 and 12 VDC. An adjustable ramp control is provided to set the acceleration/deceleration time from base to high speed settings.

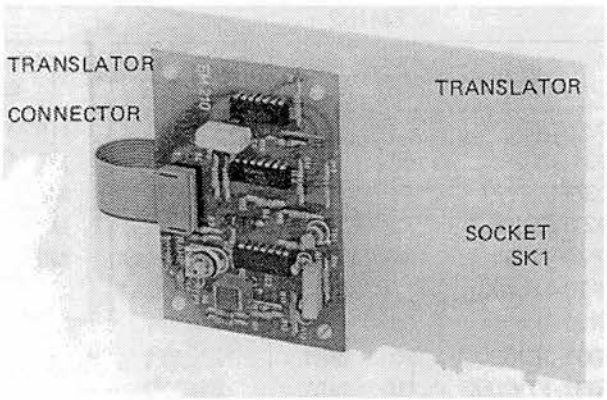


Figure 3.8 : The interface between TM 162C and VCO EM210.

3.5. Ultrasound System Unit

Ultrasound system unit is the heart of the C-scan ultrasound imaging system. Its function is similar to the well known A-mode instrument. Therefore it will be described briefly. The unit has four boards : transmitter, receiver, control and power supply boards as shown in Figure 3.9. The transmitter's task is to provide an electrical current pulse from a high voltage pulse generator to the transducer crystal when demanded by the prf controller internally or externally. Ultrasound signals reflected from tissue interfaces are converted to electrical signals and passed to the receiver. The signals are passed to a three stage tuned radio frequency (RF) amplifier with a variable gain. The control board enables the user to choose between external PRF for external interfaces or internal PRF. The internal PRF production is set by an oscillator. The last part in the unit is the power supply which can produce high voltage of 180 VDC to be used by the transmitter. The complete unit is shown in Figure 3.10.

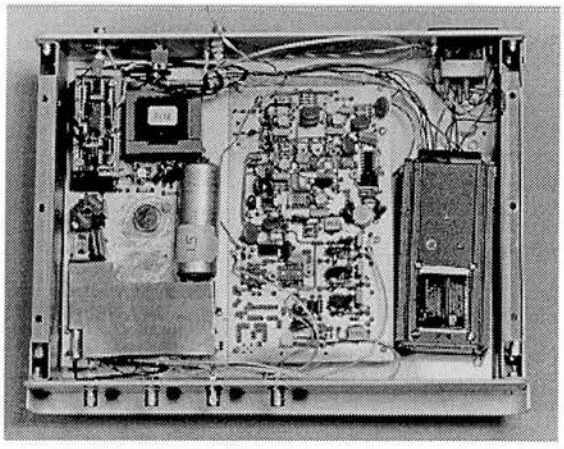


Figure 3.9 : Ultrasound system unit boards

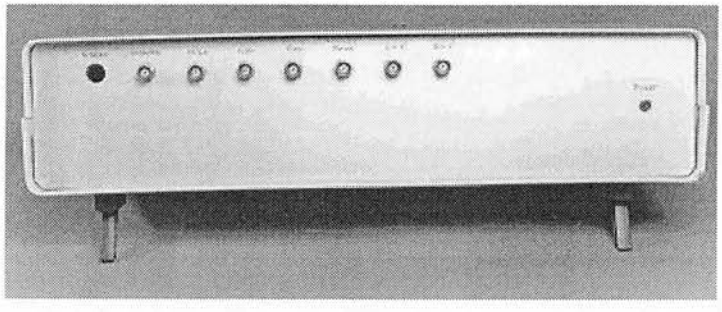


Figure 3.10 : Ultrasound system unit

3.6. Transducer Characteristics

It was planned to use a simple single element transducer with central frequency of 3.25 MHz in the C-scan system for transmitting and receiving echo signals. The transducer characteristics will be presented in the following sub-section.

3.6.1. Method

The basic measurement system used involves a hydrophone via a preamplifier connected to an oscilloscope. The hydrophone was mounted in a scanning tank and aligned with the front face of the transducer. To avoid reflection problems from the water surface, and the sides of the water tank, pieces of carpet as acoustic absorbing material were used around the sides of the tank. The transducer was mounted on a platform which could be moved in two orthogonal directions from above the water tank. The hydrophone was rigidly mounted on a holder underwater and the distance between the transducer and hydrophone could be adjusted. The hydrophone was scanned over the measurement plane to locate the position of the largest signal. A beam profile was then obtained by measuring the hydrophone signal (the peak value) along a straight line either side of the maximum.

3.6.2. Results

The pulse waveforms recorded for transmitter output setting of 20 dB from the 3.25 MHz single crystal transducer at different depths are shown in Figure 3.11(a) to (e). Table 3.4 shows the measured values of beam width at different depths at 20 dB. It was found that the transducer is focused at a depth of 6 cm. Figure 3.11 (f) shows the Fast Fourier Transform (FFT) of the waveform at depth 6 cm. When the depth was increased to more than 8 cm distorted shapes were obtained due to non-linear propagation. Also it was noted that the effect of non-linear propagation appeared when the output power was increased as shown in Figure 3.12. Table 3.5 shows that there was little affect on the beam width.

Depth (cm)	Minimum (Pa)	Maximum (Pa)	Peak-Peak (mV)	BeamWidth (mm)
1	24	31	55	10.1
2	28	36	65	6.7
3	39	47	86	5.7
4	79	72	151	3.2
5	93	84	176	2.7
6	98	88	186	2.7
7	98	84	182	3.0
8	91	75	166	3.4
9	79	60.5	144	3.8
10	69	56	125	4.5

Table 3.4 : Measured beam width at half he amplitude (-6dB) values for 3.25 MHz single crystal transducer at different depths at 20 dB transmitter output setting.

Depth (cm)	Minimum (Pa)	Maximum (Pa)	Peak-Peak (mV)	Beam Width (mm)
10	16.6	15.5	32	4.5
6	24.5	22.5	47	2.9

Table 3.5 : Measured beam width at half the amplitude (-6 dB) values for 3.25 MHz single crystal transducer at different depths at 32 dB

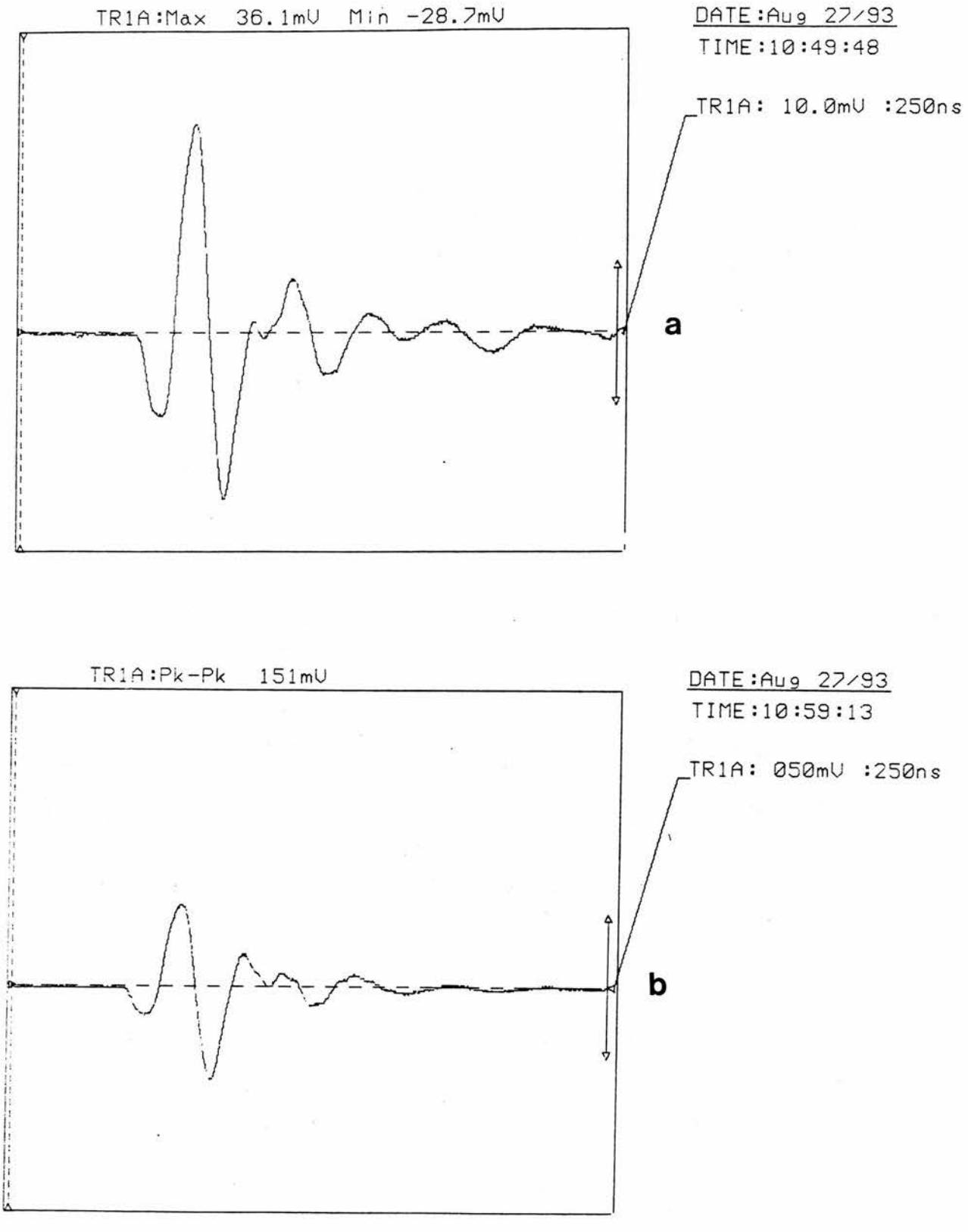
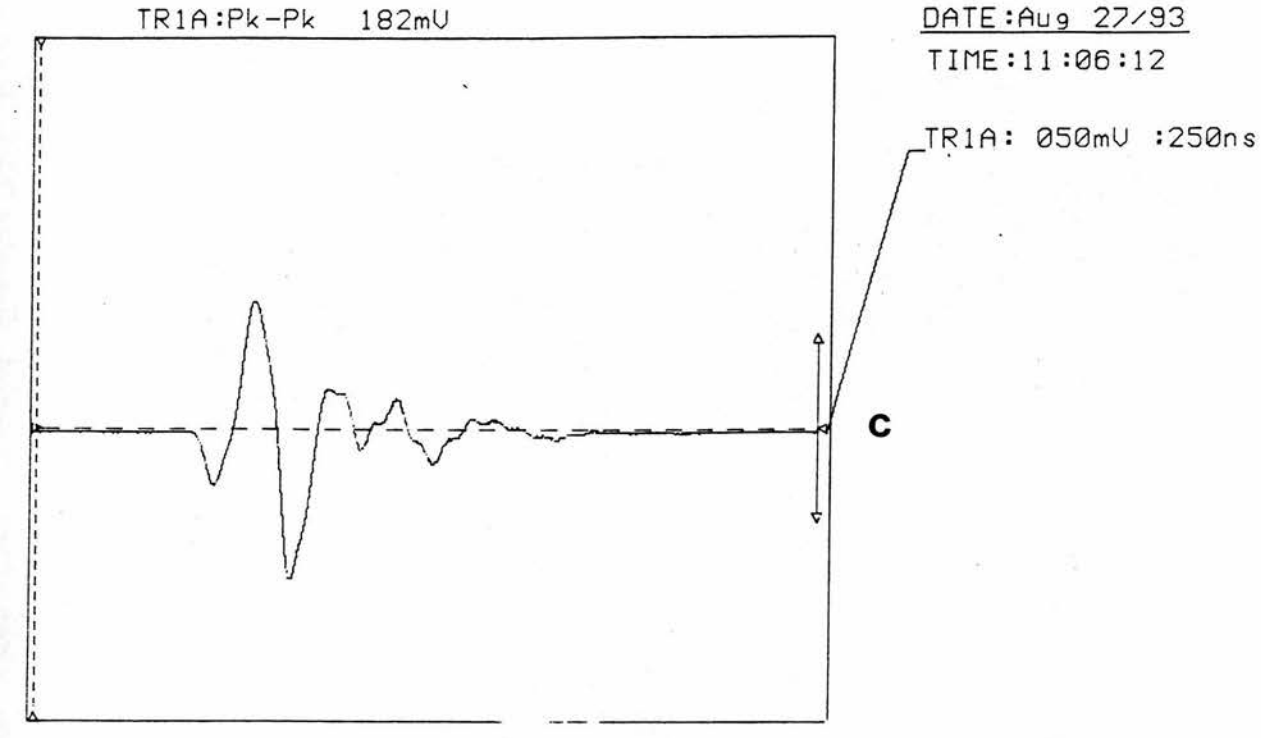


Figure 3.11 : Pulse Waveforms of 3.25 MHz single crystal transducer at 20dB at different depths (a) - 2 cm depth (b) - 4 cm depth.



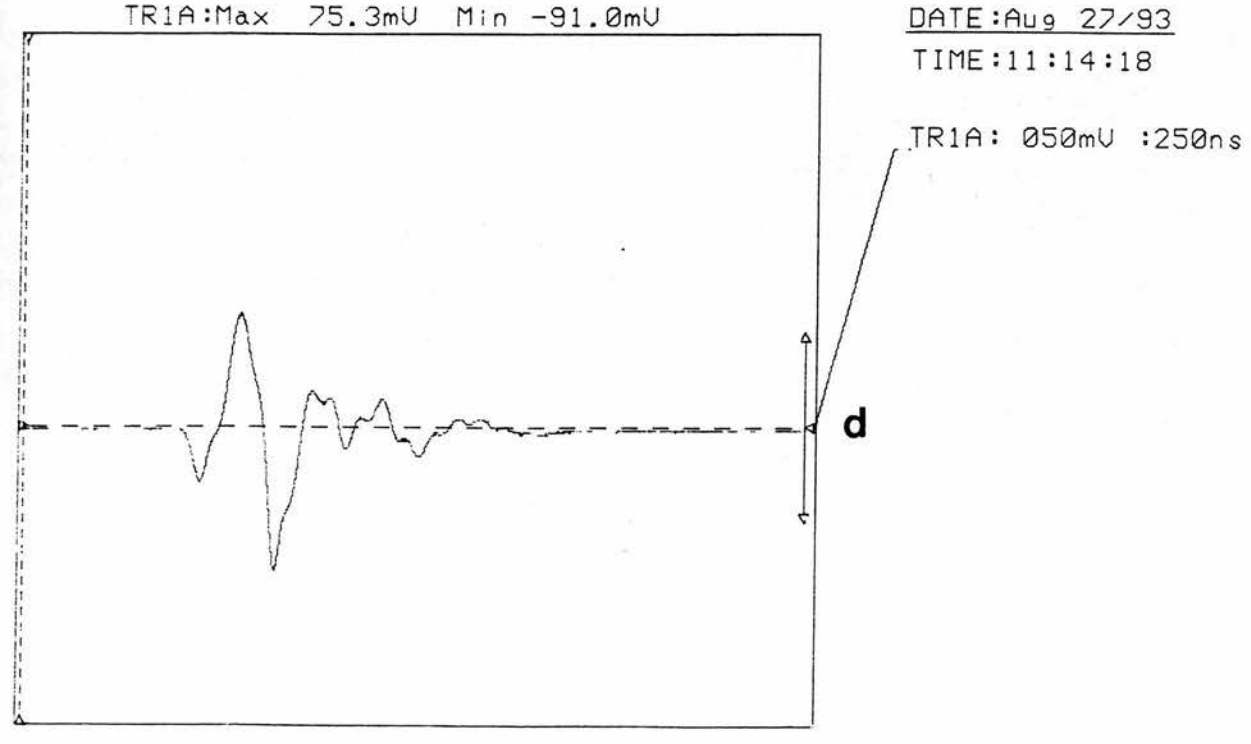
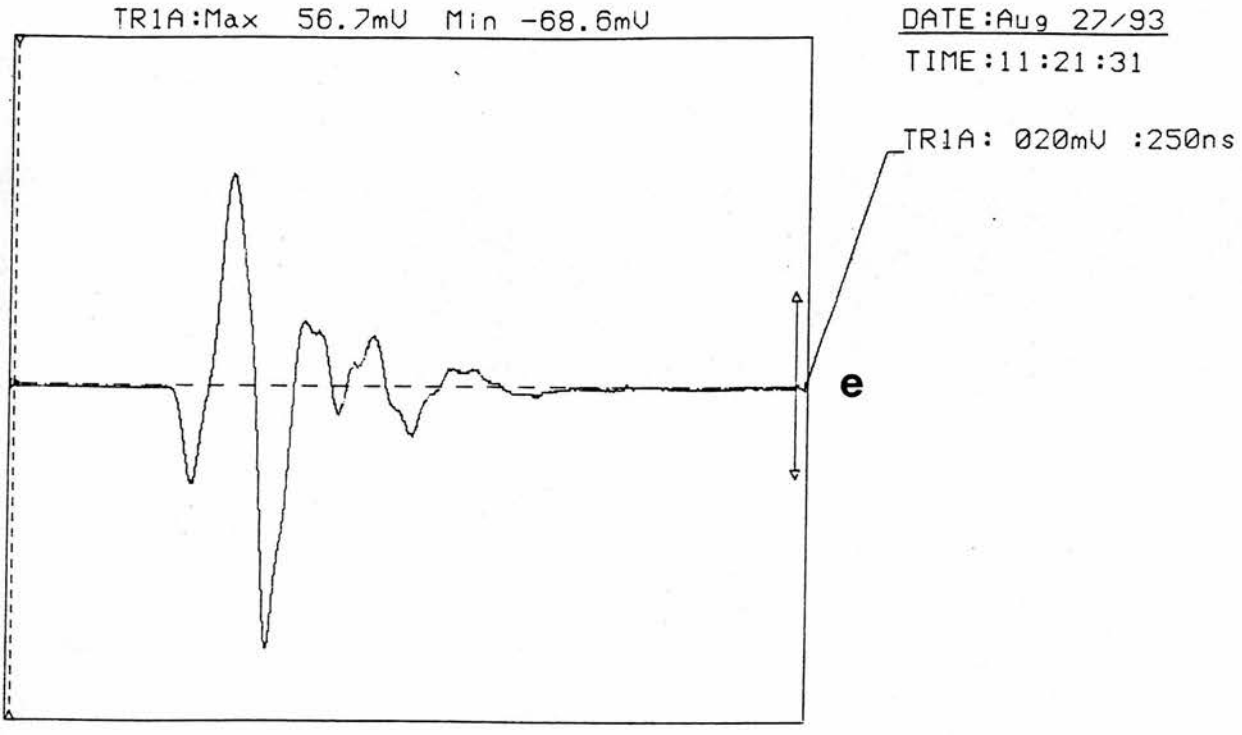


Figure 3.11 : Pulse Waveforms of 3.25 MHz single crystal transducer at 20dB at different depths (c) - 6 cm depth (d) - 8 cm depth.



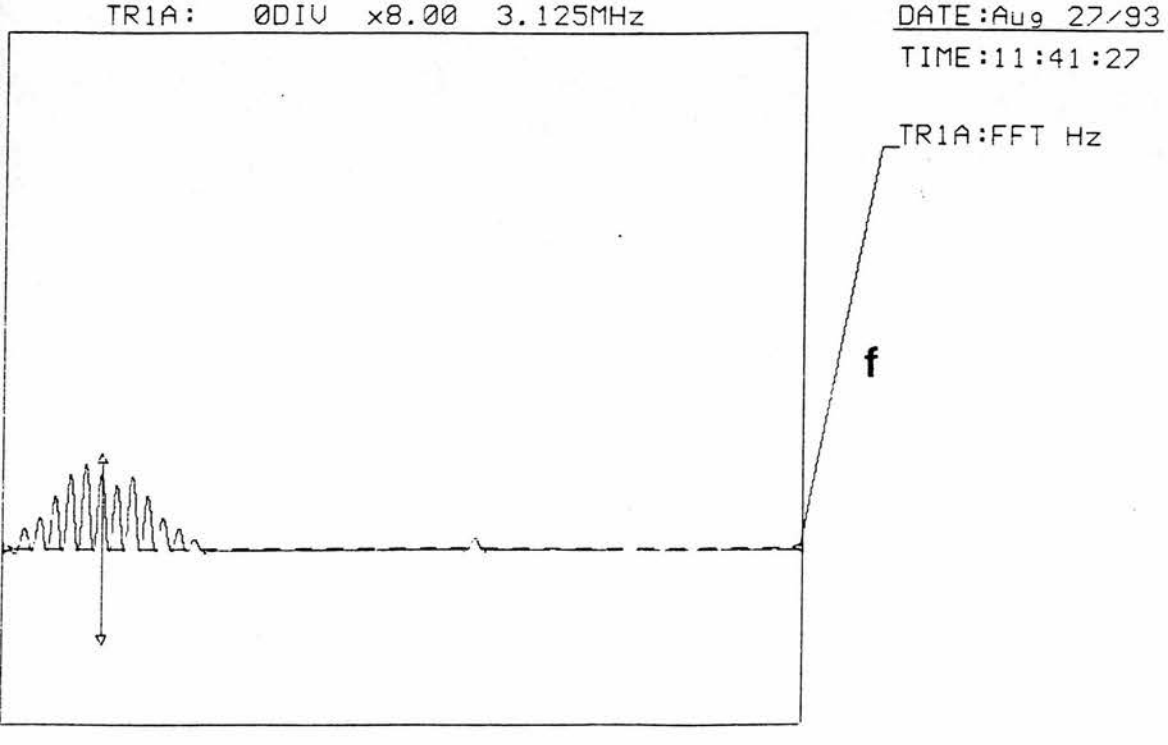
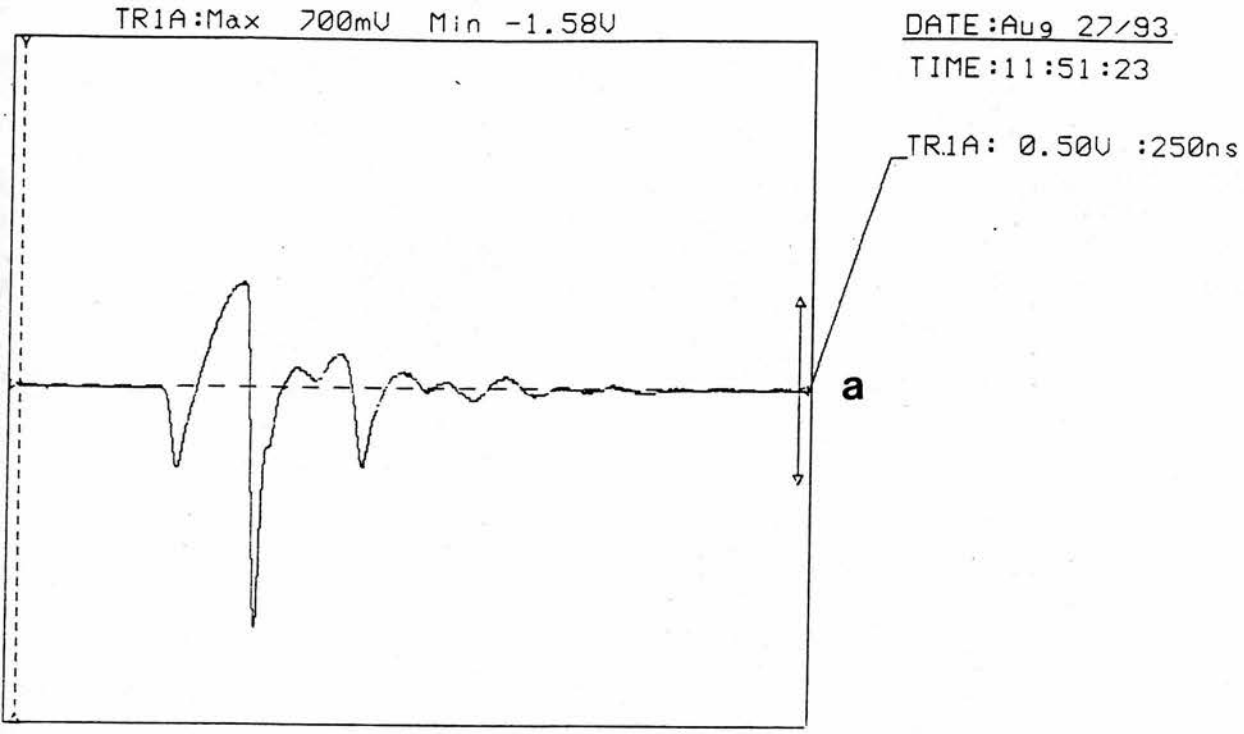


Figure 3.11: Pulse Waveforms of 3.25 MHz single crystal transducer at 20dB at different depths (e) - 10 cm depth (f) - FFT of waveform at 6 cm depth.



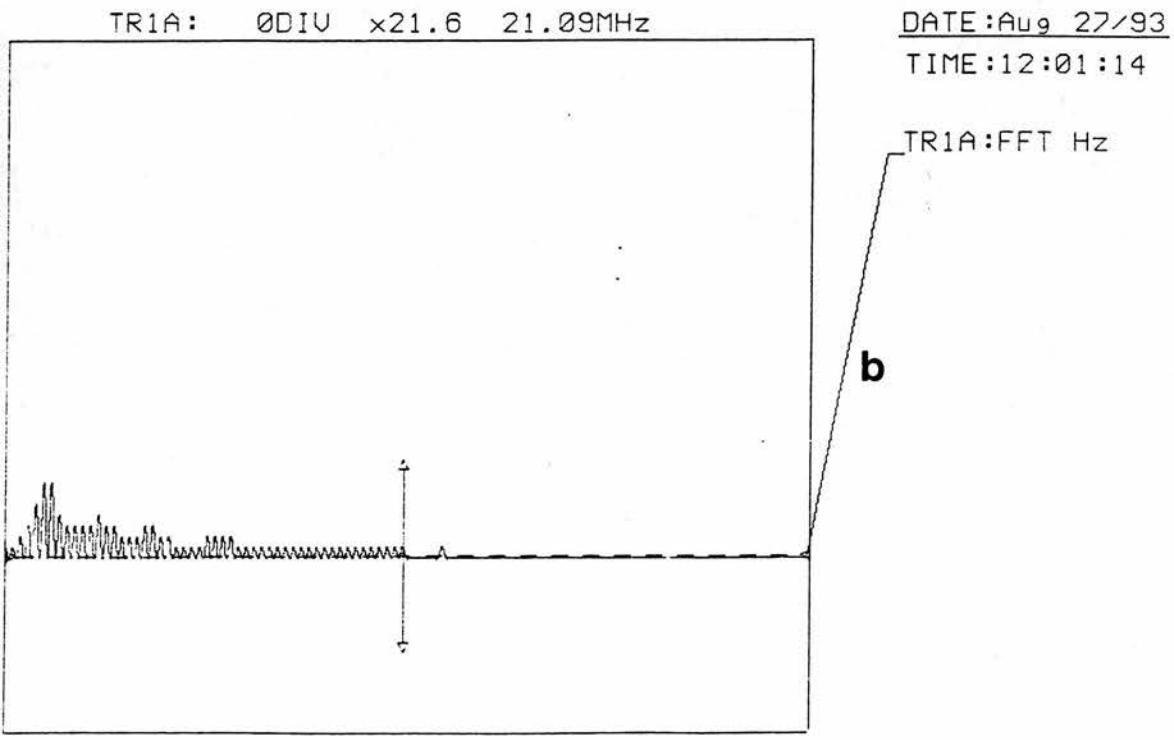


Figure 3.12: Pulse Waveforms of 3.25 MHz single crystal transducer at 32dB (a) - 6 cm depth (b) - FFT of waveform at 6 cm depth.

3.7. Drive Control and Data Sampling Unit

The drive control and data sampling unit contains four boards as shown in Figure 3.13. Two of these boards carry the data sampling part, the third carries the analogue to digital converter (ADC) and

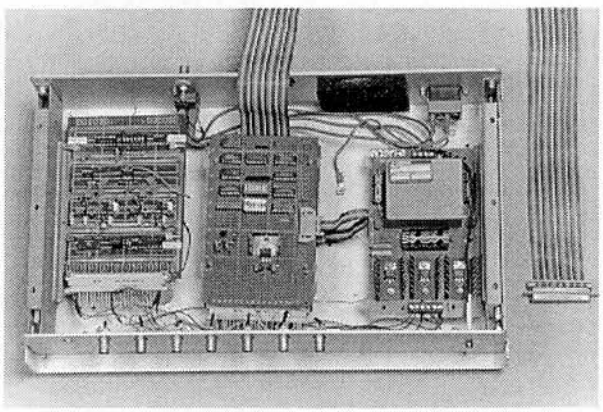


Figure 3.13 : A photograph of the drive control and data sampling unit.

the fourth carries the power supply. It was interfaced to the motor drive unit and ultrasound system via BNC cables and to the computer via a ribbon cable . The following subsections describe the function of each board.

.

3.7.1. Data sampling boards

The function of the data sampling board is to determine when to start and stop digitising ultrasound data.

The first board consists of a 16 MHz oscillator, a monostable multivibrator of duration 400 μsec to cover up to 30 cm depth, a +12V to +5V logic level converter and a short first-in-first-out buffer

(64 words x 8 bits, 35 MHz FIFO chip) used to hold data until the computer is ready to store it in memory. The second board contains a synchronised digital circuit generating up to 256 samples as selected via digital switches at the beginning of the scan. The following is a description of all the important signals:

SECR1:

The first serial clock loads data into a shift register which selects a particular line to the multiplexer. The multiplexer output triggers the ADC. In this way it is possible to vary the delay between A/D conversions.

SERC2:

This signal loads the second delay into the three cascaded counters. This determines the depth at which the first plane is imaged. After this point, planes are imaged at regular intervals, determined by the multiplexer, until the monostable multivibrator goes low.

SERD:

This is the data line on which SERC1 and SERC2 load data.

PRF:

The output from the monostable which serves to enable or disable the counters.

STEP:

A line from the stepper motor voltage controlled oscillator which is the clock for IC3 which in turn sends out a signal every 32 stepper motor pulses to set the monostable for 400 µsec. While the monostable is high, the cascaded counters are able to count. When they reach zero, the multiplexer is enabled, and analog-to-digital (A/D) conversions take place until the monostable goes low.

DN/UP:

Controls the direction of the counter.

RESET:

Enables the counter at the start of the scan.

3.7.2. Analogue to Digital Converter Board

Ultrasound data is fast and needs to be acquired and stored in a buffer for subsequent processing. The most widely used method to accomplish this is to convert the analogue voltage signal directly into digital form using a fast ADC, and then store the data in a buffer (as explained in the last section) from which the data can readily be accessed by the microcomputer.

The appropriate sampling rate must be determined. The sampling theorem states that the minimum sampling rate necessary to completely recover a bandwidth limited signal is at least twice the maximum frequency component of the signal. The presence of wideband noise and the fact that few real signals have a well defined upper frequency limit mean that it is usually necessary to sample at three to five times the maximum significant frequency.

The upper frequency limit of detected pulses from a 3.25 MHz transducer was measured using a spectrum analyser. It was positioned so as to transmit pulses into a water bath and pick up the echoes returning from a plane reflecting surface 10 cm from the probe face. The signal to the spectrum analyser was taken immediately after the detector in the receiver. The maximum frequency component detectable above the noise level was 1 MHz. Sampling at three times this frequency would give a sampling rate of 3 MHz. The specifications of the ADC are thus a minimum sampling rate of 3 MHz and 8 bit accuracy. It was decided to purchase an 8 bits parallel (flash) converter, type TDC1058 E1C evaluation board from TRW (USA) which is capable of sampling at 20 MHz. The data sampling board provides different sampling frequencies based on the desired scan. The output of the receiver (video signal) of the ultrasound system starts to saturate at 12V. The ADC can not cope with this level so the voltage level was attenuated to 1V via a potentimeter. The ADC was set up to give maximum output (255) when the output after attenuation is 1V and minimum output (0) when the output is 0V.

3.7.3. Power Supply Board

The drive control and data sampling unit needs + 5V, - 5V, + 12V, and - 12V whereas the data sampling boards are completely TTL which requires + 5v and the ADC board required + 12V, -

12V , + 5V and - 5V. The current required by the data sampling boards was 200 mA for + 5V. It was decided to use a power supply board which requires input voltage of 240 vac and gives + 5V @ 1 A, + 12V and - 12V @ 250 mA/rail. A small circuit was added to provide - 5V required by the ADC.

CHAPTER 4

DESIGN AND SPECIFICATION OF C-SCAN SYSTEM: DATA ACQUISITION AND IMAGE PROCESSING

CHAPTER 4

DESIGN AND SPECIFICATION OF C-SCAN SYSTEM: DATA ACQUISITION AND IMAGE PROCESSING

4.1. Introduction

Design of C-scan system software involves many stages. These stages are:

i. Design from software specification
ii. Software development and debugging
iii. Testing
iv. Documentation

When designing software it is desirable to consider factors such as the portability, maintainability and limitations of the equipment it will be used on.

The high level language Borland C++ (Version 4.0) was chosen for implementation, the reasons being ease of use, power (especially with the AT type PC), flexibility, availability, low level interface facilities, allowance of structured design and cost.

The package includes a compiler, an editor, a debugger, library manager and an assembler. The package gives the possibility to create executable files from a project file that contains all the program files. The executable file can be run from the Disk Operating System (DOS). The package features debugging facilities and low level interface capabilities for use in the interface between the computer and the external hardware.

A large proportion of the development time was spent planning and structuring the software in order to provide a logical and efficient construction. This is desirable for readability and maintenance.

4.2. Overall Program Design

The software was menu-driven to make it easier for the user. It was designed to control the scanner, position the stepper motors to the exact starting position for the scan, to load scan parameters, capture and display the echo data. After capturing data, the data can be displayed and examined in more detail, plotted using another program called Viewdata and stored to disc for processing and later retrieval. The main program involves many steps as follows:

1. Initialise the programmable I/O board and initialise all ports.
2. Initialise the data arrays.
3. Check if there is enough memory in RAM.
4. Check if there is enough space in the hard disk for data files.
5. Initialise all the global variables.

Then the user is given the main menu which contains different options as shown below:

A: Position the Scanner
B: Load Scan Parameters
C: Start Scan
D: Save Data
E: Display the Data
Q: Exit

Based on user's choice the program proceeds to execute the desired action. At the end Exit is chosen to end the program. More details about control, data acquisition, processing and display programs will follow in the next sections. All programs are listed in Appendix B.

4.3. Data Acquisition and Control Programs

Data acquisition and control involves the choices from A to D.

The first choice moves the motors (either x-motor or y-motor or both) to the exact point which the scan starts from. The second choice asks the user to enter the number of sampling points along the scan line, the number of scan lines making up the area, the distance between each raster position in steps, the depth of first image plane in mm (maximum of 300mm), the interval (or separation) between successive planes (1.5 mm, 3 mm, 6mm or 12mm) and finally the file name for data storage (maximum of 8 characters). The two values before the file name are converted into binary numbers and entered into the sampling board through the interfacing card. After completing the first two choices, the third choice starts the scan. The microcomputer then keeps track of the number of pulses produced by the stepper motor drive board by monitoring the output from the monostable (see circuit diagram of the C-scan ultrasound system in Appendix A) which goes low every thirty two stepper motor pulses. At the start of each scan line, the motor is accelerated to high velocity, at which point the monostable is enabled. While the monostable is high the microcomputer checks for digitised data from the ADC. When it finds data present in the short buffer (FIFO) it stores it in the memory in a three dimensional array of bytes. When the monostable goes low, all the required points along the axis of the beam have been digitised and stored. The transducer is moved to the next point along the scan line. The actual movement and positioning is controlled by the microcomputer. The fourth choice saves the echo data - collected in the previous choice - in a file on a disk. The fifth choice displays the data collected. The scan process can be repeated many times and when the user is happy with the data collected, the last choice can be selected to quit the program.

As can be seen from the above description, the program is written in a structured method to make it easy to be modified in the future. The other feature is the compatibility between this program and the other programs like the viewdata program used to show the echo data in a histogram form.

4.4. Image Processing Software Techniques

One of the important aims of our work is to implement image processing to images containing sparse amounts of data, as is obtained in real-time C-scanning. The processing should produce images of acceptable quality and should ultimately be implemented in real-time in order to gain acceptability in clinical practice.

In medical ultrasound and specifically in commercial scanners one type of image processing feature offered for noise reduction is recursive averaging. In this type of processing temporal filtering is performed. Each echo pixel is calculated from data belonging to previous frames and, therefore, obtained at different times. Another type is spatial filtering which utilises the information contained in the two-dimensional space domain defined by the x (rows) and y (columns) co-ordinates which constitute the scan image. The algorithms of the last type estimate the true grey scale level of pixel (x,y) from a combination of the pixel intensities in a predefined neighbourhood around (x,y). The neighbourhood is called the filter window or filter size and is usually a square area of M x M pixels, centred at point (x,y).

Image processing techniques fall into either of two broad categories: subjective and quantitative. Subjective processing or image enhancement is designed to improve human visual interpretation of an image. Some enhancement algorithms are image brightening, contrast stretching and contrast enhancement using filtering. Quantitative processing or image restoration is designed to reduce or eliminate the effect of degradation such as noise. Restoration algorithms are classified as linear, non-linear or adaptive according to the type of operation they perform on the data inside the window. Finally, image interpolation techniques are important in our case to increase data in images which have sparse data and to make them appear smoother and more visually pleasant. All the algorithms have been implemented in software. The corresponding programs, written in Borland C++ (Version 4.0) using the microcomputer, can be found in Appendix B.

The structure of the programs is as follows:

i. Initialise the VGA graphic card.

ii. Calculate the size of the required memory buffer.

iii. Allocate memory buffer for image data.

iv. Apply image processing algorithm.

v. Set the VGA graphics mode.

vi. Install the VGA palette required for display.

vii. Display the processed image.

viii. Save the image in a specific format.

ix. Free the memory buffer.

x. End the program.

4.4.1. Image Processing Algorithms Evaluation Methods

Ventsanopoulas and Cappellini (1986) review the hardware and software currently used for real-time image processing. They comment that,

> "The design of image processing algorithms is accomplished to a large extent on an ad hoc basis, with little or no theoretical foundation. This is due to the lack of appropriate theoretical framework for describing images and image components and their perceptual effect on human observers."

So, evaluating the performance of different filters can be a very difficult task. Ideally, a quantitative measure should be used but no agreement exists among the image processing community on which measures are the most appropriate. An interesting conclusion on this subject has been made by Mastin (1985) who performed a quantitative and qualitative evaluation of several noise smoothing algorithms. He concluded that it is the human observer's perception of quality rather than a statistical measure that defines the best filter. A possible explanation for this is that a human observer makes his judgement by scanning an image in a highly selective rather than a point-by-point manner. He chooses specific areas from which he extracts certain features and, in essence, weighs these areas more heavily than the rest of the image (Granrath, 1981). On the other hand, a quantitative measure which treats the image in a global manner with all the points having equal importance can not simulate satisfactorily the human visual system.

Finally it was decided to assess the various filters' performances using visual comparisons. An evaluation of image processing algorithms based on C-scan images is presented in Chapter 5.

Because the processed images have to be evaluated, care was taken to ensure that the quality of images presented in this thesis is the highest possible. Ultrasonic gray-shade images of the type presented in this thesis were compared when printed as a high quality photograph and by a laser printers (Figure 4.1 and 4.2 respectively). It was considered that the quality of laser printing was reasonably close to that of photography and that it could therefore be used satisfactorily for images in the thesis. This reduced the time and the cost of image reproduction considerably.

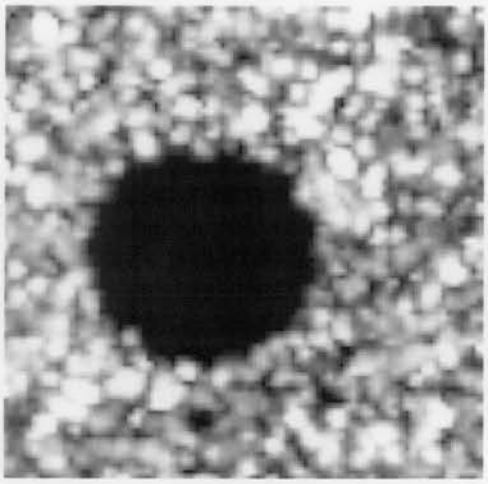
4.4.2. Image Processing Algorithms

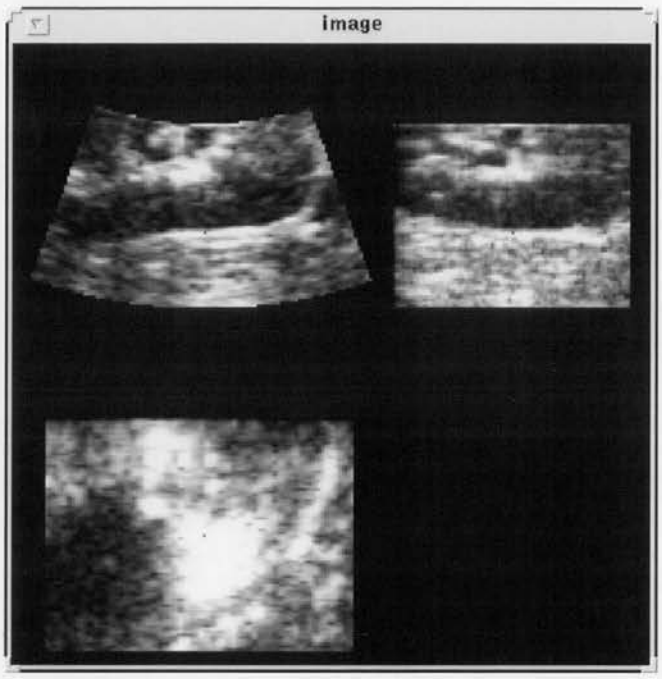
In medical images and specifically in ultrasound there are many artefacts (Kremkau et al., 1986) and one of them is speckle (Wells and Halliwell, 1981).
The suitable processing as mentioned in the begining of this section is the restoration methods. These methods can be performed either in space or in the frequency domain. The processing methods which will be used are performed in the space instead of the frequency domain because it is more flexible and easy to implement. These processing methods or what is called 'spatial filtering' techniques can suppress the noise or the irrelevant data like speckle in order to improve the ultrasound images quality. The spatial filtering techniques can be implemented in three different combinations: linear filters, non-linear and adaptive filters.

4.4.2.1. Linear Filters

Linear filters have been proposed for noise reduction in echocardiography (Hecker and Pöppl 1982).

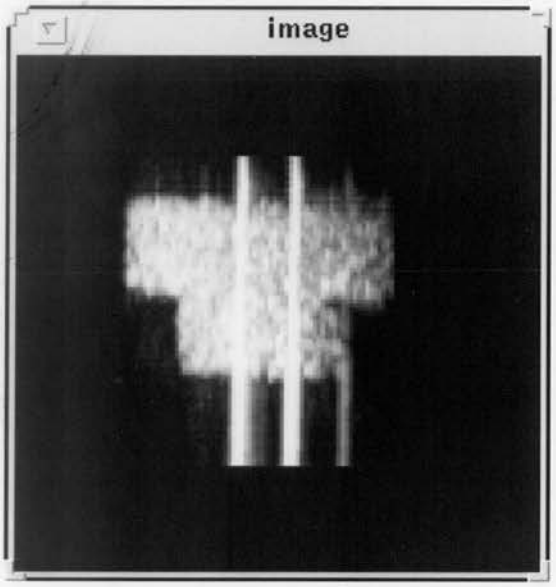
A 2D filter is defined as an operator T [•] that transforms an input sequence {I(x,y)} to an output sequence {O(x,y)}.

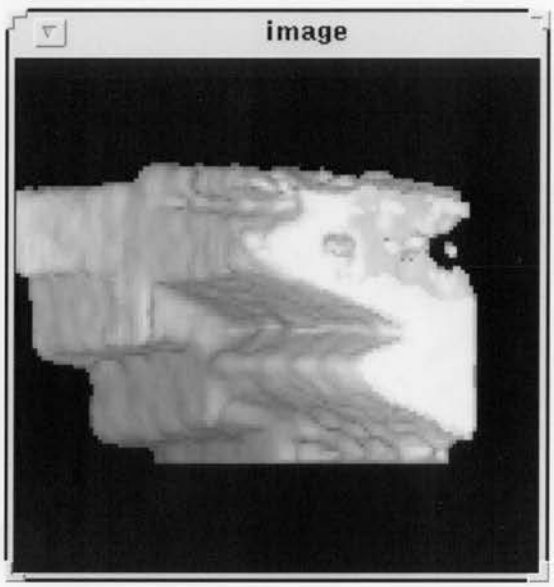




(a)

(b)

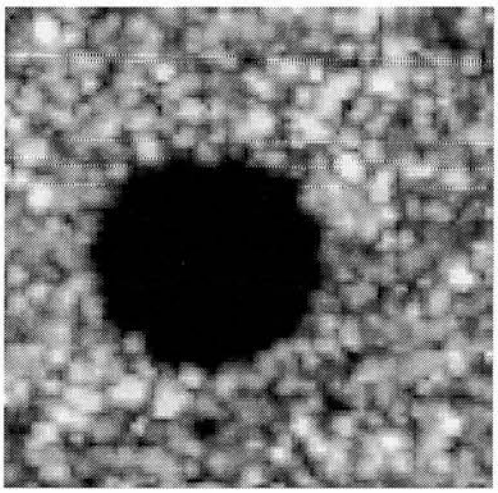




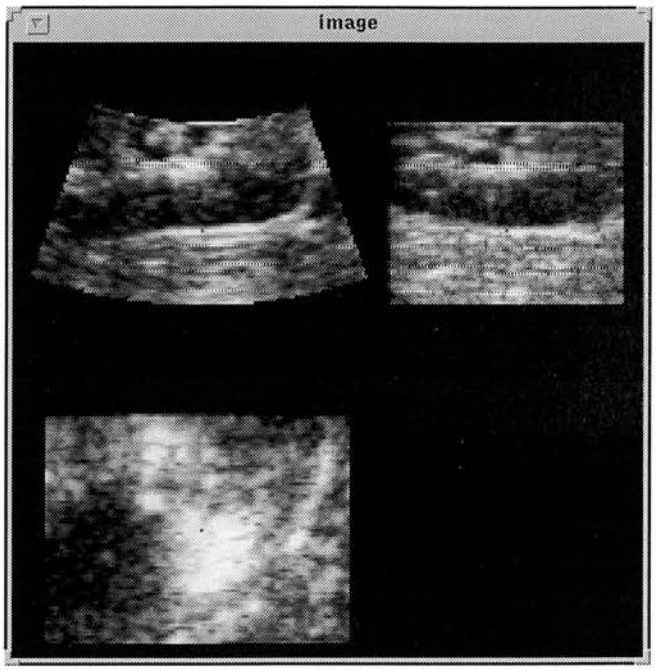
(c)

(d)

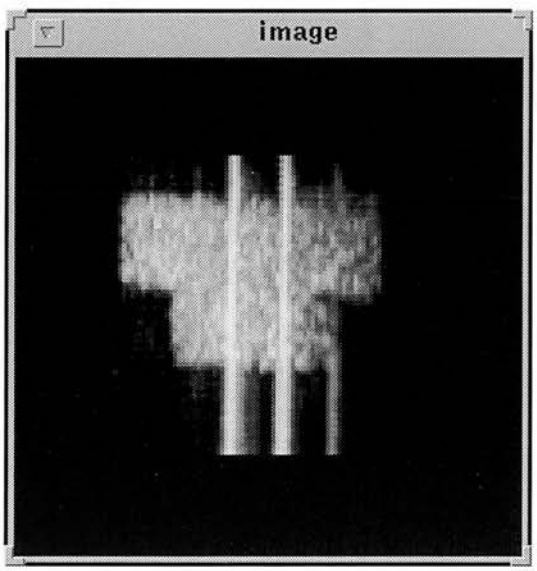
Figure 4.1 : Examples of the different types of images used in this thesis. (a) - C-scan image. (b) - Orthogonal views of 3D image. (c) - Maximum intesity projection of 3D image. (d) - Volume rendering of 3D image.



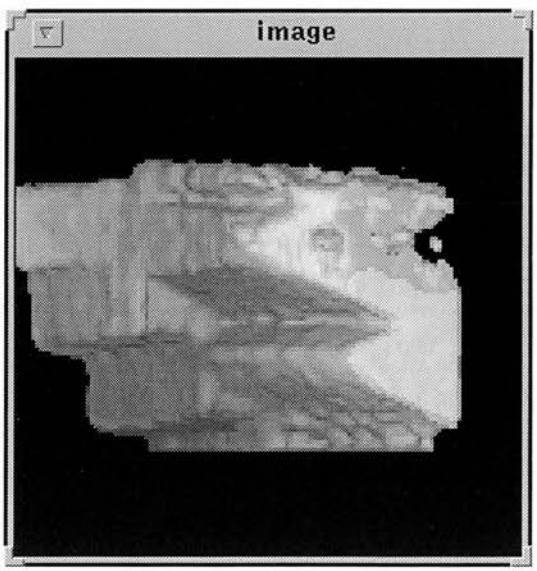
(a)



(b)



(c)



(d)

Figure 4.2 : Examples of the different types of images used in this thesis. (a) - C-scan image. (b) - Orthogonal views of 3D image. (c) - Maximum intensity projection of 3D image. (d) - Volume rendering of 3D image.

The filter is defined as linear if satisfies the principle of superposition, that is if

$$T[\{aI_1(x,y) + bI_2(x,y)\}] = a\,T[\{I_1(x,y)\}] + b\,T[\{I_2(x,y)\}] \qquad (4.1)$$

Linear filters are two types: finite impulse response (FIR) and infinite impulse response (IIR). FIR filters were chosen because they are easy to represent as matrices of coefficients, they offer linear phase characteristics, are stable and are easy to implement.

Filter design involves the calculation of the convolution mask h(x,y) so that the resulting filter has a specified frequency response. As the convolution mask is calculated, the output image O(m,n) is calculated as the convolution of the pixel intensities inside the window with h(x,y).

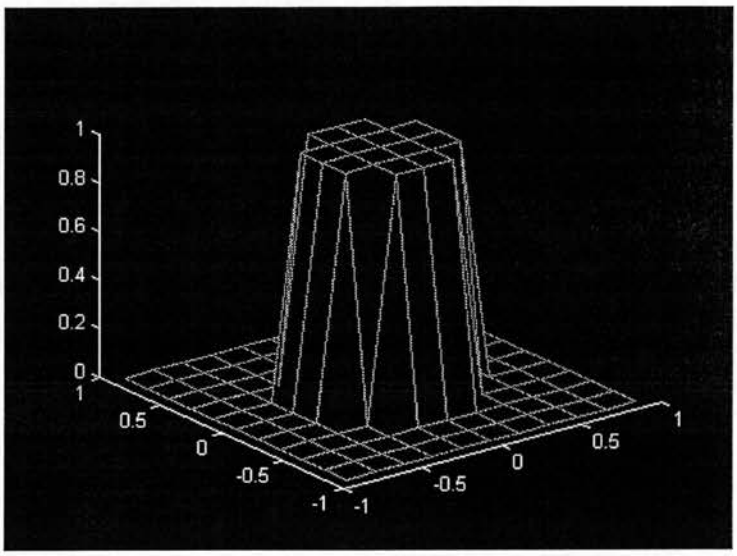
$$O(m,n) = \sum_{x=-k}^{k}\sum_{y=-k}^{k} h(x,y)\,I(m-x,n-y) \qquad (4.2)$$

The simplest FIR filter is the running average, with all the convolution coefficients equal to $1/(2K+1)^2$, where 2 k+1 x 2k+1 is the windows size. The running average filter can be expressed as

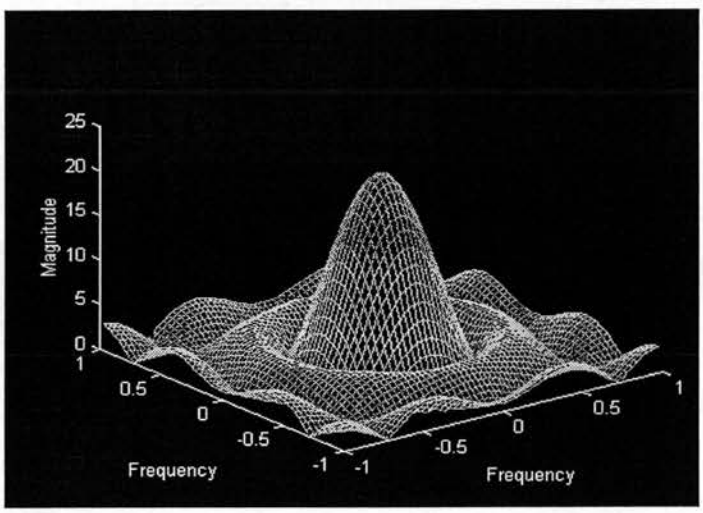
$$O(m.n) = (2k+1)^{-2} \sum_{x=-k}^{k}\sum_{y=-k}^{k} I(m-x,n-y) \qquad (4.3)$$

The impulse response of an 11 x 11 averaging filter is displayed as a 2D plot in Figure 4.3(a) and its power spectrum is plotted in Figure 4.3(b). They can be implemented without any multiplication and offer the maximum noise reduction for a given window (or kernel) size. However, as can be seen from Figure 4.3(b) their power spectrum exhibits ripple in the stopband zone and, consequently, a certain amount of high frequencies are allowed to pass through the filter.

The other type of linear filter is the Gaussian filter which is a local smoothly operator based on the probability of the Gaussin distribution (Lee, 1983).



(a)



(b)

Figure 4.3 : Linear filtering. (a) - A 2D plot of the impulse response of an 11x11 averaging filter. (b) - power spectrum.

There are many other forms of linear filter aimed at achieving improved filtering with a well-behaved frequency response (Oppenheim and Schafer, 1975; Rabiner et al., 1976; Kato and Matsumoto, 1982; Mclellan and Chan, 1977; Fiasconaro, 1979; Lee, 1980; Davis and Rosenfeld, 1978).

Finally this type of filter was used to enhance ultrasound images. (Loupas, 1987; Petrovic et al, 1986; Bartrum and Crow, 1980).

4.4.2.2. Non-Linear Filters

Non-linear filters can be median filters (Morikubo et al, 1985) or outlier removal (Schuster et al, 1986).

Median filtering uses the values of the pixels contained in the pixel neighbourhood to determine the new value given to the pixel of interest. However, it does not algorithmically calculate the new pixel. Because of its effectiveness as a noise suppressor filter, the median has found numerous applications in signal and image processing, including ultrasound imaging and other medical imaging. (Kunda et al, 1984; Loupas et al, 1988; Ritenour et al, 1984). Numerous methods to describe it theoretically and improve its speed are found. (Gallagher and Wise, 1981; Nodes and Gallagher, 1984; Pitas and Venetanopoulos, 1986; Rao and Rao, 1986; Wendt et al, 1986; Yli-Harja et al, 1988).

Median filtering is performed by replacing the value of the picture element at the centre of a small region with the median of all values occurring within the region. For example, if the window includes 2k+1 terms I_i (i = 1, ..., 2 k +1), the median O_{med} is equal to

$$O_{med} = \{ I_1 , I_2 , ..., I_{2k+1} \} = I_{(k+1)} \qquad (4.4)$$

where : $I_{(k+1)}$ = is the (k+1)th largest term.

4.4.2.3. Adaptive Filters

The major drawback of both linear and non-linear filters is that they are space-invariant, in other words they perform the same type of operation on all parts of an image. The problem induces blurring and loss of image detail. To overcome this limitation , adaptive or space-variant techniques have been proposed to reduce noise and keep the image detail (Dickinson, 1982; Bamber and Daft, 1986; Loupas et al, 1987). There are different types of adaptive filtering (Chan and Lim, 1985; Ding and Venetsanopoulos, 1987; Loupas et al, 1988; Kuan et al, 1985; Pomalaza-Raez and McGillem, 1984).

One of the common adaptive filters is the Wiener filtering. The adaptive Wiener filter is more selective than the linear filter, preserving edges and other high frequency parts of an image. However, Wiener filtering does require more compilation time. This filter tailors itself to the local image variance, i.e. the smoothing is inversely proportional to the variance. It works best when the noise is additive noise such as Gaussian White noise.

4.5. Image Display Programs

The last step of the whole process is image display. High-quality image display demands a certain minimum acceptable level of graphics adapter functionality. In our system, a Mitsubishi 640 x 480 colour monitor with VGA card is the image display. As mentioned in the introduction that the package used in programming the C-scan system is the Borland C ++ Version 4.0. Hence all access to VGA is through Borland C ++'s graphic library using the available three modes. Unfortunately, the C-scan system's resolution is 8-bit, i.e. 256 - grey shades. This mode is not supported via Borland C ++ and only supported via assembler language. This mode of operation is 256 - gray shades, 320 by 200. However, the maximum gray scale which can be used is a 64-level due to limitations in VGA card hardware. This gray scale is enough to represent our images. Another consideration is image manipulation. To be able to manipulate the images, these images should be put into a specific file format and it was decided to use the PCX format. This file format is a standard one, to allow the movement of images between applications and to provide file compression to save disk storage space. The file format is rigid, with a file header of fixed length

followed by the raster image data and an extended palette structure. The simplicity of this file format makes the code required to support PCX easy to use.

After making the bases as explained above, there were two main programs, one was included with image processing programs and the other was to compare the original raw image with the processed one. Programs listings can be found in Appendix B.

4.6. Diagnostic Programs

The construction of the C-scan system involves many stages. It is essential to test the operation of the system during all stages of construction. Many diagnostic programs were written in Borland C ++ to accomplish this objective. Software testing was done in steps. First, a routine was written for testing the operation of motors, their speed and resolution. Second, a routine was written to simulate real data to test saving and displaying data routines. Third, the data collection routine was tested using a well known signal. This test was repeated for different signals. During this stage a routine was written to calibrate the analogue to digital conversion card. Finally, after the C-scan system's construction, the whole system (hardware and software) was tested.

CHAPTER 5

EXPERIMENTAL RESULTS AND IMAGE PROCESSING TECHNIQUES FOR C-SCANNING

CHAPTER 5

EXPERIMENTAL RESULTS AND IMAGE PROCESSING TECHNIQUES FOR C-SCANNING

5.1. Introduction

The aims of this project have been laid out in section 1.2. The hardware and construction of a C-scan system have been described in detail in Chapter 3. The software design and specification of a C-scan system including data acquisition and image processing techniques have been described in Chapter 4. The feasibility of the prototype system is assessed experimentally in the present chapter, which also evaluates image processing algorithms used to improve the quality of the C-scan images. Assessment of the system's resolution is performed using a resolution phantom which was designed specifically for this purpose and generally for evaluating the prototype system.

The chapter starts by describing the resolution phantom in section 5.2. Then the system's resolution measurements based on the resolution phantom are presented in section 5.3. A number of image processing algorithms have been investigated and results can be found in section 5.4. Finally the chapter's conclusion is presented in section 5.5.

5.2. Resolution Phantom

To measure the scanner resolution a resolution phantom was constructed with two movable parallel plates of 18 cm x 23 cm made of perspex with a square hole of area of 10 cm x 10 cm in the centre of the top plate for scanning purpose. The separation between the two parallel plates can be varied in the range of 6 cm to Ø mm using a screw attached to the two parallel plates. The phantom stands on a base parallel to the two movable plates. After construction, tissue mimicking material (reticulated foam) of 18 cm x 16 cm area and height of 2.5 cm was attached to the top plate from below and another slab of tissue mimicking material of 18 cm x 16 cm area and height

of 2.5 cm was attached to the bottom plate from above with a circular hole of 3 cm diameter in the centre. Also a piece of carpet was stuck on top of the base to avoid the reflection from the hard base. The picture of the phantom is shown in Figure 5.1. The phantom features the ability to be scanned from the front position for axial measurement or from the side for lateral resolution measurement.

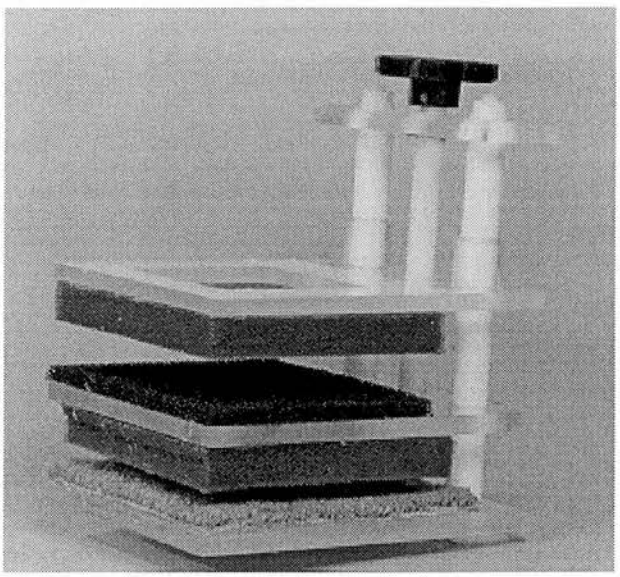


Figure 5.1 : A photograph of the resolution phantom.

5.3. Measurement of System Resolution

5.3.1. Method

To measure the system's resolution, the experimental rig was composed of 3 parts; the microcomputerised C-scan system, a large tank with probe holder and a small glass tank filled with water and fitted with a carpet in the base to avoid reflections. The height of the probe holder can be adjusted to place the face of the probe under water. To set up the axial resolution measurement, the separation between the two plates in the phantom was set to the desired distance in the range e.g. 1 mm. Then the phantom was put inside the small tank under water. After putting the phantom inside water, the tissue mimicking material stuck to the parallel plates in the phantom was

squeezed under water to get rid of bubbles. The next step was to set up the system firstly by aligning the probe to the area which is to be scanned by setting up the microcomputer controlled x and y motors followed by entering the scan parameters as explained in Chapter 4.

It is noted that the delay is chosen based on the depth of the first scan required using a sound speed of 1540 m/s which corresponds to that of soft tissue. Also the distance between lines and the distance between the samples are chosen to be compatible with the monitor resolution to keep the same aspect ratio to get the image displayed as scanned. Finally the scan is started by choosing SCAN in the main menu of the main program. When the scan finishes it gives a sound and then the stored image is examined. These steps are repeated in other axial resolution measurements using different slab separations.

5.3.2. Results

Axial resolution measurement for resolution test slab separations of 1 mm, 2 mm, 3 mm, 4 mm and 5 mm were done . It was noted that 1 mm and 2 mm can not be resolved and the axial resolution of the system is 3 mm. The lateral resolution can be controlled by changing the distance between the samples via the microcomputer controlled X-motor. Also the lateral resolution in the orthogonal direction can be controlled by the distance between raster lines via the microcomputer controlled Y-motor. The lateral resolution of an ultrasound system is normally determined by its beam width however where data is sparse the separation of the sampling points is more important. Hence the system resolution is defined by three parameters; axial resolution along ultrasound beam, lateral resolution between sample points along the scan line and lateral resolution between ultrasound raster lines.

5.4. Comparison between Raw 2D C-scan and Processed 2D C-scan Images

Since our aim is the study of the feasibility of real-time 3D ultrasound imaging, it was decided initially to start investigating the main argument mentioned in the aims of thesis using a simple C-scan image which suffers from the same sparsity of echo data as 3D scans. This section starts by describing the method and is followed by the results of slow scan imaging and simulated fast scan

imaging before and after using image processing. The comparison between raw 2D C-scan images and processed 2D C-scan images is followed by discussion of results.

5.4.1. Method

A C-scan ultrasound imaging system was used with the same arrangement used in resolution measurements in section 5.3.2. The steps to acquire the raw images are similar to the steps used to acquire images for resolution measurements in section 5.3.2. The experiment selected for this study is based on imaging the hole in the resolution phantom. To accomplish this study, sets of images must be prepared. First, slow scan images with lots of data were acquired and second, fast scan images with limited data were simulated by formatting the raw data using a written program (format program code, Appendix B). The different images with limited data were obtained by acquiring every second line, every third line, every fourth line and finally every fifth line.

Three techniques of image processing described in Chapter 4 were used; namely linear filters (averaging and Gaussian filters), non-linear filters (median filter) and adaptive filters (Wiener filter) using my programs (Appendix B) and MATLAB from the MathWorks (USA). The simulated fast scan image was pre-processed first using interpolation and then the image processing algorithms were applied to every image of the different fast scan images with limited data. The images were evaluated according to noise level, contrast, edge definition and then overall quality as explained in section 4.4.2. Evaluation was accomplished by a physicist with long experience in medical ultrasound who can evaluate the processed images from the technical point of view.

5.4.2. Results

In this section, a set of original and processed images of one of the C-scan images which was acquired in the focus of the transducer are presented. For comparison purposes Figure 5.2 shows a sketch of scanned slice, original slow scanned image with all the data and its histogram to show the frequencies at which the different shapes of gray are present. To achieve real-time scanning, scanning time must be reduced to the minimum and the reduction in data image quality compensated using processing techniques. Figure 5.3 shows the reduced image with every second line and the effect of applying the different image processing techniques. Images with more

reduced data i.e. with every third line, every fourth line and every fifth line; and the effect of applying the different image processing techniques for every image are presented in Figure 5.4 to Figure 5.6 respectively. A comparative, qualitative evaluation was carried out of the different image processing techniques applied to improve the quality of the simulated fast scanned images which suffer from sparsity of echo data. Similar studies were carried out using a triangular and a bifurcation test phantoms constructed from tissue mimicking material as shown in Figures 5.7 to 5.16.

5.4.3. Discussion

The results of the evaluation in general show acceptable images after processing even although there is sparsity of data in the unprocessed image compared to the original image with full of data. It is noted that the above is true for the different types of reduced image but the image starts to be distorted when the image has only the fifth line.

Regarding image processing techniques, it is noted that the linear filter and specifically the low-pass filter when applied as a second stage after interpolation to the data reduced image gives a smoother image than the original image which on the other hand suffers from blurring and loss of image detail.

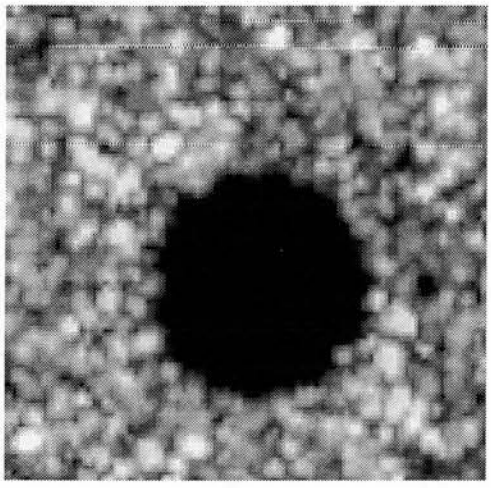
Non-linear filters and specifically, the median filter is able to suppress noise more that the linear filters do and to preserve the edge definition. However the processed scans suffer from loss of genuine image detail when the window size is increased to provide adequate noise reduction. Experimentation with a number of images shows that the median filter is able to get rid of band patterns which are obtained after using interpolation and gives smoother images with well-defined edges between areas of different echo levels. It is noted also that the edge is not blurred but the object corners are rounded. This happens when an edge is crossed, one side or the other dominates the window and the output switches sharply between these values.

Finally, the Wiener filter which is one of the adaptive filters, gives similar results to the median filter but with slower execution time.

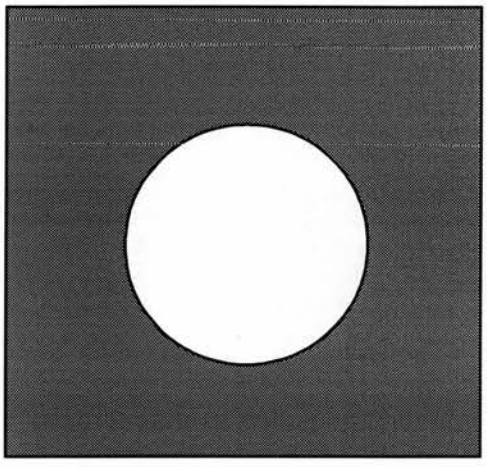
5.5. Conclusion

In this study, it has been shown that the system axial resolution is 3 mm due to the pulse length of the single crystal transducer. The comparison study of the quality of the processed 2D C-scan images, which suffer from the sparsity of echo data due to simulated real-time collection, with the original 2D C-scan images with full data shows that the image processing can overcome the problem of the sparsity of echo data to a significant extent. The comparison of the different image processing techniques suggests that the two stages of interpolation followed by a median filter is the best of the different techniques used in this study.

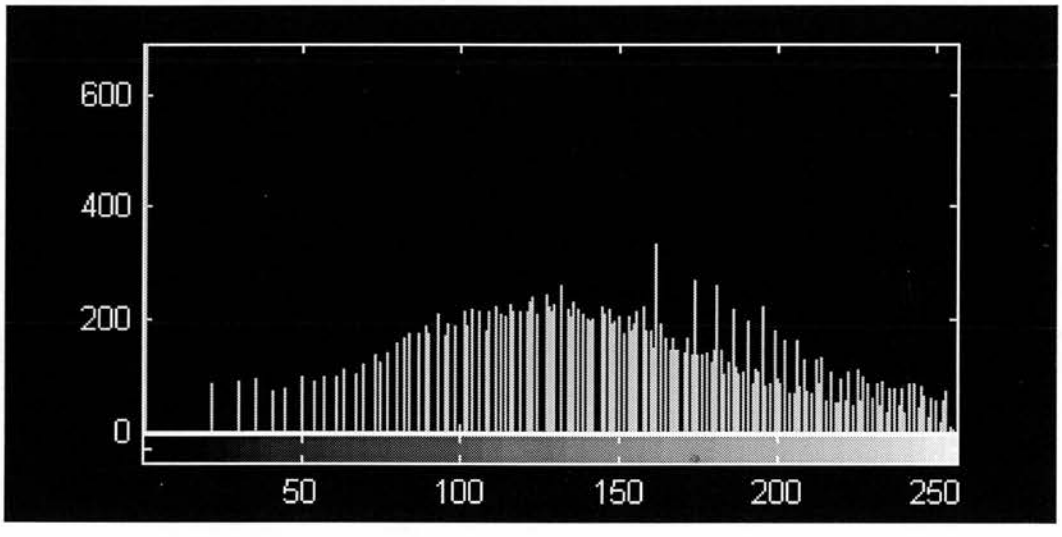
The results of this C-scan study encouraged the investigation of real-time 3D ultrasonic scanning in systems which would have a low density of echo data due to the finite time required to collect echoes.



(a)

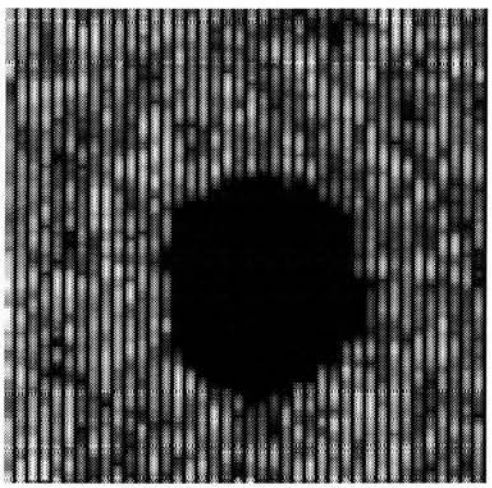


(b)

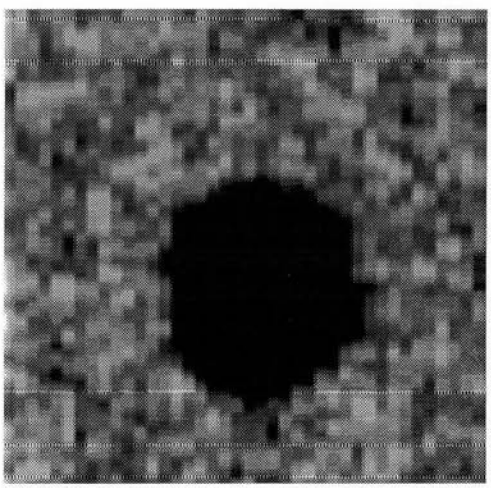


(c)

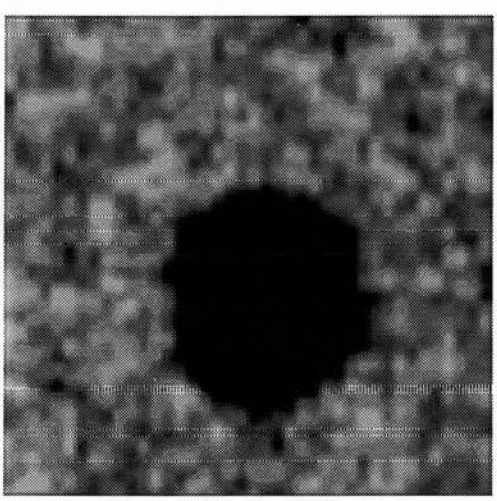
Figure 5.2 : The C-scan image of the resolution phantom. (a) - Original image (3 cm x 3 cm) acquired with every line. (b) - Sketch of the C-scan. (c) - Image histogram.



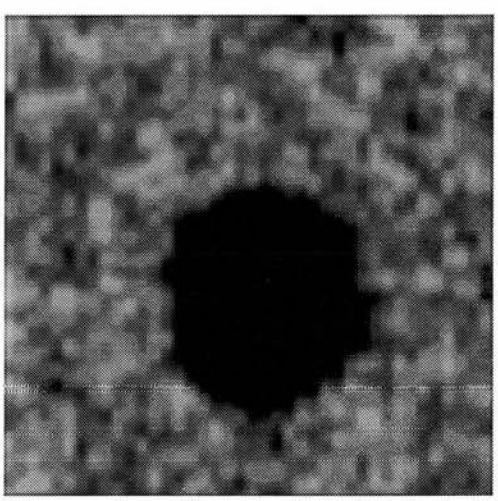
(a)



(b)

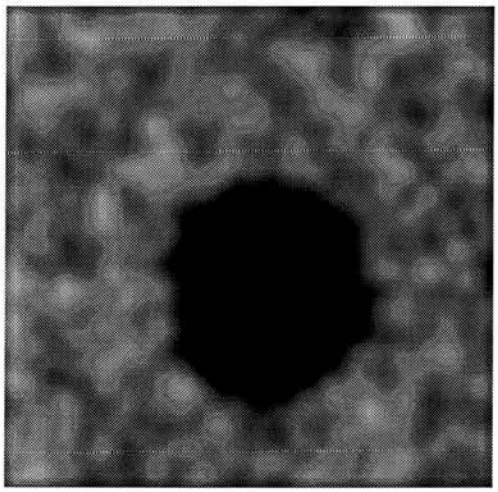


(c)

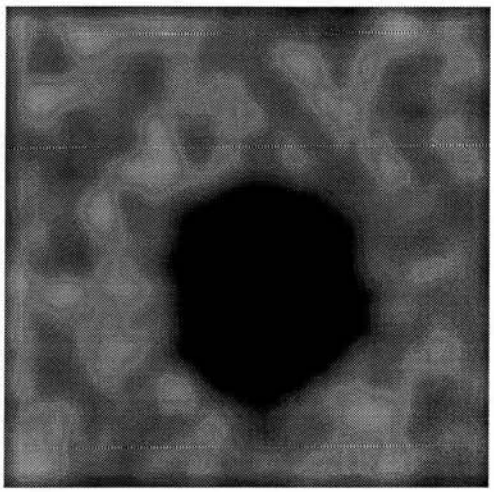


(d)

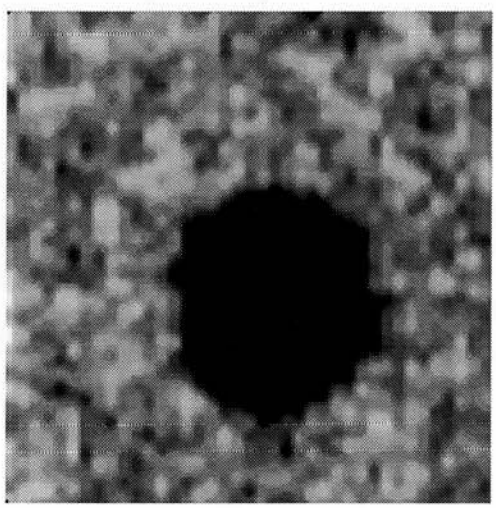
Figure 5.3 : The C-scan image of the resolution phantom. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every second line. (b) - After interpolation. (c) - After Gaussian filter with 7x7 window. (d) - After Gaussian filter with 11x11 window.



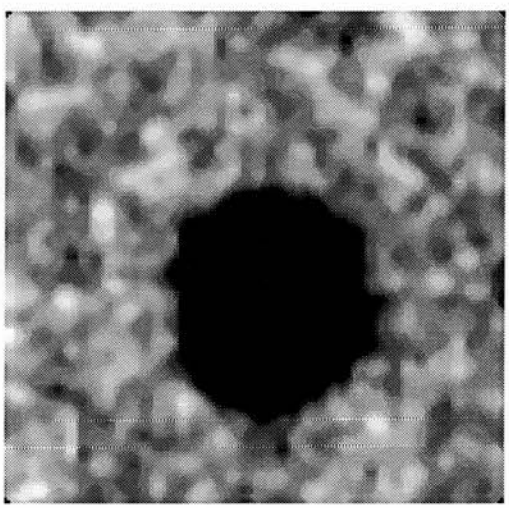
(e)



(f)

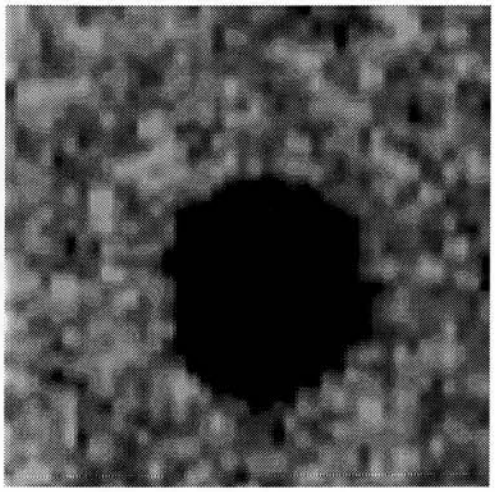


(g)

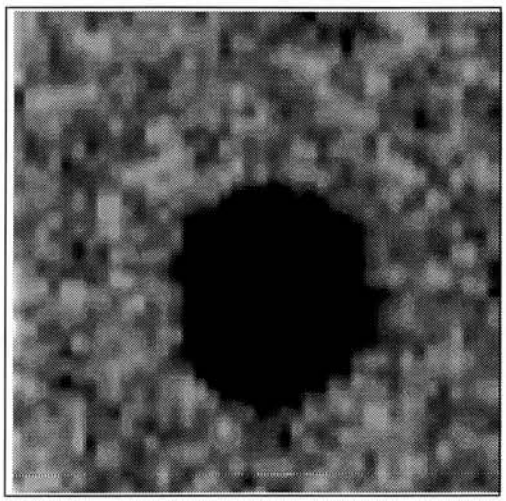


(h)

Figure 5.3 (contd.) : (e) - After averaging filter with 7x7 window. (f) - After averaging filter with 11x11 window. (g) - After median filter with 3x3 kernel. (h) - After median filter of 5x5 kernel.

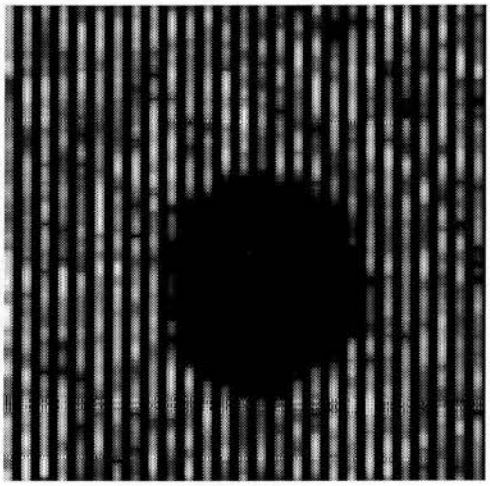


(i)

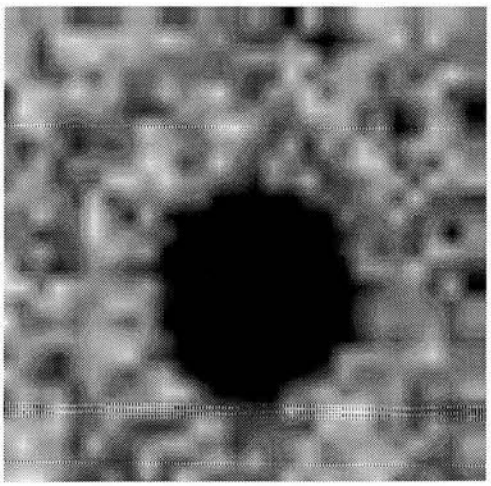


(j)

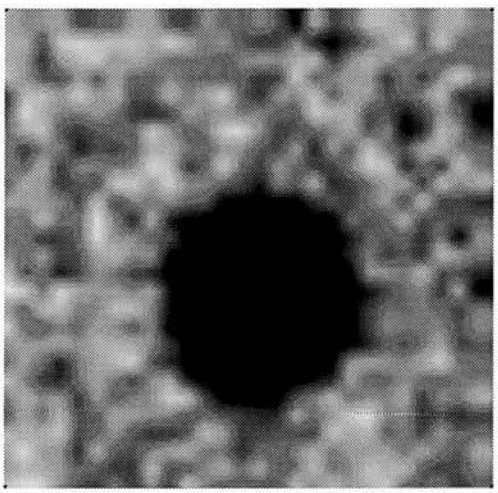
Figure 5.3 (contd.) : (g) - After wiener filter with 3x3 kernel. (h) - After wiener filter of 5x5 kernel.



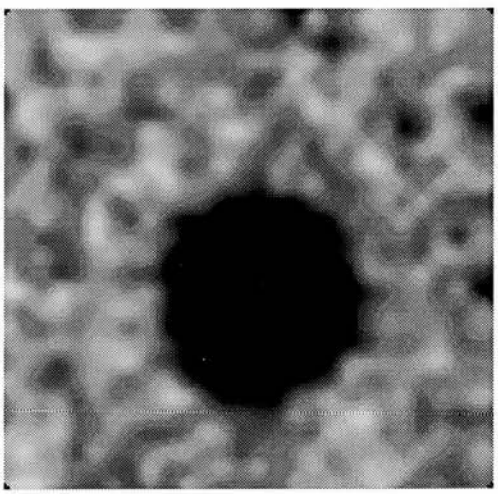
(a)



(b)

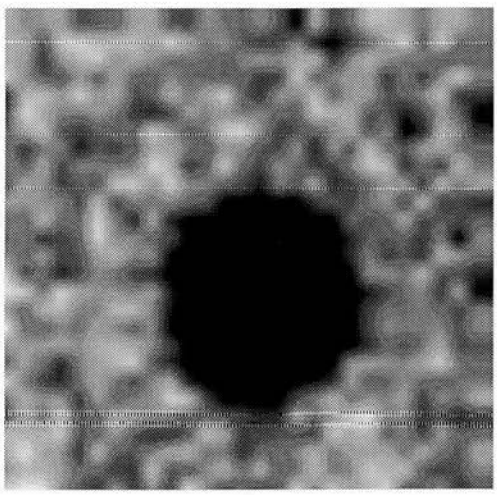


(c)

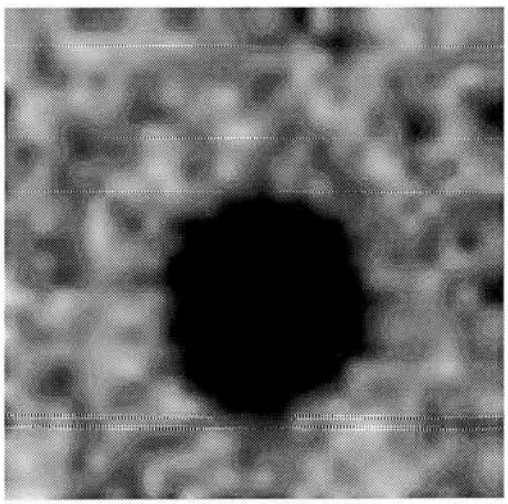


(d)

Figure 5.4 : The C-scan image of the resolution phantom. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every third line. (b) - After interpolation. (c) - After median filter with 3x3 kernel. (d) - After median filter with 5x5 kernel.

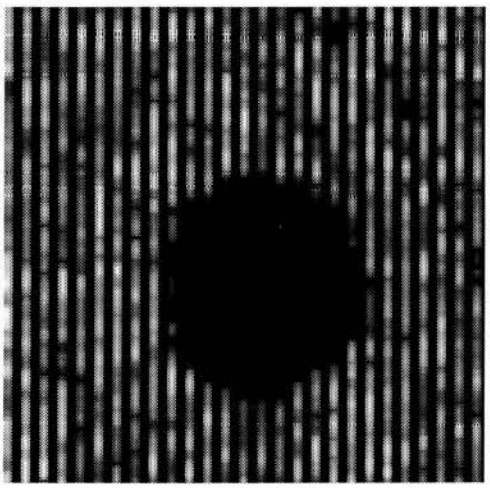


(e)

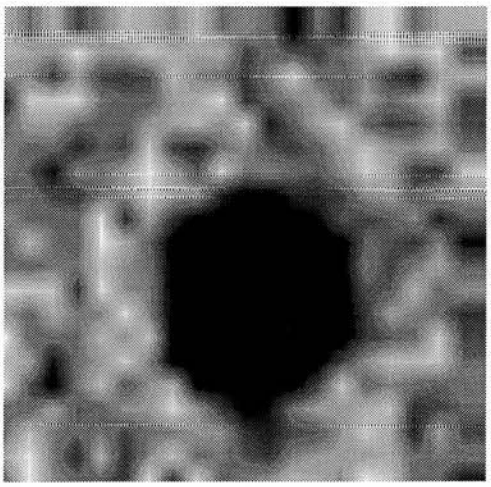


(f)

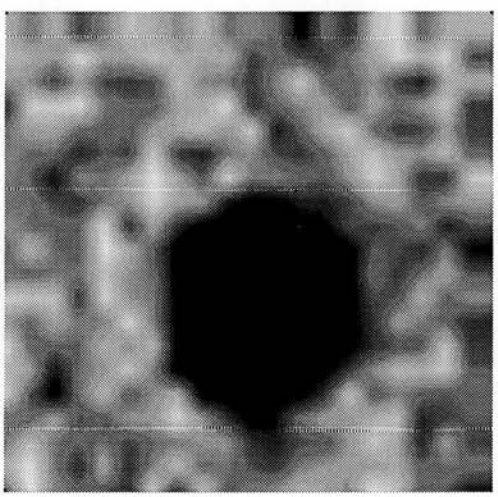
Figure 5.4 (contd.) : (e) - After wiener filter with 3x3 kernel. (f) - After wiener filter of 5x5 kernel.



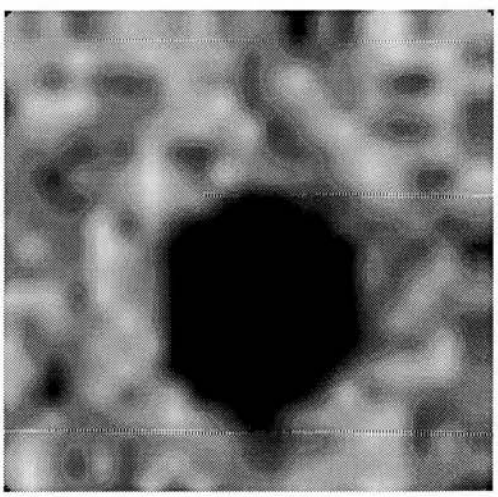
(a)



(b)

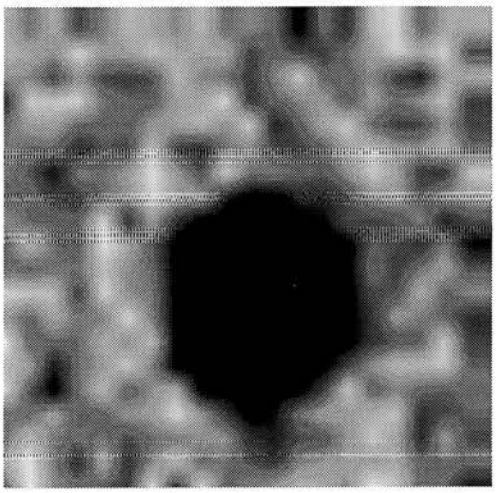


(c)

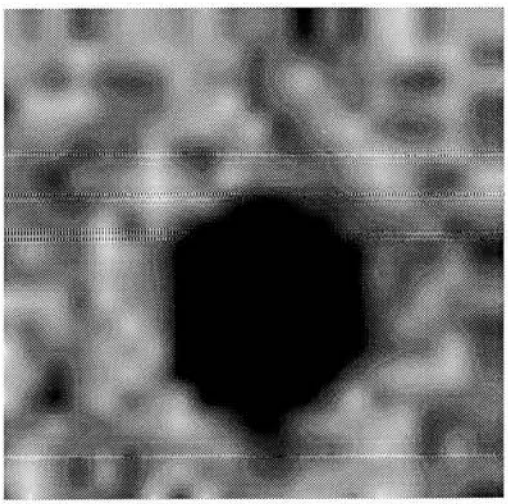


(d)

Figure 5.5 : The C-scan image of the resolution phantom. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every fourth line. (b) - After interpolation. (c) - After median filter with 3x3 kernel. (d) - After median filter with 5x5 kernel.

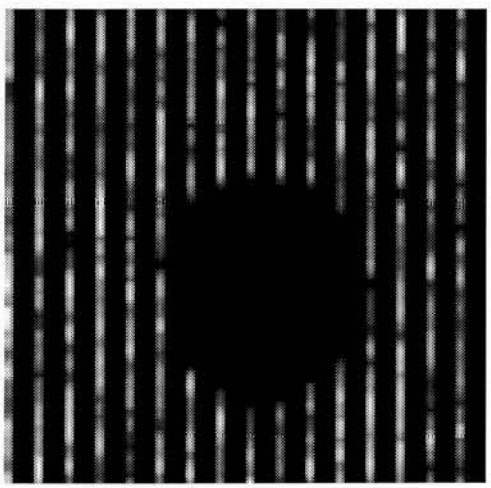


(e)

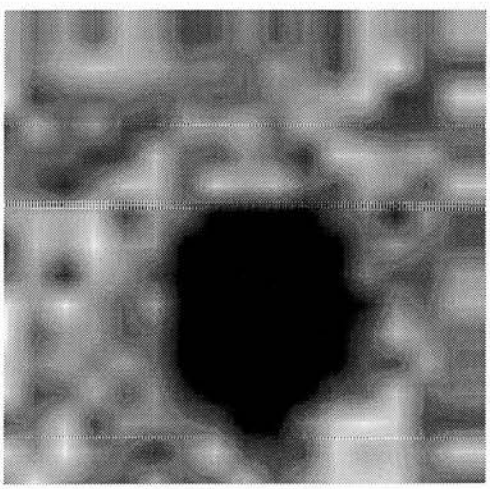


(f)

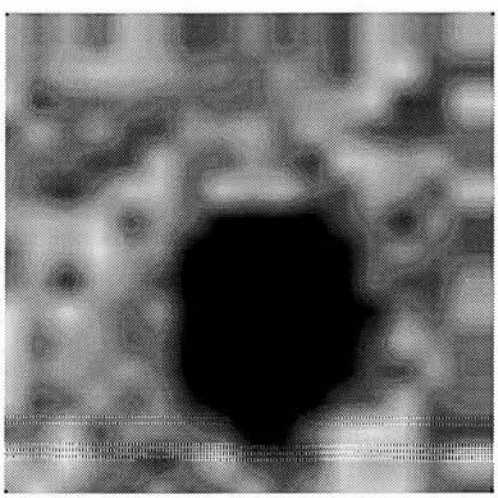
Figure 5.5 (contd.) : (e) - After wiener filter with 3x3 kernel. (f) - After wiener filter of 5x5 kernel.



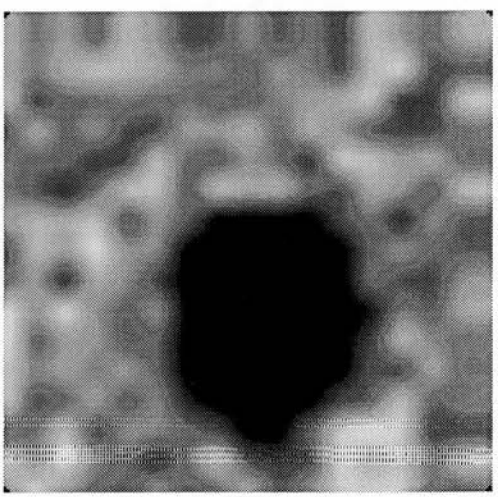
(a)



(b)

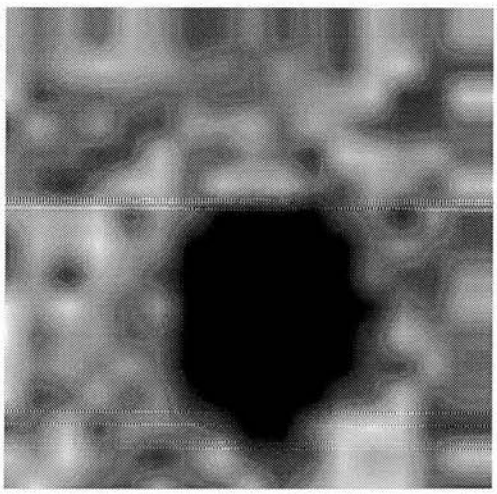


(c)

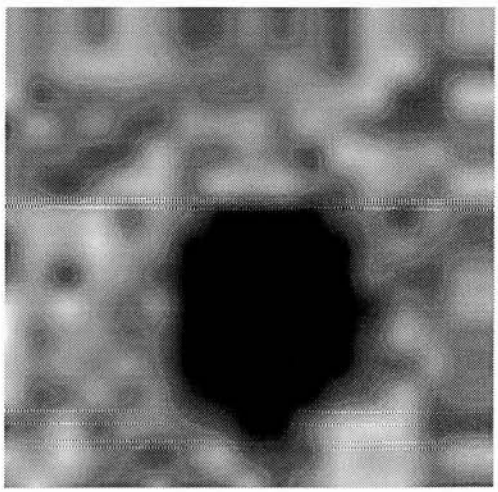


(d)

Figure 5.6 : The C-scan image of the resolution phantom. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every fifth line. (b) - After interpolation. (c) - After median filter with 3x3 kernel. (d) - After median filter with 5x5 kernel.

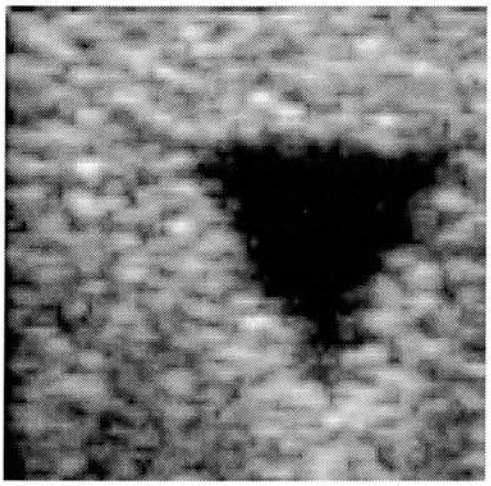


(e)

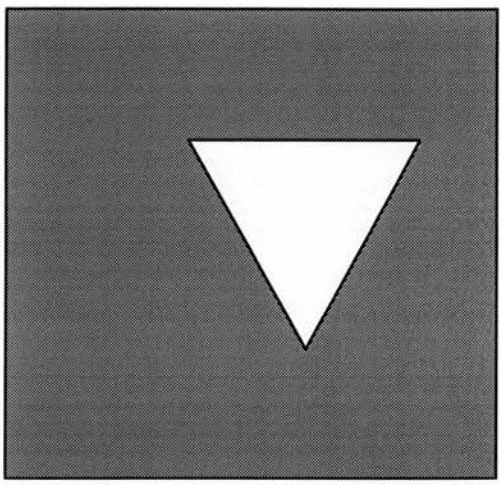


(f)

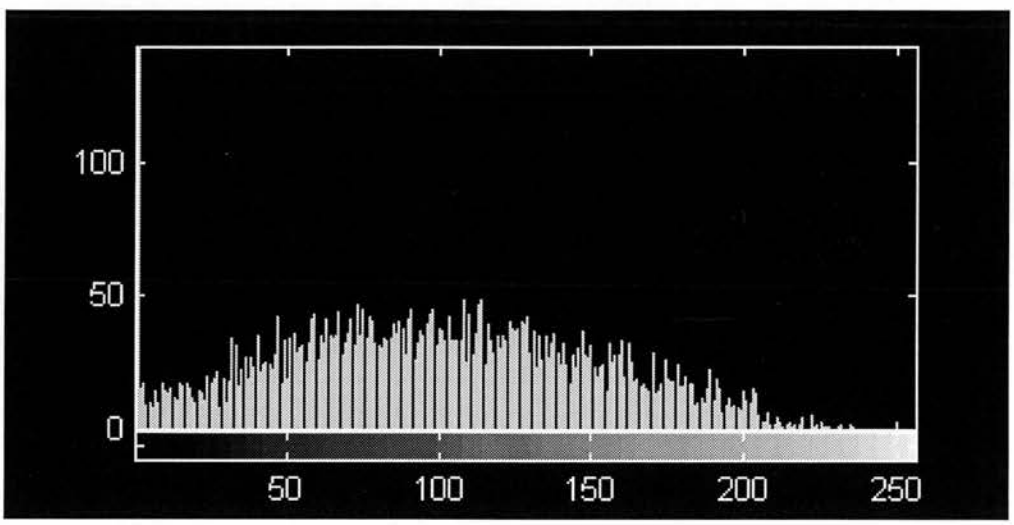
Figure 5.6 (contd.) : (e) - After wiener filter with 3x3 kernel. (f) - After wiener filter of 5x5 kernel.



(a)

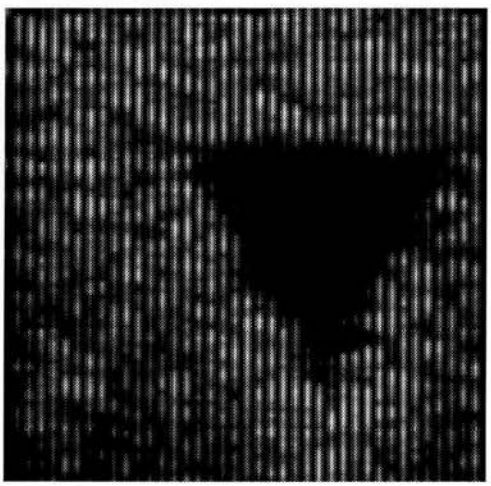


(b)

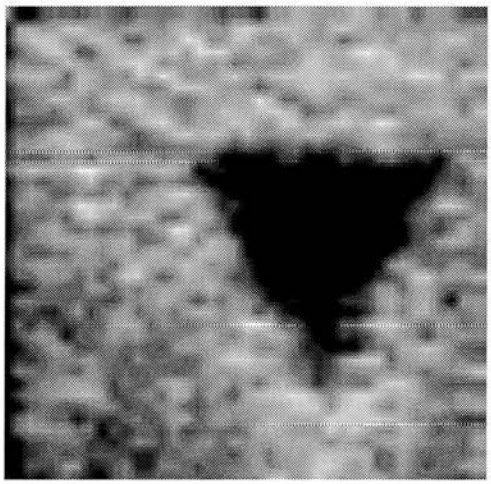


(c)

Figure 5.7 : The C-scan image of tissue mimicking material with a triangle shape. (a) - Original image (3 cm x 3 cm) acquired with every line. (b) - Sketch of the C-scan. (c) - Image histogram.



(a)

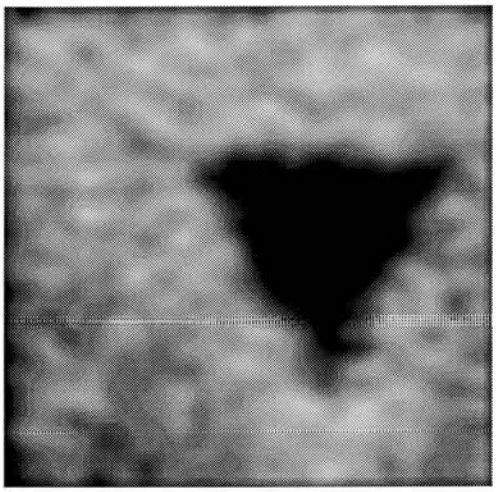


(b)

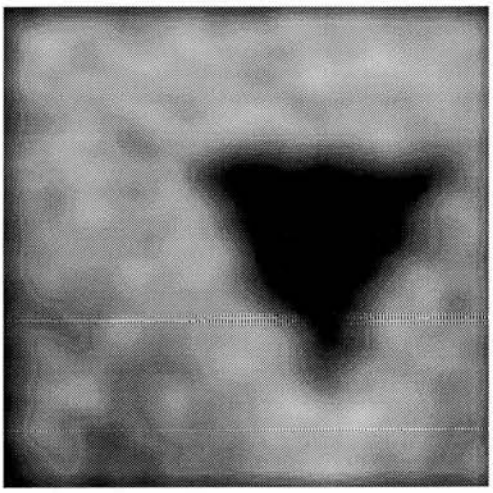
(c)

(d)

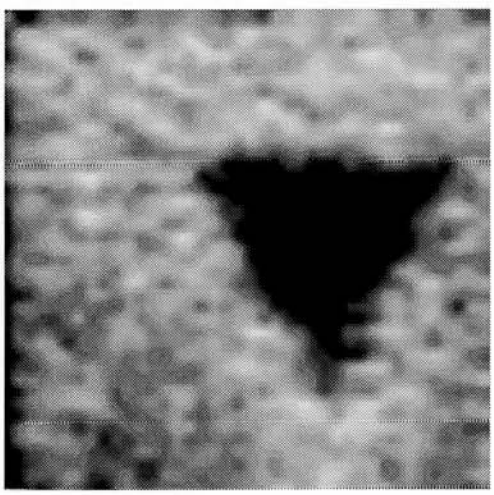
Figure 5.8 : The C-scan image of tissue mimicking material with a triangle shape. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every second line. (b) - After interpolation. (c) - After Gaussian filter with 7x7 window. (d) - After Gaussian filter with 11x11 window.



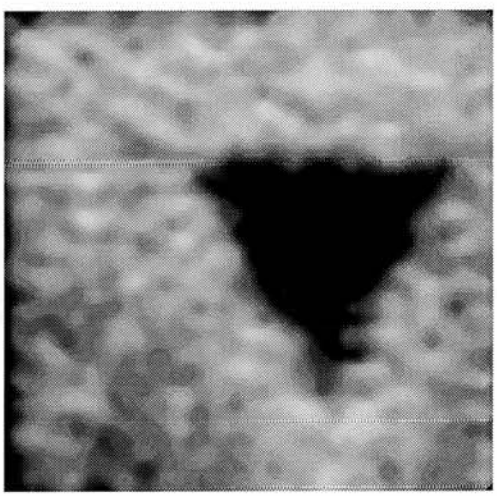
(e)



(f)

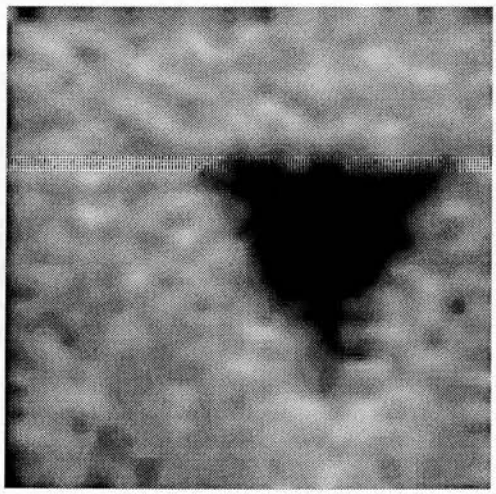


(g)

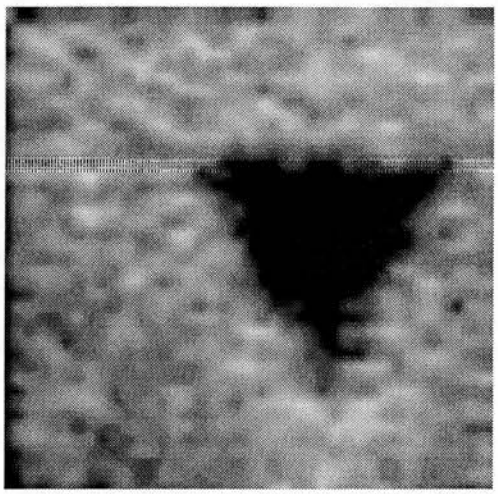


(h)

Figure 5.8 (contd.) : (e) - After averaging filter with 7x7 window. (f) - After averaging filter with 11x11 window. (g) - After median filter with 3x3 kernel. (h) - After median filter of 5x5 kernel.

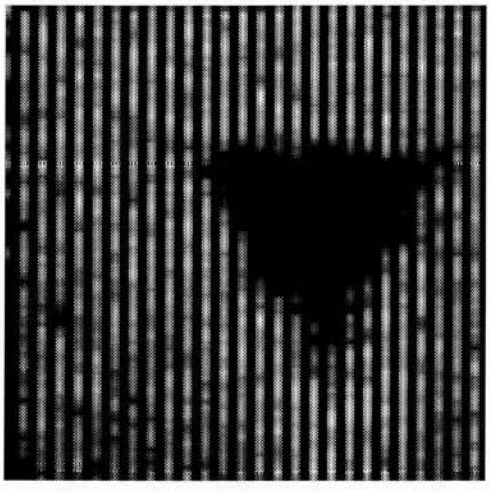


(i)

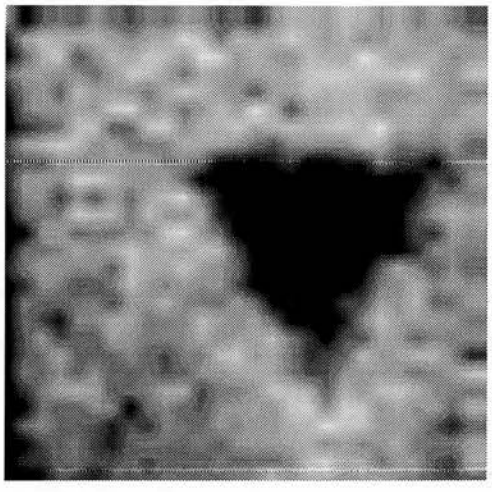


(j)

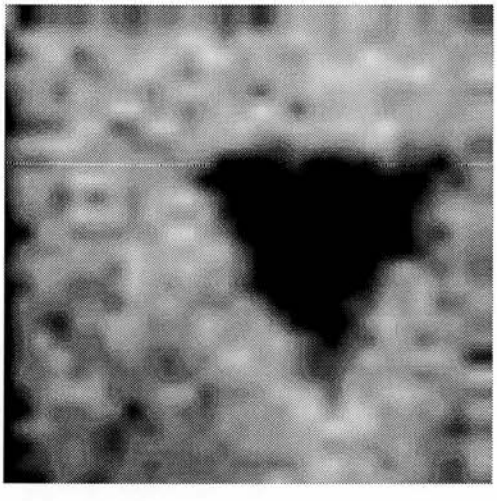
Figure 5.8 (contd.) : (i) - After wiener filter with 3x3 kernel. (j) - After wiener filter of 5x5 kernel.



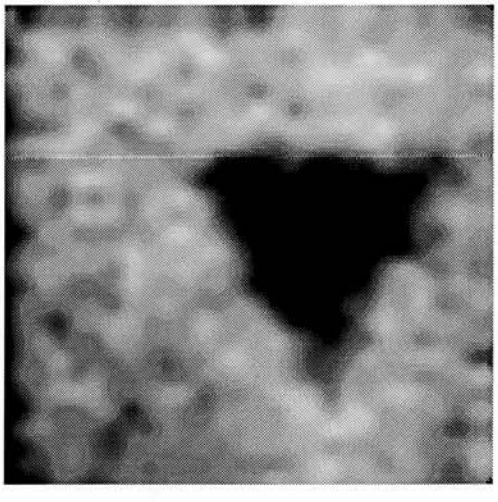
(a)



(b)

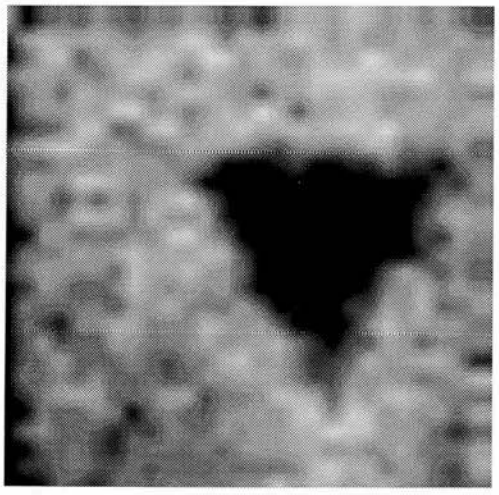


(c)

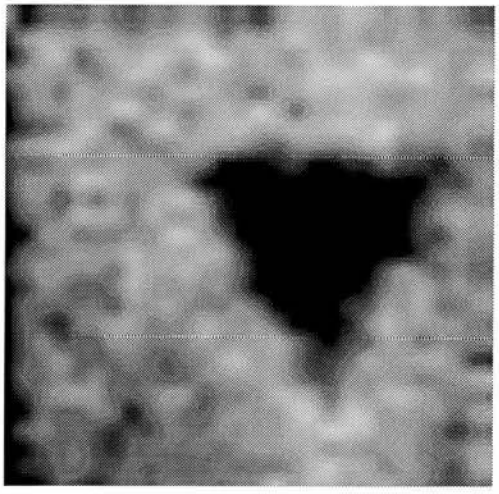


(d)

Figure 5.9 : The C-scan image of tissue mimicking material with a triangle shape. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every third line. (b) - After interpolation. (c) - After median filter with 3x3 kernel. (d) - After median filter with 5x5 kernel.

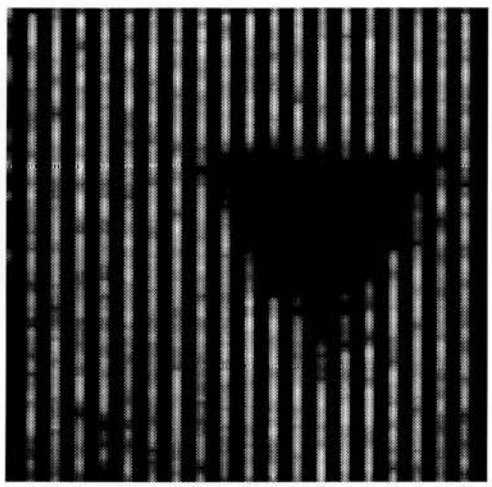


(e)

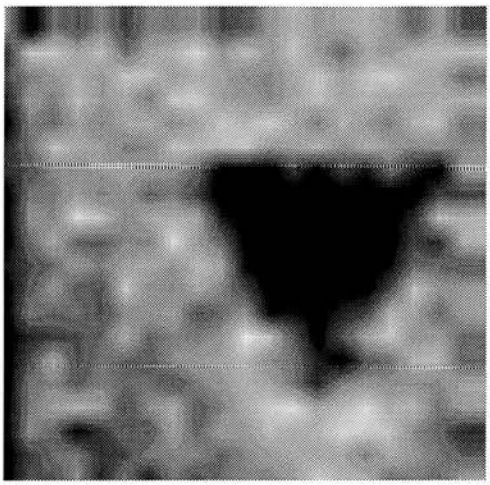


(f)

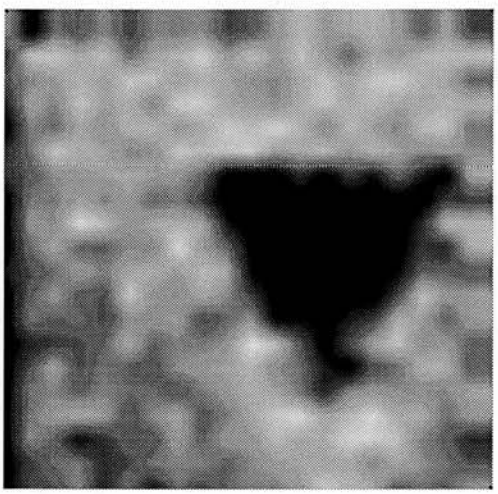
Figure 5.9 (contd.) : (e) - After wiener filter with 3x3 kernel. (f) - After wiener filter of 5x5 kernel.



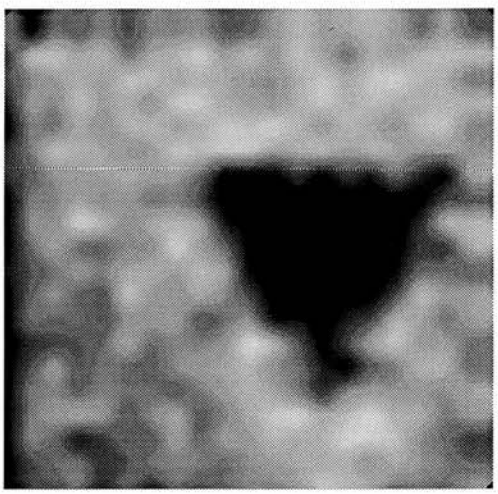
(a)



(b)

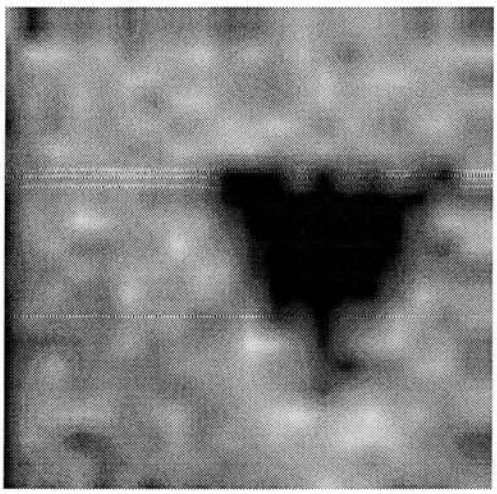


(c)

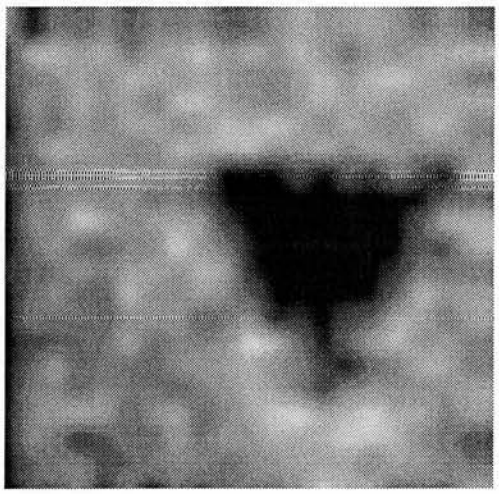


(d)

Figure 5.10 : The C-scan image of tissue mimicking material with a triangle shape. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every fourth line. (b) - After interpolation. (c) - After median filter with 3x3 kernel. (d) - After median filter with 5x5 kernel.

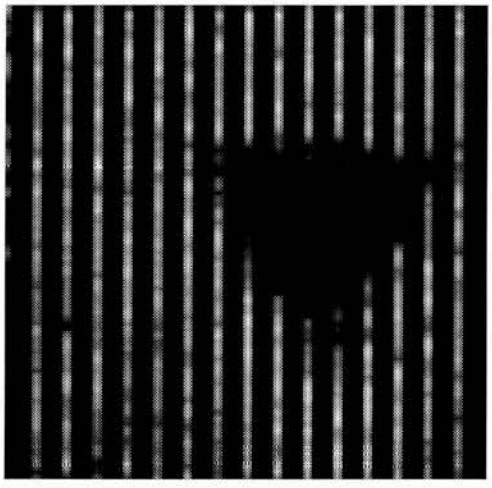


(e)

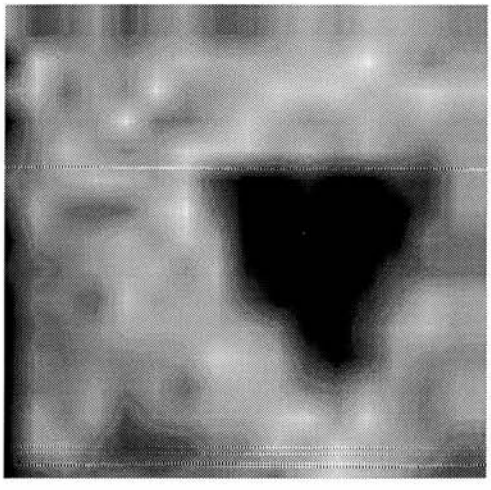


(f)

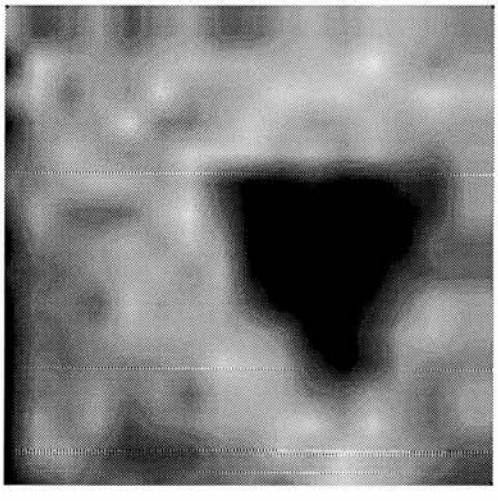
Figure 5.10 (contd.) : (e) - After wiener filter with 3x3 kernel. (f) - After wiener filter of 5x5 kernel.



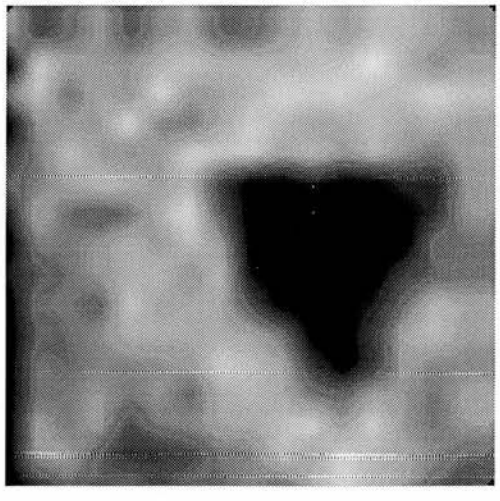
(a)



(b)

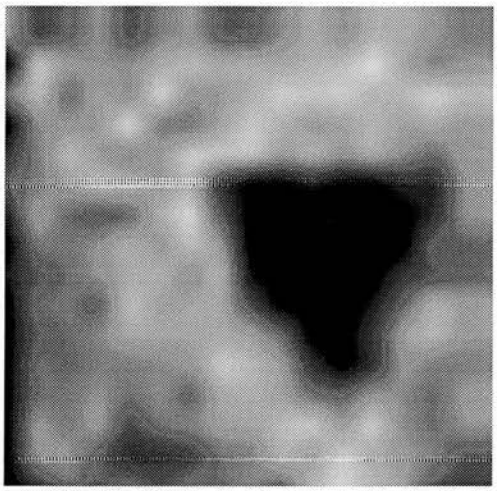


(c)

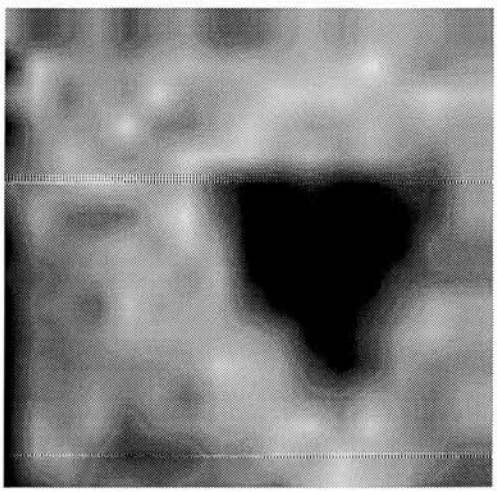


(d)

Figure 5.11 : The C-scan image of tissue mimicking material with a triangle shape. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every fifth line. (b) - After interpolation. (c) - After median filter with 3x3 kernel. (d) - After median filter with 5x5 kernel.

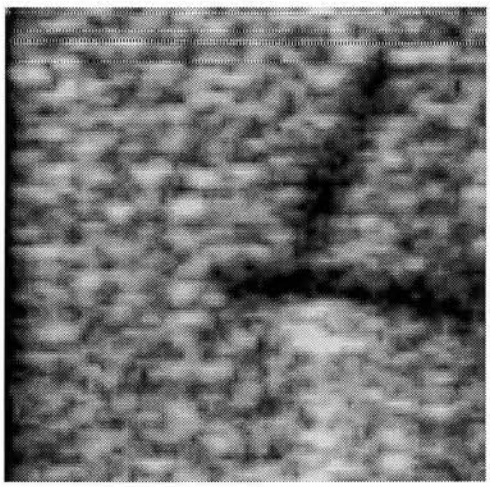


(e)

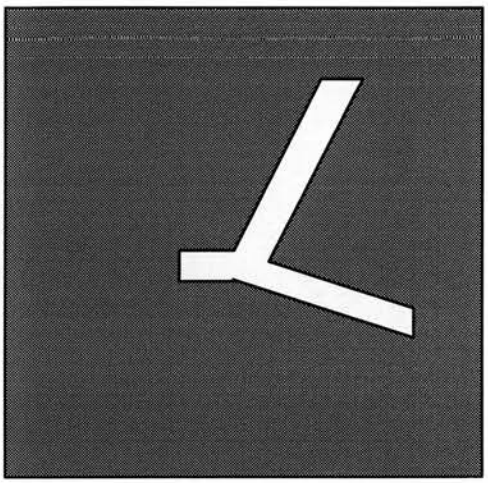


(f)

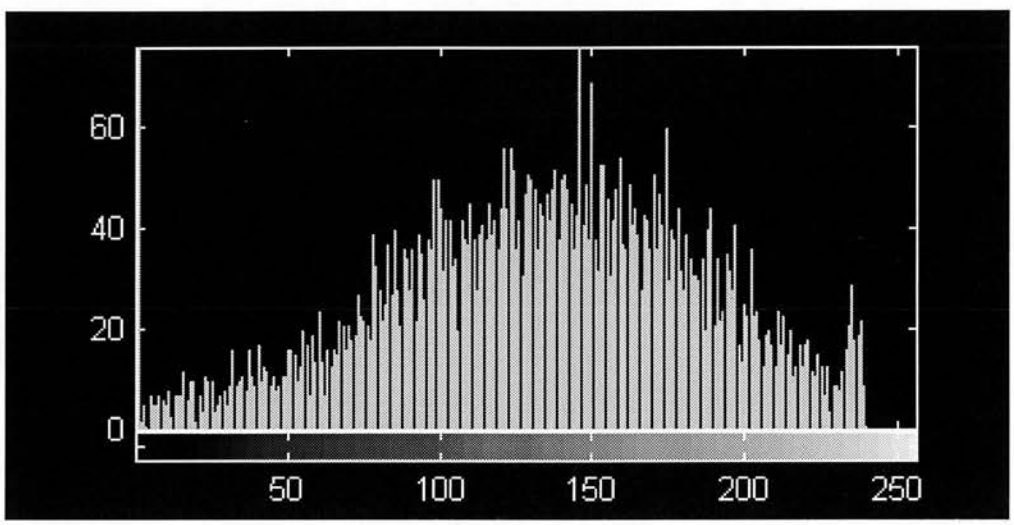
Figure 5.11 (contd.) : (e) - After wiener filter with 3x3 kernel. (f) - After wiener filter of 5x5 kernel.



(a)

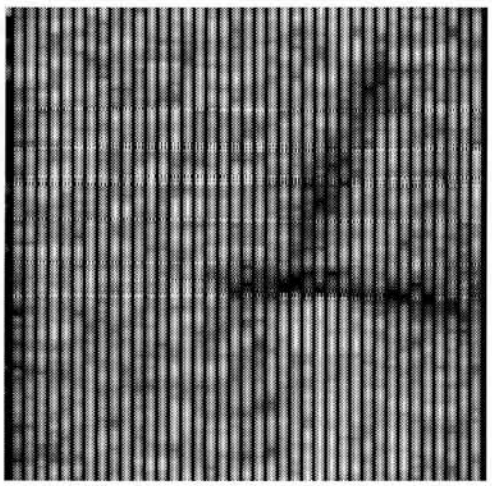


(b)

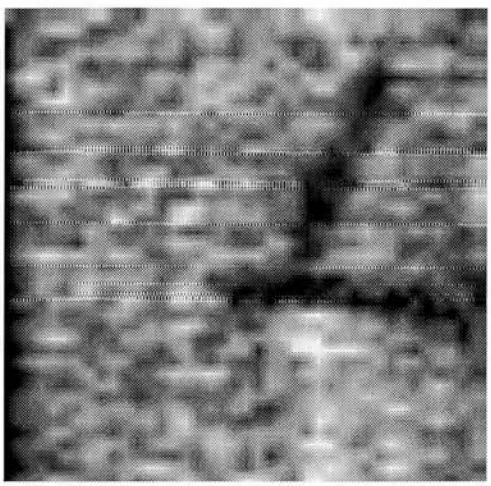


(c)

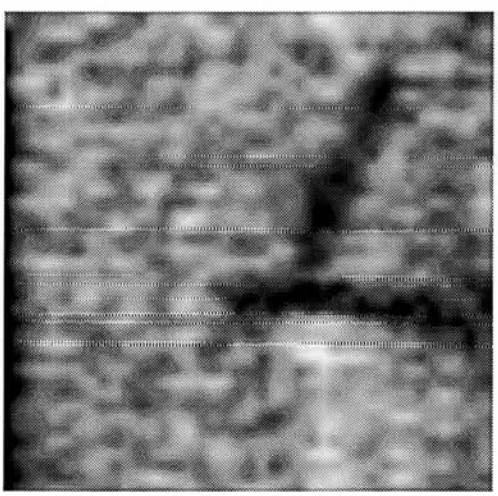
Figure 5.12 : The C-scan image of tissue mimicking material with a bifurcation shape inside. (a) - Original image (3 cm x 3 cm) acquired with every line. (b) - Sketch of the C-scan. (c) - Image histogram.



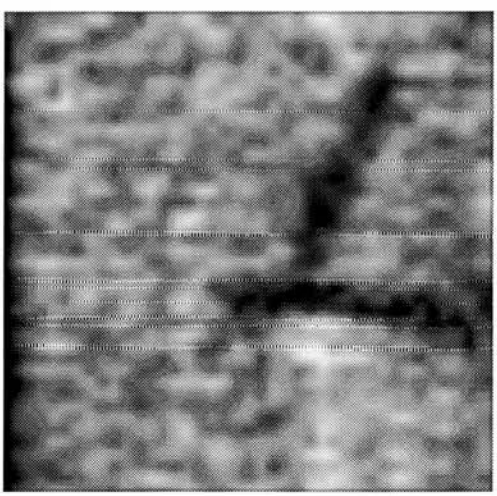
(a)



(b)

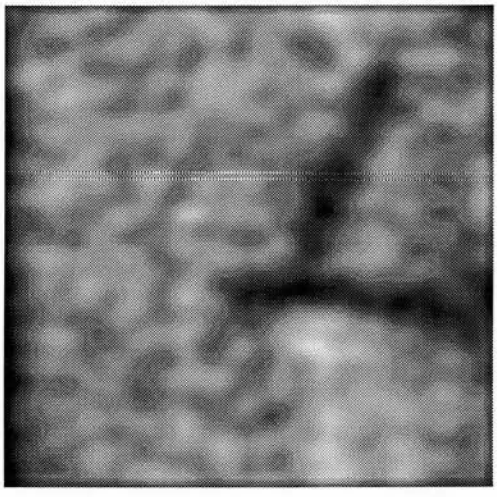


(c)

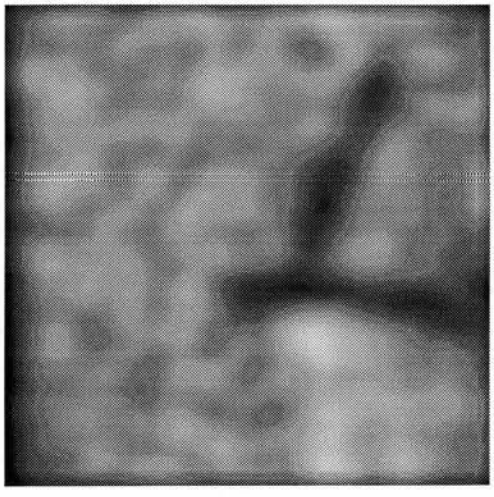


(d)

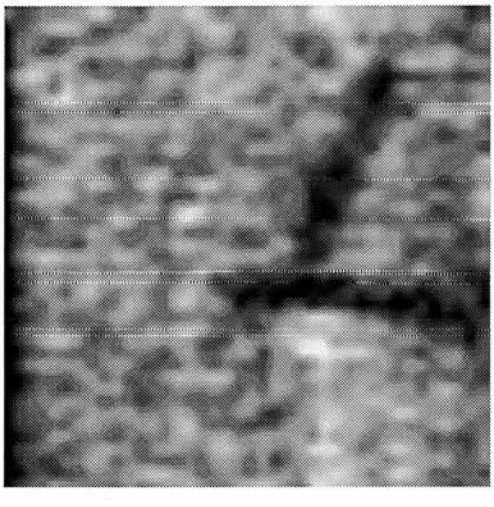
Figure 5.13 : The C-scan image of a tissue mimicking material with a bifurcation shape inside. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every second line. (b) - After interpolation. (c) - After Gaussian filter with 7x7 window. (d) - After Gaussian filter with 11x11 window.



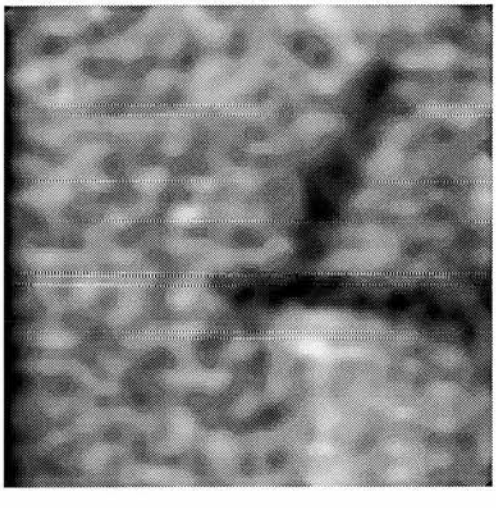
(e)



(f)

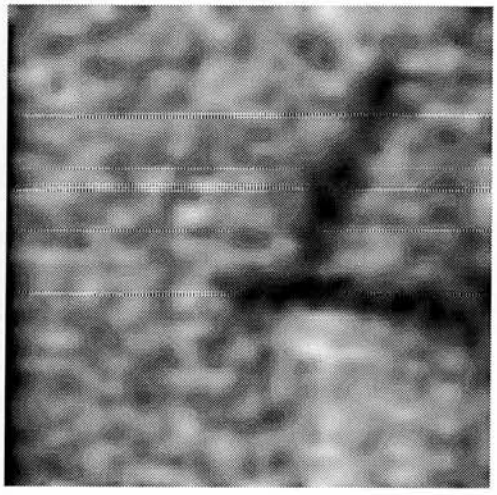


(g)

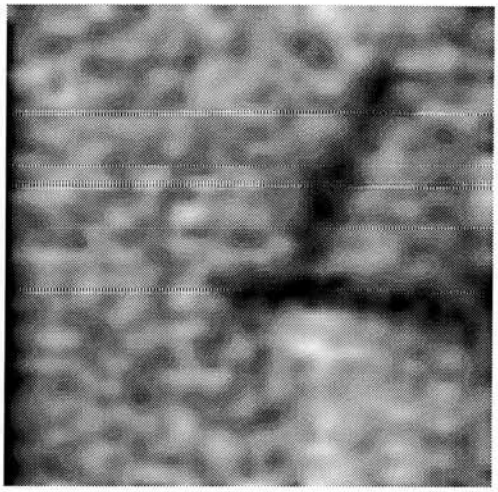


(h)

Figure 5.13 (contd.) : (e) - After averaging filter with 7x7 window. (f) - After averaging filter with 11x11 window. (g) - After median filter with 3x3 kernel. (h) - After median filter of 5x5 kernel.

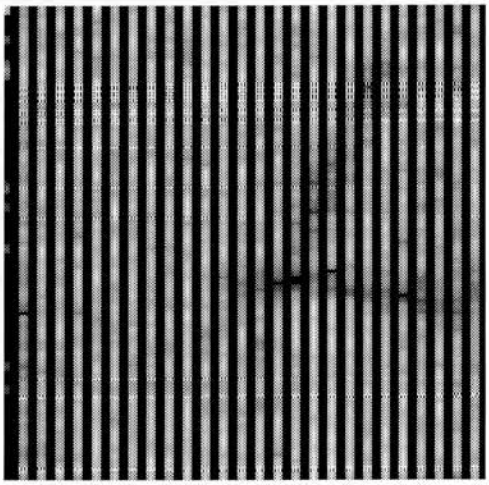


(i)

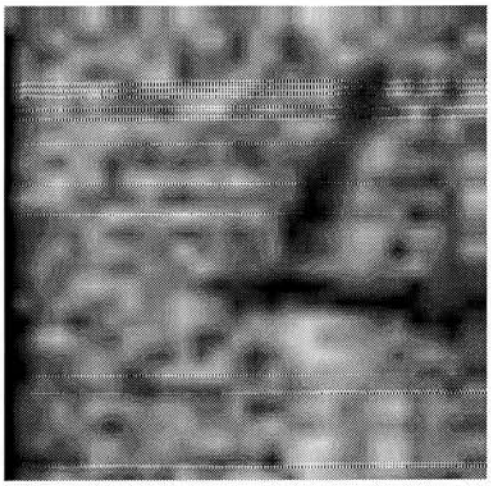


(j)

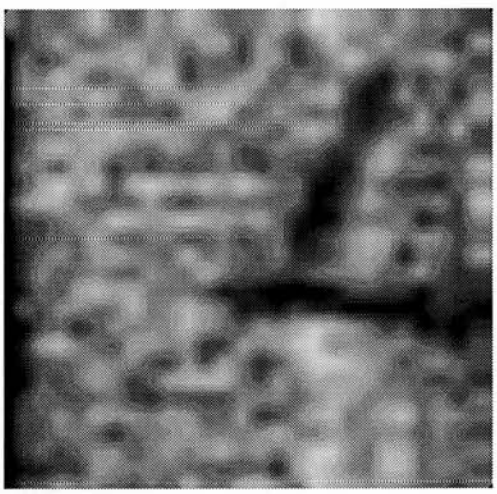
Figure 5.13 (contd.) : (g) - After wiener filter with 3x3 kernel. (h) - After wiener filter of 5x5 kernel.



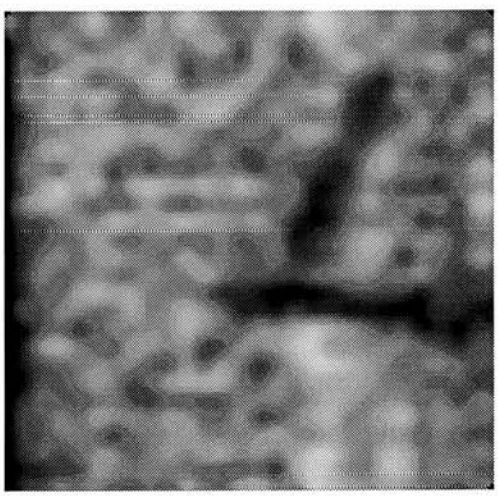
(a)



(b)

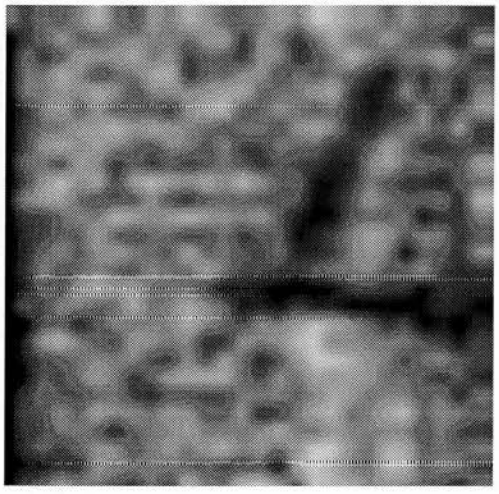


(c)

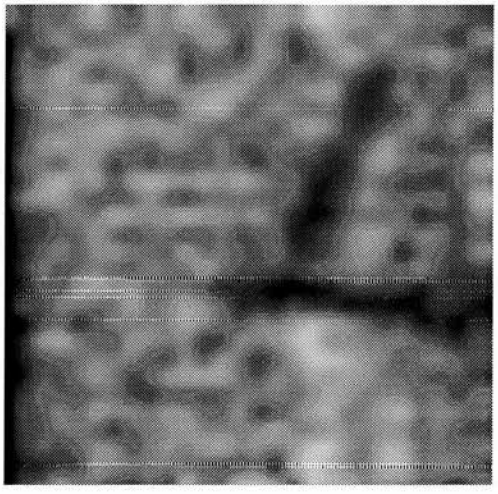


(d)

Figure 5.14 : The C-scan image of a tissue mimicking material with a bifurcation shape inside. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every third line. (b) - After interpolation. (c) - After median filter with 3x3 kernel. (d) - After median filter with 5x5 kernel.



(e)



(f)

Figure 5.14 (contd.) : (e) - After wiener filter with 3x3 kernel. (f) - After wiener filter of 5x5 kernel.

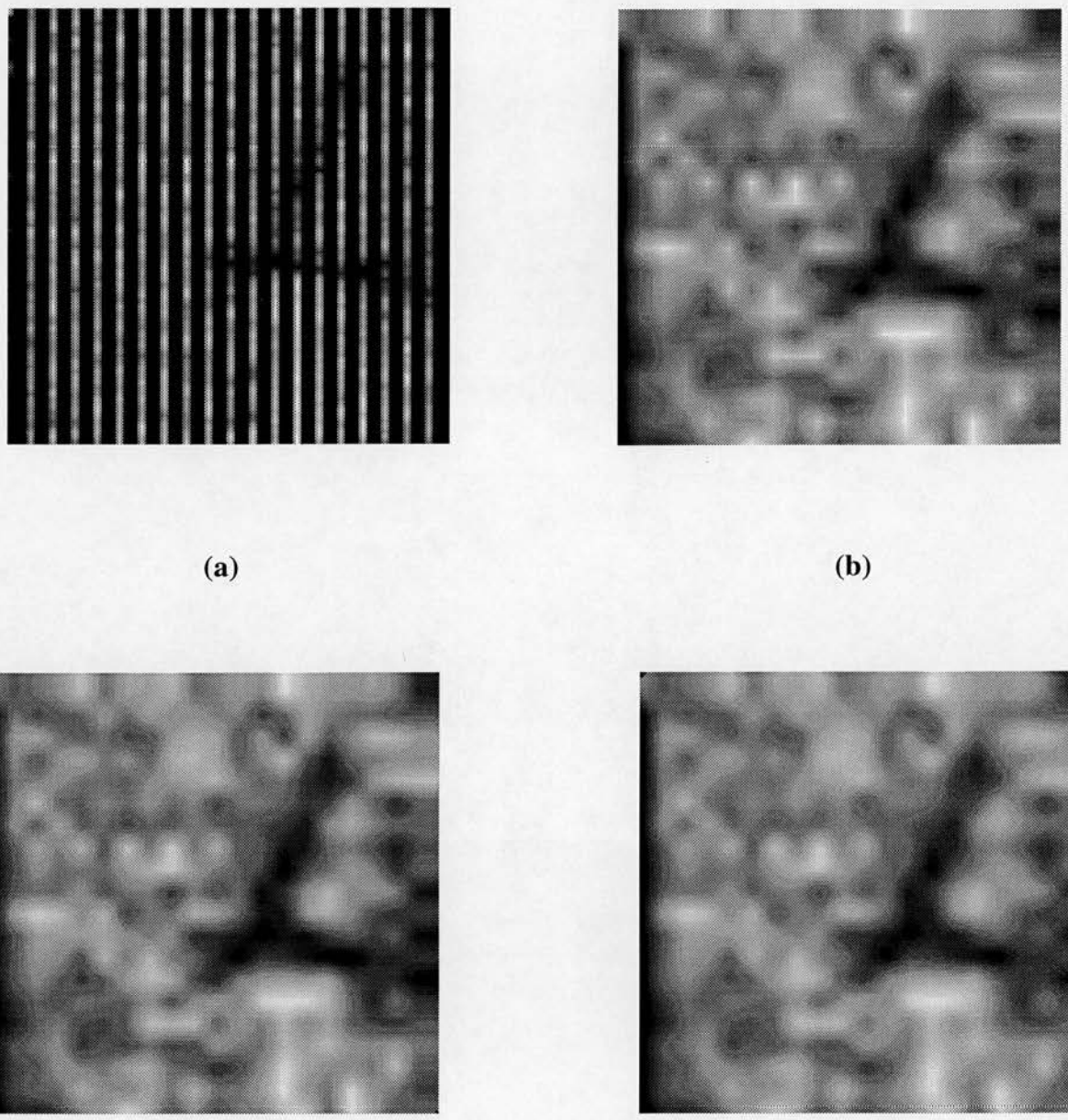
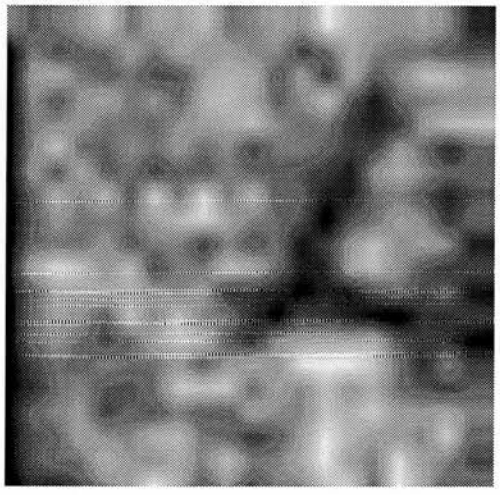
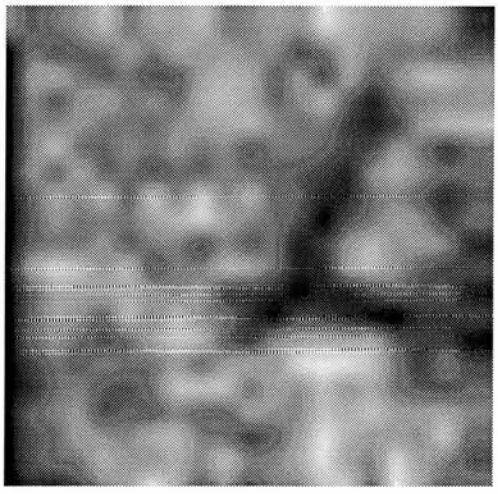


Figure 5.15 : The C-scan image of a tissue mimicking material with a bifurcation shape inside. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every fourth line. (b) - After interpolation. (c) - After median filter with 3x3 kernel. (d) - After median filter with 5x5 kernel.

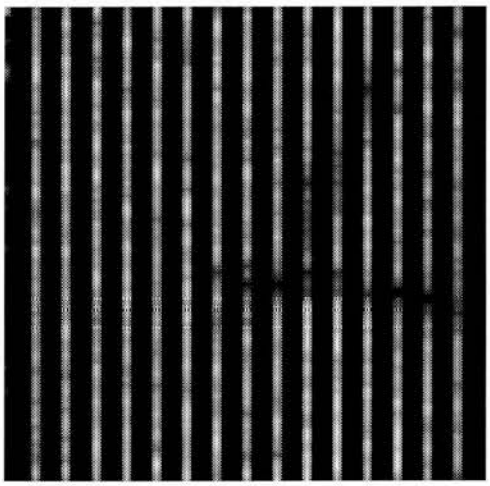


(e)

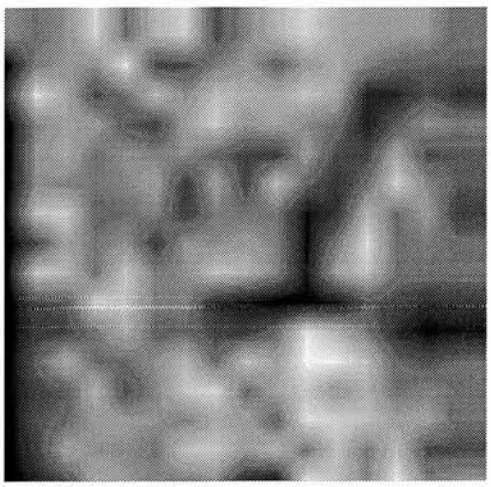


(f)

Figure 5.15 (contd.) : (e) - After wiener filter with 3x3 kernel. (f) - After wiener filter of 5x5 kernel.

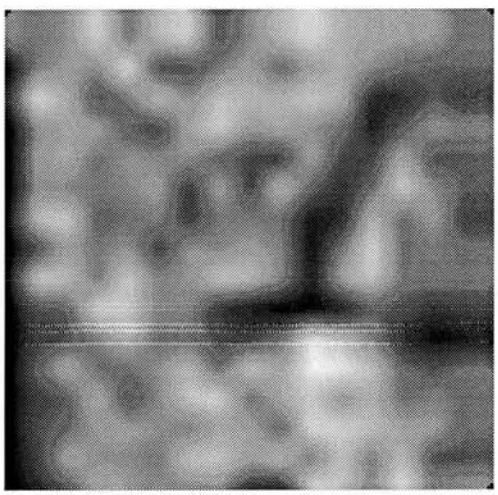


(a)



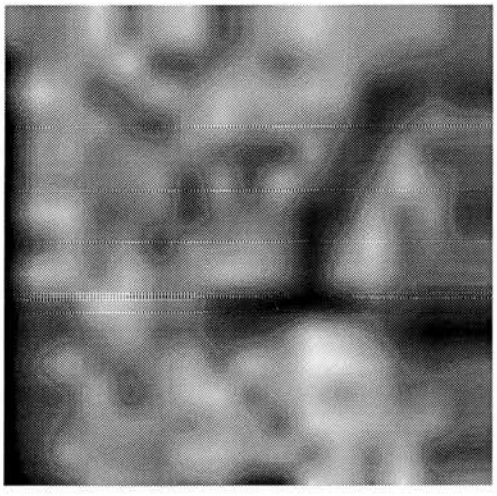
(b)

(c)

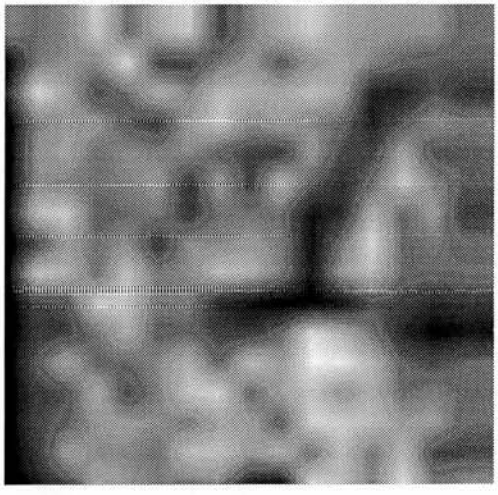


(d)

Figure 5.16 : The C-scan image of a tissue mimicking material with a bifurcation shape inside. (a) - Simulated fast C-scan image (3 cm x 3 cm) acquired with every fifth line. (b) - After interpolation. (c) - After median filter with 3x3 kernel. (d) - After median filter with 5x5 kernel.



(e)



(f)

Figure 5.16 (contd.) : (e) - After wiener filter with 3x3 kernel. (f) - After wiener filter of 5x5 kernel.

CHAPTER 6

OVERALL DESIGN CONSIDERATIONS OF THREE DIMENSIONAL ULTRASOUND IMAGING

CHAPTER 6

OVERALL DESIGN CONSIDERATIONS OF THREE DIMENSIONAL ULTRASOUND IMAGING

6.1. Introduction

In the last decade, the use of digital imaging modalities for clinical diagnosis and therapy planning has dramatically increased and also has witnessed expansion of new modalities in medical imaging (Margulis & Shea, 1986). Modalities such as magnetic resonance imaging (MRI), computerised tomography (CT), single photon emission computed tomography (SPECT) and positron emission tomography (PET) have proven extremely valuable in medicine because of their capabilities of producing spatially accurate images of internal anatomy. Although 3D techniques were established in CT initially to display body structures, such as the skull and spine in three-dimensions, the technology nowadays has advanced significantly to provide three-dimensional displays of soft-tissue organs.

New developments in 3D medical imaging such as spiral CT scanning, faster and better MRI and improved functional imaging using other modalities such as PET and SPECT encourage the specialists in medical imaging to use three-dimensional visualisation and this should open up new areas of clinical application.

Unlike other tomographic imaging techniques, ultrasound has the advantage of the low cost and real time nature as discussed in chapter 1. In this chapter, three-dimensional ultrasound and its advantages, reconstruction techniques and the basic results of reconstructed 3D images obtained using our C-scan system are presented. Then a review of three dimensional ultrasound systems with clinical applications is covered. Finally the chapter discusses 3D ultrasound system design and specification, with particular attention being paid to real-time 3D imaging.

6.2. Three-Dimensional Ultrasound and its Value

Three dimensional ultrasound imaging is a recent development. Unlike CT and MRI, ultrasound offers interactive visualisation of underlying anatomy and views images from different angles if it can be achieved in real-time.

Three dimensional ultrasonic visualisation represents a challenging task from a technological point of view. First, transducer registration; where the transducer has no frame of reference. Second, real time video digitisation. Third, image segmentation to identify the desired section and the surface of organs. Fourth, image signal-to-noise characteristics limit the ability to define organ interfaces and vascular anatomy with sufficient accuracy. In addition, display of a two-dimensional image can be presented in a display screen while a 3D image is a whole volume which is cannot be displayed on a single flat display. Therefore, 3D images are displayed using 2D screen displays, using specific techniques such as multiplaner slice projections, surface fitting, and volume rendering. Then using geometric operations to rotate the image gives a better appreciation of the three dimensional geometrical relationship.

Three dimensional ultrasound features advantages compared to 2D ultrasound. These advantages are as follows:

- Gathers all the information needed to provide different views of the same extended solid body region.

- Allows the physician to evaluate arbitrary planes not available with two-dimensional volumes and geometry.

- Allows the physician to obtain different anatomic and blood flow information.

- Reduces scanning times and hence offers more cost-effective use of equipment and sonographer's time.

- Allows complex anatomical structure to be more easily understood.

If 3D ultrasonic imaging can be made to operate in real-time, it will be more compatible with clinical use. The change from static B-scanning to real-time B-scanning not only allowed moving structures to be observed but permitted rapid searching through tissue structures and hence a greatly increased patient throughput. It is highly desirable that 3D imaging should also be in real-time.

6.3. Review of Three Dimensional Ultrasound

6.3.1. Introduction

Ultrasound imaging is used to form pictures of internal anatomy. It is a unique modality in that it is both non-destructive and can yield real-time display. Ultrasound scanners produce 2D cross-sectional images in real-time. From this 2D images collection it can be difficult to achieve comprehension of the 3D structure and relationships of objects or organs (Greenleaf,1982). To effectively visualise anatomical characteristics, it is required to have a 3D acquisition and display system of ultrasound images which features real-time and interactive process (Mills,1990). Baum and Greenwood in 1961 are the first people who described the concept of 3D ultrasound imaging (Baum and Greenwood, 1961). They created a 3D display by stacking sequential photographic plates of parallel ultrasound images of the human orbit. Since that initial work a number of researchers worked in this area to increase the feasibility of clinical 3D ultrasound images (Robinson, 1982; Greenleaf, 1982; Feigenbaun, 1982; Joynt and Popp, 1982). After these efforts advances in image acquisition, processing and display technologies have increased the clinical applications of 3D ultrasound imaging.

6.3.2. 3D Ultrasound Image Acquisition

The first studies of the 3D ultrasound imaging were to reconstruct 3D data from 2D images. Halliwell and co-workers in 1989 have used the idea of forming 3D data set from 2D images by digitising the 2D image from video tape recording and stored them in a digital memory to form 3D data set later using a commercially image processing unit (Halliwell et al., 1989). To reconstruct a 3D image from images of a 2D data, the location and orientation of the transducer must be known. This is done by many ways. Some investigators (Brinkley et al., 1978; King et al., 1990 and 1991; Levine e al., 1989; Mortiz et al.,1976, 1977 and 1983) have coupled hand-held ultrasonic

scanheads with an acoustic co-ordinate measuring system (i.e. .e the co-ordinate values are tracked acoustically). For example, in 1978 Brinkley and co-workers had studied non-invasive estimation of the volume of the heart chamber. King and co-workers in 1990 designed a system that tracks the location and orientation of 2D image slices with 6 degree of freedom. Mortiz and co-workers in 1976 developed the acoustic spark-gap system which has two spark gaps (sound sources) mounted on the scanhead and arranged microphones tracked the location and orientation of the scanhead. The co-ordinate values can be also tracked mechanically; Dekker and colleagues who are considered among the first investigators to obtain 3D ultrasound scans of the heart, used a mechanical arm to track the spatial location of an ultrasound transducer as it was moved across the chest (Dekker et al.,1974). Other researchers have used the mechanical location systems for 3D ultrasound imaging (Geiser et al.,1982a, 1982b; Greenleaf and Bahn, 1981; Hottier, 1990; Martin et al., 1990; McCann et al.,1988; Nerstrom et al., 1991; Ohbuchi and Fuchs,1990; Raichelen et al., 1986; Schmitt, 1984; Sohn et al.,1988; Steinke and Hennerici ,1989; Strickles ,1984). For example Greenleaf and Bahn in 1981 designed a 3D specialised application in breast transmission tomography. The same application was done by Schmitt and his colleagues in 1984. Nerstrom and co-workers designed a 3D mechanical system for transrectal urethra imaging application in 1991. A third way is achieved optically (Mills and Fuchs, 1990). In this method a 3D ultrasound volume is captured from a series of 2D cross-sectional images obtained with a conventional B-mode scanner. The digitised sections within the volume are located by optical tracking of the transducer using two conventional video cameras and steriopsis. The fourth method to track the location of the transducer is by using a video position and orientation measurement (POM) system (Stone ,1990). Watkin and Rubin used this method in pseudo-3D reconstruction of ultrasonic images of the tongue (Watkin and Rubin, 1989). A magnetometer-based POM device is another way to acquire 3D ultrasound images (Gardener , 1992; Detemer et al., 1994; Nelson and Pretorious, 1992). Gardener used such a position sensor (the electromagnetic device) from (Polhemus Inc.) which has a source generator and sensor receiver both with three orthogonal coils and provides position on x, y and z axes to within 0.25 mm and angular orientation accurate to about 0.2 degree (Gardener ,1992). The only limiting factor is the presence of ferrous objects distorting the field (Lees ,1992). Apart of the above systems , there are a human or a machine makes scans at specific locations and orientations. A manually guided scanner with mechanical tracking was developed at Philips' Paris Research Laboratory (Hottier and Collet Billon , 1990). In this system an annular

array transducer is rotated by a stepper motor to get a third dimension. They acquired 50 to 100 slices in a period of 3 to 5 seconds.

A few attempts have been made to produce real-time 3D ultrasound images. von Ramm's group developed a similar 3D scanner (Smith et al., 1991 ; von Ramm et al., 1991) but using a phased array instead of using an annular array. In this system a parallel processing technique (Shattuck et al. , 1984) to achieve acquisition of 3D data in real-time was used. 2D arrays also were used in conjunction with receive mode parallel processing for real-time volumetric imaging (Davidsen et al. , 1994). Real-time quasi-3D display was designed by widening the ultrasound beam of a standard transducer with use of a convex lens(Enterekin et al. , 1992; Kossoff et al. , 1994). This method does not provide true 3D imaging in that it is not volumetric, and unique tomographic planes can not be generated.

6.3.3. 3D Ultrasound Image Storage

After 3D acquisition, the 3D data set may be stored on analogue videotape. Because of the analogue videotape limitations which includes signal degradation and noise, digital storage media such as optical disk is used. In this digital storage the images are stored in a suitable format for later reconstruction. The continuous development of high speed optical disk technology reduces the limitation of the current methods. Some ultrasound scanners have digital storage of the images such as the system developed by Weyman (Wayman and Levine , 1991).

6.3.4. 3D Ultrasound Image Processing

3D ultrasound imaging represents a challenging technological task (Kossoff , 1995). One of the difficulties is to track the spatial location and orientation of the hand-held transducer and this was reviewed fully in the 3D ultrasound image acquisition (section 6.3.2). Second difficulty is the low signal-to-noise characteristics which limit the ability to define organ interfaces and to identify automatically the surfaces of organs.

There is increased availability of advanced computers and the continuous development of algorithms for image processing (Toriwaki and Yokoi , 1986). Many researchers have done a lot of research to solve the second problem which is the automatic identification of the object of

interest from the background image or what is called 'segmentation' (Sakas et al., 1995). Sakas and co-workers used off-line filtering to reduce noise and speckle and developed fast semiautomatic techniques to remove parts covering objects of interest (Sakas et al. , 1995). Also a rich literature about image segmentation has been produced by the researchers of image processing and computer vision (Niemann , 1981). Methods of image processing such as enhancing image contrast, removing noise and scaling the images have also been developed (Skorton et al., 1981 ; Song and Delp , 1990 ; Niblack , 1986 ; Gonzalez and Wintz, 1991 ; Hummel , 1977 ; Sherrier and Johnson , 1987).

6.3.5. 3D Ultrasound Image Display

The last step is to display the 3D images or what is called volume visualisation (Elvins , 1992) , which projects 3D data set onto a 2D image plane. Volume visualisation techniques must offer understandable data representation, quick data manipulation , and fast rendering which enable the physicians to change parameters and see the resultant image in real-time (Nelson and Elvins , 1993).

An early example of the display of 3D images was the work of Baum and Greenwood's stack of photographic plates (Baum and Greenwood , 1961). Dekker and co-workers collected 4D echocardiographic image sets in their work in 1974 (Dekker et al. , 1974). Optical holograms were first used to display 3D ultrasound images of the uterus in 1969 (Redman et al. , 1969). The 3D imaging using a fibre optic ultrasonic scanner has been done by McDicken and his co-workers (McDicken et al. , 1972). They have shown that it is possible with a simple technique to create 3D ultrasonic images by stereoscopic means. In 1956 , Howry and associates described a 3D and stereoscopic observation of body structure by ultrasound (Howry et al. , 1956). In 1977, Matsumoto and associates described a stereoscopic display of a wire frame model of the heart from echocardiographic images (Matsumoto et al. , 1977). Nakatani and colleagues (Nakatani et al. , 1979) extended the stereoscopic method with their use of polarised shutters. Fuchs and co-workers have used 3D vibrating mirrors (Fuchs et al. , 1982). Two different display techniques using a conventional 2D CRT screen which displays volumetric ultrasound images have been developed for use in real-time (von Ramm and Smith , 1990). McCann et al. (McCann et al., 1988) reconstructed an image volume and displayed it using a rendering technique in tools from the ANALYZE software (UK, 1990). Lalouche et al. reconstructed a scanned volume by cubic-spline

interpolation and then volume rendering was implemented (Lalouche et al., 1989). Hottier and Collet Billon, in 1990 use two visualisation techniques: re-slicing by an arbitary plane and volume rendering (Hottier and Collet Billon, 1990). In all the cases described above the standard volume rendering was used (Levoy, 1988; Sabella, 1988; Upson et al.,1988). Ohbuchi and Fuchs used an incremental, interactive, 3D ultrasound visualisation technique which visualises a 3D volume as it is incrementally updated by a sequence of registered 2D ultrasound images (Ohbuchi and Fuchs, 1990; 1991). In their paper they described a new fast ray-clipping scheme called a D-buffer algorithm that is based on the Z-buffer algorithm and another speedup scheme called hierarchical ray caching. Bajura et al., in 1992 , used a system to visualise live ultrasound echography data within a pregnant human subject using a small video camera mounted in front of a conventional head-mounted display worn by the observer. As the observer walks around the subject, the ultrasound images appear stationary in 3-space within the subject (Bajura et al., 1992). Volume rendering of 3D medical ultrasound data has also been studied using direct feature mapping which is a new algorithm to achieve volume rendering (Steen and Olstad, 1994). Some other papers for speeding up the rendering were published such as the high speed integrated rendering method described by Miyazawa (1991).

6.3.6. Clinical Evaluation of 3D Ultrasound Imaging

3D ultrasound imaging is in its earliest phase of development. Nowadays there are several 3D ultrasound imaging systems, either commercial or prototype systems as described above. Many papers are being presented describing the clinical trials of these 3D ultrasound imaging systems and also determination of optimal methods of acquisition and interactive display. Some of the clinical trials can be found in the following papers (Coy et al., 1992; Belohlavek et al., 1993; Fine et al., 1991; Nelson and Pretorius, 1992; Balen et al., 1993; Picot et al., 1993, Pretorius et al., 1992; Delcker and Diener, 1994; Gilja et al., 1994; Lee et al., 1995; Blaas et al., 1995; Merz et al., 1995; Jurkovic et al., 1995). By way of example the last three applications will be briefly described. Merz and colleagues report on their experience scanning patients with congenital anomalies using both 2D and 3D ultrasound. Blaas and colleagues examined the application of a 3D ultrasound in imaging of the brain cavities in human embryos. Jurkovic and colleagues demonstrate 3D ultrasound application for assessment of uterine anatomy.

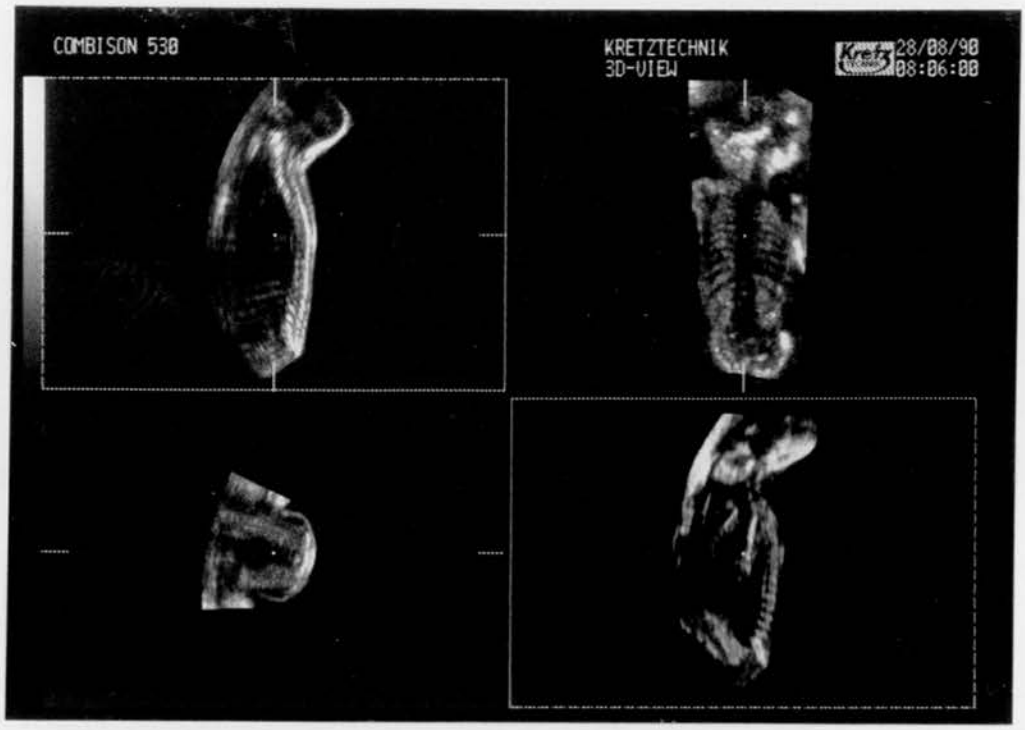
6.3.7. 3D Commercial Ultrasound Imaging System

In the 1970's Sonicaid Ltd manufactured a 3D scanner in which a single crystal transducer supported on a gantry was swept over the skin surface. Stereoscopic imaging was used. Like most early 3D scanners the lack of digital technology rendered it impractical.

Several manufacturers e.g. (Kretz and Dornier) have opted for the mechanically-driven 3D scanning devices approach in designing arrays which are swept in the third direction like sector scanner. The disadvantages are: the scanning head is bulky, the view direction is limited and there is no possibility of compounding. On the positive side these devices can provide a good controlled sampling over a volume of interest. 3D image acquired using the Kretz system is shown in Figure 6.1(a).

Another commercial product, the Echo-CT system by Tomographic Technologies, GMBH, uses the linear translation of a transducer inside a tube inserted into the oesophagus to acquire parallel slices of the heart and externally. Image acquisition is gated by respiration and an ECG to reduce registration problems (Tomographic Technologies, 1991). Some of the acquired 3D images using the rotating probe externally of the Echo-CT system in the Department of Medical Physics, Western General Hospital are shown in Figure 6.2.

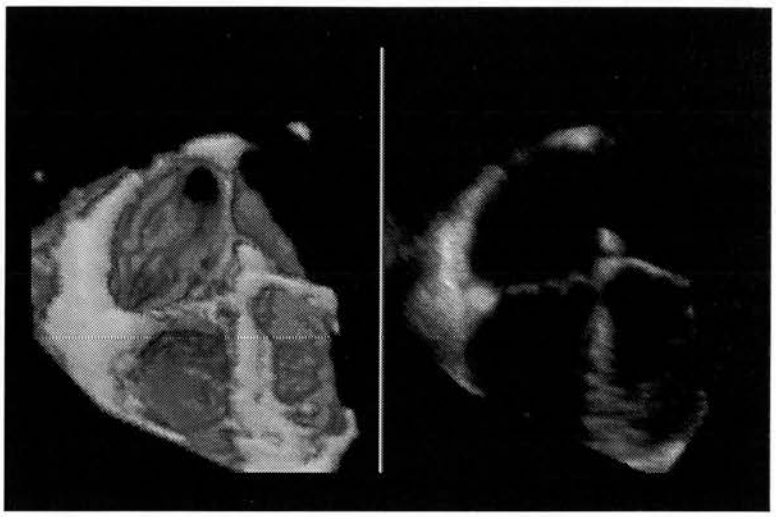
Another system from Toshiba was presented in the last conference of British Medical Ultrasound Society (BMUS, 1995). In the same conference Allott and co-workers described this technique (Allott et al.,1995). This system is based on the concept of using a thick slice beam for imaging the surfaces of structures using a fan-beam for volume imaging. One 3D image acquired using this system is shown in Figure 6.1(b). The disadvantage of this system is that the method used does not provide true 3D imaging but rather depicts the surface of a structure which lies in the scan plane.



a

b

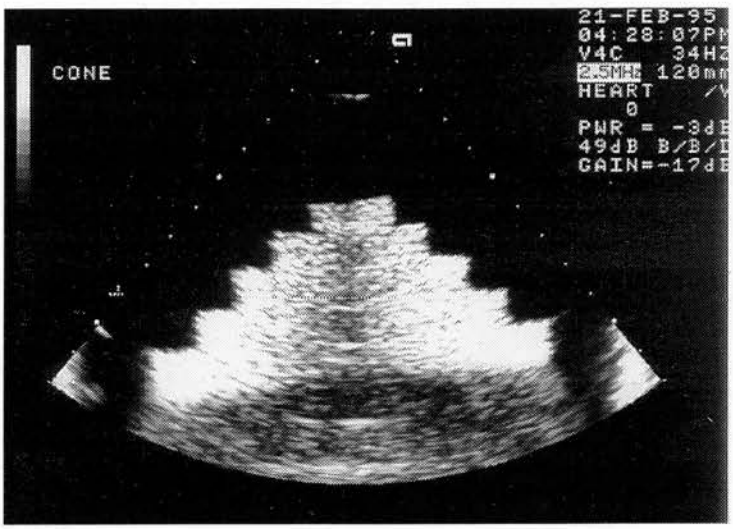
Figure 6.1 : Commercial 3D scan images. (a) - Three orthogonal scans of fetus (right top ,right top ,left bottom) and 3D reconstructed image of fetus (Courtesy of Kretz). (b) - Real-time pseudo-3D display of human hand imaged in a water tank (Courtesy of Toshiba).



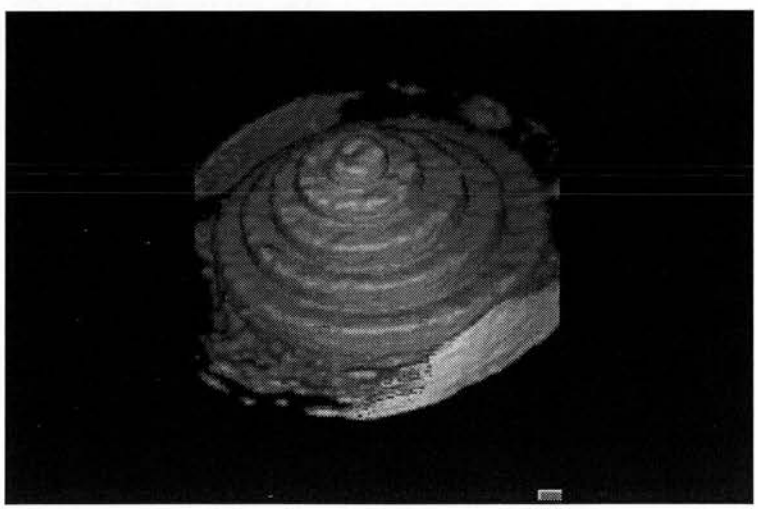
(a)

(b)

Fgure 6.2 : Commercial scanners. (a) - 3D display of the heart (left) and the B-scan of the heart (right). (b) - 3D display of the heart, left ventricle.



(c)



(d)

Figure 6.2 (contd.) : (c) - B-scan image of the tissue mimicking phantom. (d) - 3D display of the tissue mimicking phantom.

6.4. Three-Dimensional Reconstruction

With a view to designing a real-time 3D scanner, it is worth considering the methods which may be employed to acquire data and construct an image.

Three dimensional reconstruction is the step after acquiring the 3D data. There are different 3D reconstruction methods depending on the data acquisition method. Each technique has advantages, limitations and specific applications.

6.4.1 Reconstruction Techniques

- **Parallel Scan** : - Simple parallel scanning is accomplished by moving the transducer in discrete steps to cover the imaging volume (Figure 6.3). Parallel 2D images are stored as 3D images. Reconstruction consists simply of aligning the parallel slices. This method is simple but is not convenient for use with hand-held transducers since the device is normally too large. One of the applications which uses this method is intravascular ultrasound (Chandrasekaran et al, 1991; Evans et al, 1991; Mintz et al, 1991).

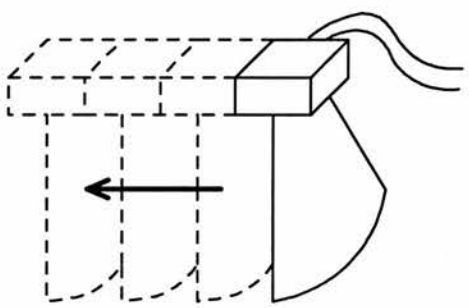


Figure 6.3 : A sketch diagram of the parallel scan reconstruction technique.

Compound Scan : - In this method a compound B-scan is obtained by sweeping the transducer across the object and then rotating the transducer a specific distance across the object to obtain another image (Figure 6.4). This operation is repeated in different positions to obtain different images of the object. Then the resulting 2D images are averaged to create a compound 2D

image. The above procedures are repeated at different levels. 3D reconstruction is done by stacking the averaged 2D images. One of the advantages is the noise reduction due to use of averaging. This method requires accurate ways to record the position of the object. One of the applications which uses this method is to generate high resolution images of the heart (McCann et al, 1988). However the method is slow and access to the heart is limited by bone and gas.

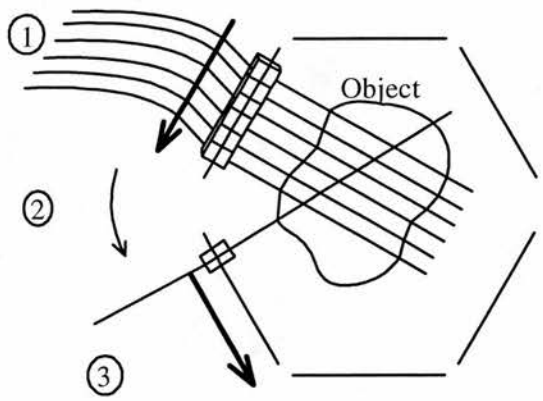


Figure 6.4 : A sketch diagram of the compound scan reconstruction technique.

- **Angular Scan** : - This is accomplished by rotating the transducer (either sector or array) in incremental steps to create a conical volume (Figure 6.5) or sweeping of the transducer through an angle to create a pyramidal volume (Figure 6.6) (Ghosh et al, 1982; McCann et al, 1987; Martin et al, 1990). This method is good for hand-held scanning, but there may well be unsampled gaps in the 3D image and this requires a way to fill the gaps when reconstructing.

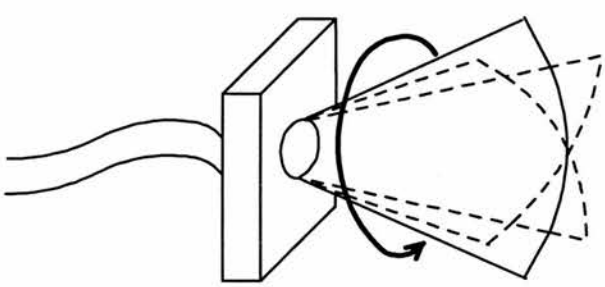


Figure 6.5 : A sketch diagram of the angular scan technique (rotating method).

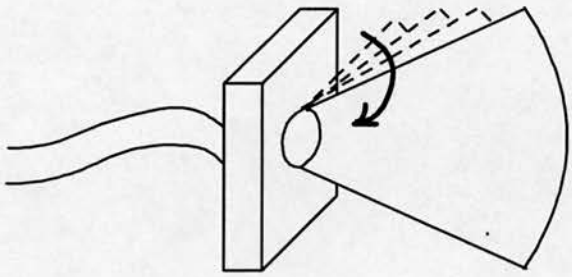
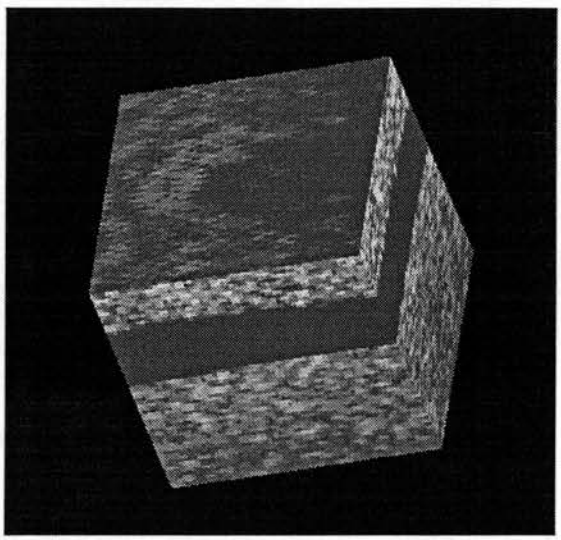


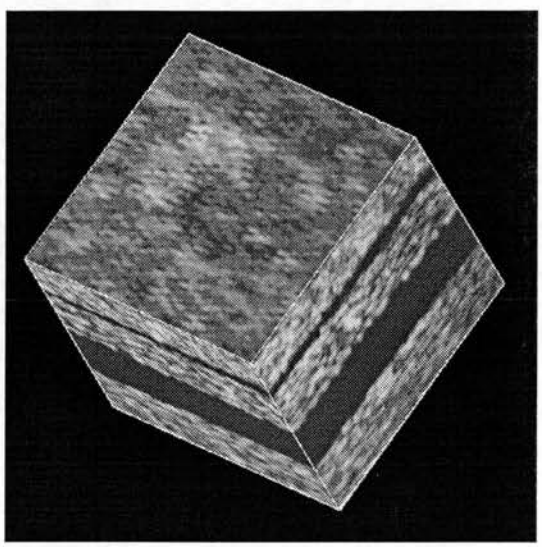
Figure 6.6 : A sketch diagram of the angular scan technique (sweeping method).

6.4.2. Reconstruction Results

Prior to investigating results from a typical hand-held 3D scanner it was decided to undertake a brief pilot study with the C-scan system. The first reconstructed 3D image using the basic C-scan system is obtained from 40 parallel slices of about 0.75 mm interval. Each 2D image acquired was of 80 x 80 pixels, which corresponds to 3 x 3 cm in real size. These data sets were acquired using our system described in chapters 3 and 4. The reconstruction was done off-line using a SUN workstation. Software was written to reformat the data and then 3D software was used to display all the information in a 3D image using volume rendering. Figure 6.7 shows the reconstructed 3D image of a resolution phantom which was presented in section 5.2.2 with different views. The 3D image of the bottom two layers of the resolution test phantom using another type of 3D display techniques called maximum value rendering is shown in Figure 6.8. Figure 6.9 shows the reconstructed 3D image with different cuts and this is useful in evaluating arbitrary planes not available with two-dimensional ultrasound. The investigation of the image processing on 2D slices is fully discussed in chapter 5, these techniques are extended and applied to 3D reconstructed images. The effect of applying 3D median filters is shown in Figure 6.10. This pilot study although not equivalent to real-time 3D imaging gave promising results so it was decided to pursue 3D imaging further.



(a)



(b)

Figure 6.7 : The 3D display of the reconstructed 3D image of the resolution phantom using texture mapped rendering . (a) - original 3D data set with the bottom two layers. (b) - Interpolated 3D data of the all three layers.

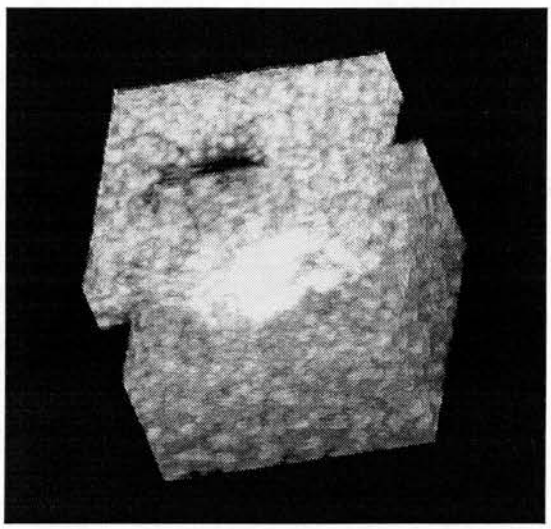
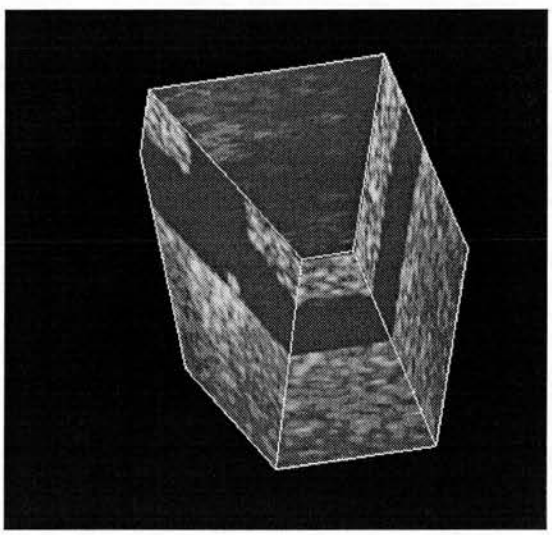
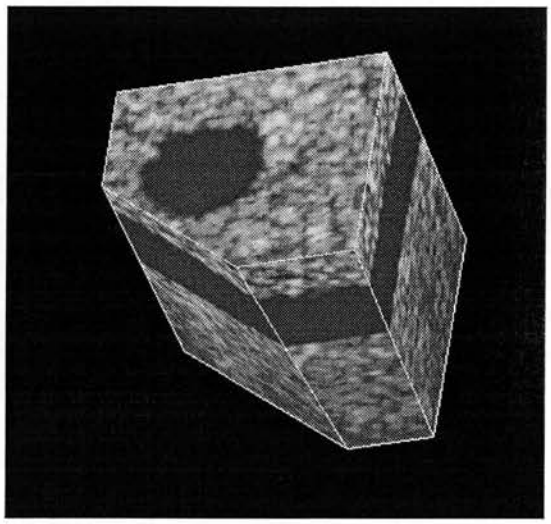


Figure 6.8 : The 3D display of the reconstructed 3D image of the resolution phantom using maximum value rendering using the original 3D data set .

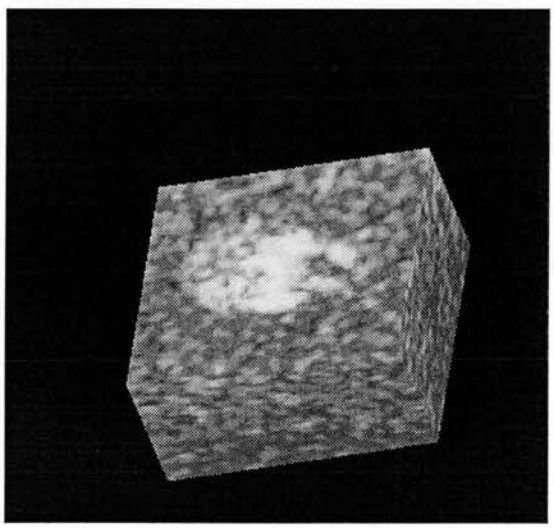


(a)

Figure 6.9 : The 3D display of the reconstructed 3D image of the resolution phantom using outlined texture mapped rendering with oblique clipping plane to show the interior of the 3D object. (a)- first view.

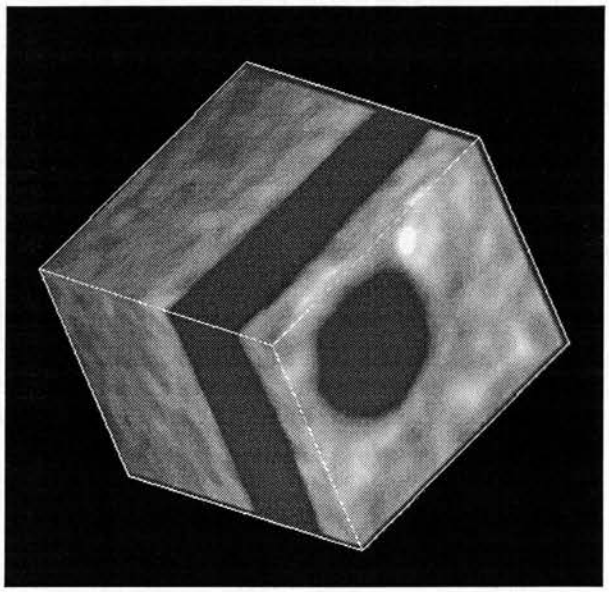


(b)

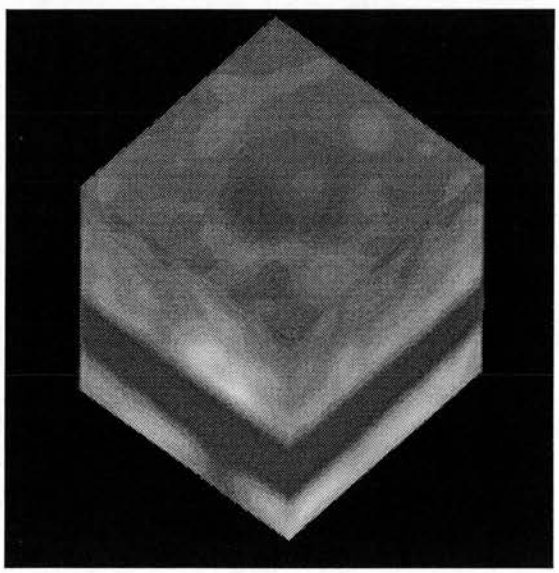


(c)

Figure 6.9 (contd.): (b) - Second view. (c) - Third view.



(a)



(b)

Figure 6.10 : The 3D display of the reconstructed 3D image of the resolution phantom using outlined texture mapped rendering after filtering . (a)- using 3x3x3 median filter. (b) - using 5x5x5 median filter.

6.5. Three Dimensional System Design

From the previous work described in chapter 3, 4 and finally in chapter 6, it is clear that the problems to implement real-time systems are the speed of collecting the data, processing the collected data and finally 3D image display.

6.5.1. Design Ideas

There are several ways in which real-time 3D imaging can be attempted. A hand-held sector scanner probe such as one with rotating transducers in it, or a wobbler could be drawn in a line perpendicular to the sector plane as was shown in Figure 6.3. One problem is that the edge of the sector describes an arc, so that if a constant time delay is used, then the surface imaged is curved. This delay can be varied to make the surface a flat plane. Alternatively, instead of drawing the transducer in a line, it could be wobbled as was shown in Figure 6.6. Again the time delay would have to be further adjusted to compensate for the curved surface, to obtain a flat plane. If a sector scanner has a frame rate of 15 f/s, and if the 3D image has say 105 2D sectors, then the volume scan rate would be 1 in 7 seconds, which is too slow to be considered real-time.

Removing mechanical movement in one dimension is possible by using an electronic scanning system such as a linear array . This may have a frame rate of 35 f/s in B-scanning, and one volume scan would take 3 seconds for a 3D image of 105 2D sectors. An array transducer has a major advantage in that it can vary its focus over an extended depth and this in turn gives a good resolution in the scanned volume. However this approach is also too slow for real-time imaging.

The process can be taken a stage further by using a two-dimensional phased array. This removes the necessity for mechanical movement, the processes of focusing and beam steering all being done electronically (refer back to von Ramm and Smith ,1983 in section 6.3). However, manufacturing such an array is both extremely difficult and expensive.

Yet another way to produce a 3D image would be to employ more than one transducer. The volume to be scanned could be split into two, each section being imaged by its own transducer. Using more than one transducer would increase the size of the system significantly, and it is always more convenient to be able to use a hand-held device without any need for an extra transmission medium such as a water bath.

Finally, it can be concluded that in a practical system the final choice needs to be a compromise between physical size, frame rate, the cost of the system, its flexibility and also the complexity in implementing it.

In the above discussion about how to scan 3D objects in real-time it has been assumed, as is normally done that fairly large volumes need to be scanned but this results in very low line density in the volume image. However scanning a limited 3D volume is often of interest from a clinical point of view and could help to overcome the low line density problem. This could be implemented firstly by making a B-scan to determine the small area of interest followed by scanning an associated small volume. This new technique could be called ' 2D/3D real-time duplex imaging'. Figure 6.11 shows a sketch which explains the concept of the 2D/3D real-time duplex imaging. Since a small volume is to be scanned it may be feasible to image it in real-time e.g. 10 volume scans per second.

If a scanner, such as the Toshiba Power Vision which has been recently marketed, is employed which emits several simultaneous ultrasound beams the size of volume scanned could be increased due to the greater rate of production of scan lines. The results of this thesis are valid for this situation.

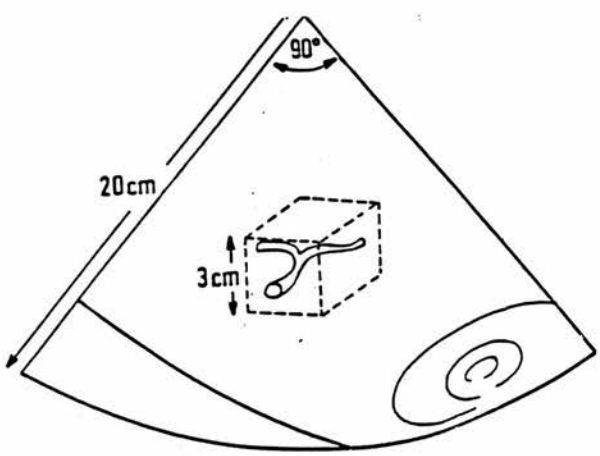


Figure 6.11 : The concept of the 2D/3D real-time duplex imaging.

This technique could be used in many applications. Some of these applications are:

1- Blood vessels
2- Fetal spine
3- Fetal kidneys
4- Ovaries
5- Gallstones
6- Heart valves
7- Head of pancreas
8- Breast tumour
9- Renal stones
10- Renal pelvis
11- Needle biopsy guidance

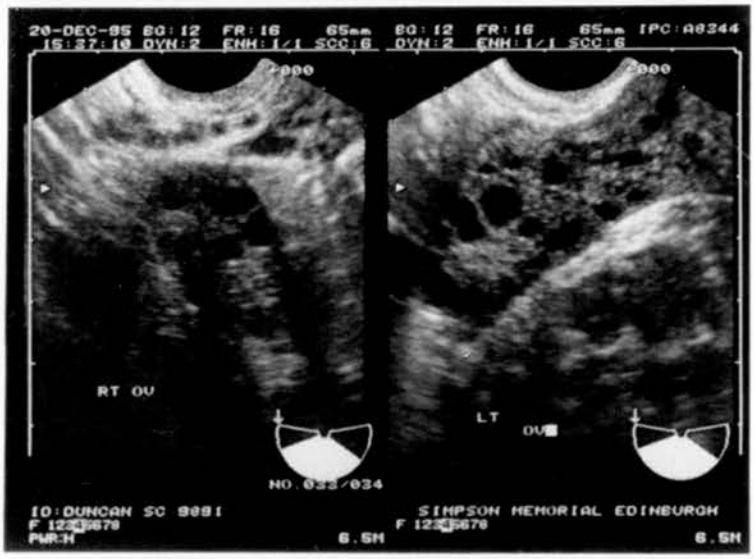
For example Figures 6.12 and 6.13 show some of the above applications in which the clinical result is obtained from a small tissue volume. In Figure 6.12 (a) fetal spine is shown and in Figure 6.12 (b) and (c) the right and left ovaries are depicted. In the last figure the small area of interest is defined with D1, D2 and D3 with dimensions of 27.3 mm, 25.1 mm and 27.2 mm respectively. Figure 6.13 shows four other applications : retina detachment, carotid plaque, renal biopsy and renal blood flow.

6.5.2. Design Specification

The main factor in design specification is discussed first. This is followed by discussing the essential stages in realisation of the 3D system . The topics of design and implementation from specification and analysing hardware and software requirements are discussed.

a

b



c

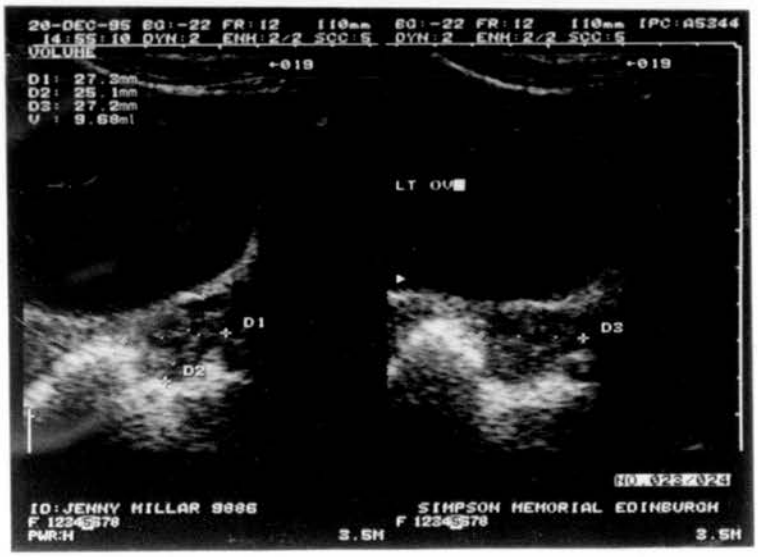
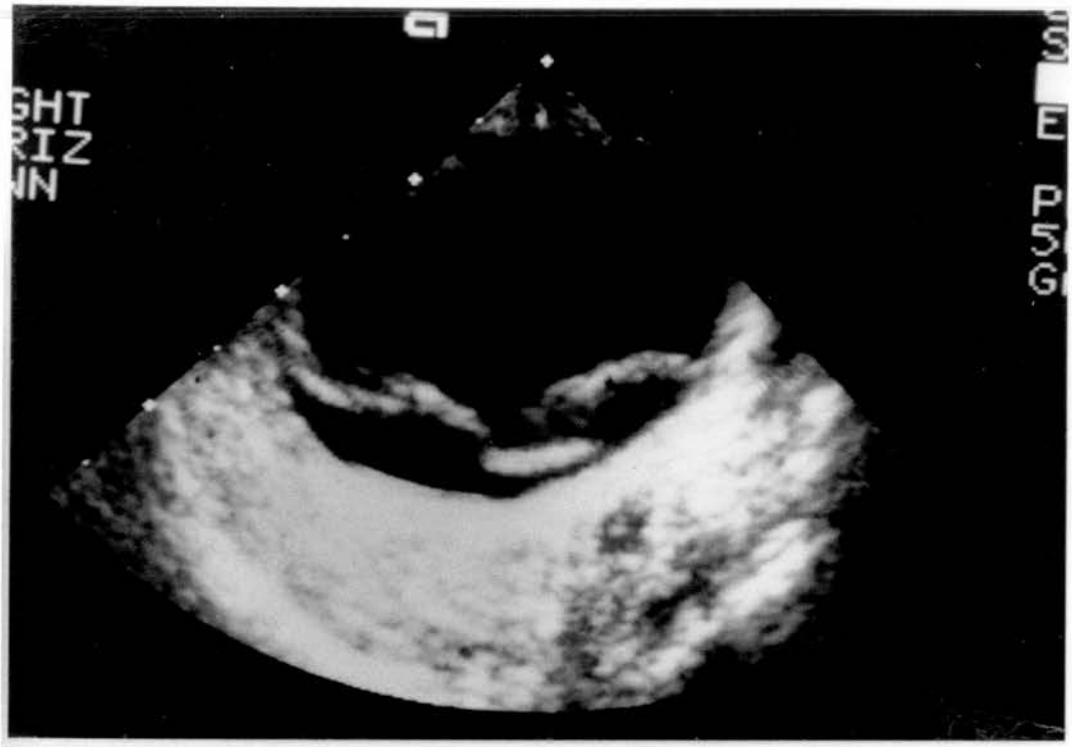


Figure 6.12 : Scans demonstrating regions of local interest. (a) - Fetal spine. (b) - Ovaries-transvaginal scan. (c) - Ovaries-transabdominal scan (Courtesy of S Chambers).

a



b

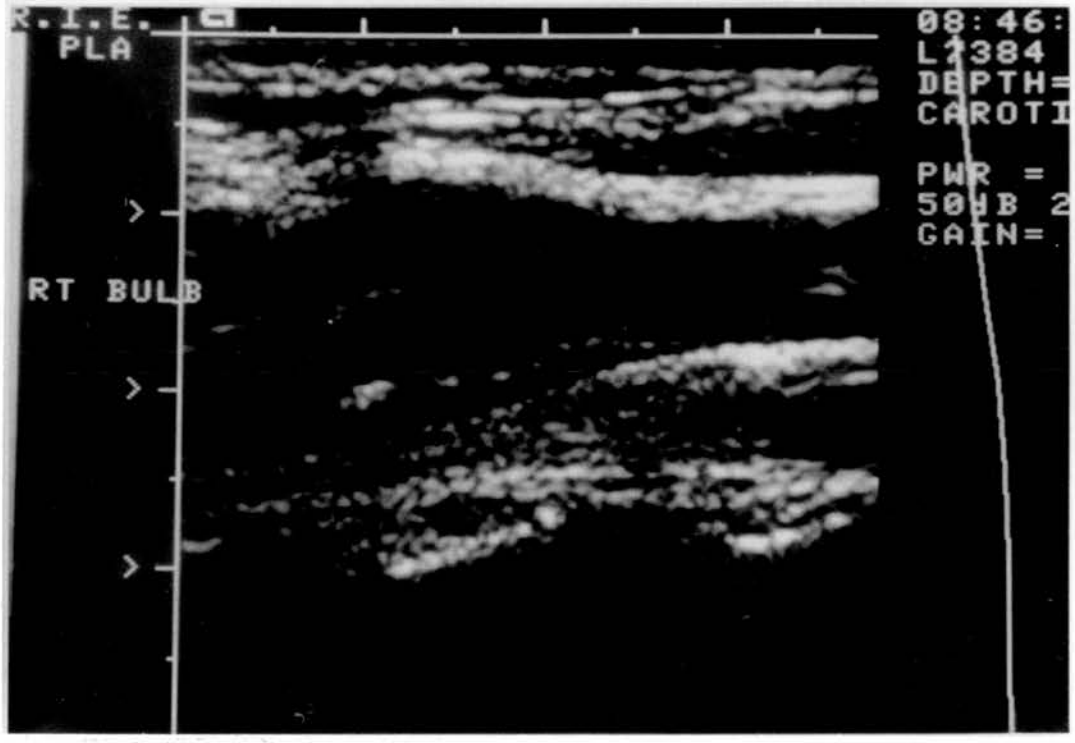
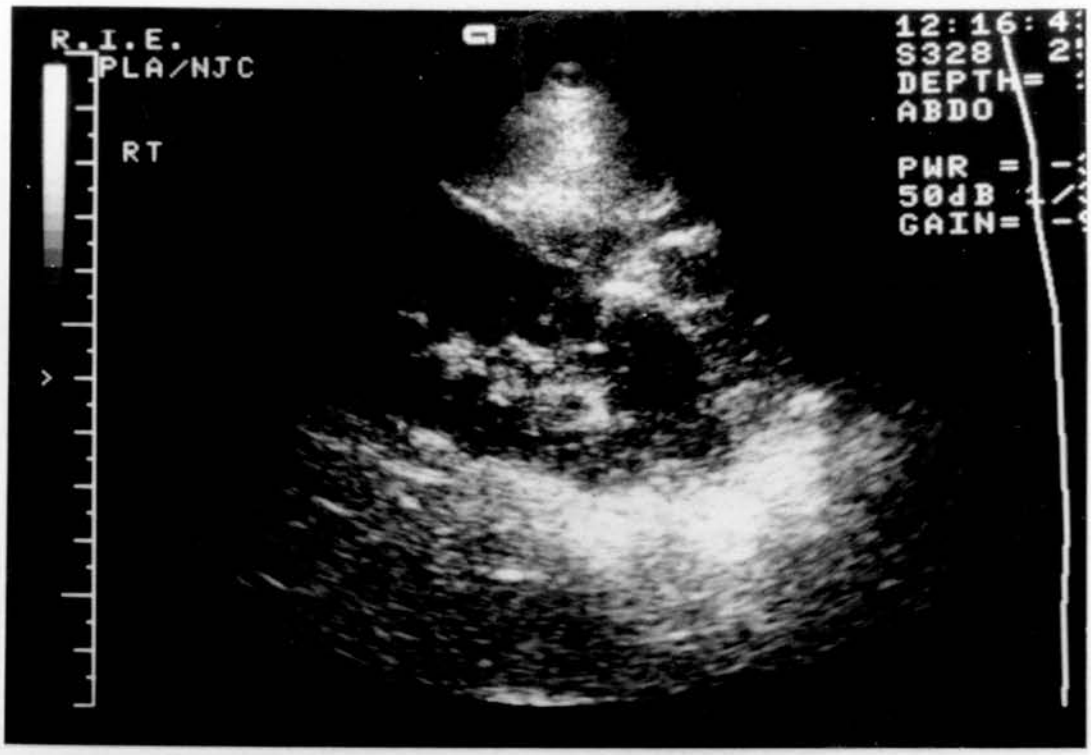


Figure 6.13 : Scans demonstrating regions of local interest. (a) - Retina detachment. (b) - Carotid plaque (Courtesy of P. Allan).



c

d

Figure 6.13 (contd.) : Scans demonstrating regions of local interest. (c) - Renal Biopsy. (d) - Renal blood flow (Courtesy of P. Allan).

Associated with real-time 3D ultrasound scanning, is the fundamental problem of acheiving an acceptable scanning rate. This arises because conventional ultrasonic scanners have an upper limit of rate of generation of scan lines of around 10 kHz , i.e. the maximum number of scan lines is 10000 per second. In real-time 2D scanning (e.g. C2000 from Dynamic Imaging) , the total number of lines in each B-scan frame is 160 lines. If the maximum PRF rate used is 10 kHz, the frame rate is 62 f/s. In 3D scanning the maximum PRF rate of 10 kHz would allow 5 vol/sec scanning with 2000 lines in each scanned volume. In other words about 44 x 44 lines through the volume which gives about 0.68 mm between lines if a small volume of 3 cm x 3 cm x 3 cm is scanned as explained in the previous section for the 2D/3D duplex technique. To increase the volume scan rate, the number of lines must be decreased which will reduce the line density, i.e. the data is sparse, hence the quality of 3D images will be affected and may not be acceptable from clinical point of view. However it may be possible to improve the quality of 3D images by fast image processing.

From this discussion it is planned to explore the specification of a scanner to be used for a 3 cm x 3 cm x 3 cm volume typically scanned at a rate of 10 vol/s. The hardware of 3D system and the required image processing techniques required to study the feasibility of the real-time 3D ultrasound imaging will be presented in the following chapters .

The 3D ultrasound test-rig developed in this project has the following components; ultrasound scanner from Dynamic Imaging Ltd in Livingston in Scotland which features portability and availability of a linear array transducer which is ideal for our system, motorised transducer, motor controller, fast data capture and control board, conversion board, fast processor card and power supply unit to provide the required power to all components. An AT type 486 Personal Computer (Dell PC486) controls the different parts in the system, to store and process the data and display the images.

CHAPTER 7

DESIGN AND CONSTRUCTION OF 3D ULTRASOUND SYSTEM: HARDWARE

CHAPTER 7

DESIGN AND CONSTRUCTION OF 3D ULTRASOUND SYSTEM: HARDWARE

7.1. Introduction

In the previous chapter, the 3D ultrasound imaging system's specification was presented. This chapter describes the design and construction of the 3D ultrasound imaging system (Figure 7.1) in detail.

The system is based on a microcomputer for the many reasons discussed in Chapter 3. It was one of the most powerful machines available at the time when the system was planned. The microcomputer of type (DELL PC486DX) has the following specification, 33 MHz speed, 8 M bytes RAM, 540 Mbytes hard disk, 3.5 inch and 5.25 inch floppy disk drive and 5 slots for the required cards.

This chapter describes the different components of the 3D ultrasound system. The format of this chapter is as follows: three-dimensional scanner design (section 7.2) which presents the two designs with their advantages and limitations, overview of the Dynamic Imaging scanner (section 7.3), image processor board (section 7.4), a programmable Input/Output card (section 7.5) and finally a detailed description of the fast data capture and control board (Section 7.6). Section 7.7 describes a hardware simulator of the ultrasound scanner.

It should be noted that the object of designing this 3D scanner was not to operate it continuously in real-time in practice but rather to enable ultrasonic echo data to be collected from a volume- field-view and with a low line density as would occur in real-time 3D imaging. It is then possible to apply real-time 3D image processing to see if images of acceptable quality can be obtained.

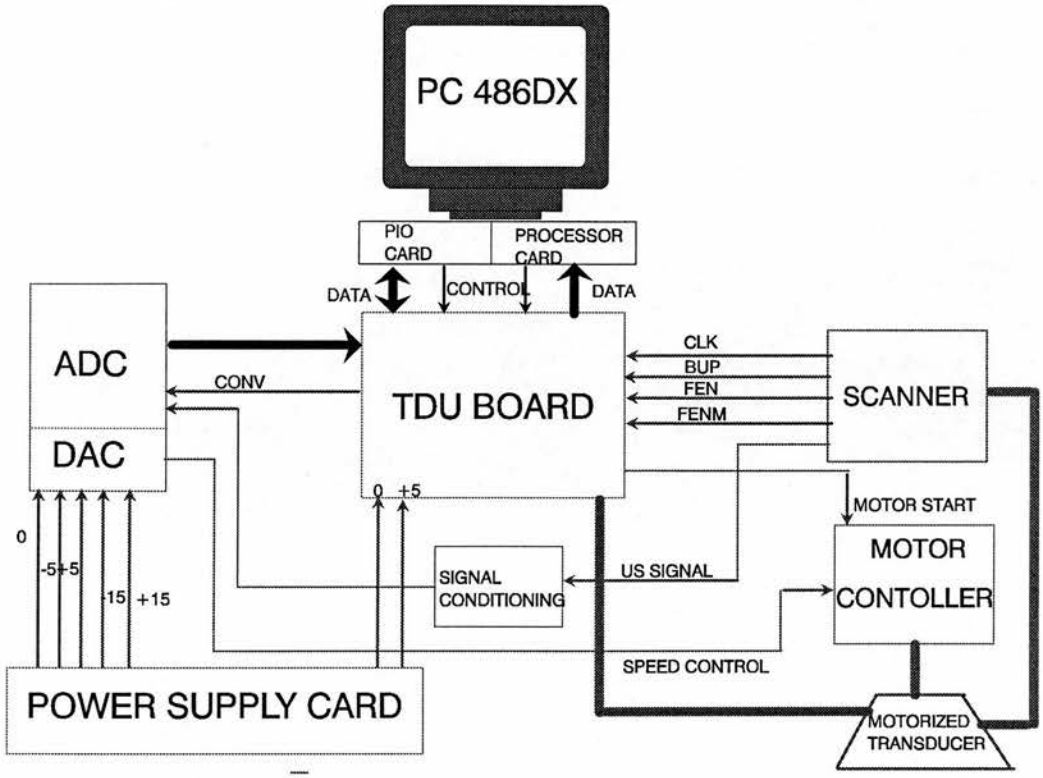


Figure 7.1 : 3D ultrasound imaging system.

7.2. Three-Dimensional Scanner Design

In this section the linear array transducer is described in general with its advantages. It is followed by two different mechanisms to achieve the main goal discussed in section 6.5. Mechanical considerations, electrical considerations and limitations and advantages of both designs are discussed.

7.2.1. Linear Array Element

The relatively simple 20-element linear array designed in 1972 provided the impetus for the development of real-time scanning (Bom et al., 1971). Each 6 mm diameter, 2 MHz transducer element was used in turn to produce one line of a 20 line B-scan. The number of scan lines could not be increased by simply using more transducers, each of smaller diameter, as the lateral resolution would be degraded due to the increased divergence in the far-field of each beam. Most

subsequent developments have been directed at improving lateral resolution and increasing the number of scan lines.

All modern designs use an array (Figure 7.2) of typically 100 rectangular elements (McDicken ,1991) , of which only a group of typically 32 adjacent elements are active at one time. This active group forms a composite transducer which produces a beamshape with properties determined largely by its overall dimensions but which can be advanced along the array by a fraction of its width (by dropping an element from one end and adding one to the other).

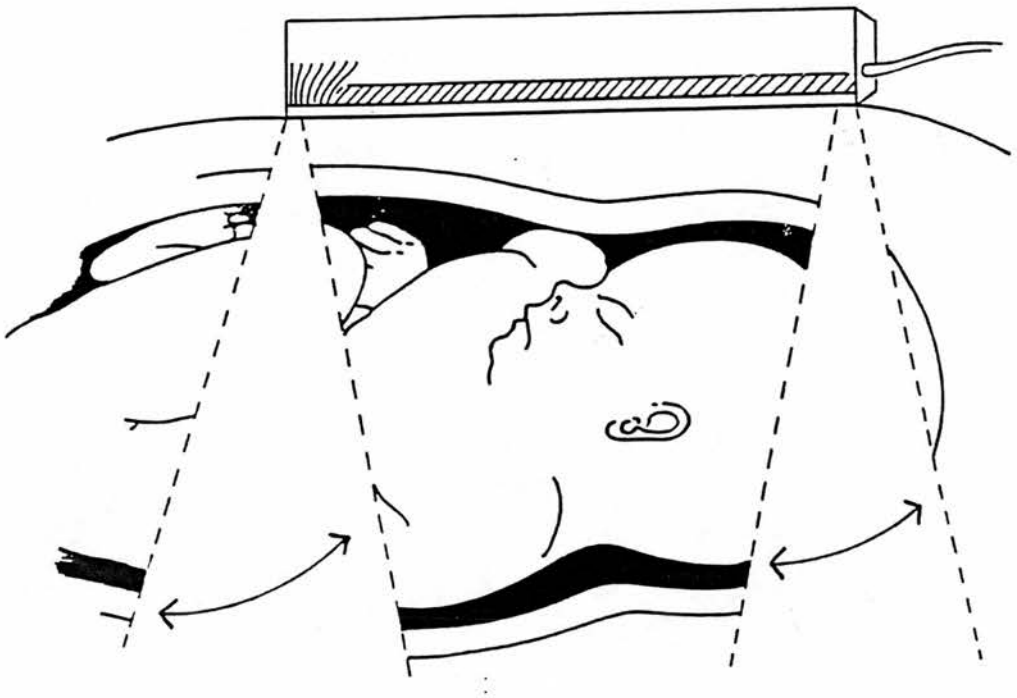


Figure 7.2 : Linear array element.

The switches that perform the element selection may be in either the probe itself (allowing a more flexible cable at the expense of probe weight and bulk), in the main console, or distributed between

probe and console. An array is constructed using independent strips of piezoelectric material. It is important that each element acts independently of its neighbours except when they are deliberately connected by electronic switches. Each of the elements has a separate electrical lead to one of its faces and another lead that connects all the elements' front faces together, usually the earth lead.

A linear-array scanner also utilises electronic focusing . The field format is rectangular. Using a linear-array, frame rates of up to 75 f/s and can be obtained. This high frame rate is one of its advantages. Linear array transducer dimensions have fall in the range of 5 to 20 cm and length is usually inversely proportional to the frequency. Figure 7.3 shows the mechanical drawing of a 5 MHz linear array transducer which was chosen to be used in the 3D ultrasound imaging system.

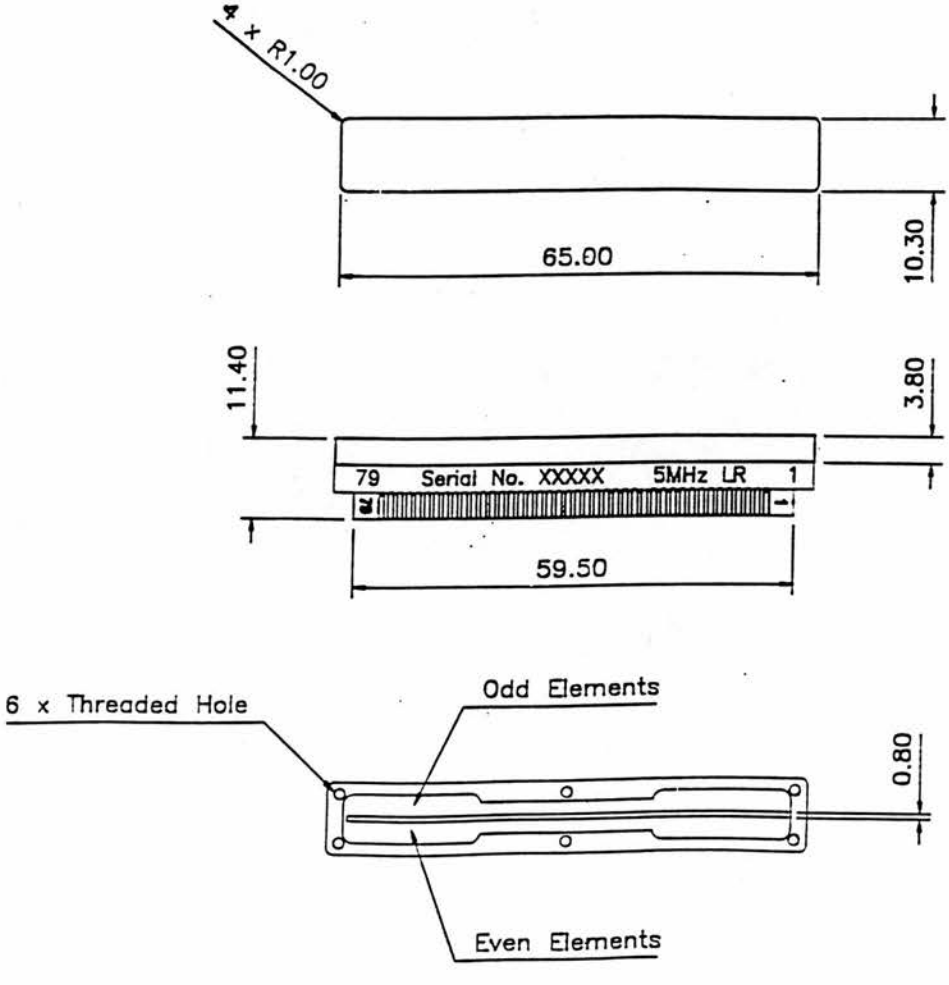


Figure 7.3 : The mechanical drawing of a 5 MHz linear array.

7.2.2. First Design

The first design mechanism is based on a dc motor to control the movement of the transducer.

7.2.2.1 Mechanical Considerations

The description of the motion of mechanical parts is useful in two important ways. First, in our system the desired motion is controlled by a motor and transmitted by some linkages. Here an analysis of the displacement, velocity, and acceleration of the motion is necessary to determine the design geometry of the mechanical parts. Furthermore, as a result of the motion generated, forces are frequently developed which must be accounted for in the design of the parts. Secondly, it is often necessary to determine the motion of mechanical parts resulting from the forces applied to them. In both types of problem the principles of kinematics have to be applied. In this section the kinematics of motion is covered and analysed as taking place in a single plane.

The mechanical system is designed to satisfy the tilting mechanism of the transducer or specifically the linear array element. This was done by attaching a motor to a circular disc via the shaft of the motor. The motion is transmitted to the desired element via a rotating slot on link BM which is attached to the transducer element at point M as shown in the simple diagram in Figure 7.4. For the purpose of system analysis, the system is drawn again as shown in Figure 7.5 (a). From the 2D graphical representation of relative motion of mechanical parts in Figure 7.5 (b) and (c) the motion equations can be derived.

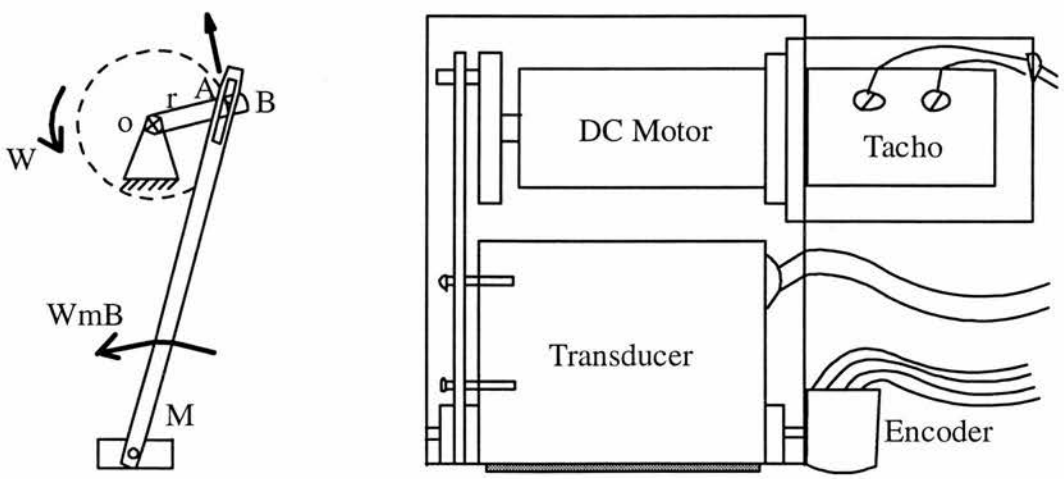


Figure 7.4 : Sketch of the mechanical arrangement of the 3D DC-based motorised transducer.

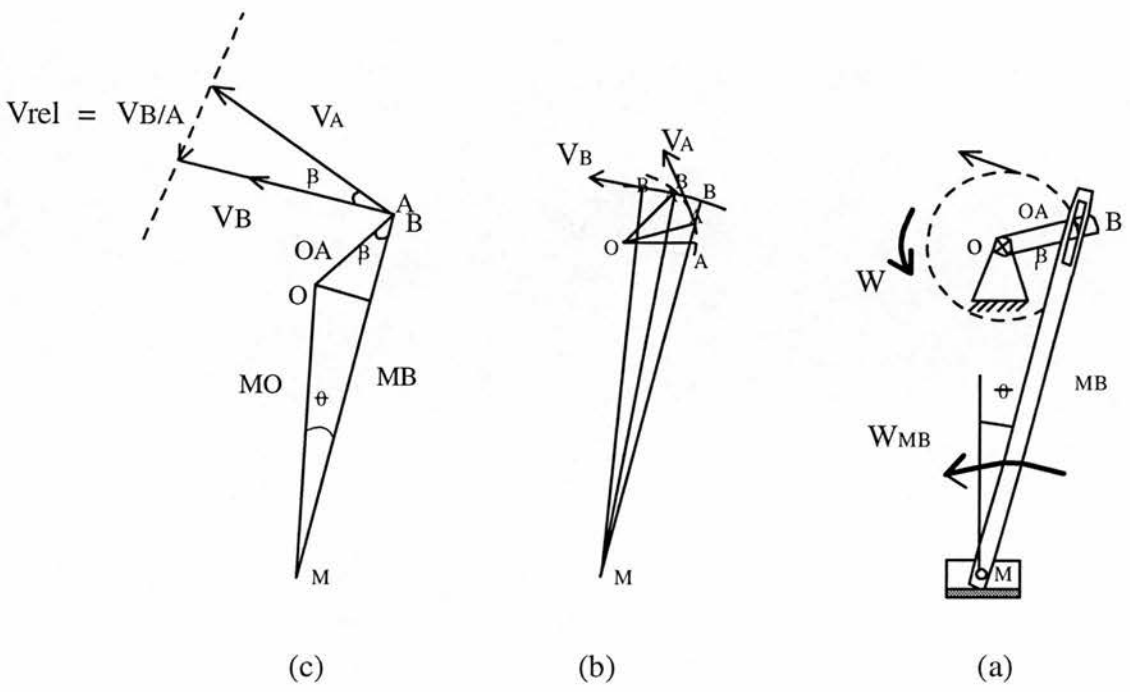


Figure 7.5 : Graphical representation of relative motion of the mechanical parts for the mechanical arrangement shown in the Figure 7.4.

Motion of a point (pin A) along a rotating path (the slot) suggests rotating co-ordinates attached to the arm MB. With the origin at M the term V_B is calculated using

$$V_B = w \times OA + V_{rel} \qquad (7.1)$$

where w is the angular-velocity vector of the circular disc (into the paper) and r_{oa} is the vector from O to point A on the arm coincident with B. The velocity of A is $V_A = w \times OA$, and the relative velocity is $V_{rel} = V_{B/A}$. Alternatively, then

$$V_B = V_A + V_{B/A} \qquad (7.2)$$

The velocity of the point A on member OA is

$$V_A = w \times OA \qquad (7.3)$$

The relative velocity $V_{B/A}$ is seen from Figure 7.5 to be along the slot toward M. This conclusion becomes clear when it is observed that A is approaching B along the slot from below before coincidence of A with B and is receding from B upward along the slot following coincidence of A with B. The vector equation may be solved since there are only two remaining scalar unknowns, the magnitude of $V_{B/A}$ and the magnitude of V_A. For the angle Ø position the figure requires

$$V_{B/A} = V_A \tan \beta = w\, OA \tan \beta \qquad (7.4)$$

$$V_B = V_A \cos \beta = w\, OA \cos \beta \qquad (7.5)$$

each in its direction shown. The angular velocity of BM is now determined

$$w_{BM} = \frac{V_B}{MB} = \frac{w\, OA \cos \beta}{MB} \qquad (7.6)$$

To find the relationship between w_{BM} and the desired angle Ø, a relationship between the angle β and the angle Ø has to be found

$$OA \sin \beta = MO \sin Ø \qquad (7.7)$$

or

$$\beta = \sin^{-1} \frac{MO}{OA} \sin Ø \qquad (7.8)$$

From equations (7.6) and (7.8), the desired speed of the motor is calculated based on the desired angle and the speed of volume scan. From the equations 7.6 and 7.8 it can be concluded that the relationship between the speed of the motor and the speed of volume scan at different angles is a sinosoidal relationship. Finally the 3D DC-based motorised transducer designed as shown in Figure 7.6.

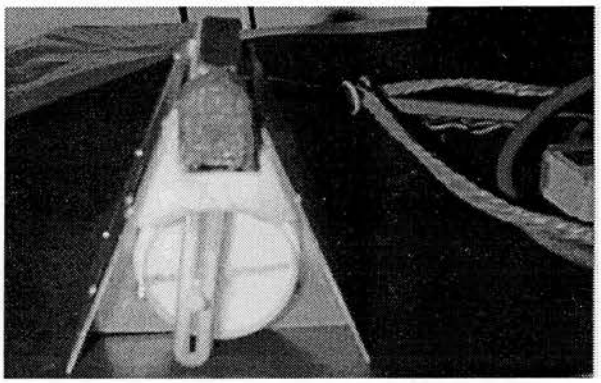


Figure 7.6 : The final design of the 3D DC-based motorised transducer.

7.2.2.2. DC Servo Motor and Servo System

The DC servo motor chosen was of type M586TE from McLennan Ltd (Surrey, England) which features an optimised torque/inertia ratio to achieve a particularly fast response coupled with a mechanical construction that ensures high levels of reliability. The specification of the DC servo motor is shown in Table 7.1.

Nominal voltage (VDC)	24 V
No load speed (rpm)	4100 rpm
Rated speed (rpm)	3250 rpm
Torque @ rated speed (Ncm)	19.5 Ncm
Starting torque (Ncm)	100 Ncm
Torque Constant (Ncm/A)	5.68 Ncm/A
Mechanical time constant (msec)	13.5 msec
Rotor inertia (gcm^2)	370 gcm^2

Table 7.1 : DC servo motor specifications.

The DC servo motor is provided with a DC tachogenerator for use in a velocity controlled DC servo system (Figure 7.7). A velocity controlled DC servo system consists of a DC servo amplifier and a DC servo motor with a DC tachometer.

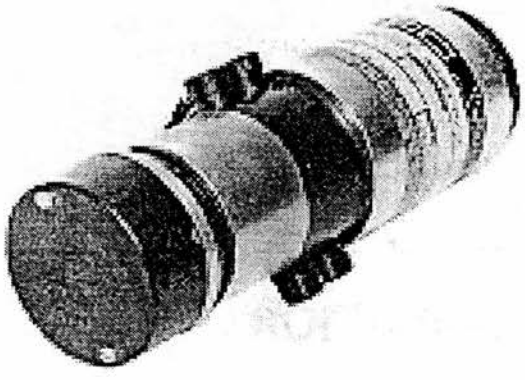


Figure 7.7 : The DC servo motor.

Figure 7.8 shows a system block diagram of the DC system which incorporates an operation amplifier (Type 741) in the input which is fed by a signal in the range (0v to 12V).

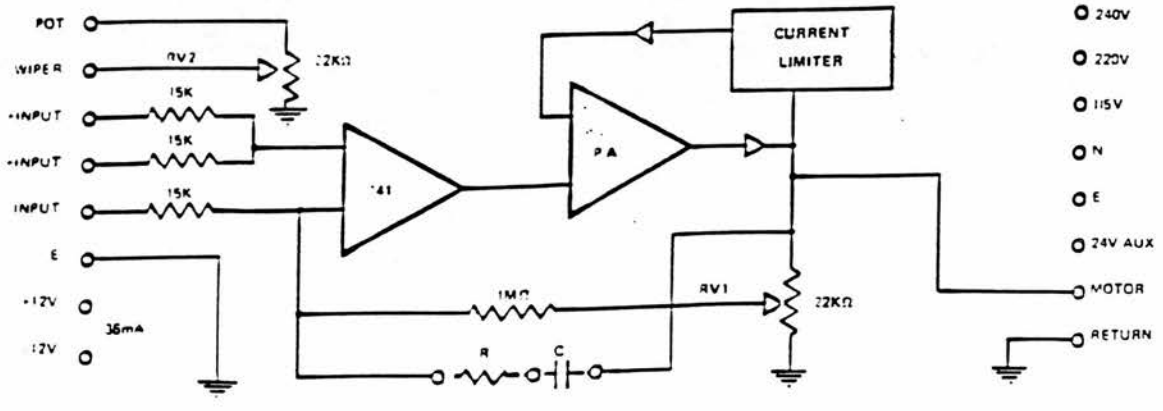


Figure 7.8 : The DC system block diagram.

A potentiometer RV2 is provided to step down the tacho output when required. The output of the operation amplifier is fed to the power amplifier which its output sensed by a current limiter to safeguard the motor and controller. To control velocity, firstly, the difference between motor speed and the input command voltage is detected, then power is supplied by the amplifier until this difference is zero. A continously variable output speed of from over 300 rpm to less than 0.2 rpm at maximum torque is available with a motor/tacho servo of this type. The system features full speed start or stop times of 30 msec, (and to provide high current for acceleration and deceleration). The DC servo system specification is shown in Table 7.2 :

Output voltage (vdc)	+/- 24 V
Continous current (A)	8 A
Peak current (A)	40 A
Control supplies rating (mA)	30 mA
Stabilisation	0.006 v/deg. C.
Aux 24 vdc rating (A)	1 A
Mains supply (vac)	240 v @ 50 Hz

Table 7.2 : DC servo system specifications.

7.2.2.3. Speed Control

The speed is controlled by an analog signal as explained in the previous section. The required speed is constant so a DC signal is required. To control speed easily, it is required to change the level of DC signal via a microcomputer to make the whole system computerised. To accomplish this, an interface between the microcomputer and the DC servo system is required. The interface consists of a digital-to-analog converter (TDC 1012) from TRW (USA). The TDC 1012 is a bipolar 12-bit monolithic digital-to-analog converter capable of converting digital data into analog current at data rates of 20 MHz. The data and control inputs are compatible with TTL.

The complementary 40 mA output current can easily produce a 1V full-scale voltage when directly driving a double-terminated 50 Ω load.

7.2.2.4. Limitations

The motorised transducer was manufactured in the Department workshop and fitted with an optical encoder which will be discussed later. An initial experiment was set up to test the accuracy of the data collection system. The data collection system was designed to read the angle position of every line of the 2D frame. The 3D data was collected and stored. The collected data was analysed and it was found that the angle changes in the range of 1-2 degrees between every 2D frame. This variable displacement between the 2D frames makes it difficult to reconstruct a 3D image. This is one of the limitations using this mechanism. It can be overcome by reducing the speed of the motor but unfortunately this can not be done because the motor requires a starting torque which can not be obtained using the slow speed. In conclusion the DC motorised transducer is good when the speed is required but it is not good when accuracy in the displacement is required. So, another mechanism required to be designed to satisfy the required specifications. This will be discussed in the following section.

7.2.3. Second Design

The limitations of the first design which is based on a DC servo motor as discussed above make it necessary to investigate another mechanism. It was decided to use a motorised transducer based on a stepper motor because the displacement can then be controlled accurately.

7.2.3.1. Mechanical Design Consideration

The first design consideration is to satisfy the tilting mechanism of the linear array element. The second design consideration is to satisfy the linear relationship between the motor angular displacement and transducer displacement to avoid the problem encountered in the first design which has a sinusoidal relationship as discussed before. This is required if a 1:1 relationship is to be obtained between the motor step and the angular displacement of the transducer. After taking into consideration all the above requirements in addition to the available components and the manufacturing of the motorised transducer, a number of transmission configurations were considered. The final arrangement is shown in Figure 7.9. The layout shown in Figure7.9 shows that the motor shaft is connected to a small diameter gear and the linear array element is connected

to a bigger diameter gear (sector of 45^0) via a pin which also is connected to a square piece of perspex connecting the motor and the linear array element. So a transmission system of ratio 5:1 between the motor and the linear array element, i.e. if the motor has a half step of a 0.9^0, an angular displacement of 0.9^0 is obtained when the motor is stepped 5 steps. To avoid loosing contact between the two gears the end part of the motor is connected to a thin circular disc with two holes which define the range of scan angle (45^0 maximum scan angle). Each end is detected by a photo-transistor located over the circular disc. Within the allowed range, the motor moves until either side is detected, then the motor is stopped. This is achieved by using the TTL output of the photo-transistor as a flag to the microcomputer when either end is reached. Figure 7.10 illustrates the prototype motorised transducer which is based on the stepper motor. The advantages of this arrangement are the small size with comparison to the first design and the ease of use either in a water bath or on patients.

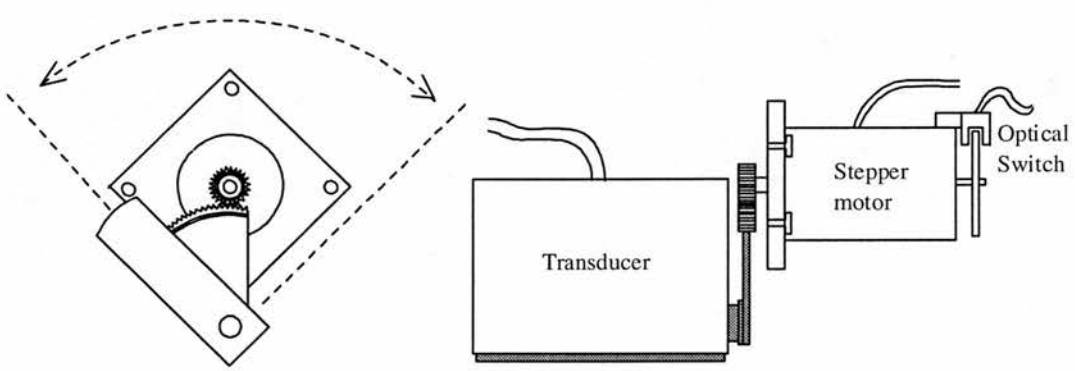


Figure 7.9 : Sketch of the mechanical arrangement of the 3D stepper-based motorised transducer.

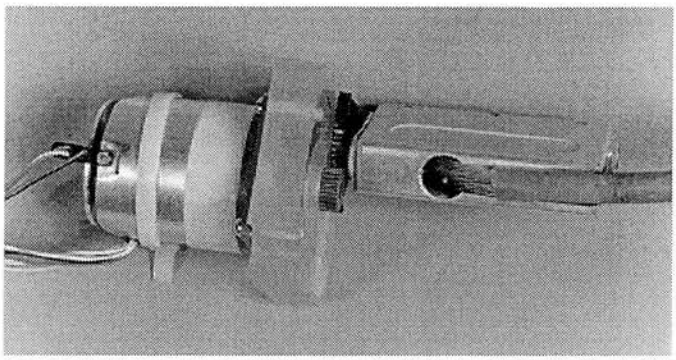


Figure 7.10 : The final design of the 3D stepper motor-based motorised transducer.

7.2.3.2. Stepper Motor and Drive Unit

The stepper motor chosen is similar to the one used on the previously described C-scan system. For more information on the stepper motor and drive unit refer to Chapter 3.

7.3. Overview of Dynamic Imaging Scanner

The Dynamic Imaging scanner of type C2000 (Figure 7.11) is a comprehensive sector/convex/linear ultrasound system capable of accepting different transducers of different frequency (3.5 MHz to 10 MHz). The machine can be used on different applications from routine upper abdominal studies to specialist transvaginal scanning. It has many features such as post processing, 256 grey levels, post freeze magnification, cardiac and obstetric software packages to allow different measurements and alternate focusing. The focusing feature, when a linear or convex transducer is used, gives good resolution along the length of the ultrasound beam. In addition to the above features, the machine has 4 gain controls: initial, near, far and overall gain. It has a video output to enable a fast means of recording on-screen images and information when connected to a suitable video printer.

When using the linear array of frequency 5 MHz, a field of view 80 mm wide and 130 mm deep can be obtained. The frame rate is 16 f/s.

The system transmits focused ultrasound beams from the transducer by applying pulses in a suitable pattern to several of its 79 elements at a time. An ultrasound frame is made up of 160 beams. The system receives the echoes from along the central axis of the beams, amplifies and processes them before passing each line to a digital scan converter for conversion. The converted samples are stored in a memory. The data are then read out and mixed with TV synchronising pulses. Finally, the mixed data is fed into a digital to analogue converter to produce a composite video signal suitable for display on the monitor.

Figure 7.11 : The Dynamic Imaging Scanner type C2000 (courtesy of Dynamic Imaging Ltd.)

7.4. Image Processor Board

The image processor board is based on the DSP32C digital signal processor from AT & T (USA). It is 25 MFLOPS processing engine. It has a frame grab/display subsystem. The DSP32C is capable of executing arithmetic and logical operations in a single 80 nsec instruction cycle. A 3 x 3 convolution can be completed in less that 1.5µs/pixel. Calculations can be performed on 8,16 or 24 bit integers or 32 bit floating-point data values if a wide dynamic range is required. The block diagram (Figure 7.12) shows that in addition to the DSP32C's on-chip memory of 1.5 K x 32 Static Random Access Memory (SRAM), it has a bank of SRAM of a total of 128 K x 32 of zero wait state with 55 nsec access times. The board has also Video Random Access Memory (VRAM) and this with SRAM can be used to capture, store and display using the pixel port on the

VRAM, while processing of images is performed in parallel on images previously transferred to the board's faster SRAM using the VRAM's parallel port. The board has also a video acquisition system which consists of three software-selectable input sources. These are multiplexed to a single, user-selectable, video filter. The video signal is digitised using an on-board 8-bit video ADC of 15 MHz sample rate. It has also a pixel clock that can be operated at a slower rate as required. In addition to that the board provides for both external and internal sync generation. The board has a video output system which is based on a combined triple 8-bit DAC and programmable look-up table. Finally and most important the board can be interfaced to external hardware off-board via a DSPLINK 16 bit I/O port.

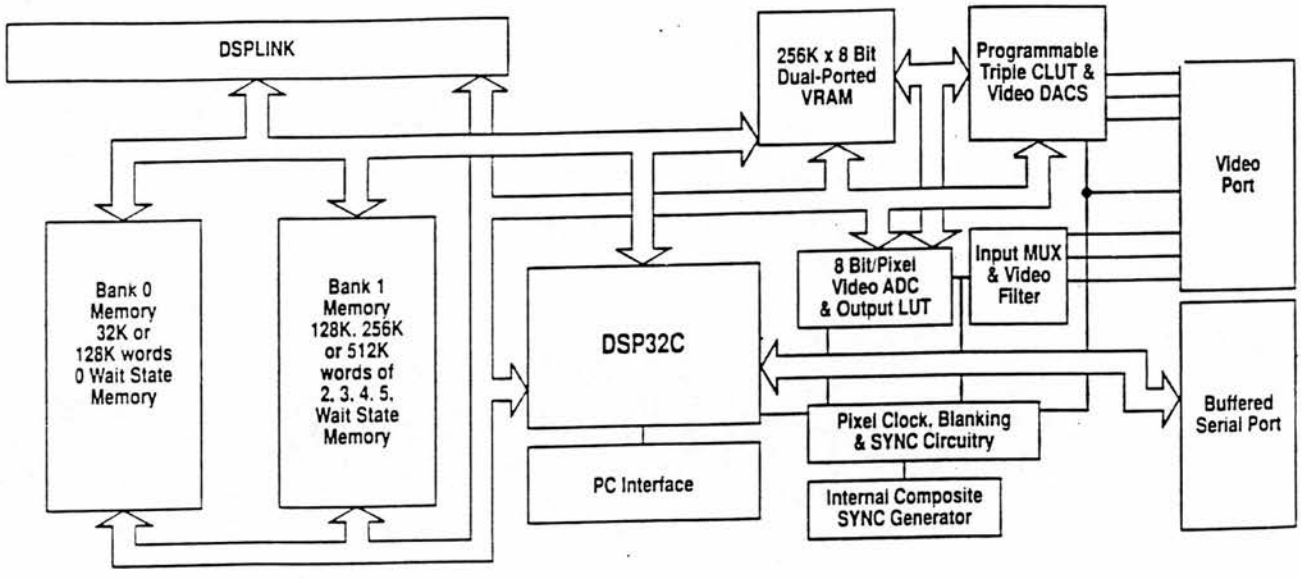


Figure 7.12 : The DSP32C Image Processor block diagram.

7.5. 96 PIO Board

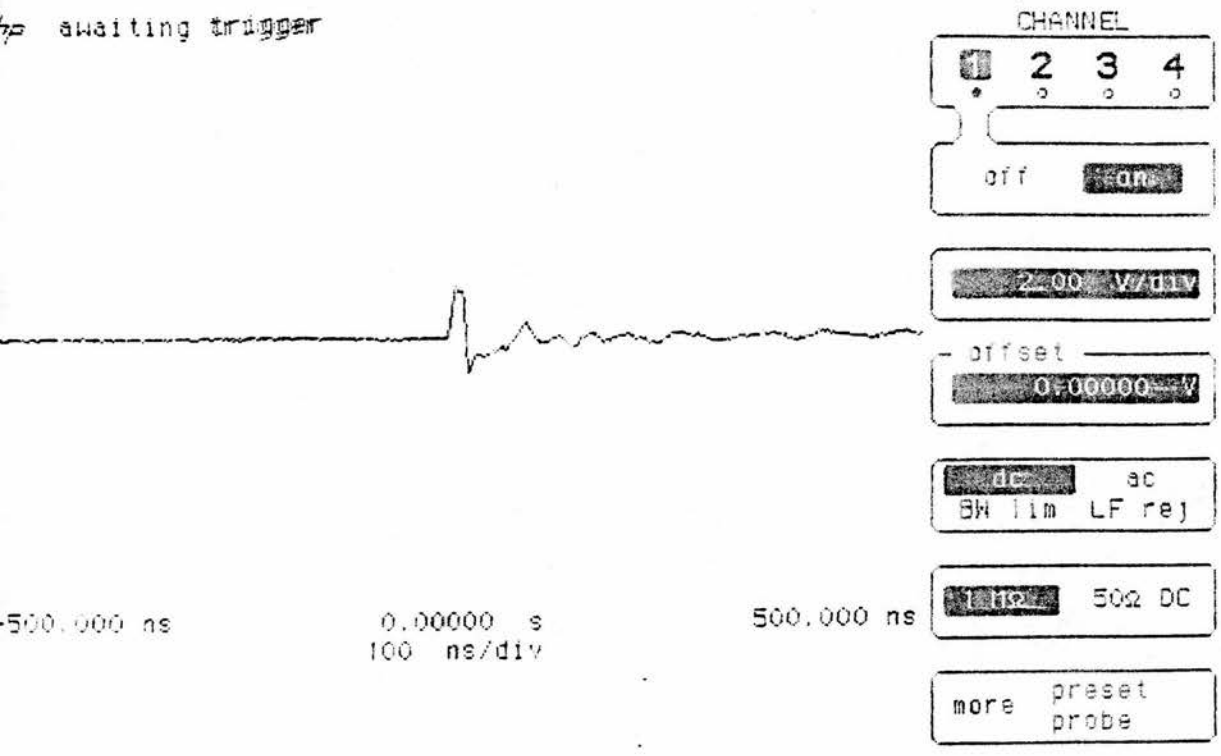
The CIO-DIO96 is a digital I/O board. It has four 8255 digital I/O chips, each controlling 24 digital lines, the board may be used for byte wide or bit I/O of TTL level signals. The interface to the CIO-DIO96 is via two ribbon cable connectors, each cable carries 48 digital I/O lines, + 5V and ground. Each 8255 has 24 I/O lines. The chip is configured as 3 ports. Two ports, A & B, are 8 bits wide. Port C may be an 8 bit port or two 4 bit ports. Individual ports may be configured as Input or Output and are written to and read from as a unit. The addresses of the chips are:

Chip U2 = BASE address + 0
Chip U1 = BASE address + 4
Chip U8 = BASE address + 8
Chip U7 = BASE address + 12

7.5.1. Noise Considerations

Noise is unavoidable in PC based data acquisition systems. There are many sources of noise. The first source of noise is the board itself. The second source of noise is signal wires, especially single ended inputs subject to Electro-Magnetic Interference (EMI) and Radio Frequency Interference (RFI), both of which can induce noise on the wires carrying the external data signal to the board. There is also noise at the signal source itself. All these sources of noise combine to create a region of uncertainty around the signal value. Although small noise levels are of concern to analog readings, sometimes noise from electro mechanical sources such as motor driver may be great enough to cause false triggering of digital inputs.

The noise is discussed because all of the problems encountered when the board was interfaced to the whole system were through false triggering of digital inputs (Figure 7.13). After characterising the noise signal and its frequency, a hardware filter was designed and built and put in the inputs which carry critical signals. The response of filtered system to a step-function is shown in Figure 7.14.



	Sensitivity	Offset	Probe	Coupling
Channel 1	2.00 V/div	0.00000 V	10.00 :1	dc (1M ohm)

Trigger mode : Edge
Pos Positive Edge Of Chan1
Trigger Level
Chan1 = 1.00000 V (noise reject OFF)
Holdoff = 40.000 ns

Figure 7.13 : The noise signal.

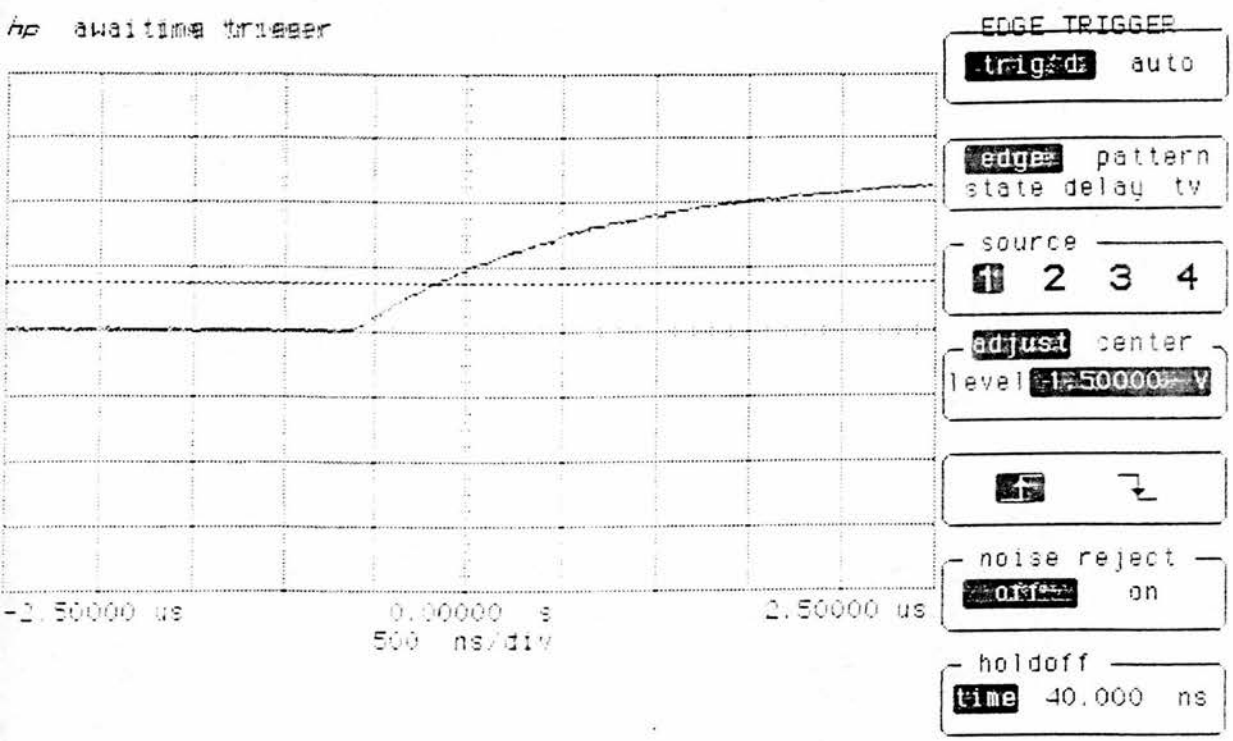


Figure 7.14 : Response of filtered system to a step-function.

7.6. Fast Data Capture Unit

The fast data capture unit is considered the heart of the system since all different parts of 3D ultrasound imaging system are connected to. The unit comprises four boards as shown in Fig 7.15. These boards are the main board (TDU board) which has the digital control and synchronisation circuits, the second is the analog conditioning circuit board, the third is the analog-to-digital (ADC) converter board and the fourth is the power supply. The unit is interfaced to the motor drive via BNC cables, to the scanner via SMA coaxial cables, and to the signal processor and programmable parallel input/output boards in the microcomputer via two ribbon cables. The following sections describe every part and its function in detail.

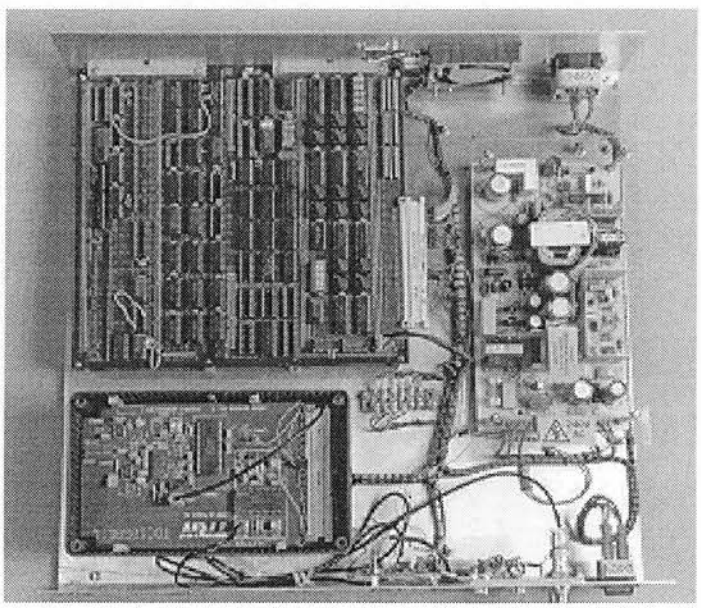


Figure 7.15 : The fast data capture unit boards.

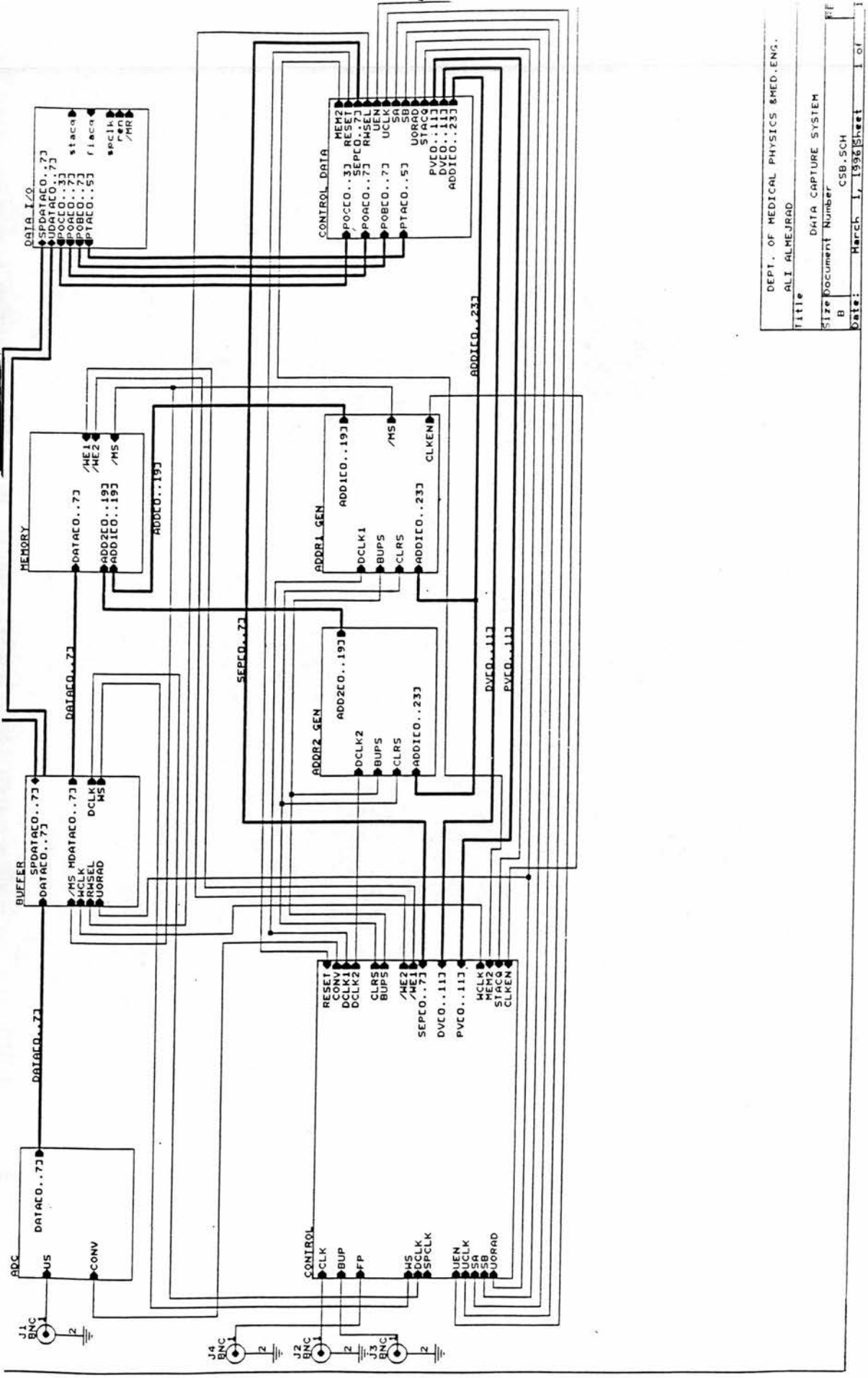


Figure 7.16 : The fast data capture board block diagram.

7.6.1. Fast Data Capture Board

The fast data capture board as shown in the block diagram in Figure 7.16 is split up into different sections as shown below:

a) Buffer section.
b) Memory section.
c) Address 1 generation section.
d) Address 2 generation section.
e) I/O data section.
f) Control data section.
g) Control section

The board is operated in two modes: the first mode is run mode when the system is scanning the object, and the second mode is test mode when the system is in standby or the microcomputer is testing the different parts in the system and especially the memory.

Buffer section

The function of this section is to buffer data which comes from the ADC board when the board is in run mode and data which flows between the memory and the microcomputer when in test mode. This buffer is designed to avoid clashes on the data bus when different parts of the system are running simultaneously.

Memory section

This section contains two memories of 1M x 8 bits of static random access memory (SRAM) modules which is the maximum amount required for storage. The two memories work alternately way i.e. when one memory is being written the other is being read and vice versa for maximising the time for the digital processor to store and process the data.

Address 1 generation section

The function of this section is to provide the address to the first memory which every acquired or read sample needs if the memory is in write or read mode. The address counters are started from

the first address every 2D frame. If the maximum address is reached a flag is produced which in turn triggers a specific circuit in the control section, which will be described later, to end acquisition. The software then shows an error message indicating a failure in the acquisition of data.

Address 2 generation section

The function of this section is to provide the address to the second memory which every acquired or read sample needs if the memory is in read or write mode. The address counters are started from the first address every 2D frame.

I/O data section

The function of this section is to interface between the processor board, the PIO board and the microcomputer. Also the I/O data section interfaces between the microcomputer and the other sections which deal with the written or read data . Most of the circuit in this section is to provide up to 16 bi-directional parallel I/O ports and a pair of 74LS138 to decode four I/O ports. In this circuit, the first port is used. Many strobes are produced by the processor board in combination with a W/R signal which is used to enable write clocks in the other related parts in the board. The other data provided by this section is to control the start and end of acquisition.

Control data section

The function of this section is to provide, firstly, all control signals which control the operation of the board and, secondly, all set-up values to run the system as required. All control signals and set-up values can be changed by microcomputer or manually by using hex switches. The last feature enables testing of the system without the need for the microcomputer. This excludes one source of error as explained before, namely the noise problem on the PIO board. The set-up values are loaded by the microcomputer via two 8-bit latches (74LS373) and by using a suitable control signal to load the required value into appropriate counters in the control section. There are five control signals: µclk, µORAD, W/R, Clr and MotDir and four set-up values: PV (12 bit wide), DV (12 bit wide), STEPS (8 bit wide) and NLINES (8 bit wide). All the counters are loaded with the conjugate value to produce the required delay to count up and then a preset value is loaded again. These steps are repeated as required until the scan is completed.

Control section

The control section provides controls of all the timing and selection of all signals in the board. This section contains subsections as follows:

Clock generation :

The master frequency of 8 MHz is used which gives 125 nsec clock period. This main frequency is used to synchronise all signals of the board.

Programmable delay :

the objective of this part is to determine the depth at which the object is imaged. This delay will be repeated every ultrasound line. The value is pre-set using the microcomputer as explained in the previous section.

Ultrasound line counter:

This counter is used to count the number of the required lines. This is also loaded by the microcomputer.

Motor steps production :

The function of this part is to control the motor movement down to a resolution of 0.2^0.

Processor interrupt production :

The objective of this part is to produce the processor interrupt in the right time and after synchronisation with all parts of the system.

Others:

Apart from the above subsections, the rest are used to control the operation of the board.

To complete the discussion of the electronics, there is an additional circuit to read scan angle with every ultrasound line as explained previously in addition to the original seven sections of the fast data capture board. This circuit was used with the DC-based motorised transducer. The encoder is mounted directly on the motor, then the shaft is rotated and the outputs monitored with a scope to align the encoder body. The encoder has three outputs. The two-channel output (channel A and

channel B) with 90^0 phase shift permits direction sensing while the third channel (Channel I) produces a zero-reference position. These channels are interfaced to a quadature decoder counter (HCLT-2020) (Hewllet Packard Optoelectronics Designer's Catalogue) which consists of a quadrature decoder, a binary up/down state counter and an 8-bit bus interface. The other parts of the circuit are to synchronise between reading the acquired data and angle encoded data by reading the angle and then the acquired data for each ultrasound line.

7.6.2. Data Conversion Board

The sampling theorem states that the maximum sampling rate necessary to completely recover a bandwidth limited signal is at least twice the maximum frequency component of the signal. The presence of wideband noise and the fact that few real signals have a well defined upper frequency limit mean that it is usually necessary to sample at three to five times the maximum significant frequency.

The scanner C2000 from Dynamic Imaging (C2000 manual) was set up by the manufacturer in such a way that the ultrasound is digitised at a specific sample rate depending on which frequency of probe is used. A higher sample rate is used for the higher frequency transducers in order to get all useful information. After experimenting with the scanner using the linear array transducer which is used in the 3D ultrasound system, it was found that the ADC (6 bit resolution) used in the frequency scaler board in the scanner uses a 2.5 MHz sample rate and the input video ultrasound signal is 2.5 Vp-p with - 0.5V offset. Based on the level of these measurements, it was decided to sample at 2 MHz and 8-bit accuracy which is compatible with the typical data transfer rate over DSPLINK (DSP32C PC image processor board user manual). It was found that the 8-bit parallel flash converter (TDC1058 E1C) from TRW(USA) which is used in the C-scan system is a good choice for the above specification. The ADC is capable of sampling up to 20 MHz. The ADC was set up to give maximum output (255) when the output (after conditioning using a two stage amplifier with variable gain and offset) is 1V and minimum output (0) when the output is 0V.

7.6.3. Power Supply Board

The power supply board is required to provide the power to all parts of the system: fast data capture board, signal conditioning board, ADC board and the optical switch. Most of them are TTL compatible which requires +5V. The ADC board and signal conditioning board require -15V, +15V and -5V in addition to +5V. Hence a 60 watt open frame switch mode power supply was chosen to provide the required power, even if additional circuits are added in the future. The specification of the board is shown in Table 7.3:

Maximum Power	60 Watt
Output 1	+5V @ 6A
Output 2	+15V @ 2A
Output 3	-15V @ 0.7A
Output 4	-5V @ 0.7A
Input Voltage Range	220 VAC
Regulation 1	± 4%
Regulation Others	± 5%
Noise Peak-Peak	1%

Table 7.3 : The power supply board specifications.

7.7. Scanner's Hardware Simulation

The 3-D ultrasound prototype system has many components connected to each other. This means more sources of error. For easy testing, easy timing checks and easy detection of signal levels and spikes in the fast data capture unit during construction, a scanner simulator was required (Figure 7.17) which has stable signals of line pulses, frame pulses and a synchronised clock with well defined timing. Timing of line pulses and frame pulses can be defined as in the actual scanner. Also a number of lines can be set manually as required. In addition to the above signals, an ultrasound signal was simulated by using a signal generator making it easy to test the data acquisition system.

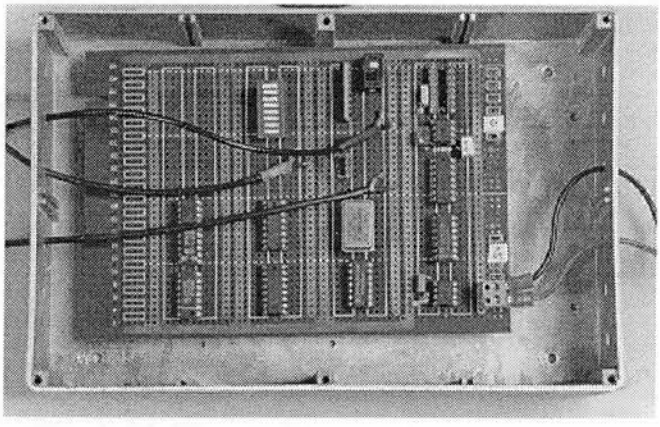


Figure 7.17 : The scanner simulator.

CHAPTER 8

DESIGN AND SPECIFICATION OF 3-D ULTRASOUND SYSTEM: CONTROL, ACQUISITION AND PROCESSING PROGRAMS

CHAPTER 8

DESIGN AND SPECIFICATION OF 3-D ULTRASOUND SYSTEM: CONTROL, ACQUISITION AND PROCESSING PROGRAMS

8.1. Introduction

The microcomputer based instrumentation requires hardware and software to control it and provide an easy user interface. The development of software involves many stages as explained in chapter 4 which are:

i) Design and software specification
ii) Software development and debugging
iii) Software testing
iv) Documentation

Due to limitations in the availability of memory encountered when the DOS application was used in the C-scan system discussed in Chapter 4 especially when dealing with images, it was decided to use a Windows application. The Windows application is based on Object Oriented Programming (OOP) methods. Testpoint software based on OOP was used in addition to using the high level language Borland C++ (Version 4) and its tools which were described previously on Chapter 4. Use of an image processor from AT & T based on the DSP32C digital signal processor chip requires specific software to run it. The software support library (WE DSP32C Support Software Library User Manual, 1988) from AT & T includes assembler, linker, simulator and LSI high level language interface library (LSI DSP32C user manual 1991).

One of the aims of this thesis as outlined in Chapter 1 is the study of the feasibility of real-time image processing as applied to 2D and then 3D ultrasound images. Hence, the first step in such a study is to investigate the real-time image processing as applied to 2D images. This study is

presented in section 8.2. This is followed by descriptions of the Windows application data acquisition control and display programs in section 8.3 and section 8.4 respectively. Reconstruction of 3D ultrasound images using open-windows on a SUN workstation is described in section 8.5. Finally, test and diagnostic programs are described.

8.2. Investigation of Real-Time Image Processing

The first step towards applying image processing algorithms in real-time is to investigate implementing image processing algorithms using IMP32C which was described in section 7.4.

The arrangement of this experiment consists of 25 MFLOPS DSP32C, a video ADC and DAC with look-up tables to allow non-linear transfer functions to be programmed. Also there are two distinct areas of memory, 256 x 256 bytes of VRAM which is used for capturing and displaying images on the monitor and an area of zero wait state faster SRAM which is used for processing. In a real-time application, could be used to process a previously grabbed image in SRAM whilst grabbing a new image into VRAM. The processed image then can be displayed on the video monitor.

To make the hardware work, two separate programs are used , one running on the PC and one running on the DSP32C. These are written in C using Borland C++ (Version 4) and DSP32C assembly language using AT & T's Assembler. The DSP32C program takes care of all the image transfers and image processing. The program has three parts. The first part is responsible for the initialisation of the interrupt vector table pointer and the allocation of a few specific memory locations to store flags. Then the program initialises the colour look-up-table and the input look-up-table. The program enters an idle loop which contains the repeated tests for any flags having been set in DSP memory by the PC program. There are three flags, one to initiate a 3 x 3 convolution of the image contained in SRAM and this is used for most of the image processing algorithms and two flags to control the transfer between VRAM and SRAM (i.e. VRAM to SRAM or SRAM to VRAM). The second part is the convolution routine which is used to perform a 3 x 3 convolution on the 256 x 256 image in memory. The third part is the interrupt service routine which is responsible for refreshing VRAM every 8 ms at the start of each horizontal scan line of an

image. This routine must be active as a background processor task to maintain an active display, capture and store, or refresh the image memory.

The PC program is a menu driven program for DOS (the window version is compatible with the data acquisition and control program which will be discussed later in this chapter) and is responsible for initialising the DSP32C, downloading and starting the DSP assembly program. After the DSP program is running, the PC program uses the C library functions (LSI library) to set flag values in DSP memory to obtain the choice of operation, the convolution routine which is used to implement the image processing algorithm - the aim of this study.

Experimentation with this system shows that the 2D image processing algorithms used can be implemented in real-time.

8.3. Data Acquisition and Control Program

The program is a window application program based on Object-Oriented Programming methods using Testpoint software. The program was designed to be used for two purposes, one is used as interface between the IMP32C board and the PC's user (i.e. as IMP32C driver) similar to the objective of the previous section and the other is to control the operation of the 3D ultrasound imaging system. Figure 8.1 shows the main window of the program with the two user's interfaces. Both the two user's interfaces have child windows which control the operation of the two interfaces. Hence the IMP32C driver has a similar function as discussed in the previous section, this part of the program will not be discussed further. The data acquisition and control program has two child windows in addition to the main window, one as shown in Figure 8.2 is to initialise the files required to run the DSP program and store the data with its header. The other as shown in Figure 8.3 is to initialise the PIO and the fast data capture boards, load the different parameters (depth, number of samples, lines and frames) which control the operation of the fast data capture board to obtain the required scan. Also in this window, there is a control panel for the 3D motorised transducer. This control involves the 3D motorised transducer set-up to step it to the angular position where the scan is started from and the speed of the movement. The number of steps to move during set-up and scan can also be set prior to the process.

To make a working program, the program required two programs to be written, first a Dynamic Linkage Library (DLL) which is linked at run time as a part of the IMP32C driver which was mentioned in the beginning of this section and second a DSP assembly program which takes care of data acquisition. These two programs will be described in the following subsections.

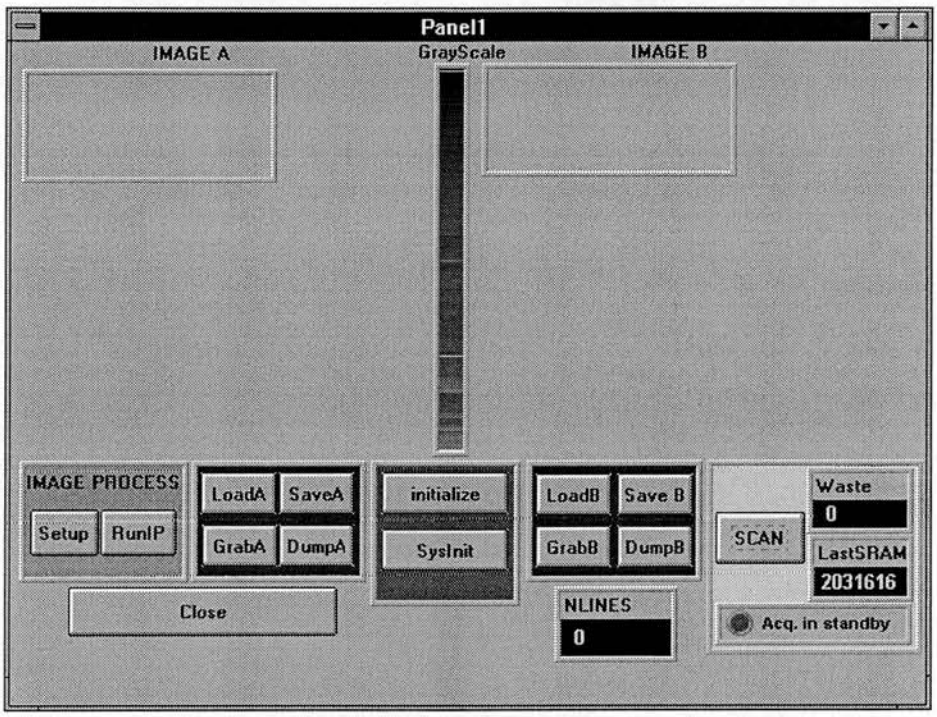


Figure 8.1 : The main window of the data acquisition and control program user interface.

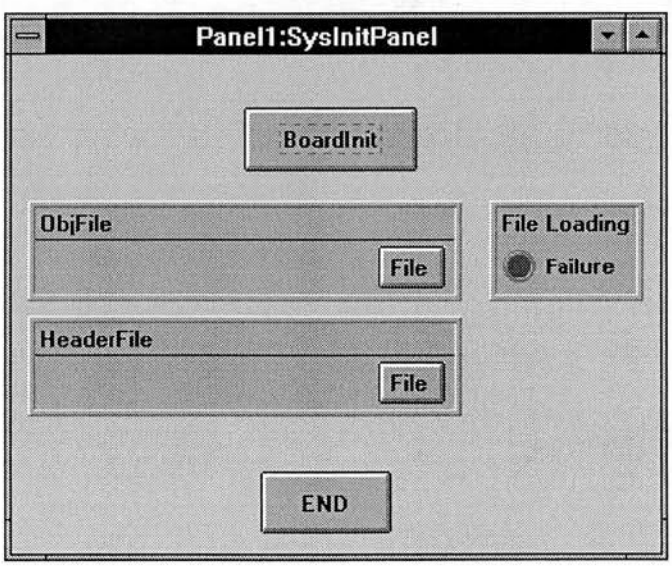


Figure 8.2 : The files set-up window of the data acquisition and control program user interface.

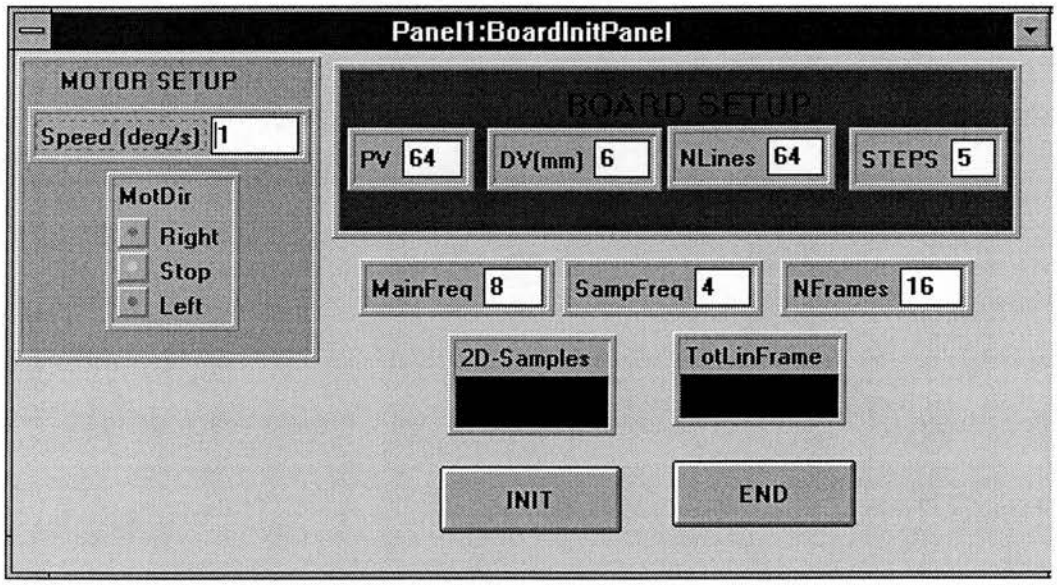


Figure 8.3 : Boards initilisation window of the data acquisition and control program user interface.

8.3.1. IMP32C Dynamic Linkage Library

There are a number of approaches about how to build DLLs. The approach which was used is by using normal C ++ language. The DLL will not be object orientated because it may be used for systems that do not support this feature. Hence only normal C/windows-API functions were used. In order to make the DLL functions accessible to other windows applications such as executed (exe) or other DLL, they must be exported from the DLL.

To create the required DLL four files were required:

- The definition file (Impdrv.def):

This file holds essential information about the current project (Borland C++ user's Guide)

- The DLL shell file (Shelldrv.cpp):

It holds the definition of function defining the DLL entry/exit points. These functions are essential when building DLL by this approach.

- The header file (DSP32C.h) :

It holds the declaration of all exportable functions in this file.

- The main file (Impdrv.cpp) :

It holds the actual definition of all functions necessary to control the IMP32C board.

These above files were included in the project using the Borland C++ 4.0 IDE. This project has been created with a target to DLL, large memory model with explicitly exportable functions option. The last option makes the compiler export only those functions that are marked with the flag 'WINAPI-export'. After compiling this project the DLL file DSPDRIVE.DLL and the import library DSPDRIVE.LIB were generated.

This is a brief description of how the DLL file was built. Now a description of how the DLL was used in the application follows. To use a function from DLL the code is not linked into the application, but rather uses a two steps method:

1. At link time, TLINK binds import records which contain DLL and procedure location information to the executed file. This will satisfy any external references to DLL functions in the code. These import records are supplied by module definition files or import libraries.

2. At run time, the import-record information is used to locate and bind the DLL functions to the application program.

Because of this arrangement the applications are smaller. Also, many applications can access DLL functions which are loaded when the first application starts and stay there until the last application ends.

8.3.2. The DSP32C Assembly Program

The DSP32C assembly program is the fast interface between the 3D ultrasound imaging system's hardware and the microcomputer. The filename of the source code for the program is TDU-DSP.S and its purpose is to take in digitised 3D data from the ADC and place the results into a buffer that is read by the microcomputer and saved in a file for later use such as image display and off-line construction for 3D image display.

To make this file into a complete down-loadable program, first the file TDU-DSP.S is assembled using the AT & T assembler. Then the AT & T linker is used to produce the down-loadable object file which can be downloaded to the board and executed. For convenience, a batch file is used to assemble and link the assembly program.

This is an overview of how data is read into a buffer in the DSP32C's memory space. The actual working part of the code in TDU-DSP.S is split into two main parts : the main routine and the interrupt service routine (ISR). The main routine starts by defining the address where the microcomputer communicates with the DSP and vice versa. This is followed by setting the interrupt vector table pointer (IVTP), v22, to point to the location (ØXØØØ1Ø) which contains the base address of IVTP. The IVTP contains eight pairs of 32-bit words starting at the location specified in the IVTP register, v22 which was mentioned above. Then the program jumps to START where different registers are initialised and the flags are cleared. The external interrupt is enabled via the processor control word (PCW). This is followed by setting the SRAM address where to save the acquired 3D data. The scan parameters (number of lines and number of frames) are loaded in two of the available registers. Then the program enters in a wait loop which is labelled by WAIT which the program always loops back to after collecting a complete ultrasound line. This loop will continue until the total number of lines in all 2D frames are acquired and then a flag from the microcomputer (PCFLAG) is set to indicate the end of 3D scan as required by the user. After that the program enters in an idle loop which is labelled IDLE which disables the

interrupt and the DSP sets the DSPFLAG. This flag is to acknowledge the end of 3D scan and this in turn enables the 3D data saving by the microcomputer via the window based application program. The other routine is the ISR which starts on the line labelled ISR which is called every ultrasound line. The first task is to save the contents of used registers. This is done by using low memory locations as immediate address operands. Then a loop starts by reading the samples byte by byte along the ultrasound line from the ADC via the DSPLINK which is defined by the address LINKADDR until the end of all required samples. This is followed by testing if all the total lines in all frames are acquired. If the test is true, the routine jumps to the line which is labelled by EndSub and disables the interrupt and then restores the registers which are used in the ISR and returns to the main routine. But if the test is false only the registers are restored and the register which stores the current total lines is decremented and then saved to be restored in the next call to the ISR until the test becomes true after storing the last 2D frame. In this case the main routine takes care of the program to end the DSP program as explained in the main routine discussion.

8.4. Image Display Program

To evaluate the 3D data which was acquired using the 3D ultrasound imaging system and check if the 3D data acquisition is successful or not, a high quality image display is required. The best available system at this phase of this project is a Dell 486 with 33 MHz speed with ultrascan colour monitor. The system has a video chip set which can be programmed up to a maximum resolution of 1024 x 768 pixels, 256 colours (noninterlaced) (Dell 486 Systems User's Guide, 1993). The system was set to a resolution of 486 x 640 pixels , 256 colours. As mentioned in the introduction of this chapter the programs were built using windows, using OOP methods. The user's interface of the image display program is shown in Figure 8.4. The interface has many objects : image display, 3D data parameters (number of samples, number of lines and number of frames), preconversion data format push-button, frame counter, control push-button and close push-button. To display the acquired 2D frames one by one in a continuous mode, some stages are required. Since the program is a windows based application the image format has to be compatible with the available windows image format. It was decided to use the common bitmap (BMP) format which has a resolution of 8-bit, i.e. 256 gray shades. This requires one routine to format the acquired data to be a raster format, i.e. row by row since the data was acquired column by column. This is

accomplished using the preconversion push-button object. The other routine is to convert the formatted data in the previous step to BMP format image. These two routines are executed using PreConv.DLL functions which were built for this purpose. From the displayed 2D frames, it can be checked if the 3D data acquisition is successful or unsuccessful and then the experiment can be repeated again by specifying the right 3D scan parameters. This file is one of the TDU.SYS group which has all 3D ultrasound imaging system programs.

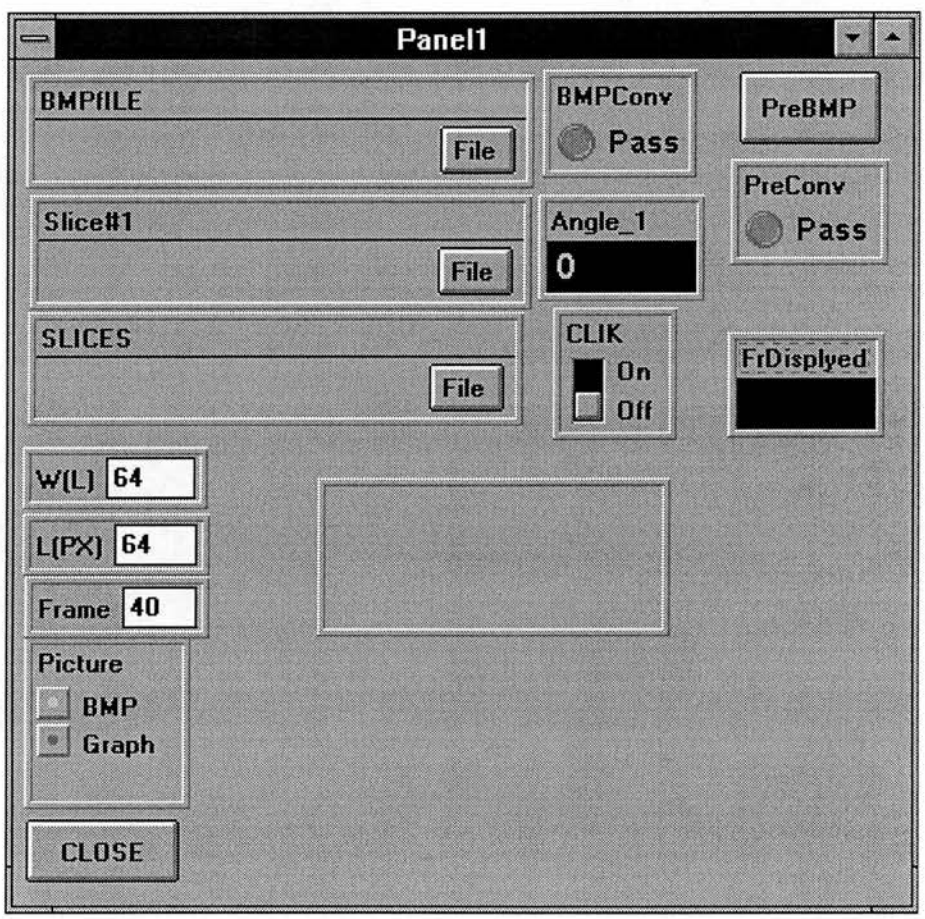


Figure 8.4 : The image display program user interface window.

8.5. 3D Reconstruction, Image Processing and Display Programs

The final stage to obtain the 3D ultrasound images was decided to be done using the SUN workstation because of its suitability for this type of work especially the availability of the 3D imaging software.

To accomplish this objective three steps are required. The first step is the 3D construction of the 3D raw data. This was accomplished using a specific program written using C language. The objective of the program is to reformat and scan-convert the 3D data set which were acquired and checked as explained in the previous sections. To execute the program four parameters are required to be entered: number of samples, number of lines, number of frames and depth of scan with 0.9^{0} between the frames. The converted file was saved in another new file. The second step is to simulate the fast 3D scan by producing the modified 3D data set which suffer from scarcity of data. This was done firstly by reducing the number of lines with the same number of frames and secondly with the reduced number of frames. For convenience a batch file was used to produce the whole set of modified 3D data. These files were then filtered. The 3D median filter was written and run using the same workstation. The third step is to display the reconstructed 3D objects - either the original or the processed images. This was accomplished by using custom software which was written in the Department of Medical Physics and Medical Engineering. The user interface of this program is shown in Figure 8.5. The program offers different display techniques, such as volume rendering and Maximum Intensity Projections (MIP), and many other features.

Finally, it is worth mentioning that two other types of software were considered for comparison with the custom software discussed above. The first one is the SunVision software which runs under Open Windows. One of its tools is the Sun Voxel which consists of window-based volume rendering function that enables the user to generate images from volumetric data which can be manipulated and viewed. This is good when a display of multiple planer slices of raw data such as 2D images is required as shown in the results in chapter 6. Using this software the other display techniques: MIPS and the orthogonal views are not available. For these reasons this software was not suitable for our purpose. The other software is the multidimensional imaging ANALYZE

(Robb et al, 1987) which provides all the image analysis features needed for 3D viewing and processing. It has many features but with a high cost of purchase. The experimentation with our data using ANALYZE was not successful because the 3D images were blurred too much with comparison to the true 3D display of the custom software which was discussed in the beginning of this section.

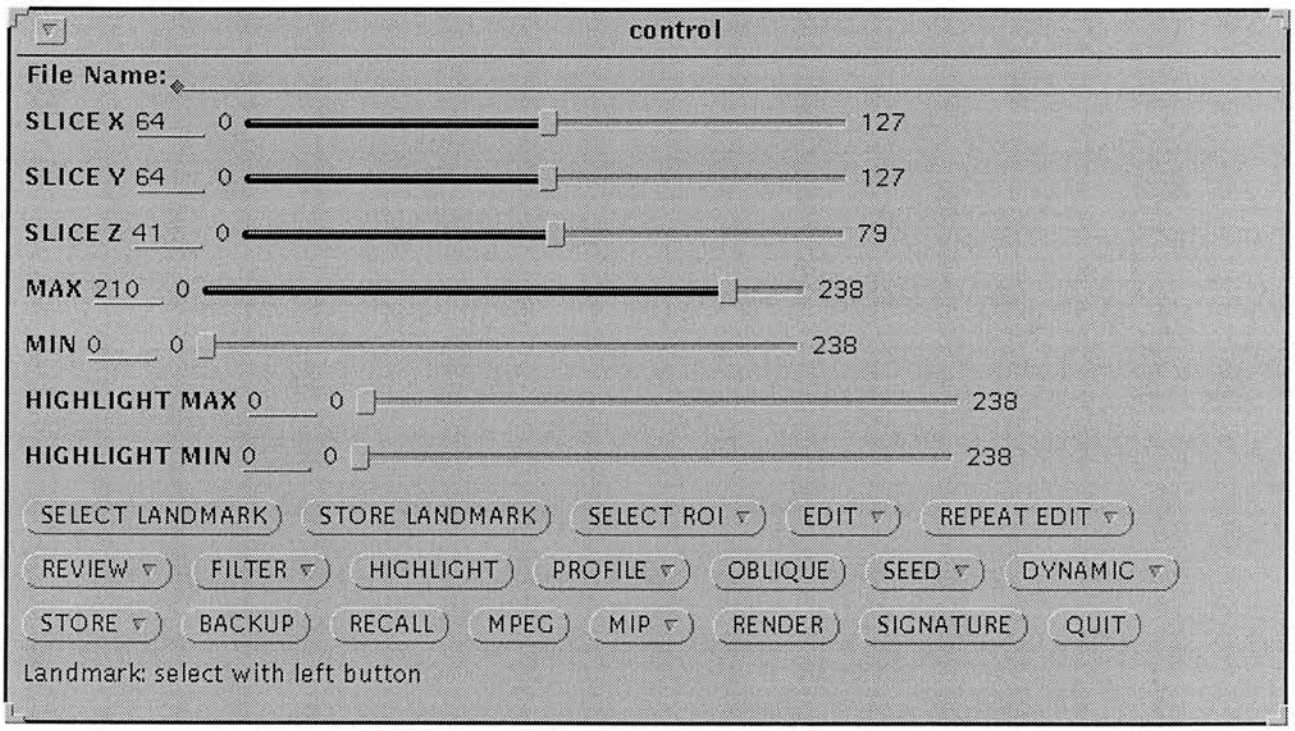


Figure 8.5 : 3D display program user interface.

8.6 Diagnostic Programs

The construction of a 3D ultrasound imaging system involves many stages. It is essential to test the operation of the system during all stages of construction. Many diagnostic windows-based

programs were written using OOP methods. Four of them were built to test the hardware PIO board, DSP32C, memory and motors. For example Figure 8.6 shows the user's interface of PIO board test. Memory test was done using the user interface shown in Figure 8.7. This was done to test if the memory reads and writes correctly by using well known signals with known values of the magnitude and frequency of these signals. The DSP32C assembly programs were tested using internal interrupt with well-known data to enable checking this important part of the software before using it in the final system. Finally there were many small programs written but they are not mentioned here because they are not important compared to the main ones which were discussed above. The results obtained using these programs are presented in Chapter 9.

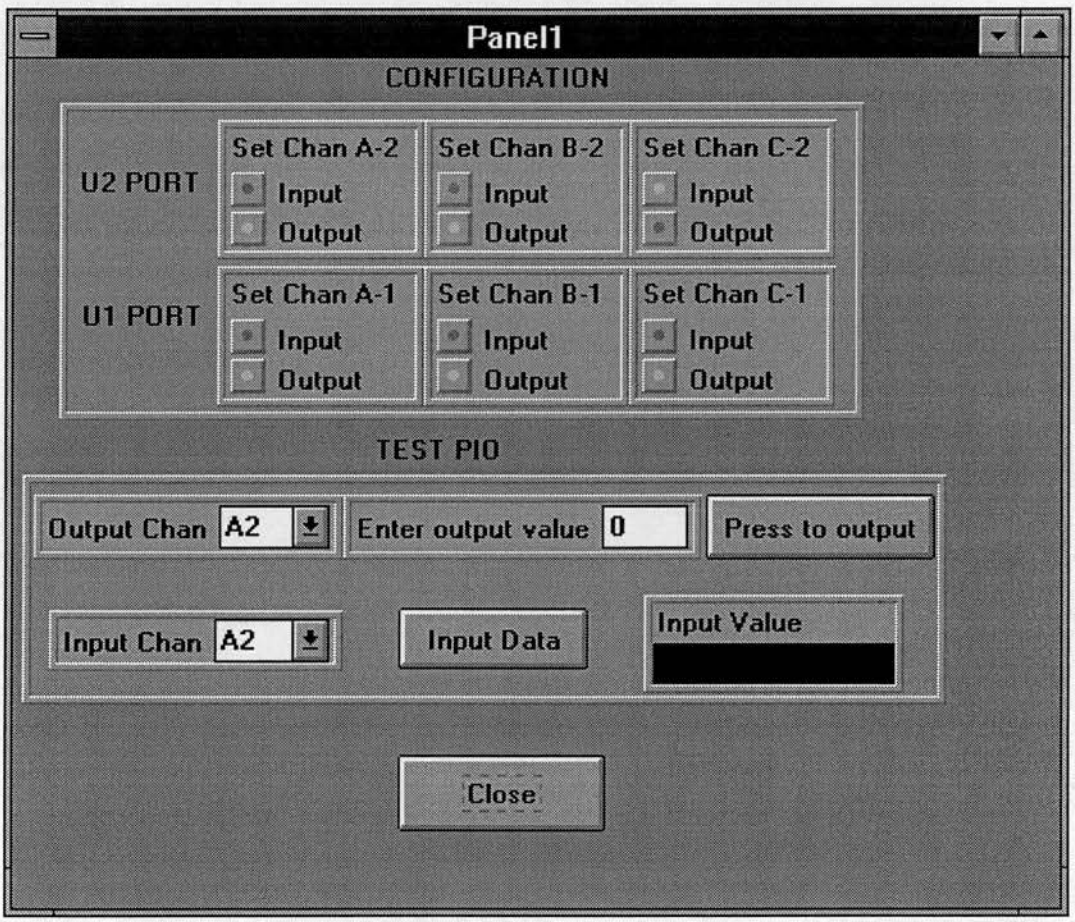


Figure 8.6 : The user interface of PIO diagnostic program.

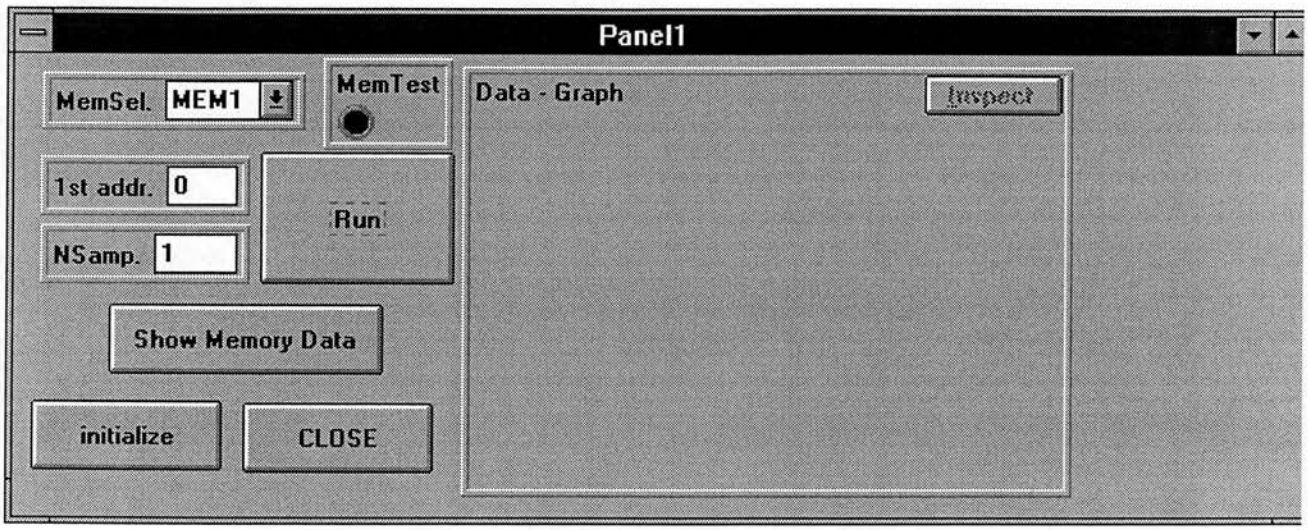


Figure 8.7 : The user interface of memory test program.

CHAPTER 9

EXPERIMENTAL RESULTS AND DISCUSSION OF 3D ULTRASOUND IMAGING SYSTEM

CHAPTER 9

EXPERIMENTAL RESULTS AND DISCUSSION OF 3D ULTRASOUND IMAGING SYSTEM

9.1. Introduction

The aims of this project have been laid out in section 1.3. The main aim is to study the feasibility of achieving real-time 3D ultrasonic imaging by scanning a limited volume and by using modern electronic and image processing techniques. The manufacture of a 3D linear array transducer assembly and design and construction of the test-rig 3D ultrasound imaging system have been described fully in Chapters 6 and 7. The user interface including the software to control the system, acquire, process, reconstruct and display 3D ultrasound images is described in Chapter 8. The current chapter starts in section 9.2 by describing all test objects used to study the feasibility of the real-time 3D ultrasonic imaging . The study methodology used in this chapter is described in section 9.3. The comparison of slow scan 3D images and 3D simulated real-time images, before and after image processing are presented in section 9.4. This is followed by section 9.5 which includes the results of the comparison study of the different 3D display techniques. Finally section 9.6 presents a general discussion of all results presented in this chapter.

9.2 Test Phantoms and B-scan Images

In addition to clinical tests, three different types of test phantom were used in the 3D ultrasound imaging assessment.

The tissue mimicking test phantoms, shown in Figure 9.1, were formed from different shapes of tissue mimicking material and some of these targets were used.

1- Cubic shape (4 cm x 4 cm x 2 cm) with a hole of an equilateral triangular shape of 1 cm.

2- Cubic shape (4 cm x 4 cm x 2 cm) with a hole of a squared shape of 1 cm and 2 cm.

3- Cubic shape (4 cm x 4 cm x 2 cm) with a hole of a circular shape of 1 cm ,2 cm and 3 cm diameter.

4- Cubic shape (4 cm x 4 cm x 2 cm) with a hole of burfication shape.

5- Three circular shapes of different diameter, joined together by a straight tube.

6- Different shapes with many circular holes of different diameter.

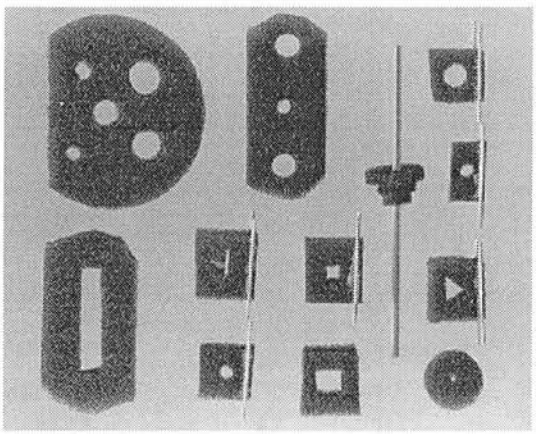


Figure 9.1 : Tissue mimicking phantoms.

The second type of test phantom is the anatomical tube burfication as shown in the experimental arrangement in Figure 9.2. This phantom is made from latex tubes of 1 cm diameter.

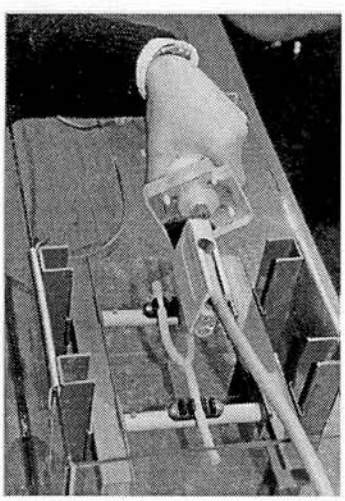


Figure 9.2 : Latex tube phantom experimental arrangement.

The third type of test phantom is the standard Cardiff test phantom shown in the experimental arrangement in Figure 9.3. This phantom is filled with tissue mimicking material which enables a gray scale test of the imaging system.

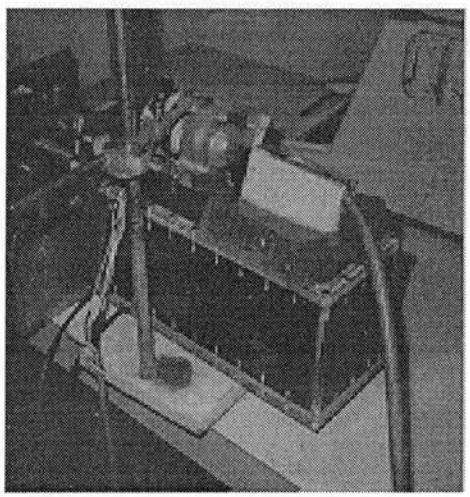


Figure 9.3 : The Cardiff test phantom experimental arrangement.

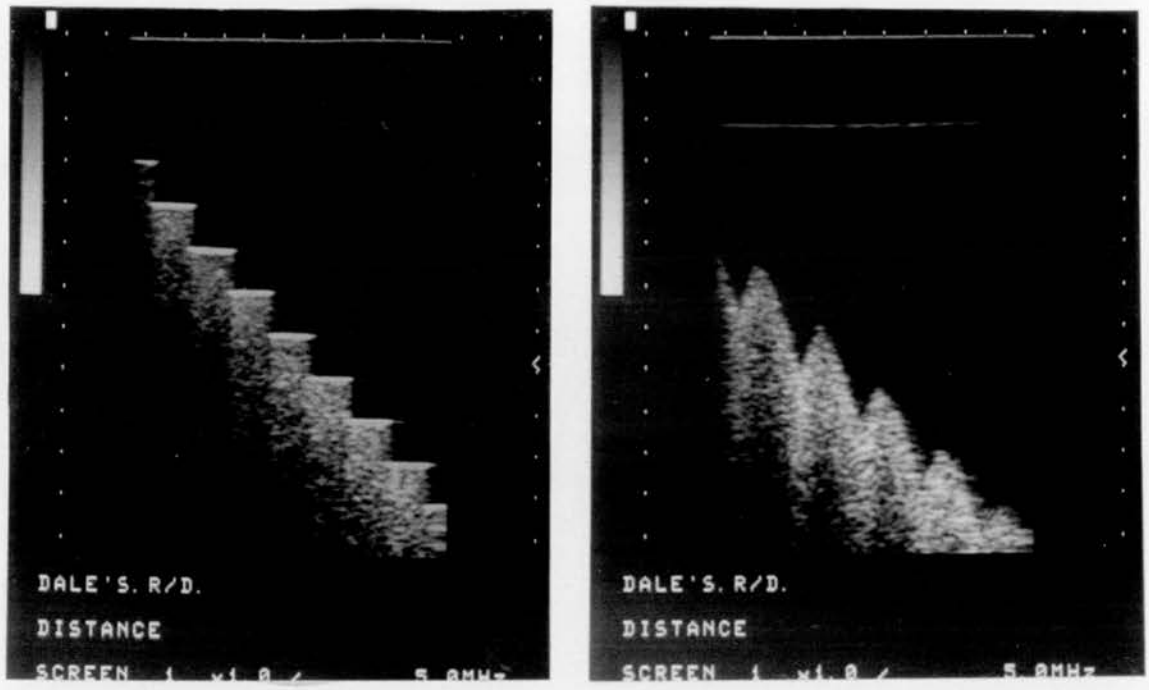
The B-scan images of the test phantoms used in this study are shown in Figure 9.4 and the B-scan images of the clinical tests are shown in Figure 9.5. These B-scan images are referred to when the 3D images are reconstructed later in this chapter.

9.3. Methodology

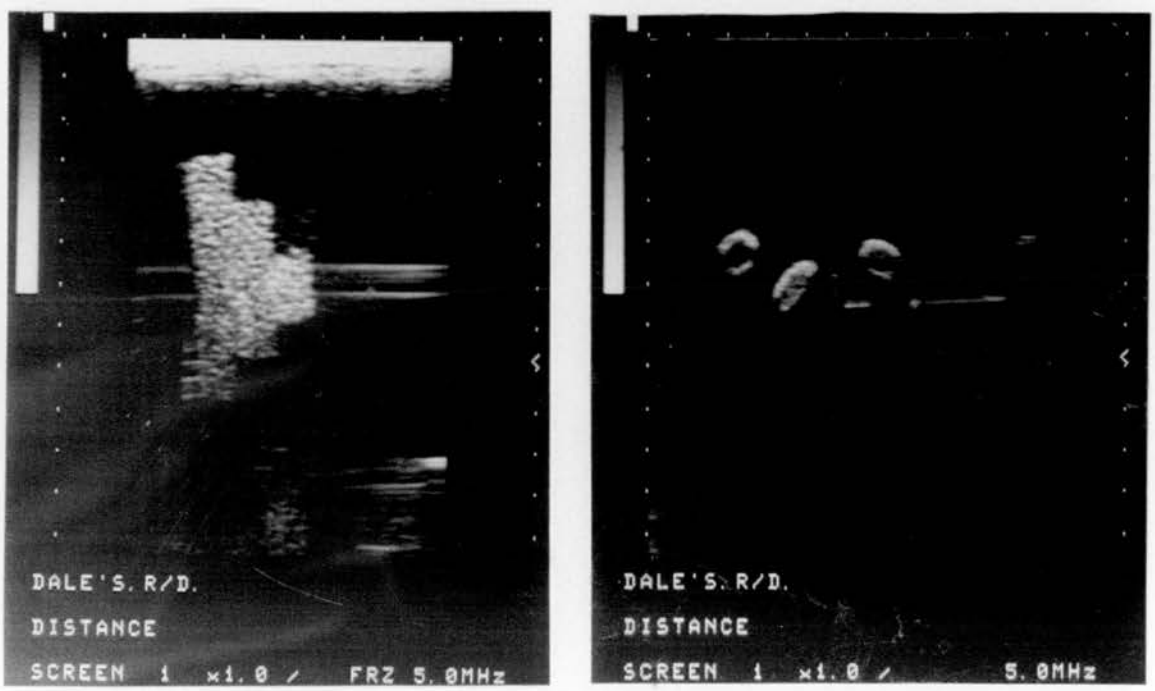
In this section the methodology followed in this chapter to acquire the 3D set, process and display the 3D images is presented.

To acquire the 3D data set, the 3D ultrasound imaging system was used. The experimental arrangements are dependent on the scanned object. There are four different types of experiments: tissue mimicking and tube test phantoms , Cardiff test phantom and clinical experiments.

The first type of study was of the tissue mimicking test phantoms. The experimental rig was composed of three parts: the microcomputerised 3D ultrasound imaging system, a small glass tank

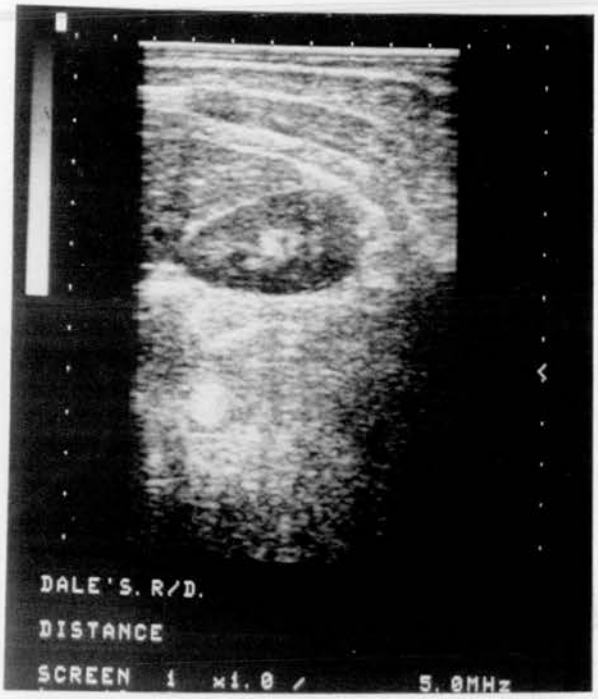


a

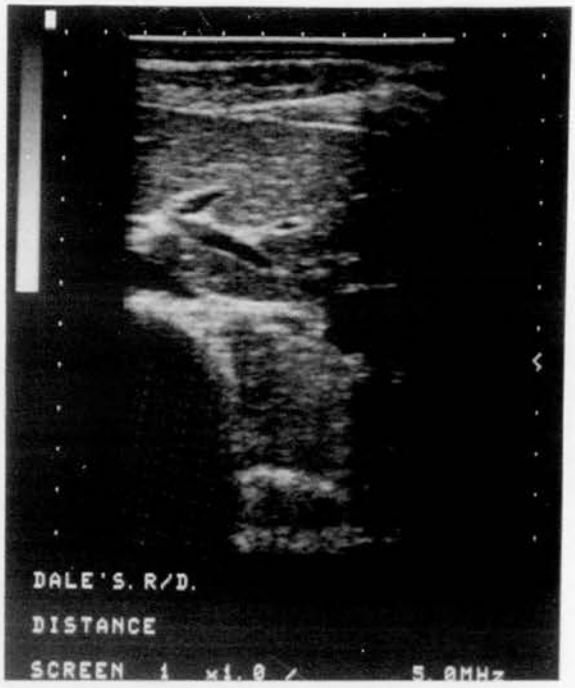


b c

Figure 9.4 : B-scan images. (a) - Tissue mimicking test phantom (Cardiff test-object). (b) - B-scan of tissue mimicking reticulated foam phantom. (c) - B-scan latex tubes with contrast agent in central tube.



a



b

c

Figure 9.5 : B-scan images of soft tissue structures. (a) - Kidney. (b) - Liver blood vessels. (c) - Carotid artery.

filled with water and fitted with a carpet in the base to avoid reflections and the immersed test phantom. The height of the transducer holder could be adjusted to place the face of the transducer under water. The tissue mimicking test phantom was put inside the tank under water and then was squeezed to get rid of bubbles. The system was set-up as described in the above experiments and then the scan was started. When the scan was finished, a 3D data set of the volume of interest was stored. The same steps were repeated for all tissue mimicking test phantoms.

The second type of study was of the latex tube phantom. The experiments were carried out in a manner similar to that of the tissue mimicking test phantoms. In one of these experiments an albumin microspheres contrast agent was also used in the tubes. The contrast agent was prepared by filling a syringe with water, getting rid of excess bubbles, injecting the water into powder in a vial, shaking it and then withdrawing the contrast agent suspension into the syringe. After preparing the contrast agent, it was injected to the latex tube which was clamped at one end and then the other end was clamped to keep the contrast agent inside the tube for scanning.

The third type of study was of the Cardiff test phantom. In this experiment, the 3D ultrasound imaging system, the Cardiff phantom and a jelly transmission medium were used. To obtain the scan the experiment was arranged as shown in Figure 9.3 and the system was set-up as described above in the experiments. When the scan was finished, a 3D data set of the volume of interest was stored.

In clinical experiments, the volunteers laid down on a bed and then the hand held 3D motorised transducer was moved until the desired volume of interest was obtained. This can be achieved by looking at the B-scan image until the area of interest is found using what may be called a ' 2D/3D Duplex ' technique which was mentioned in Chapter 6. After choosing the volume of interest, the system was set-up. The set-up involves the initialisation of the file required to run the DSP program and the file of where to store the 3D data set with its header, loading parameters (depth, number of samples, number of lines and number of frames) which control the operation of the fast data capture board and finally setting the 3D motorised transducer position and speed. Then the volume of interest was scanned and 3D data set was stored. Many clinical experiments were done, but three of them (carotid arteries, kidney and liver arteries) were chosen to be included in this study.

To accomplish our study, sets of images must be prepared off-line using the SUN workstation from the 3D raw data sets. First, 3D images with all the data were reconstructed and second , 3D images which suffer from scarcity of data were reconstructed. The second type of image was required to study simulations of real-time scanning. Fast 3D images with limited data were simulated by formatting the 3D raw data using a program as described in Chapter 8. The different images with limited data were obtained by dropping every second line, with every frame and with every second frame. The same process was repeated for 3D images using only every third line and then every fourth line. The obtained images were pre-processed using interpolation. For convenience this process was done using a batch file. The effect of using 3D imaging processing and specifically the 3D median filter which was found to be the best in Chapter 5, was then studied for some of the images.

Finally methods were assessed of how to display the 3D images to enable them to be interpreted and analysed. Three display techniques were included in the computer program described in Chapter 8.

The different 3D images were evaluated by a physicist with long experience in medical ultrasound.

9.4. Image Processing Results

In this section, the results are presented in a sequence starting from simple tissue mimicking and latex tube phantoms in Figures 9.6 to 9.9 followed by the standard Cardiff phantom in Figures 9.10 and 9.11 and finally some clinical results in Figures 9.12 to 9.15.

Looking at the results in more detail, Figure 9.6 shows the orthogonal slices through the reconstructed 3D image of the tissue mimicking phantom from a slow scan of 80 lines and 80 samples per line (s/l)and 45 frames (i.e. total of 3600 lines) which was acquired at a volume rate of 1 vol/sec in Figure 9.6 (a) and as interpolated fast scan 3D images in Figures 9.6 (b) to 9.6 (h). The data-reduced 3D image with every line and every second frame which has total of 1860 lines is shown in Figure 9.6 (b). Looking at the three orthogonal views of this image, it is noted that the image is not affected too much regarding image detail, boundary definition and contrast except the

little blurring due to the reduction of the number of frames to half. This effect is clear in the longitudinal and the horizontal sections. Figure 9.6 (c) shows more reduced lines (i.e. using every second line) with every frame which produce a 3D image with a total of 1800 lines. Comparing this 3D image to the slow scan 3D image in Figure 9.6 (a) regarding the image characteristics such as image detail, boundary definition and contrast, shows that both images have similar characteristics. The 3D image in Figure 9.6 (d) is reduced more by reducing the number of frames to half producing a 3D image with a total of 880 lines. The resulting 3D image has the same image characteristics as the 3D image shown in Figure 9.6 (b) except for a little more blurring because both images have the same reduced number of frames but different number of lines. More reduced lines giving a 3D image down to a third of the total lines in the slow scan 3D image with the same number of frames (i.e. 990 lines) is produced in Figure 9.6 (e). The result of this figure shows the effect of this reduction which reduced the contrast in all three orthogonal views and produced less image detail in the transverse view compared to the slow scan 3D image in Figure 9.6 (a). The blurring effect is noticed in the horizontal and the longitudinal views in Figures 9.6 (f) which has every third line and every second frame (i.e. 572 lines). Considering images with about one quarter of the total lines with the same number of frames producing a 3D image of 900 lines, they have less contrast and less image detail in all three views but the image in general keeps the boundary definition and the object shape. The image in Figure 9.6 (h) shows a reduced data 3D image with every fourth line and every second frame which has 440 lines. The resulting 3D image still has some detail but with less defined boundaries and more blurring to the three orthogonal views of the 3D image. This image is the last processed because it was felt that the 3D image reached the possible lowest line density which can keep acceptable image detail, boundary definition and good contrast.

The study is extended by applying the image processing to the interpolated fast scan 3D image to see if the image quality can be improved more. The results of applying the median filter with 3x3 x3 kernel and 5x5x5 kernel to the slow scan 3D image in Figure 9.6 (a) and to the interpolated fast scan 3D image with every third line and every frame in Figure 9.6 (e) are presented Figure 9.7. The results show that the median filters improve the image quality with comparison to the unfiltered 3D image. The median filter of both 3x3x3 and 5x5x5 kernels show significantly improved results. However, small detail starts to disappear as the filter kernel increases as shown in Figure 9.7 (b) which shows the results of the filtered slow scan 3D image using 5x5x5 kernel median filter.

Figure 9.7 (c) shows an improved image which features reduced noise and acceptable boundaries compared to using interpolation only (Figure 9.6(c)).

The results of the same study as applied to the latex tube phantom are shown in Figures 9.8 (a) to 9.8 (h). Figure 9.8 shows the orthogonal views of the reconstructed 3D image of the latex tube phantom from a slow scan and from interpolated fast scan 3D images. The reduced data 3D images (i.e with every second line, every third line and every fourth line with every frame and with every second frame) are presented in Figures 9.8 (b) to 9.8 (h). The effects of applying the median filter with a 3x3x3 window and a 5x5x5 window to the original slow scan 3D image in Figure 9.8 (a) and to the interpolated fast scan 3D image with every third line and every frame in Figure 9.8 (e) are presented in Figure 9.9. Because the orthogonal views of this 3D image do not show a lot of detail about the tubes except cross-sections. The effect of the processing can not be distinguished, but in general it seems that similar results to those of the tissue mimicking phantom are obtained.

The standard Cardiff phantom has a lot of detail such as sharp edges, angle-shaped edges and noise on the top surface which permit study of techniques to produce a 3D image of good quality which can be obtained in real-time. Figure 9.10 shows the orthogonal views of the reconstructed 3D image of the phantom from a slow scan in Figure 9.10(a) and then as interpolated fast scan 3D images. The reduced data 3D images (i.e. with every second line, every third line and every fourth line with every frame and with every second frame) are presented in Figures 9.10 (b) to 9.10 (h). The effect of applying the median filter with 3x3x3 window and 5x5x5 window to the original slow scan 3D image in Figure 9.10 (a) and to the interpolated fast scan 3D image with every third line and every frame in Figure 9.10 (e) are presented in Figure 9.11. It is noted from the results in Figure 9.10 that the 3D images keep the image detail, boundary definition and good contrast in all three orthogonal views when the lines are reduced down to third of the total lines with all frames as shown in Figures 9.10 (a), 9.10 (c) and 9.10 (e). When the number of frames are reduced to half of the total frames, the blurring effect starts to appear in the longitudinal and the horizontal views (Figures 9.10 (b), 9.10 (d) and 9.10 (f)). When the lines are reduced to a quarter of the total lines, the blurring effect starts to appear in all three orthogonal views as shown in Figure 9.10 (g). When the line density is reduced to half of the data included in the previous image, the blurring effect is increased more, the boundaries are distorted but not much and the contrast is reduced but the final

shape of the object in all three orthogonal views keeps the similar general shape as the slow scan image which has the maximum line density.

The results of applying median filters of both 3x3x3 and 5x5x5 kernels show significantly improved results as shown in Figure 9.11 (a) ,(b) and (c), compared to the unfiltered 3D images, hence the filtered 3D images have less noise and with well-defined boundaries.

The study is extended to more a practical clinical study. The results of the clinical scans: the carotid arteries are shown in Figure 9.12, the kidney is shown in Figure 9.14 and the liver arteries are shown in Figure 9.15. The results of the effect of applying median filtering to the slow scan and the interpolated fast scan with every third line and every frame of the carotid arteries are shown in Figure 9.13 (a) to (c). In all clinical results the scans have diagnostic information in addition to the images characteristics which requires medical experience to help interpretation of the scans. This requirement is increased when interpreting the 3D images displayed as 2D projections. In this study, the results are discussed from the technical point of view. The results of the carotid arteries are similar to the results obtained from the study of the tissue mimicking, latex tube and the standard Cardiff phantoms. For the reduction of line density to half, third and fourth of the total number of lines as shown in Figures 9.12 (c), 9.12 (e) and 9.12 (g), the quality of the 3D images obtained is comparable to the quality of the slow scan 3D image in Figure 9.12 (a). This means that a real-time scan volume rate can be obtained without loosing much of the image characteristics. On the other hand , the blurring effect appears when the number of frames are reduced to half the total number of frames as shown in Figures 9.12 (b), 9.12 (d), 9.12 (f) and 9.12 (h). This effect is increased when combined with the maximum reduction of lines as shown in Figure 9.12 (h) in which the total number of 440 lines is very low as discussed in detail in the results of the tissue mimicking phantom.

The effect of applying median filters using both 3x3x3 and 5x5x5 as applied to the slow scan 3D image in Figure 9.13 (a) and (b) respectively show improved 3D images regarding the noise reduction and hence smoother images with well-defined boundaries. Similar results are obtained when the median filter with 3x3x3 kernel is applied to the fast scan 3D image of a low line density image of 990 lines as shown in Figure 9.13 (c).

Preprocessed images showing the effect of reducing line density of kidney tissue and liver artery scans are presented in Figure 9.14 and Figure 9.15 respectively. They exhibit results similar to those discussed earlier.

9.5. Display Techniques Results

To complete our study, it is desirable to compare the different display techniques of reconstructed 3D images. One of the display techniques is the orthogonal views display technique which was used in the previous section because of its suitability for the comparison study. This technique is useful to navigate through the 3D image displaying 2D sections in real-time. Two other 3D display techniques were included in the study which are the Volume Rendering (VR) and the Maximum Intensity Projections (MIP) with different thresholds. Because of the large number of images produced with all objects (clinical and test phantoms objects), it was decided to present one of every type (i.e. one clinical, the Cardiff phantom, one tissue mimicking phantom and one latex tube phantom) . Every figure of 9.16 to 9.19 shows the two 3D display techniques for comparison purposes for the tissue mimicking phantom, the latex tube phantom, the standard Cardiff phantom and the kidney respectively. Also in this section the effect of 3D image processing for improving the 3D display was presented for the standard Cardiff phantom and one clinical (liver arteries) as shown in Figure 9.20 and Figure 9.21 respectively. In the results of the Cardiff phantom, it is shown how the median filter is effective in removing the small surface artefacts covering objects of interest and hence improving the 3D visualisation. The full effectiveness of VR and MIP displays can only be appreciated by rotating the images on a display screen.

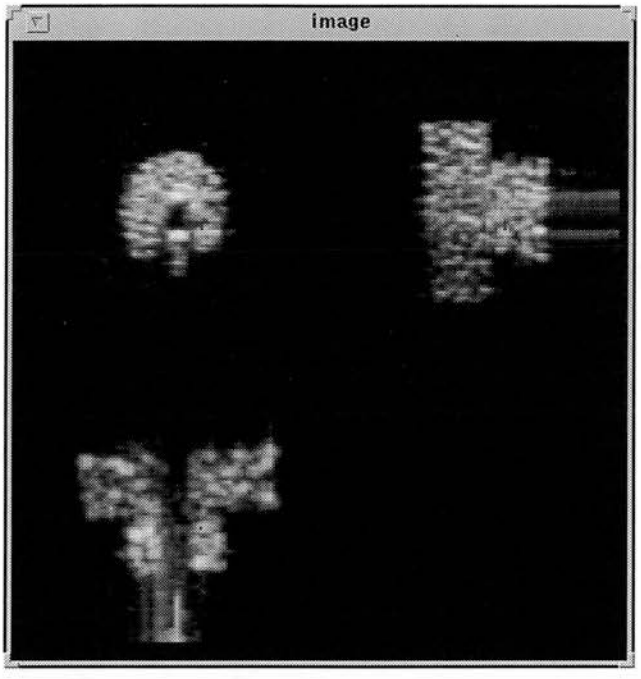
9.6 Discussion

The results of the comparison study of slow scan 3D images and the simulated fast scan 3D images after processing show that the 3D images can be obtained in a real-time volume rate of up to 22 vol/sec at a line density of 440 lines per volume of 27 cm^3 provided a 10 kHz prf rate is used which is possible in the ultrasound scanners. As trade-off between achieving a required real-time volume rate of 10 vol/sec as discussed in Chapter 6 and keeping good quality image, a fast scan image of a

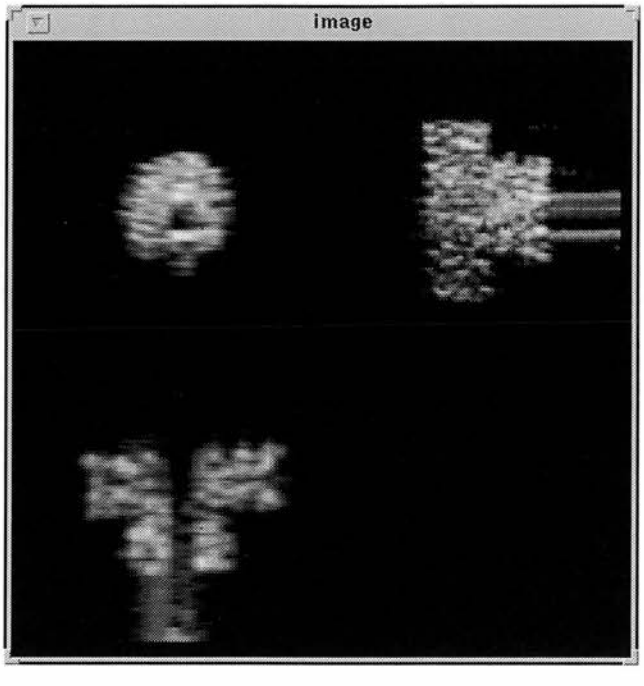
line density of 990 lines has been shown to be desirable. The results of this show acceptable 3D images either interpolated alone or interpolated and filtered.

The quality of the image at this high speed is acceptable as shown in the results of the study as applied to all different types of objects from simple phantom to clinical scans. In this study, it is noted that the acceptable 3D images which features defined boundary, image detail and good contrast can be obtained using only simple interpolation as shown in the figures in section 9.4. Regarding the noise reduction, using the median filter shows smoother images after reducing the noise and removing parts covering objects of interest as was shown in the Cardiff phantom with a minimum loss of detail of the 3D image which is considered important from the clinical point of view. Using the median filters, it is noted that 5x5x5 window produces heavy filtering while 3x3x3 window produces acceptable filtering which is a compromise between reducing the noise and keeping image detail.

The 3D display results show that the orthogonal views display technique is good because it enables the user to navigate through the 3D image by changing the land mark in real-time. It is noted that the MIP display technique is good for displaying the clinical 3D images but not for the test phantoms. On the other hand, the VR is good for displaying the known test phantom but not for the clinical 3D data which require sufficient clinical experience for interpretation. With our present computing system, MIP and VR require a few seconds to be executed but this problem will be discussed in the next chapter on further developments to accomplish them in real-time . Results shown in Figures 9.20 and 9.21 show that the median filter improves the quality of the displayed 3D image compared to the interpolated 3D image which has a band pattern due to gap filling between lines and frames.

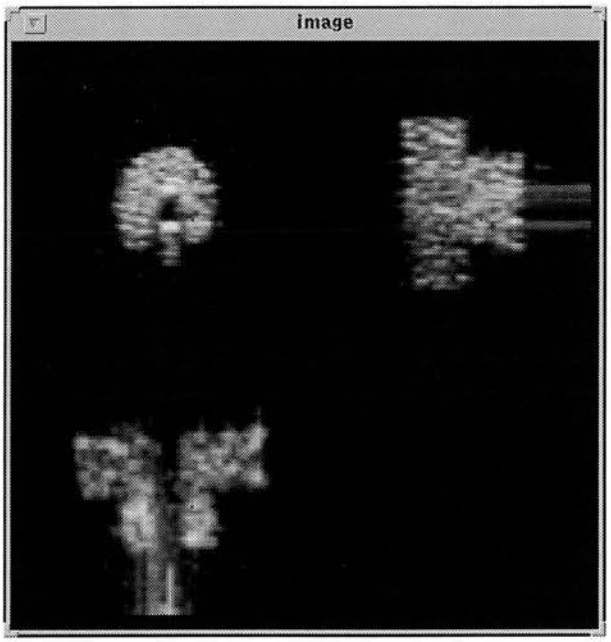


(a)

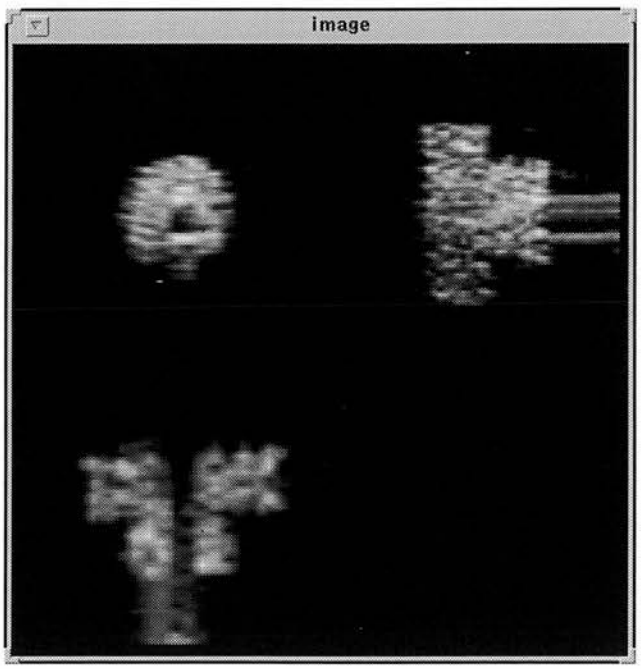


(b)

Figure 9.6 : Orthogonal views of the reconstructed 3D image of tissue mimicking material. (a) - Original slow scan 3D image acquired with every line and every frame. (b) - Pre-processed fast scan 3D image acquired with every line and every 2nd frame.

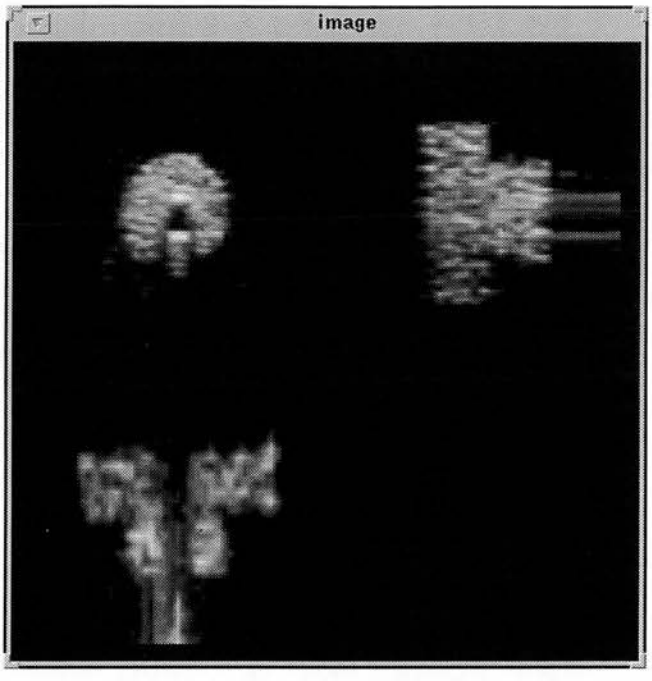


(c)

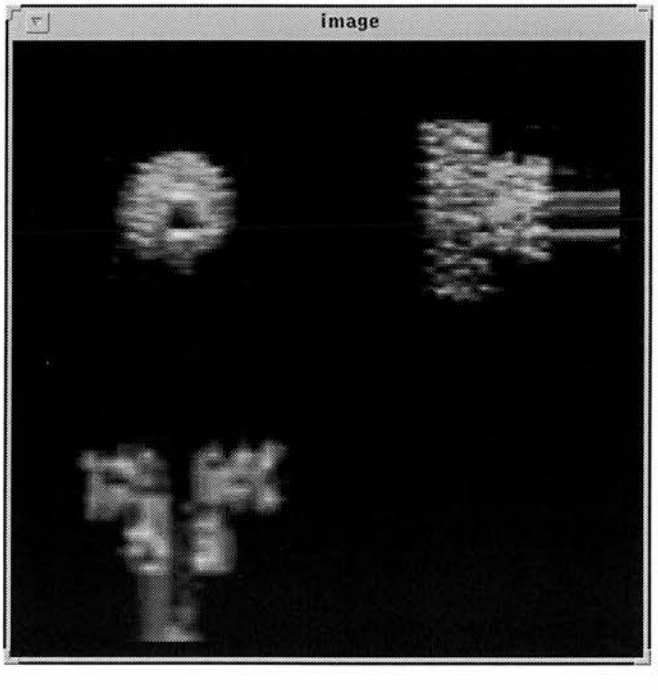


(d)

Figure 9.6 (contd.) : (c) - Pre-processed fast scan 3D image acquired with every 2nd line and every frame. (d) - Pre-processed fast scan 3D image acquired with every 2nd line and every 2nd frame.

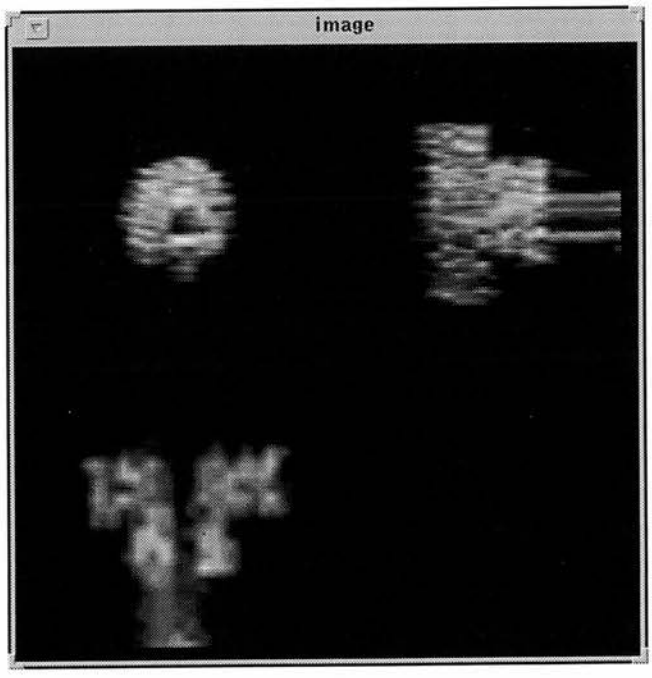


(e)

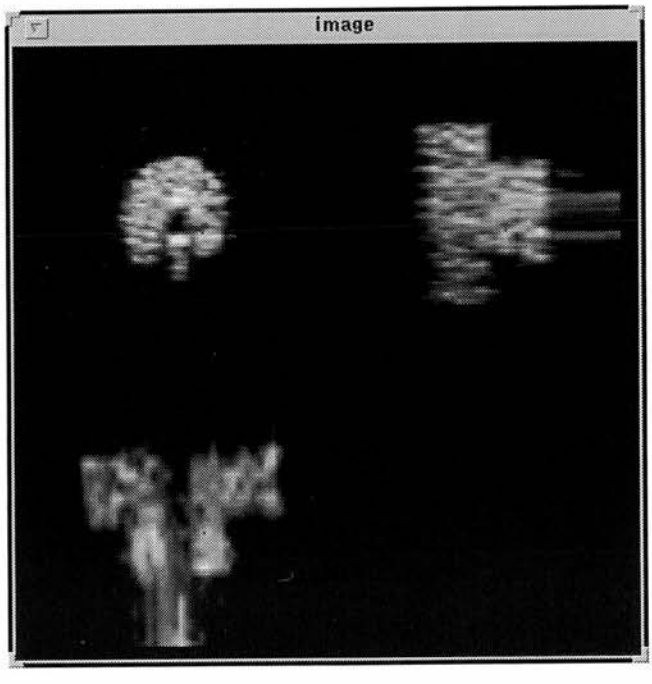


(f)

Figure 9.6 (contd.) : (e) - Pre-processed fast scan 3D image acquired with every 3rd line and every frame. (f) - Pre-processed fast scan 3D image acquired with every 3rd line and every 2nd frame.

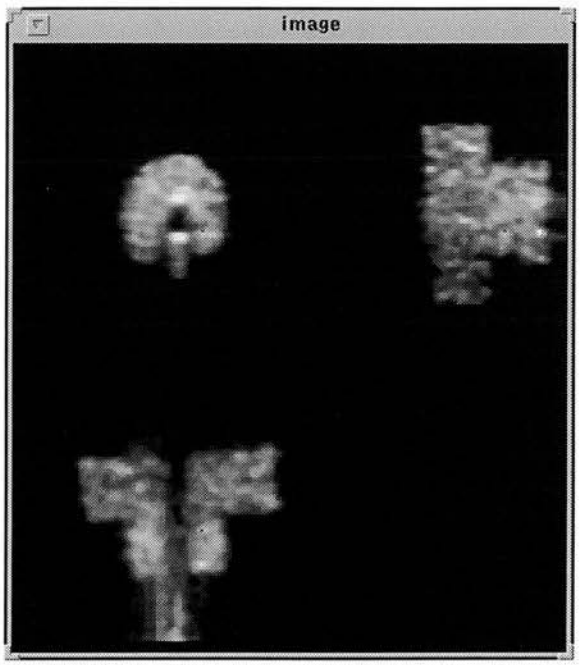


(g)

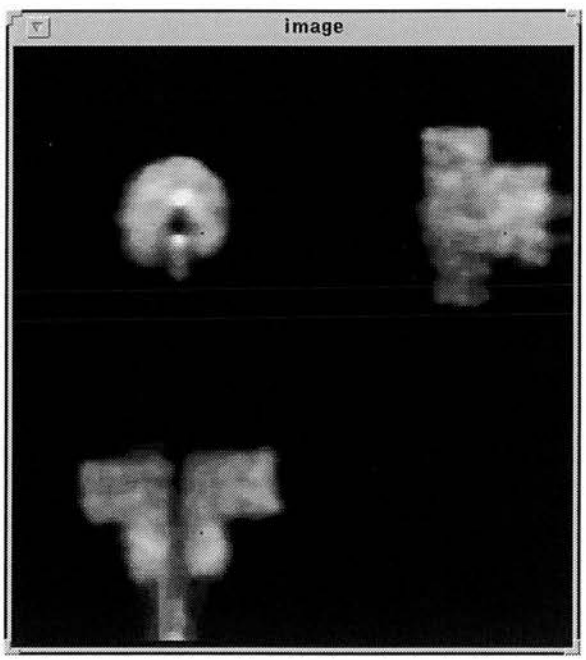


(h)

Figure 9.6 (contd.) : (g) - Pre-processed fast scan 3D image acquired with every 4th line and every frame. (h) - Pre-processed fast scan 3D image acquired with every 4th line and every 2nd frame.

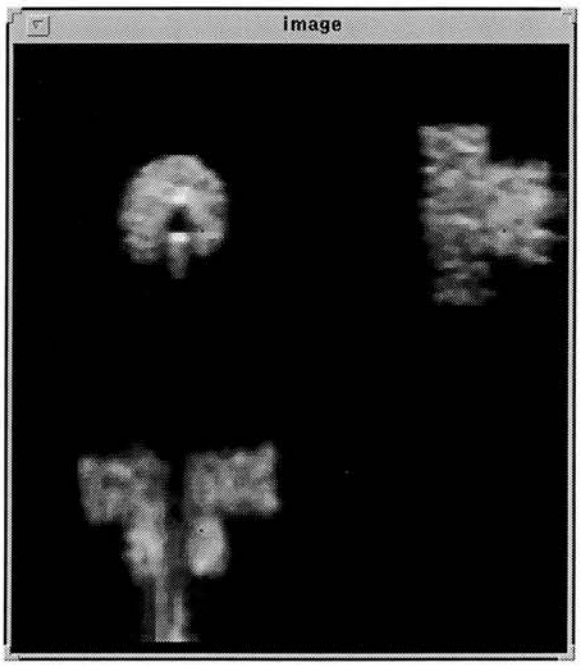


(a)



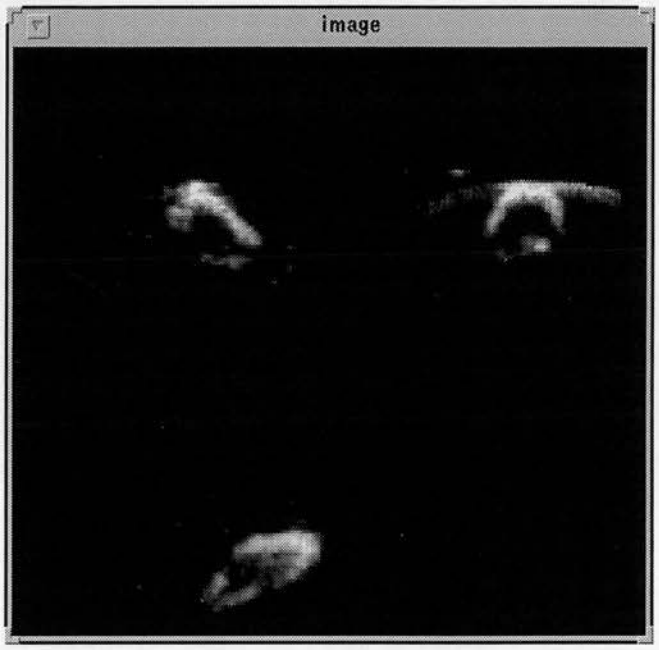
(b)

Figure 9.7 : Orthogonal views of the reconstructed 3D image of tissue mimicking phantom after applying 3D median filter to some of the 3D images in Figure 9.6. (a) - 3D filtered image using 3x3x3 kernel of the 3D image in Figure 9.6(a). (b) - 3D filtered image using 5x5x5 kernel of the 3D image in Figure 9.6(a).

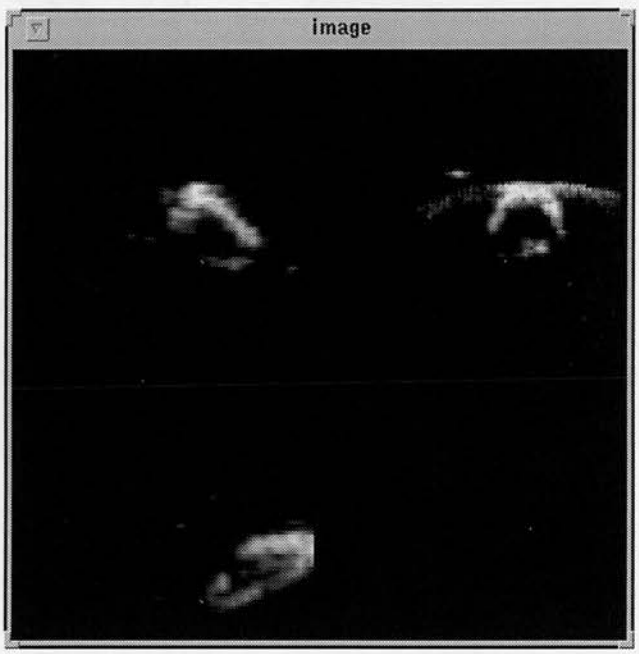


(c)

Figure 9.7 (contd.): (c) - 3D filtered image using 3x3x3 kernel of the 3D image in Figure 9.6(e).



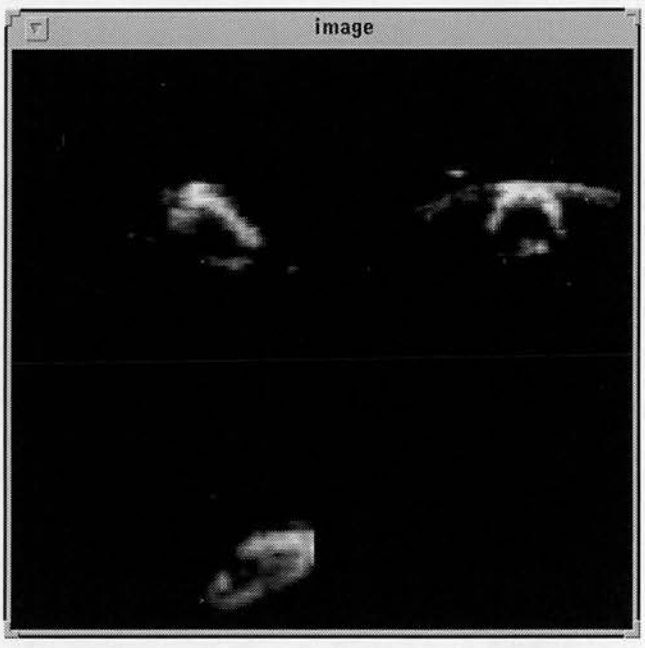
(a)



(b)

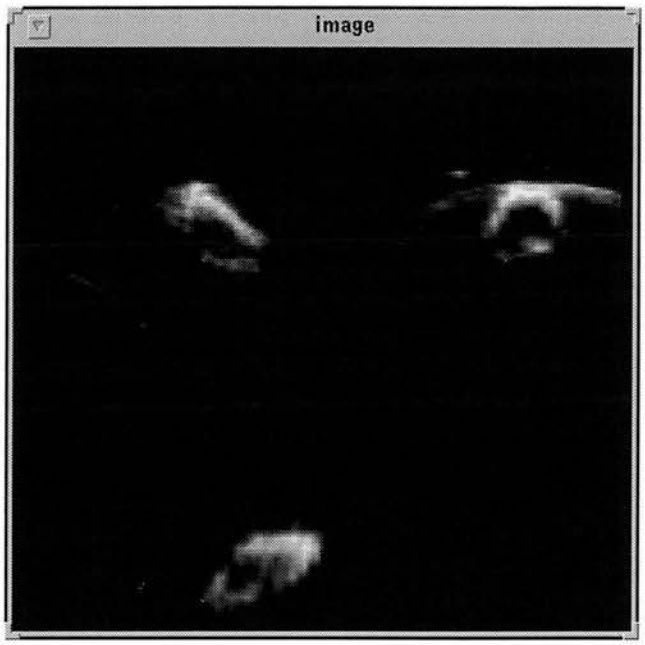
Figure 9.8 : Orthogonal views from the reconstructed 3D image of latex tube phantom. (a) - Original slow scan image acquired with every line and every frame. (b) - Pre-processed fast scan image acquired with every line and every 2nd frame.

(c)



(d)

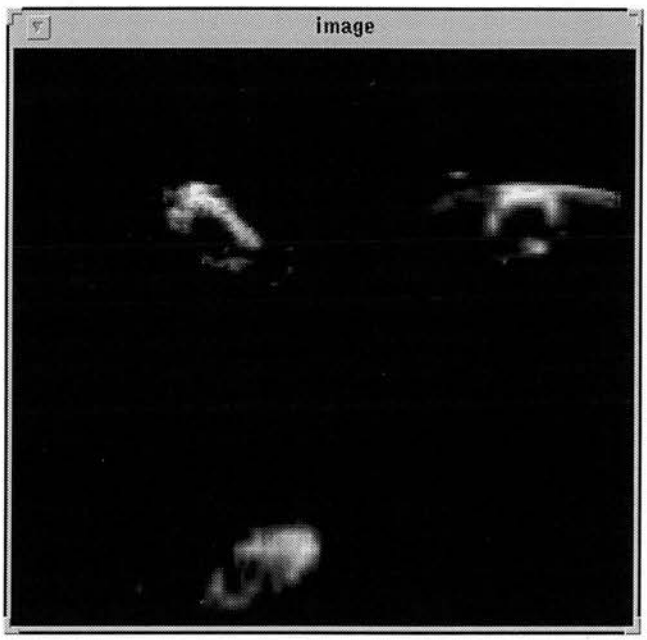
Figure 9.8 (contd.) : (c) - Pre-processed fast scan image acquired with every 2nd line and every frame. (d) - Pre-processed fast scan image acquired with every 2nd line and every 2nd frame.



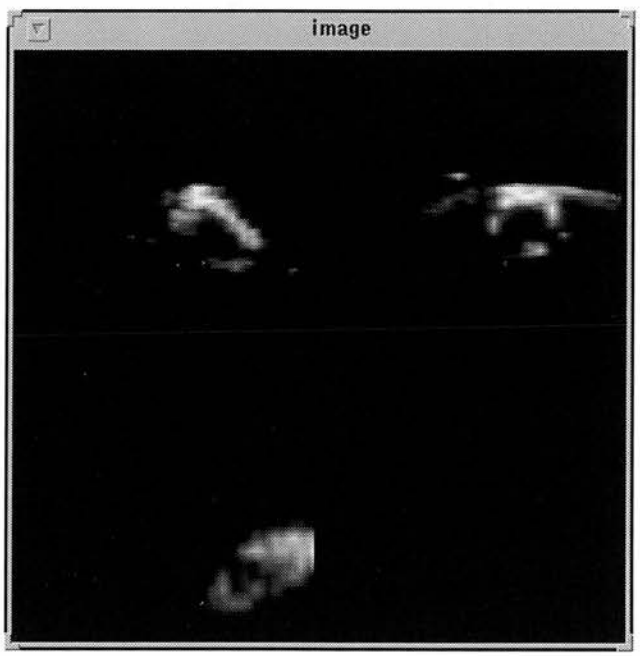
(e)

(f)

Figure 9.8 (contd.) : (e) - Pre-processed fast scan image acquired with every 3rd line and every frame. (f) - Pre-processed fast scan image acquired with every 3rd line and every 2nd frame.



(g)



(h)

Figure 9.8 (contd.) : (g) - Pre-processed fast scan image acquired with every 4th line and every frame. (h) - Pre-processed fast scan image acquired with every 4th line and every 2nd frame.

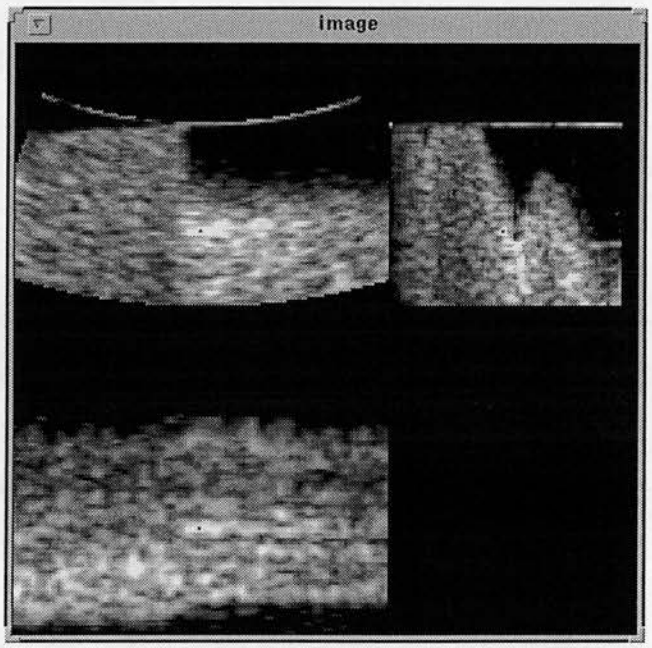
(a)

(b)

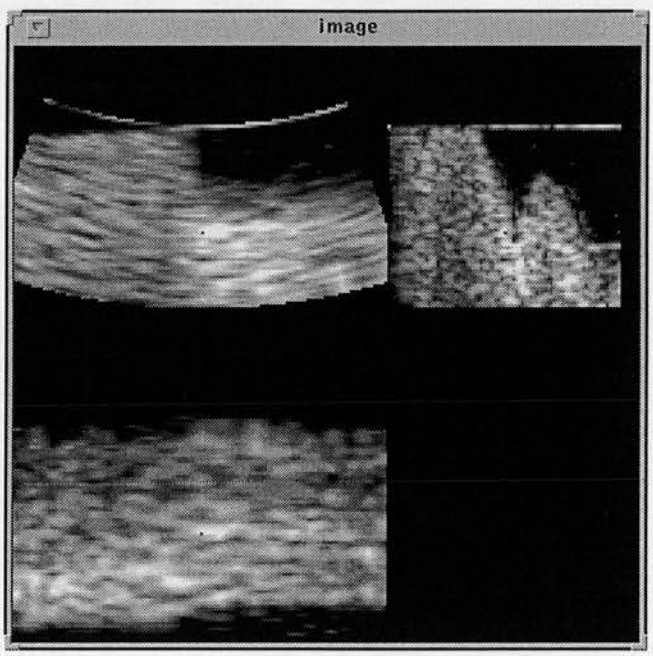
Figure 9.9 : Orthogonal views of the reconstructed 3D image of the latex tube phantom after applying 3D median filter to some of the 3D images in Figure 9.8. (a) - 3D filtered image using 3x3x3 kernel of the 3D image in Figure 9.8(a). (b) - 3D filtered image using 5x5x5 kernel of the 3D image in Figure 9.8(a).

(c)

Figure 9.9 (contd.): (c) - 3D filtered image using 3x3x3 kernel of the 3D image in Figure 9.8(e).

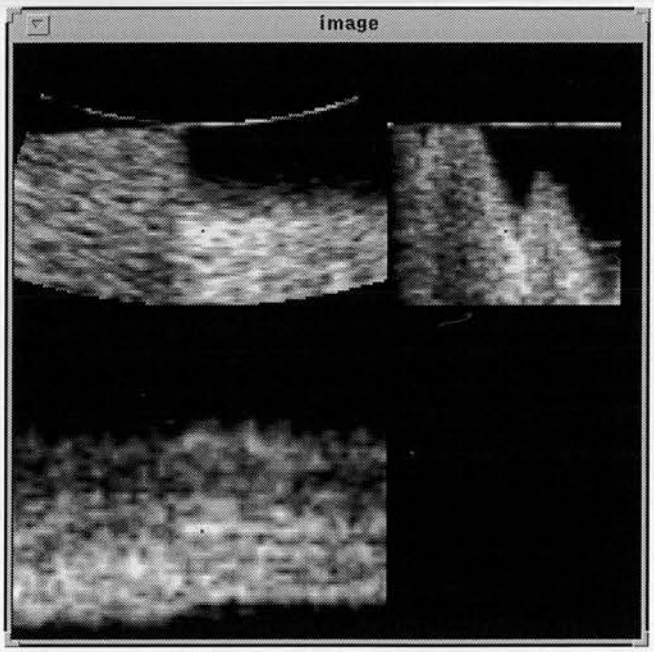


(a)

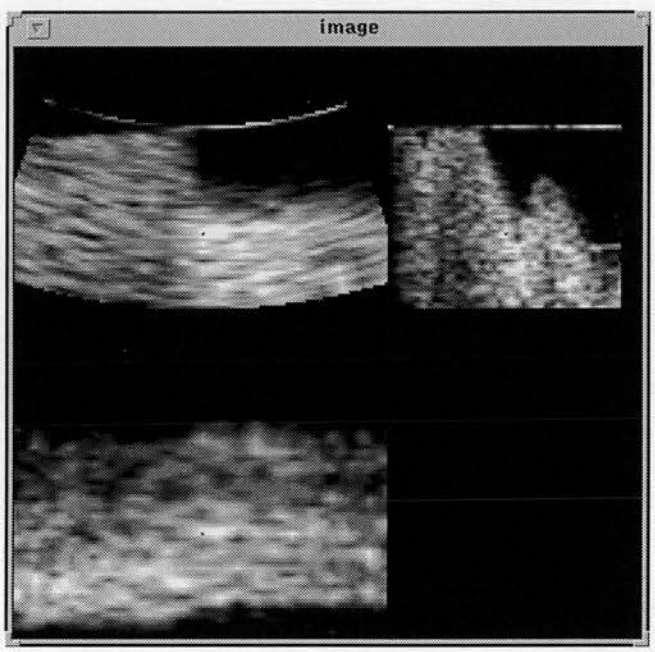


(b)

Figure 9.10 : Orthogonal views of the reconstructed 3D image of standard Cardiff phantom. (a) - Original slow scan 3D image acquired with every line and every frame. (b) - Pre-processed fast scan 3D image acquired with every line and every 2nd frame.

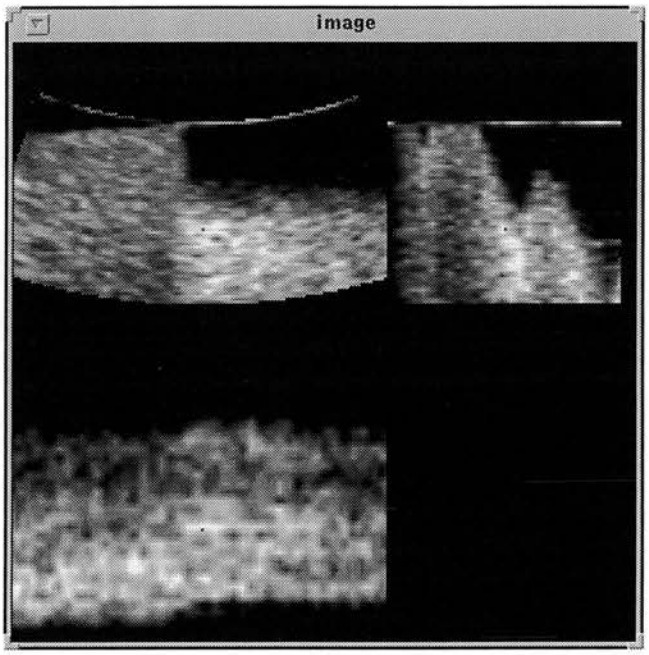


(c)

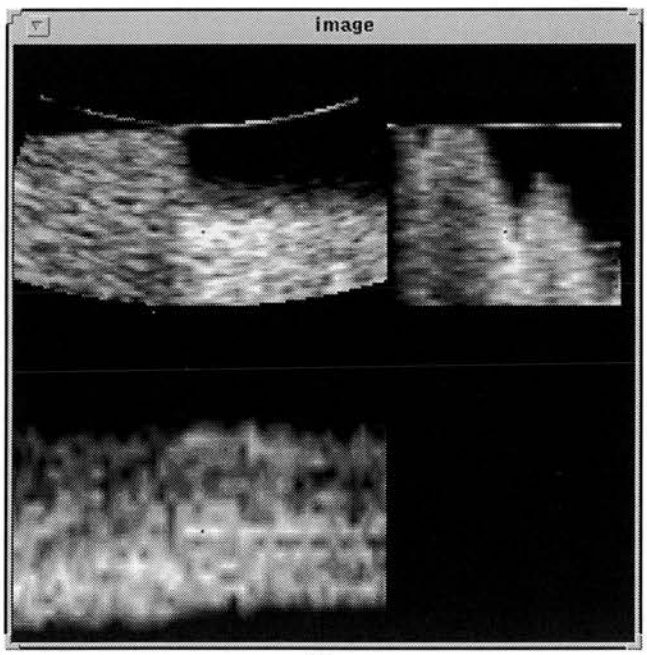


(d)

Figure 9.10 (contd.) : (c) - Preprocessed fast scan 3D image acquired with every 2nd line and every frame. (d) - Preprocessed fast scan 3D image acquired with every 2nd line and every 2nd frame.

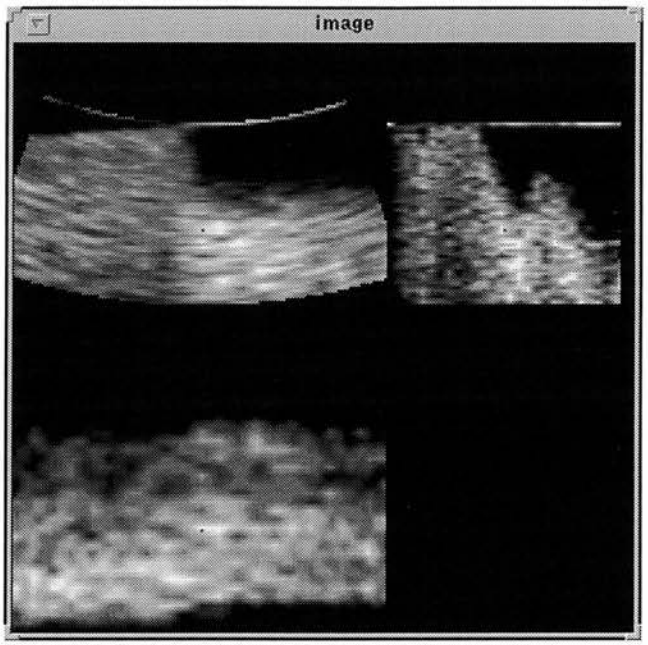


(e)

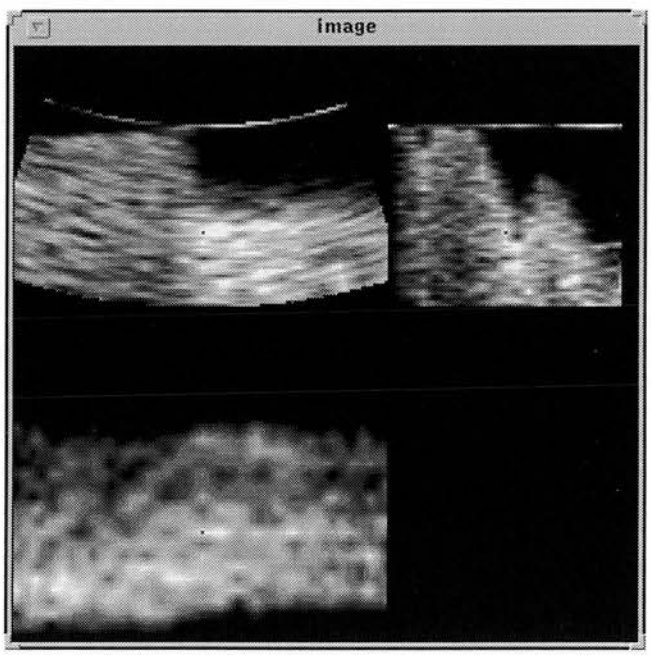


(f)

Figure 9.10 (contd.) : (e) - Preprocessed fast scan 3D image acquired with every 3rd line and every frame. (f) - Preprocessed fast scan 3D image acquired with every 3rd line and every 2nd frame.

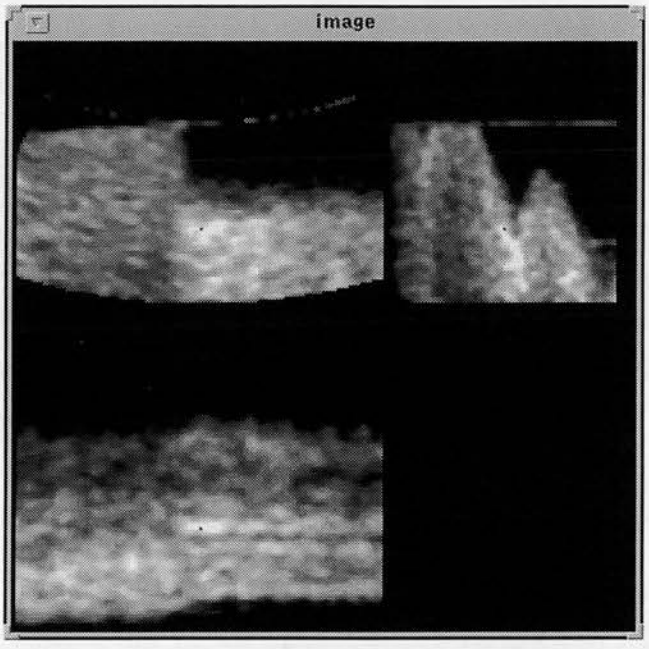


(g)

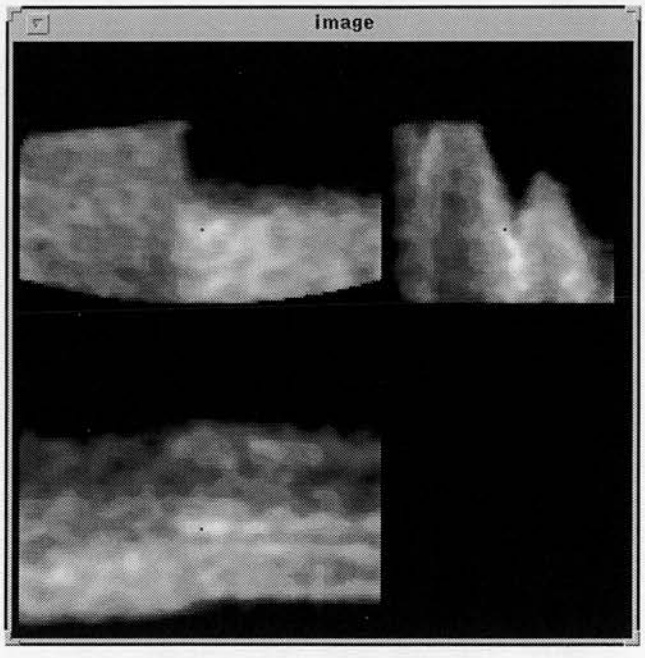


(h)

igure 9.10 (contd.) : (g) - Preprocessed fast scan 3D image acquired with every 4th line and every frame. (h) - Preprocessed fast scan 3D image acquired with every 4th line and every 2nd frame.

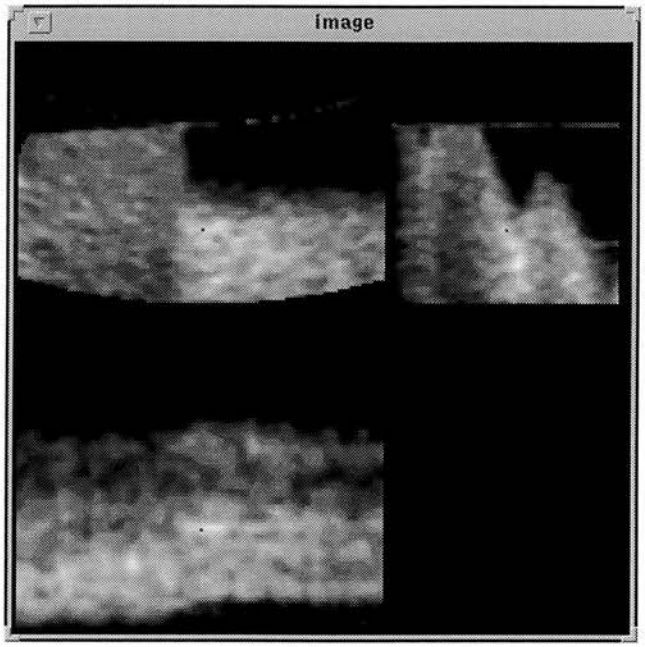


(a)



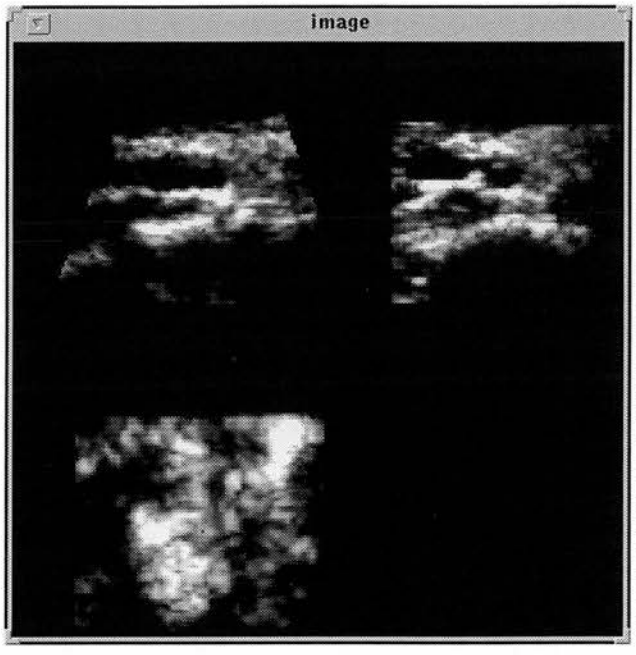
(b)

Figure 9.11 : Orthogonal views of the reconstructed 3D image of standard Cardiff phantom after applying 3D median filter to some of the 3D images in Figure 9.10. (a) - 3D filtered image using 3 x 3 x3 kernel of the 3D image in Figure 9.10(a). (b) - 3D filtered image using 5 x 5 x5 kernel of the 3D image in Figure 9.10(a).

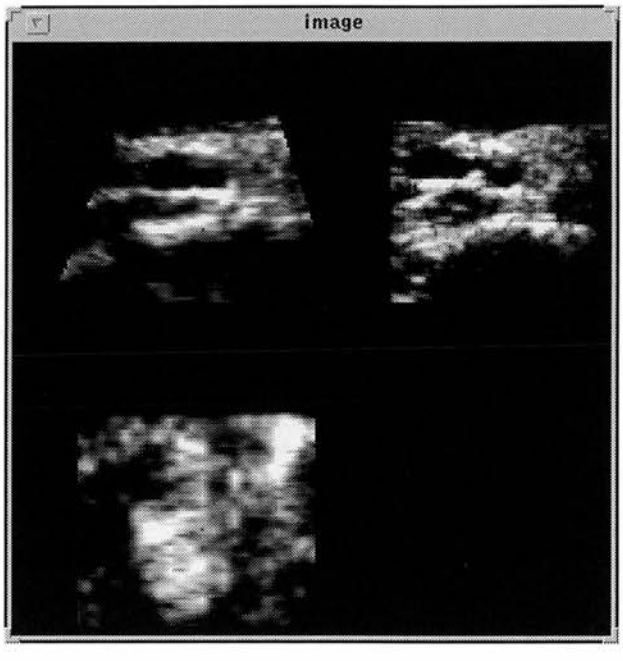


(c)

Figure 9.11 (contd.): (c) - 3D filtered image using 3 x 3 x3 kernel of the 3D image in Figure 9.10(e).

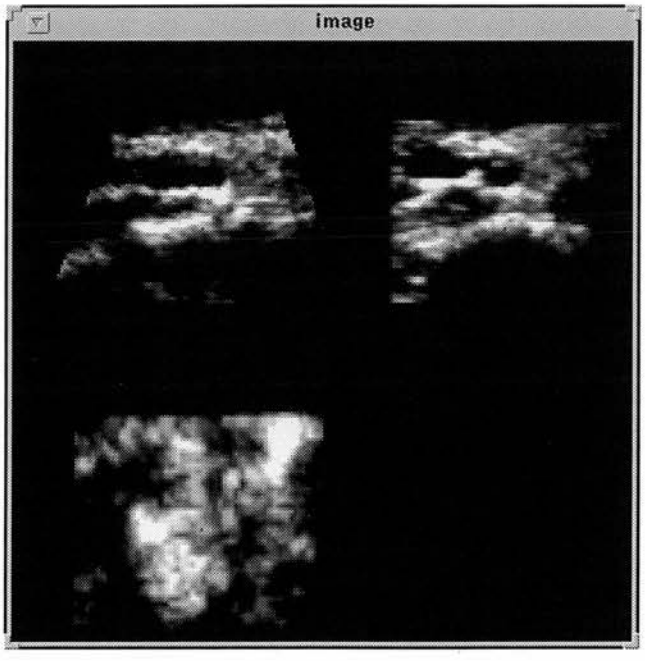


(a)

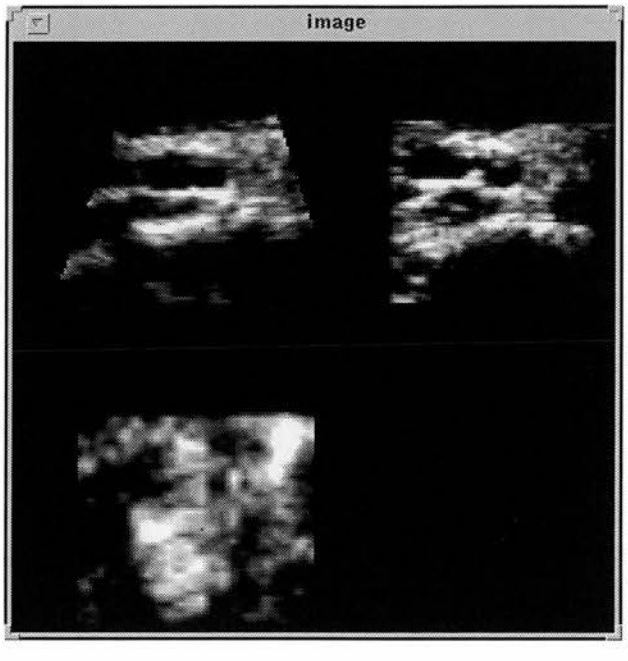


(b)

Figure 9.12 : Orthogonal views from the reconstructed 3D image of carotid artery. (a) - Original slow scan 3D image acquired with every line and every frame. (b) - Pre-processed fast scan image acquired with every line and every 2nd frame.

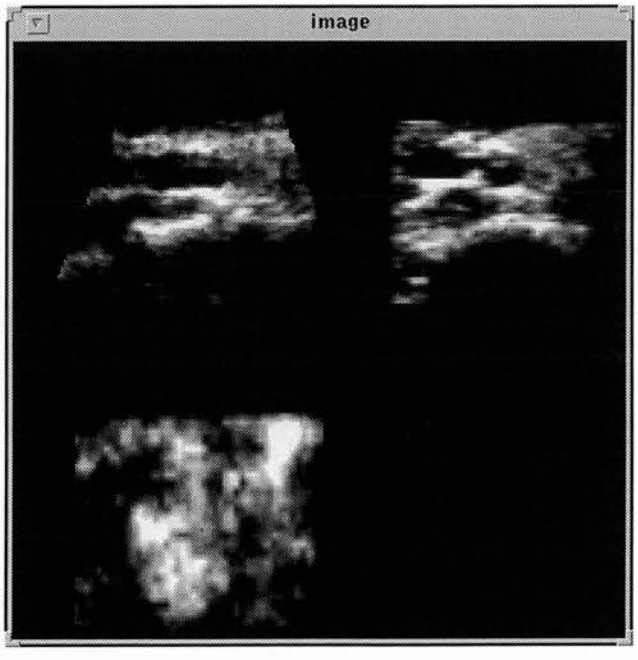


(c)

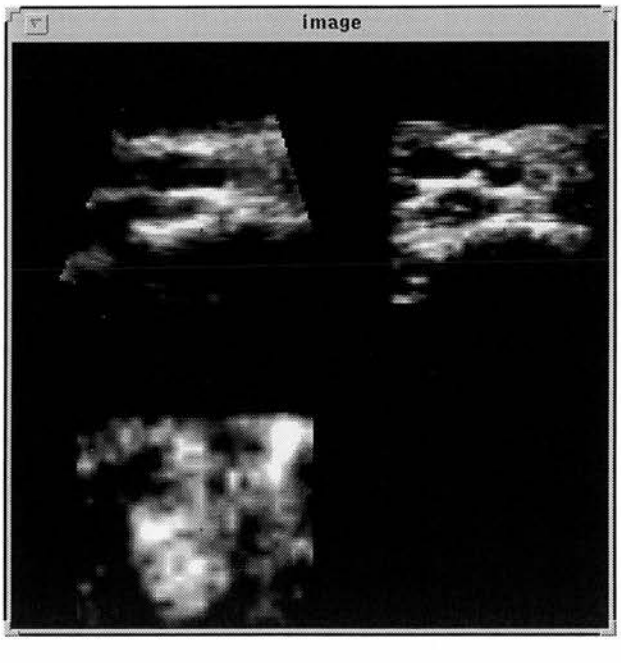


(d)

Figure 9.12 (contd.) : (c) - Pre-processed fast scan image acquired with every 2nd line and every frame. (d) - Pre-processed fast scan image acquired with every 2nd line and every 2nd frame.

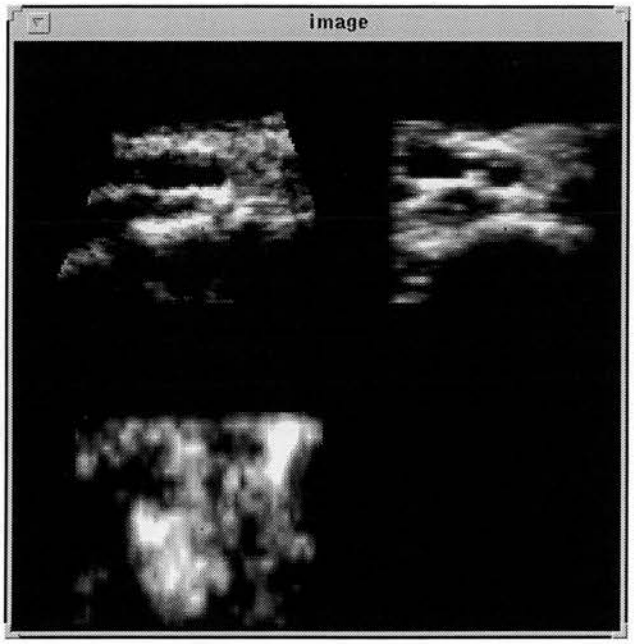


(e)

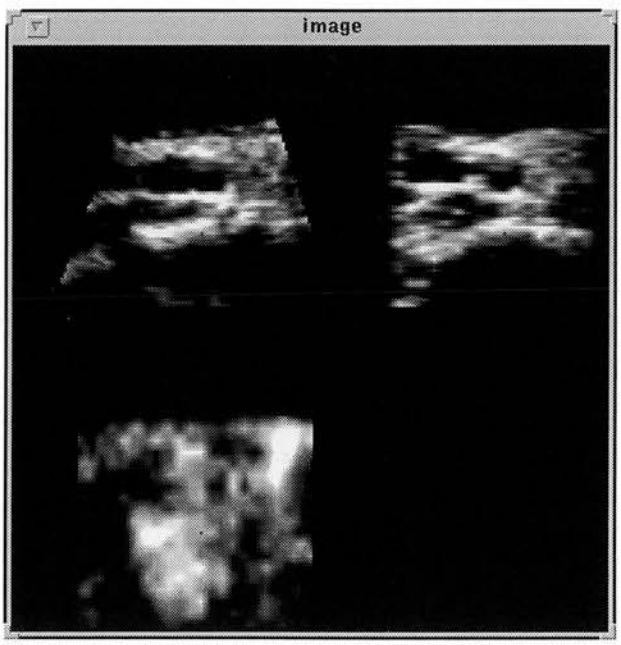


(f)

Figure 9.12 (contd.) : (e) - Pre-processed fast scan image acquired with every 3rd line and every frame. (f) - Pre-processed fast scan image acquired with every 3rd line and every 2nd frame.

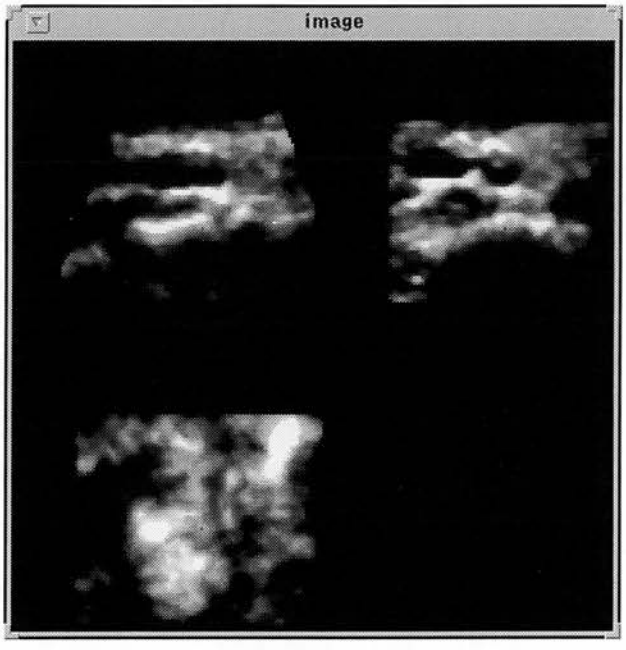


(g)

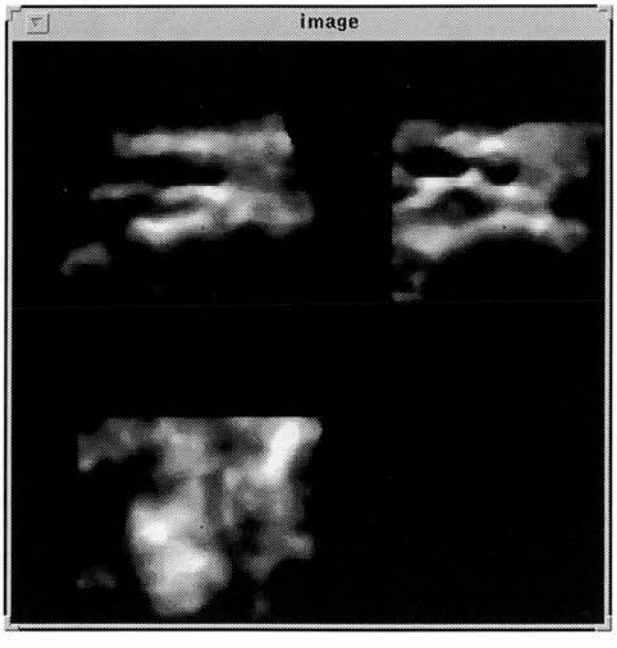


(h)

Figure 9.12 (contd.) : (g) - Pre-processed fast scan image acquired with every 4th line and every frame. (h) - Pre-processed fast scan image acquired with every 4th line and every 2nd frame.

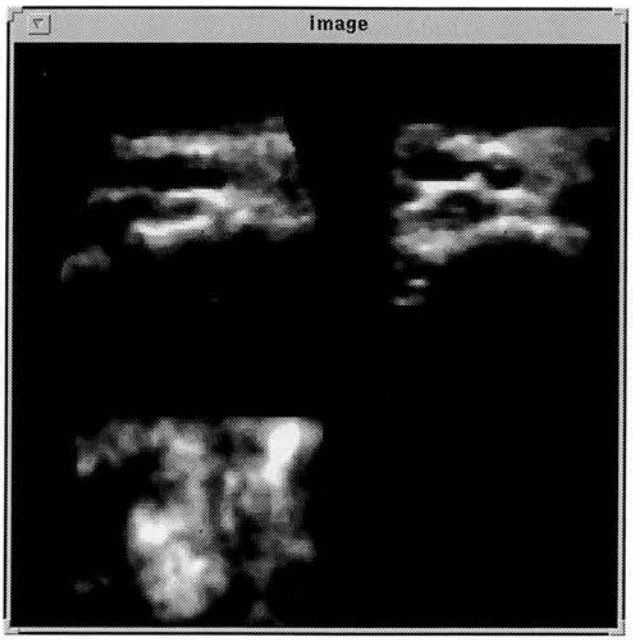


(a)



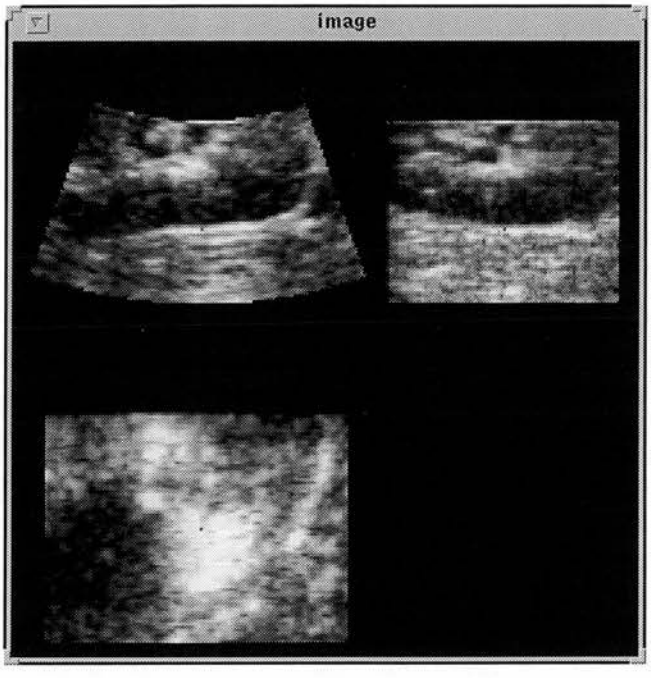
(b)

Figure 9.13 : Orthogonal views of the reconstructed 3D image of carotid artery after applying 3D median filter to some of the 3D images in Figure 9.12. (a) - 3D filtered image using 3x 3x3 kernel of the 3D image in Figure 9.12(a). (b) - 3D filtered image using 5x5x5 kernel of the 3D image in Figure 9.12(a).

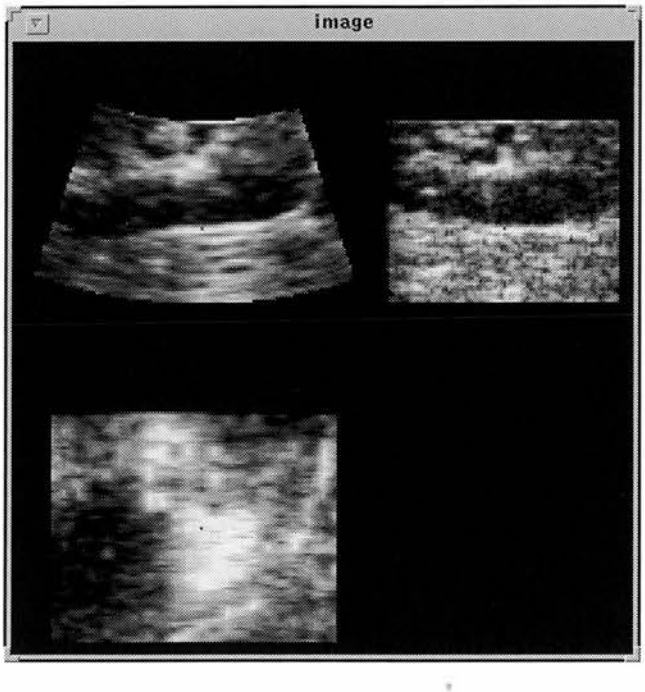


(c)

Figure 9.13 (contd.): (c) - 3D filtered image using 3x3x3 kernel of the 3D image in Figure 9.12(e).

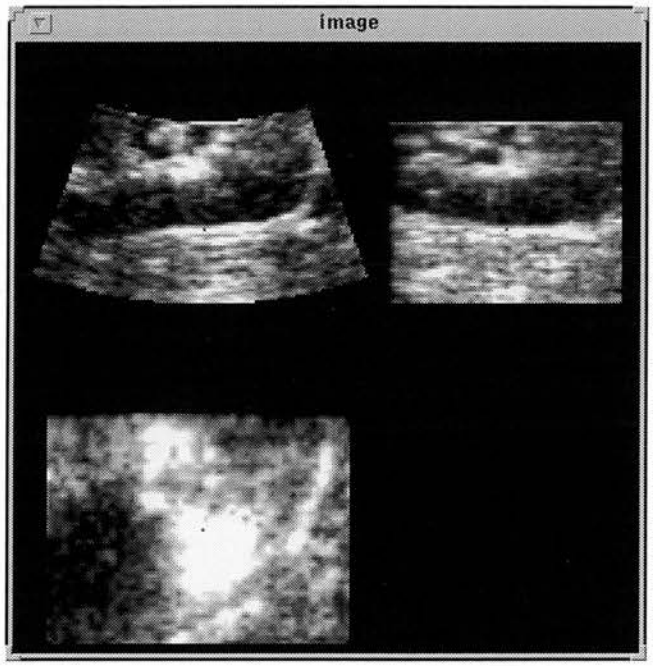


(a)

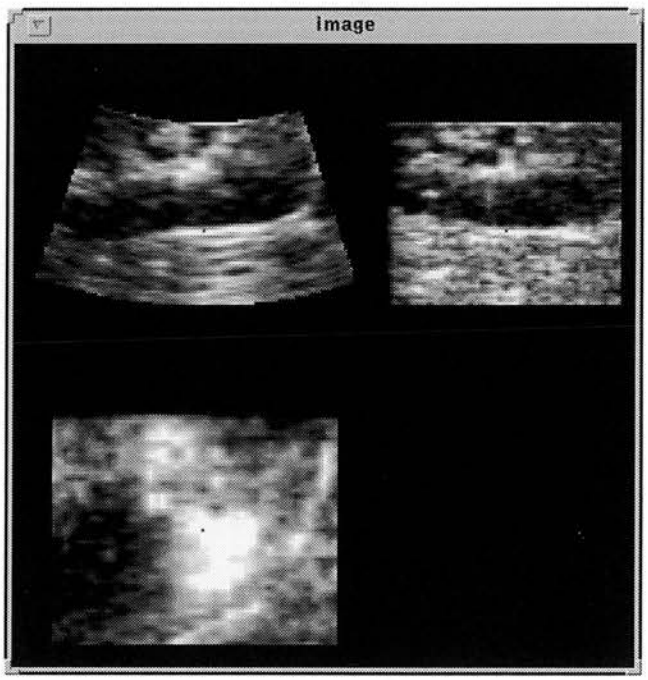


(b)

Figure 9.14 : Orthogonal views from the reconstructed 3D image of kidney. (a) - Original slow scan 3D image acquired with every line and every frame. (b) - Pre-processed fast scan 3D image acquired with every line and every 2nd frame.

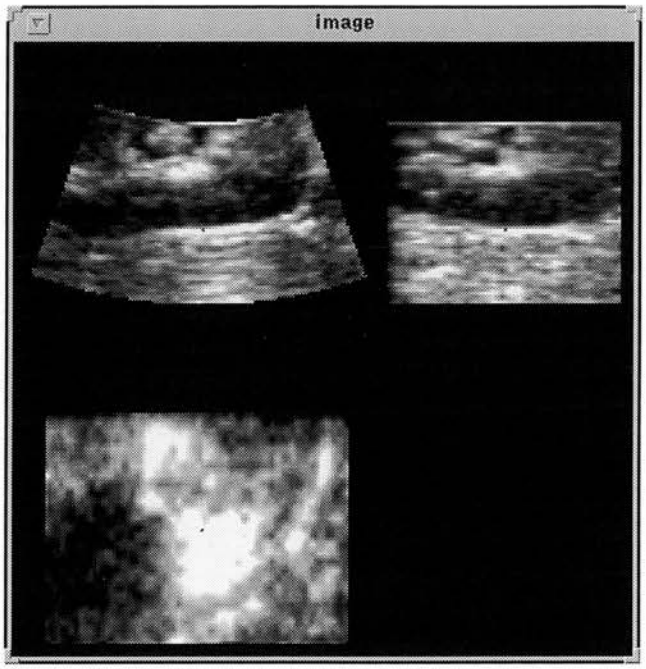


(c)

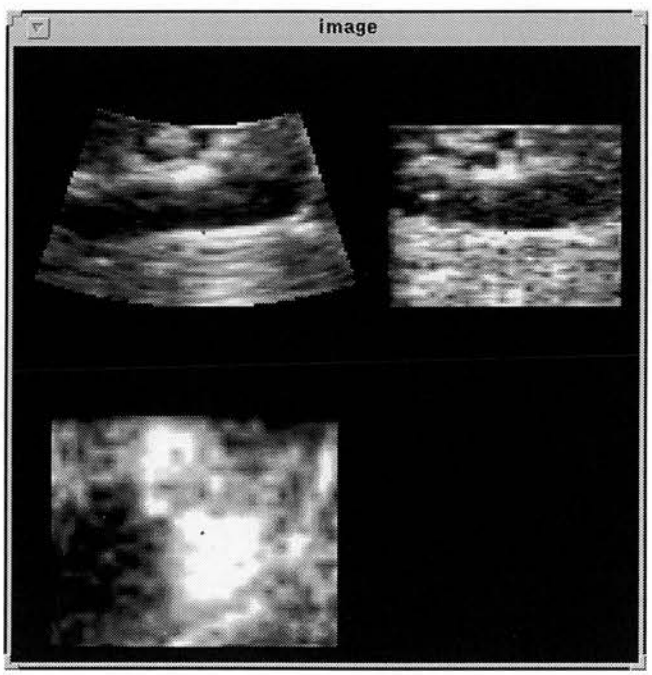


(d)

Figure 9.14 (contd.) : (c) - Pre-processed fast scan 3D image acquired with every 2nd line and every frame. (d) - Pre-processed fast scan 3D image acquired with every 2nd line and every 2nd frame.

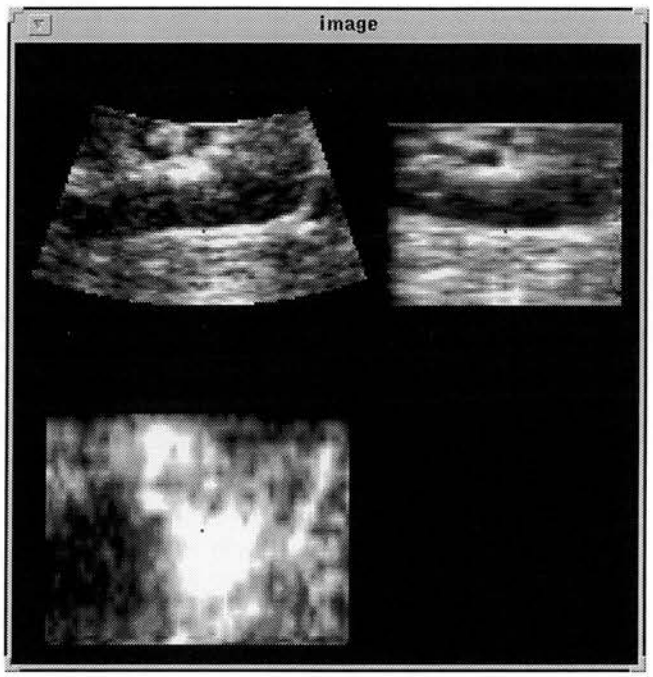


(e)

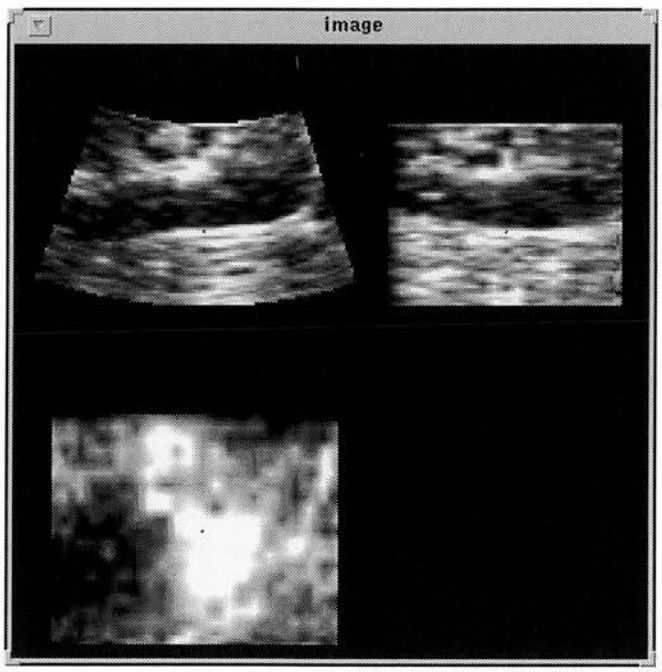


(f)

Figure 9.14 (contd.) : (e) - Pre-processed fast scan 3D image acquired with every 3rd line and every frame. (f) - Pre-processed fast scan 3D image acquired with every 3rd line and every 2nd frame.

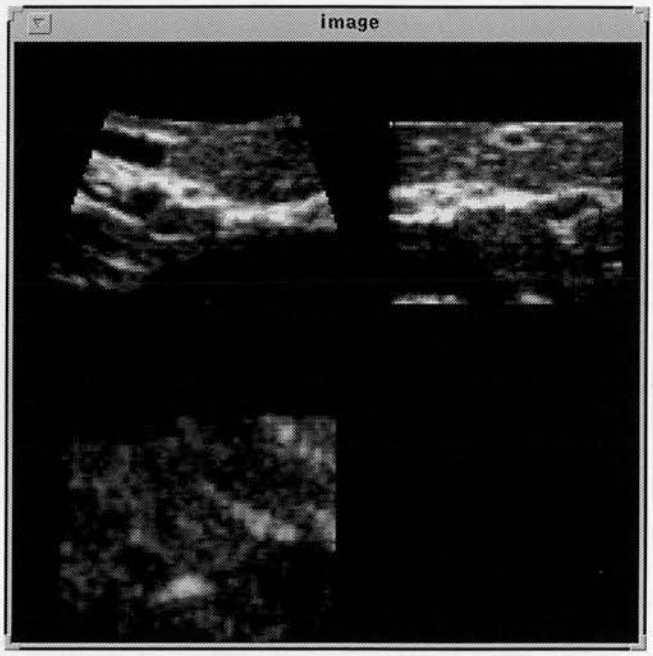


(g)

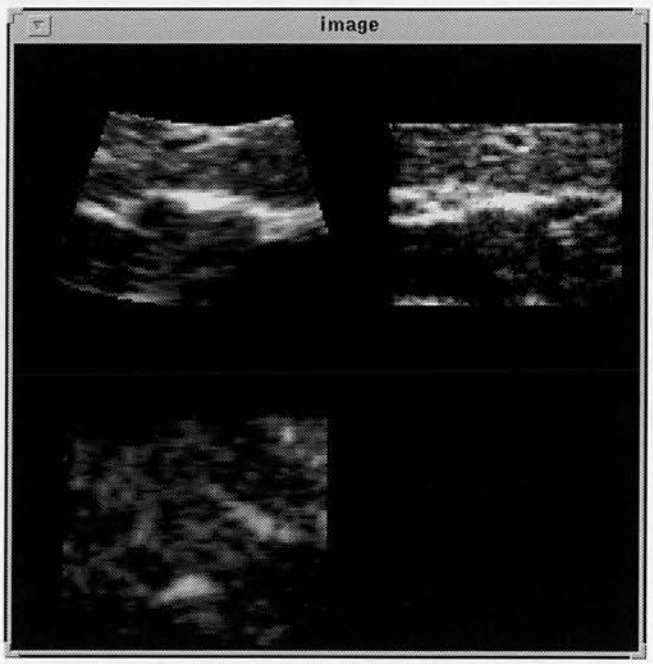


(h)

Figure 9.14 (contd.) : (g) - Pre-processed fast scan 3D image acquired with every 4th line and every frame. (h) - Pre-processed fast scan 3D image acquired with every 4th line and every 2nd frame.

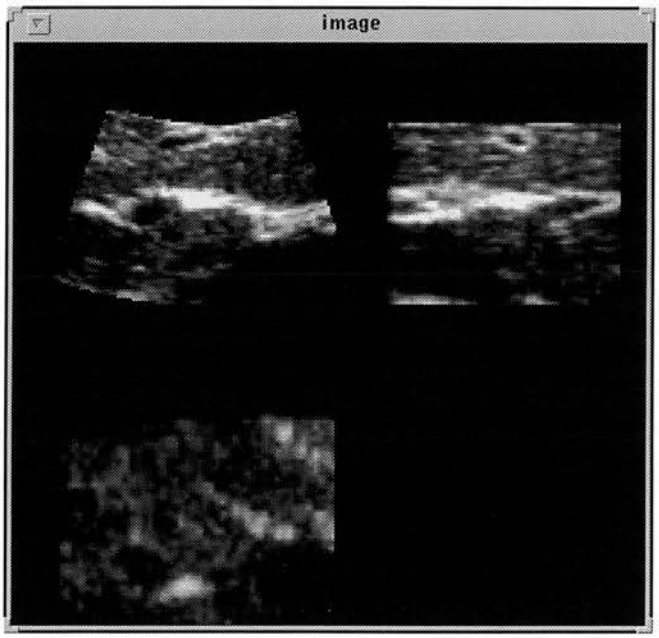


(a)



(b)

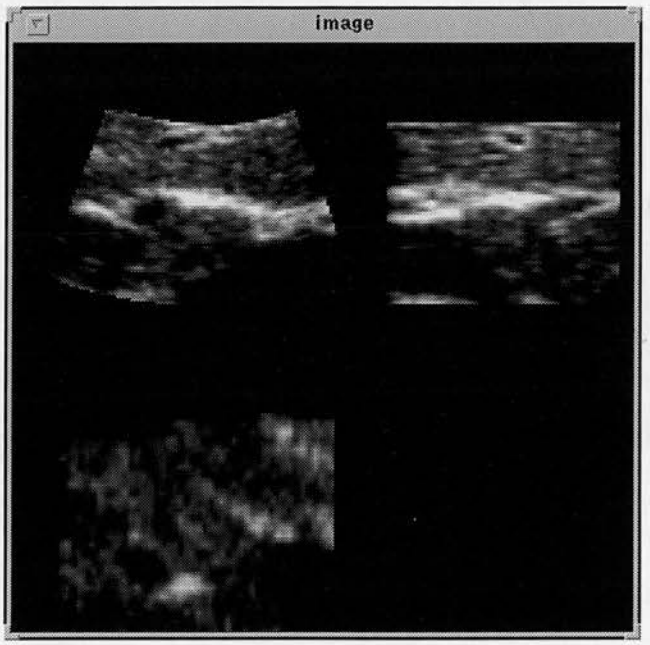
Figure 9.15 : Orthogonal views from the reconstructed 3D image of liver artery. (a) - Original slow scan 3D image acquired with every line and every frame. (b) - Pre-processed fast scan 3D image acquired with every line and every 2nd frame.



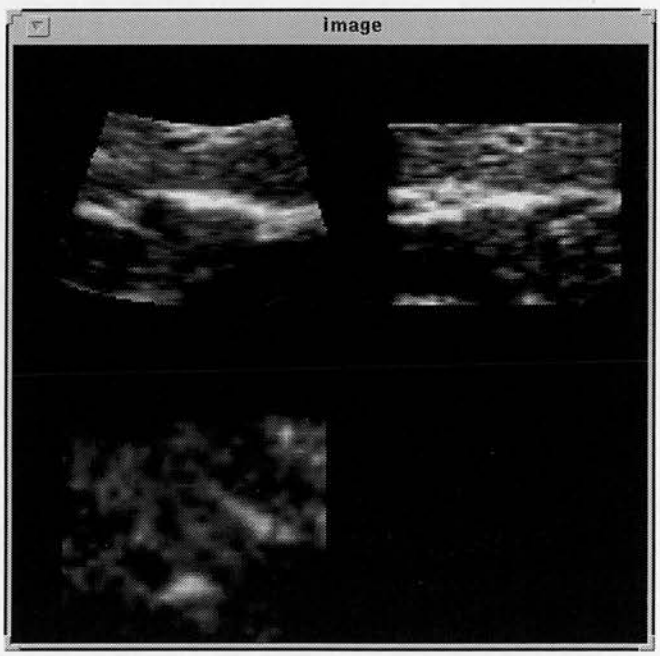
(c)

(d)

Figure 9.15 (contd.) : (c) - Pre-processed fast scan 3D image acquired with every 2nd line and every frame. (d) - Pre-processed fast scan 3D image acquired with every 2nd line and every 2nd frame.



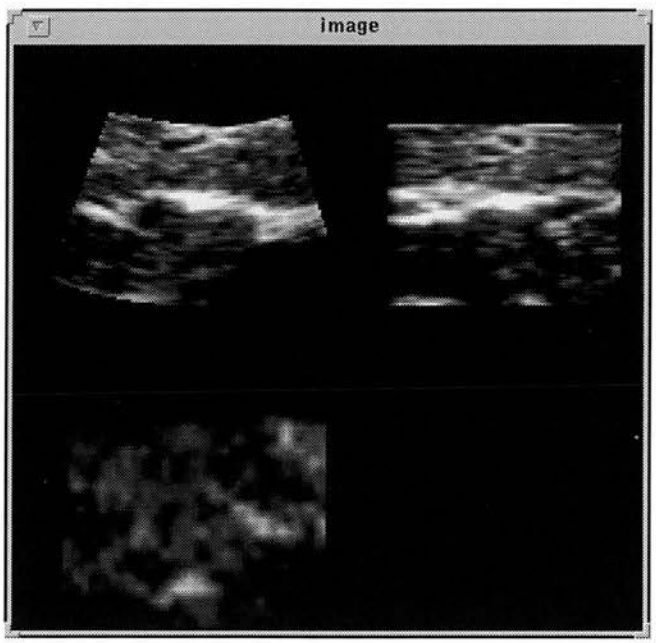
(e)



(f)

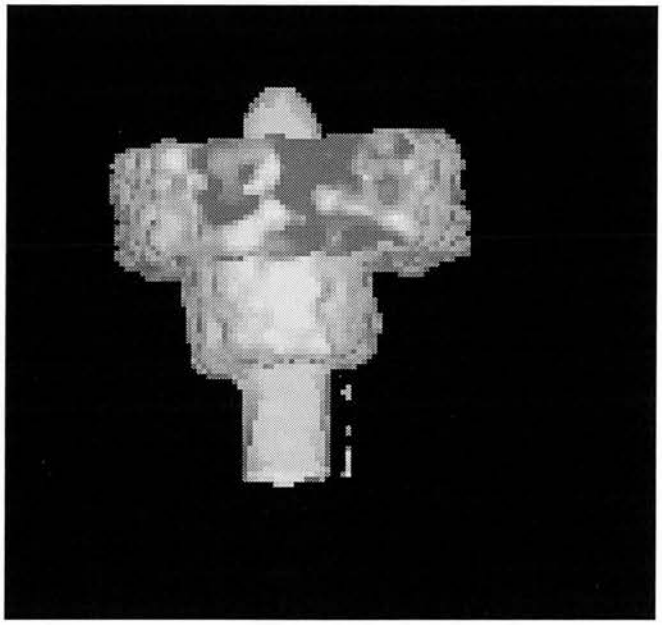
Figure 9.15 (contd.) : (e) - Pre-processed fast scan 3D image acquired with every 3rd line and every frame. (f) - Pre-processed fast scan 3D image acquired with every 3rd line and every 2nd frame.

(g)

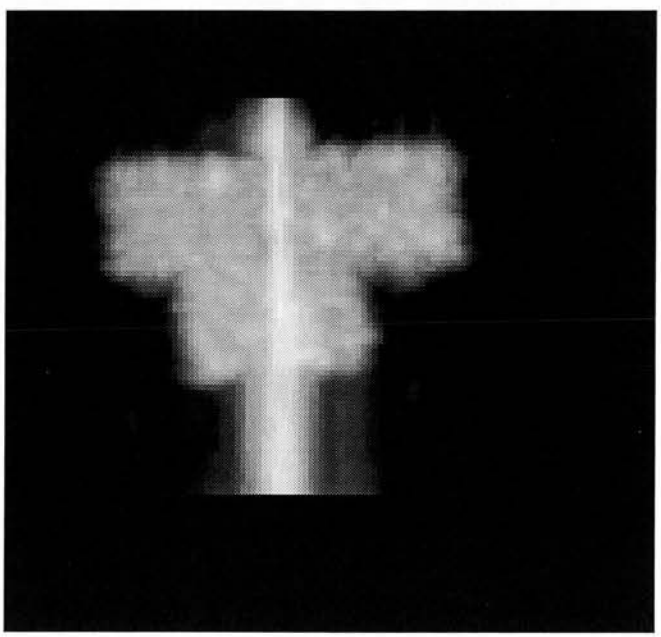


(h)

Figure 9.15 (contd.) : (g) - Pre-processed fast scan 3D image acquired with every 4th line and every frame. (h) - Pre-processed fast scan 3D image acquired with every 4th line and every 2nd frame.

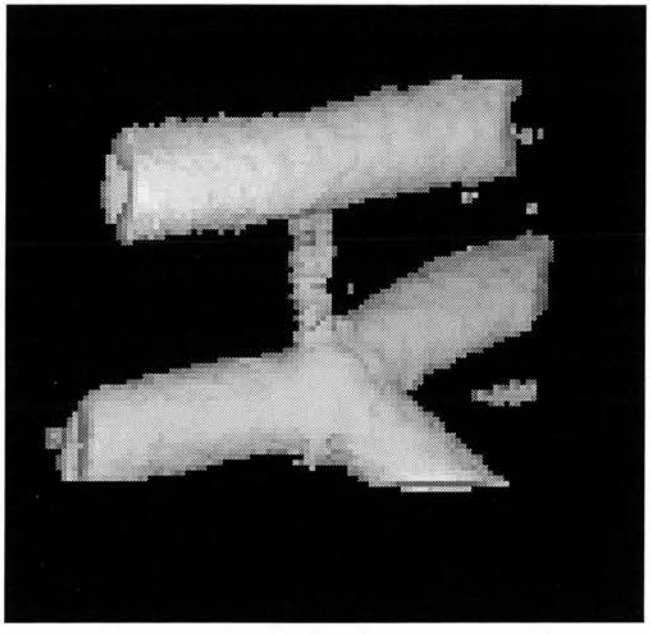


(a)

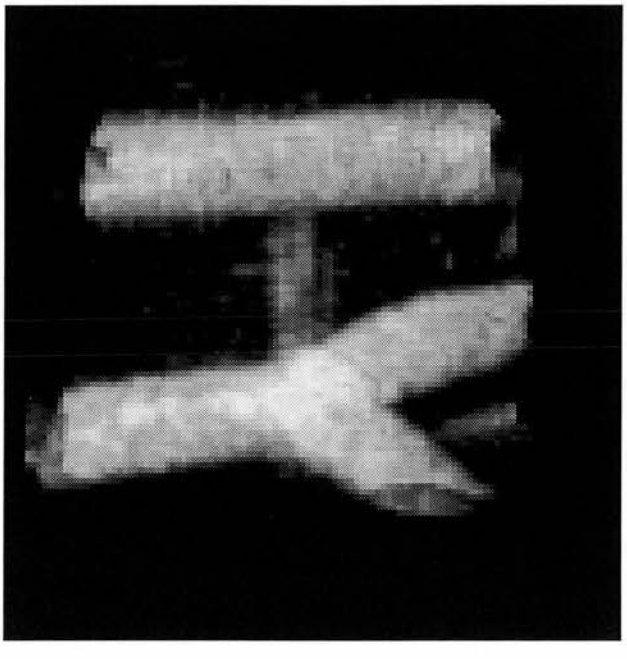


(b)

Figure 9.16 : 3D image display techniques of the filtered 3D image with 3x3 x3 kernel of the tissue mimicking phantom. (a) - Using VR technique. (b) - Using MIP technique.

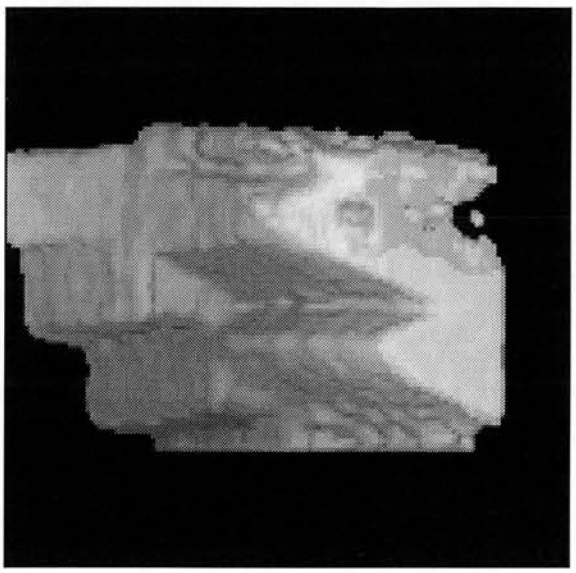


(a)

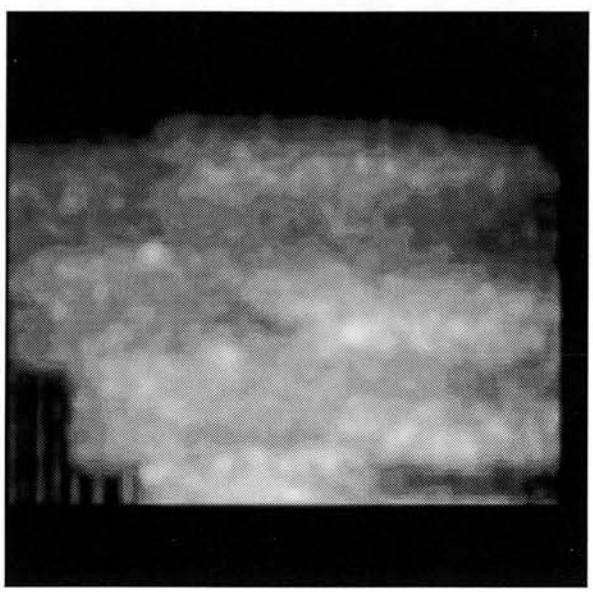


(b)

Figure 9.17 : 3D image display techniques of the filtered 3D image with 3x3x3 kernel of the latex tube phantom. (a) - Using VR technique. (b) - Using MIP technique.

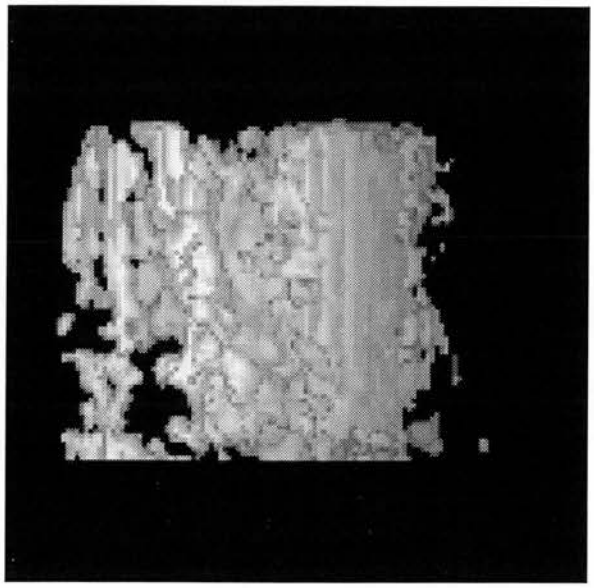


(a)

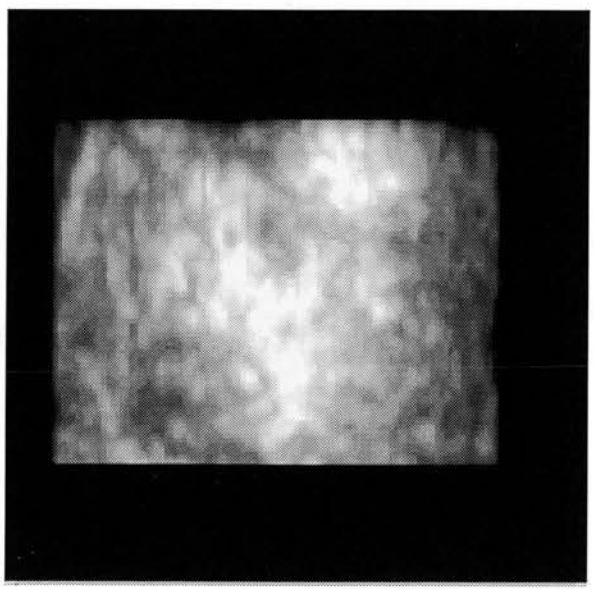


(b)

Figure 9.18 : 3D image display techniques of the filtered 3D image with 3x3 x3 kernel of the standard Cardiff phantom. (a) - Using VR technique. (b) - Using MIP technique.

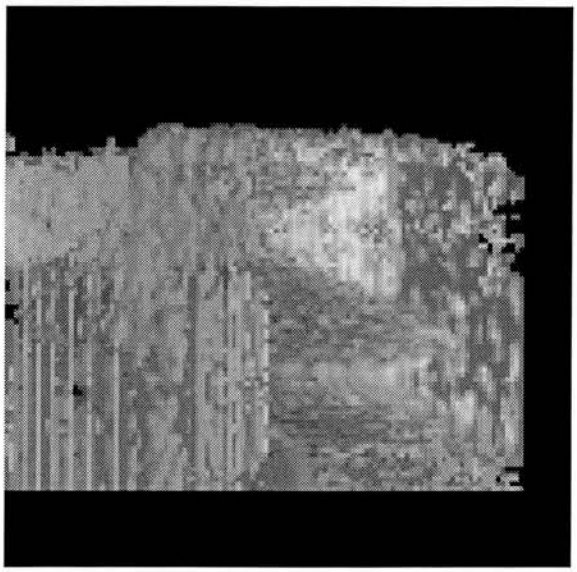


(a)

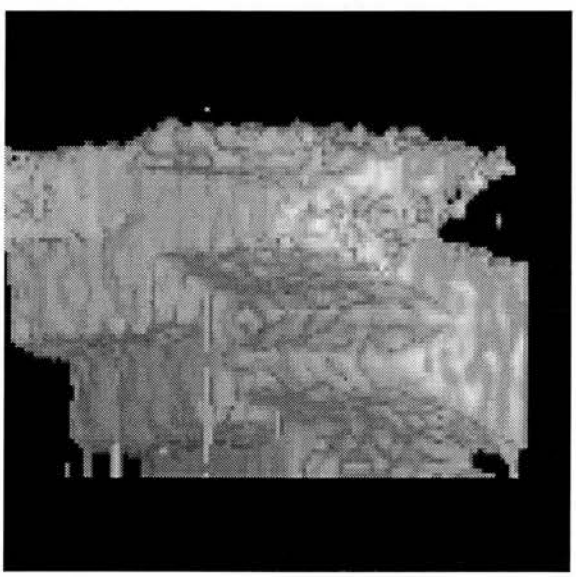


(b)

Figure 9.19 : 3D image display techniques of the filtered 3D image with 3x3x3 kernel of the kidney. (a) - Using VR technique. (b) - Using MIP technique.

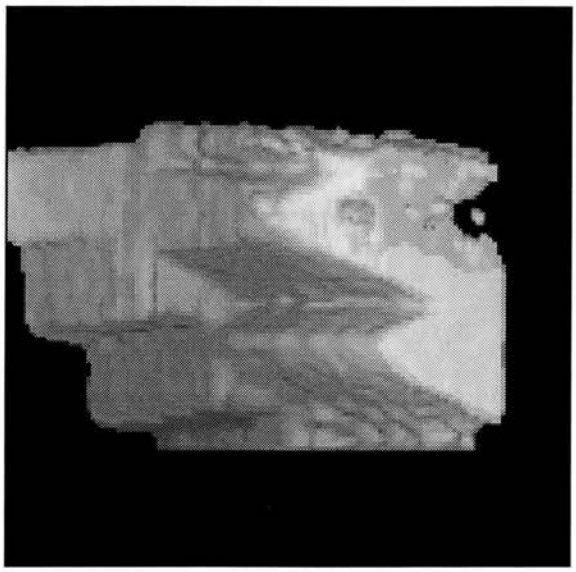


(a)

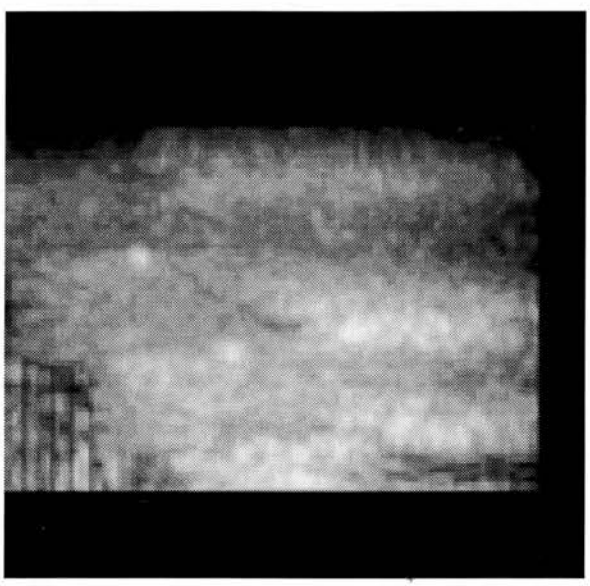


(b)

Figure 9.20 : The effect of 3D image processing on the quality of the different types of the 3D display techniques of the standard Cardiff phantom. (a) - Volume rendered 3D unfiltered image . (b) - Volume rendered 3D filtered image using 3x3x3 kernel.

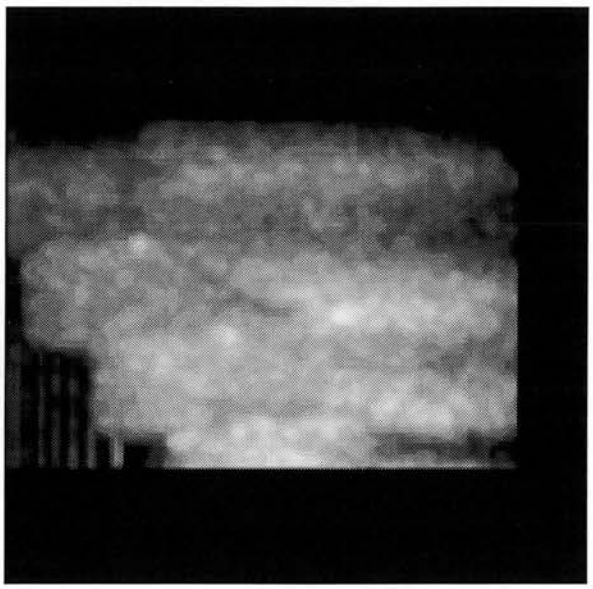


(c)

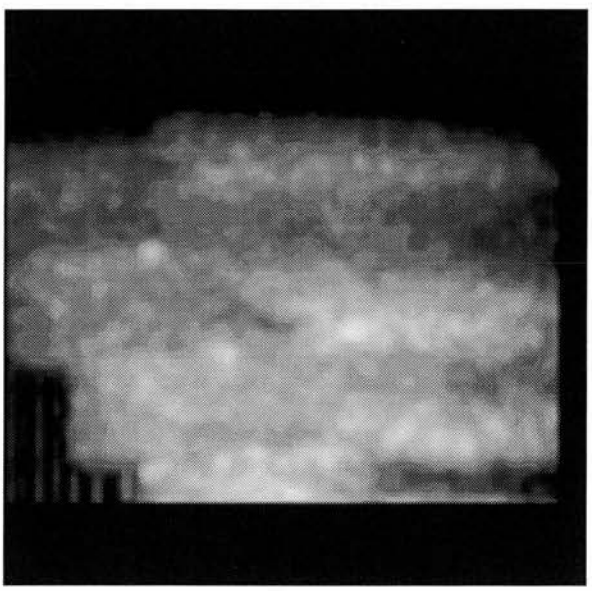


(d)

Figure 9.20 (contd.) : (c) - Volume rendered 3D filtered image using 5x5x5 kernel. (d) - Maximum intensity 3D unfiltered image.

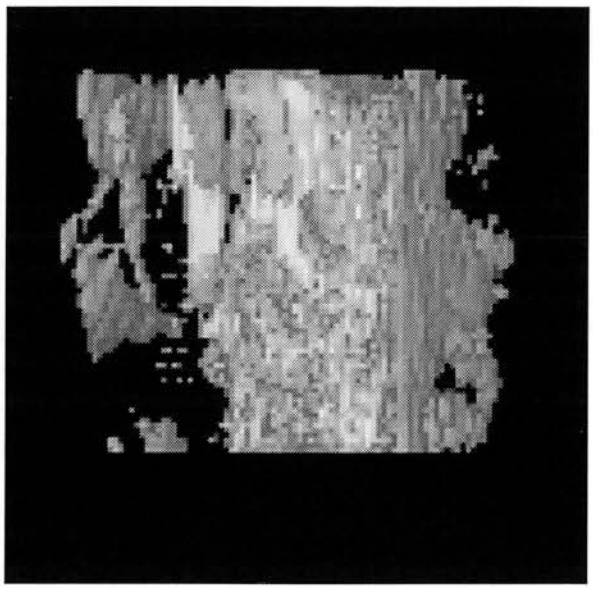


(e)

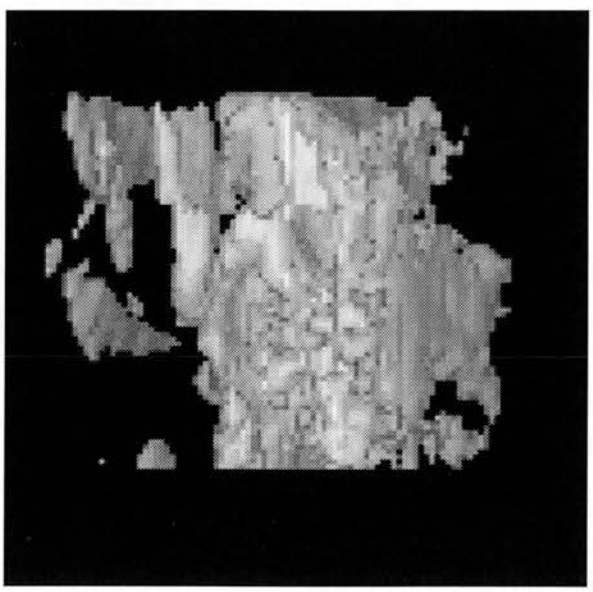


(f)

Figure 9.20 (contd.): (e) - Maximum intensity 3D filtered image using 3x3x3 kernel. (f) - Maximum intensity 3D filtered image using 5x5x5 kernel.

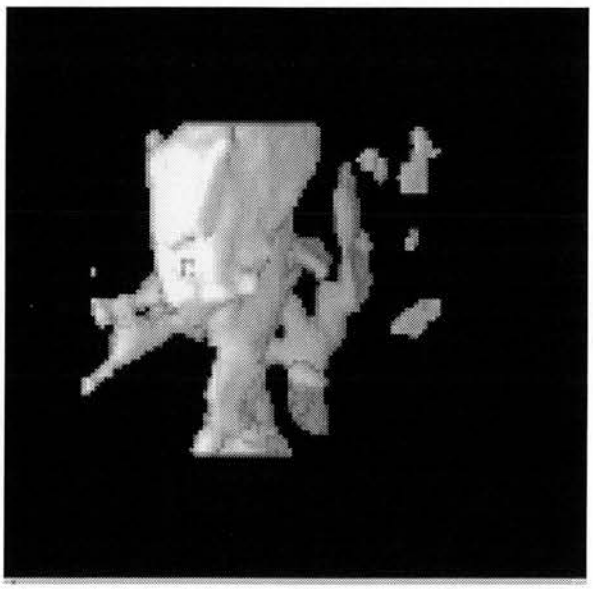


(a)

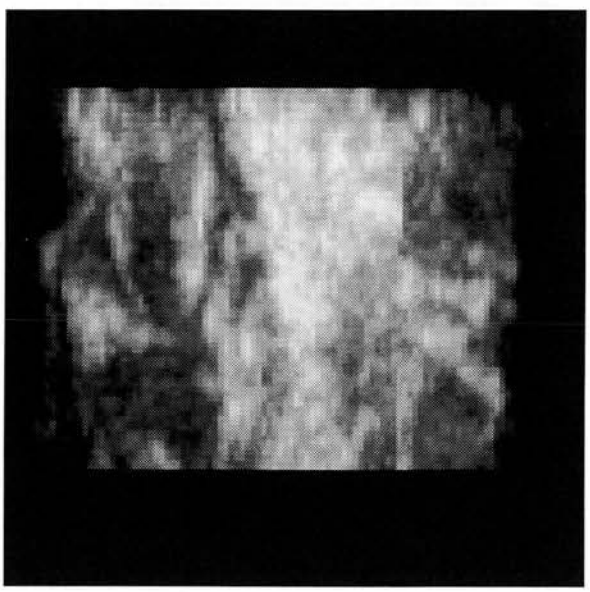


(b)

Figure 9.21 : The effect of 3D image processing on the quality of the different types of the 3D display techniques of the liver vessels. (a) - Volume rendered 3D unfiltered image . (b) - Volume rendered 3D filtered image using 3x3x3 kernel.

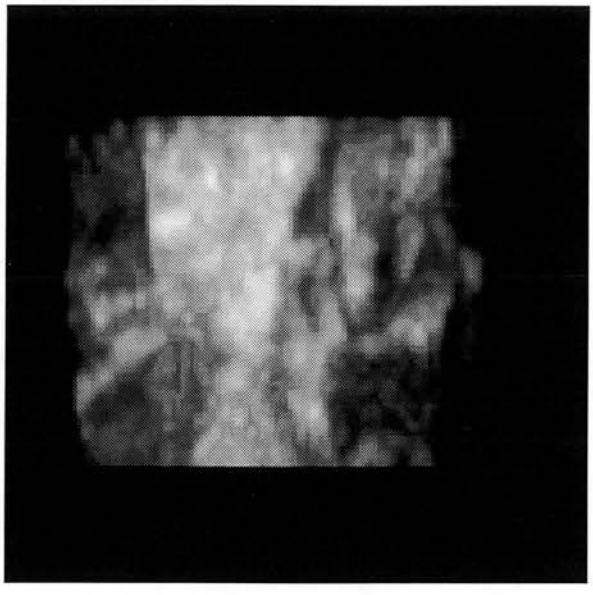


(c)



(d)

Figure 9.21 (contd.) : (c) - Volume rendered 3D filtered image using 5x5x5 kernel. (d) - Maximum intensity 3D unfiltered image.



(e)

(f)

Figure 9.21 (contd.): (e) - Maximum intensity 3D filtered image using 3x3x3 kernel. (f) - Maximum intensity 3D filtered image using 5x5x5 kernel.

CHAPTER 10

FINAL CONCLUSIONS AND FURTHER DEVELOPMENT OF REAL-TIME 3D ULTRASOUND IMAGING

CHAPTER 10

FINAL CONCLUSIONS AND FURTHER DEVELOPMENT OF REAL-TIME 3D ULTRASOUND IMAGING

10.1. Conclusions

In this thesis, the development of two systems, image processing and experimental work have been undertaken to study the feasibility of real-time 3D ultrasonic imaging to see if ways can be found to overcome the fundamental problem of sparsity of echo line data. The echo line density is sparse when a volume is scanned in real-time because of the upper limit of rate of generation of scan lines which is around 10 kHz.

A C-scan ultrasound system which is simpler but similar to a 3D ultrasound system has been assessed. A microcomputer based C-scan test-rig system including hardware and software has been constructed to investigate the effectiveness of real-time image processing in compensating for the fundamental problem of sparsity of echo data. In this study a comparison study of the 2D slow C-scan images and the processed 2D simulated real-time C-scan images using different image processing techniques has been undertaken. The results of this study have shown that the image processing can overcome the problem of the sparsity of echo data in C-scanning. Hence these results encouraged the investigation of real-time 3D ultrasonic scanning in a system which would have a low density of echo data.

A preliminary result of 3D reconstruction from a 3D data set acquired in the C-scan system is presented to show the usefulness of 3D ultrasound images compared to 2D ultrasound images.

A comprehensive review of 3D ultrasound systems regarding 3D image acquisition, 3D image processing, 3D image display techniques and clinical evaluation of the commercial and prototype 3D ultrasound systems was completed.

Based on this review of the advantages and disadvantages of these systems, a hand-held 3D ultrasound system including hardware and software based on a commercial B-scanner (Dynamic Imaging C2000) has been constructed to extend our study to 3D using a number of test objects and volunteers. The results of the experimental 3D scans of many test objects and volunteers have shown that real-time 3D ultrasound imaging of a limited volume size is feasible using a hand-held transducer with specific hardware to do the simple image processing and to display 3D images in real-time. Real-time 2D/3D Duplex imaging could be of clinical value when a diagnosis is to be made from a limited volume of tissue. The concept could also be extended to 3D Doppler.

In conclusion, it can be said that the objectives set at the beginning of the thesis have been satisfied in full and that the research has demonstrated the feasibility of developing real-time 3D ultrasound imaging.

10.2. Future Development

Based on the research work carried out on this thesis, a real-time 3D ultrasound imaging system can be implemented. To achieve this, the three main stages of the 3D ultrasound imaging system: 3D image acquisition, 3D image processing and 3D image display must be implemented in real-time.

First of all, 3D image acquisition can be done in real-time. In our system, a 3D data set is collected in real-time using a fast data capture board as presented in chapter 7. Our hand-held scanner was built as a test-rig. However a refined real-time scanner based on the same principle could readily be built. Even faster volume scan rates may be possible using multi-beam arrays which are now appearing on the market.

3D image processing can be implemented in real-time using dedicated hardware such as that needed for 2D real-time ultrasound imaging (Loupas et al.,1994) because 3D is an extension of 2D. Interpolation or gap filling can be done in real-time because all scanners use dedicated hardware for real-time 2D imaging (e.g. C2000 from Dyamic Imaging Ltd.). In the thesis off-line 3D median filtering has been used to reduce noise and speckle which cover objects of interest to improve the quality of the displayed 3D images. Hence, a real-time 3D median filter is required and can be made by extending the dedicated hardware of 2D median filters (Oflazer, 1983; Jaggernauth and Venetsanopoulos, 1985; Hatamian, 1986; Perlman et al., 1987; Vainio et al., 1989; Cypher and Sanz, 1989; Hartley and Corbett, 1990; Richards, 1990; Catthoor and De Man, 1990; Siu et al., 1993) either by cascading a 2D median filter and a 1D median filter or implementing a 3D median filter in dedicated hardware. Real-time scan conversion can be achieved by extending the technique used in 2D real-time imaging.

Our 3D image display volume rendering technique required several seconds to execute, now a 3D rendering processor for low cost PC designs developed by Egham- based 3Dlabs and a powerVR chip from videologic/NEC (Electronics Weekly, 1996) will be available the summer of 1996 to solve this aspect of 3D imaging by a single-chip processor. In this instance, we can thank the games world for a service to medical technology.

REFERNCES

References

Ahmed M, Wade G & Wang K (1980). Ultrasonic imaging modalites for medical applications. IEEE Trans on Nuclear Science, 27 (3), 1156-1167.

Allott C P, Arundel P A, Barry C D, John N W, Mellor P M, Thomson D S & Waterton J C (December 1995). Freehand, quantitative 3-dimensional ultrasound imaging and its application. BMUS, 27th Annual meeting, Torquay.

ANLYZE SOFTWARE, CNSSoftware Ltd, West Sussex, UK, 1990.

Bailey D & Paglione A (1986). Development of a portable resonance C-scan system. Abstract of proceedings, Material Evaluation, 44, part 9, 134.

Bajura M, Fuchs H & Ohbuchi R (1992). Merging virtual objects with the real world: seeing ultrasound imagery within the patient. Computer Graphics, 26, 2, 203-211.

Balen F G, Allen C M, Gardener J E, Siddle N C & Lees W R (1993). 3-dimensional reconstruction of ultrasound images of the uterine cavity. British Journal of Radiology, 66, 588-591.

Bamber J C & Daft C (1986). Adaptive filtering for reduction of speckle in ultrasonic pulse-echo images. Ultrasonics, 24, 41-44.

Barber F E, Baker D W, Nations A W S, Strandness D E & Reid J M (1974). Ultrasonic duplex echo Doppler scanner. IEEE Trans Biomedical Engineering, 21, 109-113.

Bartrum R J & Crow H C (1980). Ultrasound echo averaging: a simple method for improving image perception. Journal of Clinical Ultrasound, 8, 63-64.

Baum G & Greenwood I (1958). The application of ultrasonic locating techniques to opthalmology. Archives of Opthalmology, 60, 263-279.

Baum G & Greenwood I (1961). Orbital lesion localisation by three-dimensional ultrasonography. N Y State J Med, 61,4149-4157.

Belohlavek M, Foley D A, Gerber T C, Greenleaf J F & Seward J B (1993). Three-dimensional ultrasound imaging of the atrial septum: normal and pathologic anatomy. Journal Amer Coll of Cardiology, 22, 6, 1673-1678.

Blaas H G, Eik-Nes S H, Kiserud T, Berg S, Angelsen B & Olstad B (1995). Three-dimensional imaging of the brain cavities in human embryos. Ultrasound Obstet Gynecol, 5, 228-232.

Bom N, Lancee C, Honkoog J & Hugenholtz P (1971). Ultrasonic viewr for cross-sectional analysis of moving cardiac structres. Biomed Engineering, 6, 500.

Brinkley J F, Moritz W E & Baker D W (1978). Ultrasonic three-dimensional imaging and volume from a series of arbitrary sector scans. Ultrasound in Med & Biol, 4, 317-327.

Catthoor F & de Man H J (1990). Application-specific architectural methodologies for high-throughput digital signal and image processing. IEEE Trans on Acoustic, Speech and Signal Processing, 38,2, 339-349.

Chandrasekaran K, Porter T, D'Adamo A, Sehgal C M, Taylor D & Mohanty P (1991). Three-dimensional volumetric imaging of pulmonary arteries in humans by intravascular ultrasound: feasibility and clinical potential(abstract). Circulation, 84(suppl 2): II-685.

Chan P & Lim J S (1985). One-dimensional processing for adaptive image restoration IEEE Trans on Acoustic, Speech and Signal Processing, 33, 117-126.

CIO-DIO96 user manual, Computer Boards Inc. (1990).

Clinics in Diagnostic Ultrasound. Churchill Livingstone, New York (Large series of clinical texts).

Coy K M, Park J C, Fishbein M C, Laas T, Diamond G A, Adler L, Maurer G & Siegel R J (1992). In vitro validation of the three-dimensional intravascular ultrasound for the evaluation of arterial injury after ballon angioplasty. Journal Amer Coll of Cardiology, 20, 3, 692-700.

Cypher R & Sanz J L C (1989). SIMD architectures and algorithms for image processing and computer visio n. IEEE Trans on Acoustic, Speech and Signal Processing, 37,12, 2158-2173.

Davidson R E, Jensen J A & Smith S W (1994). Two-dimensional random arrays for real time volumetric imaging. Ultrasonic Imaging, 16, 143-163.

Davis L S & Rosenfeld A (1978). Noise cleaning by iterated local averaging. IEEE Trans on Systems Mans and Cybernetics, 8, 705-710.

Dekker D L, Piziali R L & Dong E (1974). A system for ultrasonically imaging the human heart in three dimensions. Comput Biomed Res, 7, 544-553.

Delcker A & Diener H C (1994). Quantification of atherosclerotic plaques in carotid arteries by three-dimensional ultrasound. British Journal of Radiology, 67, 672-678.

Detmer P R, Bashein G, Hodges T, Beach K W, Filer E P, Burns D H & Strandness D E (1994). 3D ultrasonic image feature localization based on magnetic scanhead tracking : in vitro calibration and validation. Ultrasound in Med & Biol, 20,9,923-936.

Dickinson R J (1982). Reduction of speckle in ultrasound B-scan by digital signal processing. Acoustic Imaging Vol 12, Edited by E A Ash & C R Hill (Plenum Press: New York), 213-224.

Ding R & Venetsanopoulos A N (1987). Generalizes hormophic and adaptive order statistics for the removal of impulsive and signal-dependent noise. IEEE Trans Circuits and Systems, 34, 948-955.

Djordjevic B B (1983). Computerized robotic ultrasonic C-scan. Abstract of proceedings, Material Evaluation, 41, part 2, 10.

Donald I, McVicar J & Brown T G (1958). Investigation of abdominal masses by pulsed ultrasound. Lancet , 1, 1188-1195.

Dynamic Imaging C2000 user manual (1986).

Electronics weekly (February 28, 1996). No 1756.

Elvins T T (1992). A survey of algorithms for volume visualization. Computer Graphics, 26, 3, 194-201.

Entrekin R, Keller P & Robinson B (1992). Real time 3-D ultrasonic imaging with a 1-D fan beam transducer array. Proc SPIE 1733:264-272.

Evans J L, Ng K H, Vonesh M J, Radvany M G, Kane B J & Meyers S N (1991). Spatialy correct 3-D reconstruction of intravascular ultrasound data (abstract). Circulation, 84(suppl 2): II-685.

Evans D H, MCDicken W N, Skidmore R & Woodcock J P (1989). Doppler Ultrasound: Physics, instrumentation and clinical applications. (Wiley: Chichester, England).

Fiasconaro J G (1979). Two-dimensional nonrecursive filters. Picture processing and digital filtering, Edited by T S Huang (Springer-Verlag: Berlin).

Fine D, Perring S, Herbetko J, Hacking N, Fleming J S & Dewbury K C (1991). Three-dimensional ultrasound imaging of the gallbladder and dilated biliary tree: reconstruction from real-time B-scans. British Journal of Radiology, 64, 1056-1057.

Fish P (1972). Visualising blood flow vessels by ultrasound. In Roberts VC (ed) Blood flow measurements. (Sector publishing: London).

Fish P (1990) Physics and Diagnostic medical ultrasound. (Wiley : England).

Frieden B R (1979). Image enhancement and restoration. Picture processing and digital filtering, Edited by T S Huang (Springer-Verlag: Berlin).

Fry W J, Leichner G H, Okuyama D, Fry F J & Fry E K (1968). Ultrasonic visualization system employing new scanning and presentation methods. Journal of the Acoustical Society of America, 44, 1324-1338.

Fuchs H, Pizer S M, Heinz E R, Tsai L C and Bloomberg S H (1982). Adding a true 3-D display to a raster graphic system. IEEE Trans Computer Graphics and Applications, 2, 73-78.

Gallangher N C Jr & Wise G L (1981). A theoretical analysis of the properties of median filters. . IEEE Trans on Acoustic, Speech and Signal Processing, 29, 1136-1141.

Gardener J E (November 1992). The principles of three-dimensional ultrasound imaging, BMUS Bulletin.

Geiser E A, Ariet M, Conetta D A, Lupkiewiez S M, Christie L G & Conti C R (1982). Dynamic three-dimensional echocardiographic reconstruction of the intact human left ventricle: Technique and initial observations in patients. American Heart Journal, 103,6,1056-1065.

Geiser E A, Christie L G, Conetta D A, Conti C R & Gossman G S (1982). Mechanical arm for spatial registration of two-dimensional echographic sections. Cathet Cardiovasc Diagn, 8, 89-101.

Ghosh A, Nanda C N & Maurer G (1982). Three-dimensional reconstruction of echo-cardiographics images using the rotation method. Ultrasound in Med & Biol, 8,6, 655-661.

Gilja O H, Thune N, Matre K, Hausken T, Odegaard S & Berstad A (1994). In vitro evaluation of three-dimensoinal ultrasonography in volume estimation of abdominal organs. Ultrasound in Med & Biol, 20,2,157-165.

Gonzales R C & Wintz P (1987). Digital image processing.(Addison-Wesley: Reading, Massachusetts).

Gordon G A, Canumalla S & Tittmann B R (1993). Ultrasonic C-scan imaging for material characterization. Ultrasonics, Vol. 31, 5, 373-380.

Granrath D J (1981). The role of human visual models in image processing. Proceedings of the IEEE, 69, 552-561.

Granz B & Oppelt R (1986). Real-time ultrasonic transmission imaging with a two-dimension PVDF receiver array. Abstract of proceedings, Eleventh International Symposium on Ultrasonic Imaging and Tissue Characterization, June 1986, 40.

Greenleaf J F & Bahn R C (1981). Clinical imaging with transmissive ultrasonic computerized tompgraphy. IEEE Trans Biomedical Engineering, 28,177-185.

Greenleaf J F (1982). Three-dimensional imaging in ultrasound. J Med Syst, 6,579-589.

Halliwell M (1987). Ultrasonic imaging in medical diagnosis. IEE Proceedings Part A, 134, 179-187.

Halliwell M, Key H, Jenkins D, Jackson P C & Wells P N T (1989). New scans from old: digital reformatting of ultrasound images. British Journal of Radiology, 62,824-829.

Harltey R & Corbett P (1990). Digit-serial processing techniques. IEEE Trans on Circuits and Systems, 37, 6, 707-719.

Hatamian M (1986). A real time two-dimensional moment generating algorithm and its simple chip implementation. IEEE Trans on Acoustics, Speech and Signal Processing, 34, 546-553.

Hecker R & Pöppl S J (1982). Structured noise removal by using a high flexible 2D FIR digital filter. SPIE Proceedings, 375, 87-91.

Hewllet Packard optoelectronics Designer's catalogue, 1990.

Hill C R (1986). Introduction. Physical principles of medical ultrasonics, Edited by C R Hill (Ellis Horwood: Chicchester, Sussex).

Holton W & Djordjevic B B (1985). A low cost C-scan system with multicolor computer graphics. Abstract of proceedings, Material Evaluation, 43, part 2, 12.

Hoskins P R (1990). Measurement of arterial blood flow by Doppler Ultrasound. Clin Phys Physiol Meas, 11, 1-26.

Hottier F & Collet Billon A (1990). 3D Echography: status and perspective, 3D imaging in medicine. (Springer-Verlag: UK).

Howry D H, Posakony G J, Cushman C R & Holmes J H (1956). Three dimensional and stereoscopic observation of body structure by ultrasound. J appl physiol, 9, 304-306.

Hummel R (1977). Image enhancement by histogram transformation. Comuter Graphics and Image Processing, 50, 308-328.

Hussey M (1985). Basic physics and technology of medical diagnostic ultrasound. (MacMillan: London).

Ito K, Itoh M, Yuta S, Yokoi H & Inouye T (1979). C-mode scan and resolution improvement techniques for ultrasound diagnosis. IEEE Trans on Biomedical Engineering, 26, 1, 11-17.

Ito K, Shibuya N, Tamura K & Semuma N (1980). A real-time ultrasonic diagnostic equipment for B, C and F mode tomograms. Abstract of proceedings. physics in Medicine and Biology, 25(4), 788.

Itoh M & Yokoi H (1979). A computer-aided three-dimensional display system for ultrasonic diagnosis of a breast tumor. Ultrasonics, 261-268.

Jaggernauth J, Loui A C P & Venetsanopoulos A N (1985). Real-time image processing by distributed arithmetic implementation of two-dimensional digital filters. IEEE Trans on Acoustics, Speech and Signal processing , 33, 6, 1546-1555.

Jonas W E (1972). First a meat slicer - then a C-scan unit. Materials Evaluation. September 1992, 23A-26A.

Joynt L & Popp R L (1982). The concept of three dimensional resolution in echocardiographic imaging. Ultrasound in Med & Biol, 8,237-247.

Jurkovic D, Geipel A, Gruboeck K, Jauniaux E, Natucci M & Campbell S (1995). Three-dimensional ultrasound for the assessment of uterine anatomy and detection of congenital anomalies: a comparison with hystersalpingography and two-dimensional sonography. Ultrasound Obstet Gynecol, 5, 233-237.

Kato H & Matsumoto G (1982). Design of circularly symmetrical 2-D FIR digital filters using a Fourier reconstruction technique. IEEE Trans on Acoustics, Speech and Signal processing, 30, 505-108.

Kerr A T, Patterson M S, Foster F S & Hunt J W (1986). Speckle reduction in pulse echo imaging using insensitive and phase sensitive signal processing techniques. Ultrasonic Imaging, 8, 11-28.

King D L, King D L & Shao M Y (1990). Three-dimensional spatial registration and interactive display of position and orientation of real-time ultrasound images. Journal of Ultrasound in Medicine, 9, 525-532.

King D L, King D L J & Shao M Y (1991). Evaluation of in vitro measurement accuracy of a three-dimensional ultrasound scanner. Journal of Ultrasound in Medicine, 10, 77-82.

Kossoff G, Griffiths K A & Warren P S (1994). Real-time quasi-three-dimensional viewing in sonography, with conventional, grey-scale volume imaging, Ultrasound in Obstetrics and Gynaecology, 4, 211-215.

Kossoff G, Kaye A & Anthony P (1995). Transducer Rotation: A useful scanning manoeuvre in three-dimensional ultrasonic volume imaging, Radiology, 195, 870-872.

Kremkau F W & Taylor K J W (1986). Artifacts in ultrasound imaging. Journal of Ultrasound in Medicine, 5, 227-237.

Kuan D T, Sawchuk A A, Strabd T C & Chavel P (1985). Adaptive noise smoothing filter for images with signal-dependent noise. . IEEE Trans on Pattern Analysis and Machine Intelligence, 7, 165-177.

Kundu A, Mitra S K & Vaidyanathan P P (1984). Application of two-dimensional generalized mean filtering for removal of impulse noise from images. IEEE Trans on Acoustics, Speech and Signal processing, 32, 600-609.

Lalouche R C, Bickmore D, Tessler F, Mankovitch H K & Kangaraloo H (1989). Three-dimensional reconstruction of ultrasound images. Proc SPIE, Medical Imaging, 59-66.

Lee A, Kratochwil A, Deutinger J & Bernaschek G (1995). Three-dimensional ultrasound in diagnosing phocomelia. Ultrasound in Obstetrics and Gynaecology, 5, 238-240.

Lee J S (1980). Digital imaging enhancement and noise filtering by use of local statistics. IEEE Trans on Pattern Recognition and Machine Intelligence, 2, 165-168.

Lee J S (1983). Digital image smoothing and the sigma filter. Computer Vision, Graphics and Image Processing, 24, 255-269.

Lees A R (March 1992). 3-D ultrasound images optimize fetal review. Diagnostic Imaging, 15, 69-73.

Lerski R A (1988). Practical ultrasound . (IRL Press Limited: Oxford).

Levine R A, Handschumacher M D, Sanfilippo A J, Hagege A A, Harrigan P & Marshall J E (1989). Three-dimensional echocardiographic reconstruction of the mitral valve, with implications for the diagnosis of mitral valve prolapse. Circulation, 80, 589-598.

Levoy M (1988). volume rendering: display of surfaces from volume data. IEEE Computer Graphics and Applications,8, 29-37.

Loupas T, McDicken W N & Allan P L (1987). Noise reduction in ultrasonic images by digital filtering. British Journal of Radiology, 60, 389-392.

Loupas T, McDicken W N & Allan P L (1988). An adaptive weighted median filter for speckle suppression in medical ultrasonic images. IEEE Trans on Circuit and Systems 36: 129-135.

Loupas T, McDicken W N, Anderson T & Allan P L (1994). Development of an advanced digital image processor for real-time speckle suppression in routine ultrasonic scanning. Ultrasound in Med & Biol , 20, 3, 239-249.

LS DSP32 image processor board user manual (1991).

Margulis A R & Shea W J (1986). Advances in imaging technology and their impact on medicine. British Journal of Medicine, 59, 309-315.

Martin R W, Bashein G, Detmer P R & Moritz W E (1990). Ventricular volume measurement from a multiplaner transesophageal ultrasonic imaging system: An in-vitro study. IEEE Trans Biomedical Engineering , 37, 442-449.

Mastin G A (1985). Adaptive filters for digital noise smoothing: an evaluation. Computer Vision, Graphics and Image Processing, 31, 103-121.

Matsumoto M, Matsuo H, Kitabatake A, Inoue M, Hamanaka Y & Tamura S (1977). Three-dimensional echocardiogram and two--dimensional echocardiographic images at desired planes by a computerised system. Journal of Clinical Ultrasound, 3, 163-178.

McCann H A, Sharp J S, Kinter T M, McEwan C N, Barrilot C & Greenfield J F (1988). Multidimensional ultrasonic imaging for cardiology. Proc IEEE, 76,9, 1063-1073.

McClellan J H & Chan D S K (1977). A 2-D FIR filter structure derived from the Chebyshev recursion. IEEE Trans on Circuits and Systems, 24, 372-378.

McCready V R & Hill C R (1971). A constant depth ultrasoic scanner. British Journal of Radiology, 44,747-750.

McDicken W N (1991). Diagnostic Ultrasonics - Principles and Use of Instruments. (Willey: New York).

McDicken W N, Lindsay M & Robertson D A R (1972). Three-dimensional images using a fibreoptic ultrasonic scanner. British Journal of Radiology, 45, 70-71.

McDonald D A (1974). Blood flow in arteries. (Edward Arnold: London).

Merz E, Bahlmann F & Weber G (1995). Volume scanning in the evaluation of fetal malformations: a new dimension on prenatal diagnosis. Ultrasound in Obstetrics and Gynaecology, 5, 222-227.

Mezrich R (1977). High resolution, high sensitivity ultrasonic C-scan imaging system. Acoustical Holography, Vol. 7, 53-64.

Milan J (1972). An improved ultrasonic scanning system employing a small digital computer. British Journal of Radiology, 45, 911-916.

Mills P H & Fuchs H (1990). 3D ultrasound display using optical tracking. First Conference of Visualization for Biomedical Computing, Atlanta G A, IEEE, 490-497.

Mintz G S, Potkin B N, Keren G, Eldredge S L & Satler L F (1991). Cross-sectional and three-dimensional intravascular ultrasound analysis of coronary artery geometry after rotational atherectomy (abstract). Circulation, 84(suppl 2): II-155.

Miyazawa T (1991). A high-speed integrated rendering for interpreting multiple variable 3D data. Proc SPIE, 1459,5.

Morikubo H, Takada M & Shida S (1985). Image processing by microcomputer to the ultrasonogram of the thyroid, the breast and the abdominal organs (abstract). Proceedings of Fourth Meeting of the World Federation fro Ultrasound in Medicine and Biology and First World Congress of Sonographers, Sydney, Australia, 561.

Moritz W E & Shreve P L (1976). Analysis of an ultrasonic spatial locating system. IEEE Trans Instrum Meas, 25, 43-50.

Moritz W E & Shreve P L (1977). A system for locating points, lines and planes in space. IEEE Trans Instrum Meas, 26, 5-10.

Moritz W E, Pearlman A S, McCabe D H, Medema D K & Ainsworth A K (1983). An ultrasonic technique for imaging the ventricle on three dimensions and calculating its volume. IEEE Trans Biomedical Engineering, 30, 482-491.

Nakatani H, Tamura S, Tanaka K, Kitabatake A & Inoue M (1979). A binocular stereoscopic display system for echocardiography. IEEE Trans Biomedical Imaging, 26,65-67.

Nelson T & Elvis T (1993). Visualization of 3D ultrasound data. IEEE Computer Graphics and Applications, 13, 6, 50-57.

Nelson T R & Pretorius D H (1992). Three-dimensional Ultrasound of Fetal Surface Features. Ultrasound in Obstetrics and Gynaecology, 2, 1-9.

Nestrom H, Holm H H, Christensen N E, Movild A F & Nolsoe C (1991). 3-dimensional ultrasound based demonstration of the posterior urethra during voiding combined with urodynamics. Scand J Urol Nephrol Suppl, 137, 125-129.

Niblack W (1986). An introduction to digital image processing. (Englewood Cliffs,Prentice Hall).

Niemann H (1981). Pattern Analysis. (Springer-Verlag: UK).

Nodes T A & Gallagher N C (1984). The output distribution of median type filters. IEEE Trans on Communications, 32, 532-541.

Oates C P & Whittingham T A (1984). An experimental investigation of the 'hosepipe' technique of real-time C-scanning. Acoustical Imaging, 12, 175-184.

Oflazer K (1983). Design and implementations of a single-chip 1-D median filter. IEEE Trans on Acoustics, Speech and Signal processing, 31, 1164-1168.

Ohbuchi R & Fuchs H (1991). Incremental volume rendering algorithm for interactive 3D ultrasound imaging. Information Processing in Medical Imaging, (Lecture Notes in Computer Science, Springer -Verlag, Wye, UK), 486-500.

Ohbuchi R& Fuchs H (1990). Incremental 3D ultrasound imaging from a 2 D scanner, First Conference on Visualization in Biomedical Computing, Atlanta, GA, IEEE, 360-367.

Omoto R (ed) (1987). Color atlas of real-time two-dimensional Doppler echocardiography. 2nd edn Shindan-To-Chiryo, Tokyo.

Ophir J & Maklad N F (1979). Digital scan converters in diagnostic ultrasound imaging. Proceedings of the IEEE, 67, 654-664.

Oppenheim A V & Schafer R W (1975). Digital Signal processing. (Prentice-Hall: Englewoood Cliffs, New Jersey).

Perlman S S, Eisenhandler S, Lyons P W & Shumila M J (1987). Adaptive median filtering for impulse noise elimination in real time TV signals. IEEE Trans on Communications, 35, 646-652.

Petrovic O, Feigenbaum H, Armstrong W F, Ryan T, West S R, Green-Hess D, Stewart J, Mattson-Friedmeyer J L & Fineberg N (1986). Digital averaging to facilitate two-dimensional echocardiographic measurements. Journal of clinical ultrasound, 14, 367-372.

Picot P A, Rickley D W, Mitchell R, Rankin R N & Fenster A (1993). Three-dimensional Colour Doppler Imaging. Ultrasound in Med & Biol, 19, 2, 95-104.

Pitas I & Venetsanopoulos A N (1986). Nonlinear mean filters in image processing. IEEE Trans on Acoustics, Speech and Signal Processing, 34, 573-584.

Pluygers E, Burion J, Takehara Y & Takamizawa K (1984). C-mode ultrasonography in breast disease. J. Belge Radiol, 67, 4, 235-240.

Pomalaza-Raez C A & McGillem C D (1984). An adaptive nonlinear edge-preserving filter. IEEE Trans on Acoustics, Speech and Signal Processing, 32, 571-576.

Pretorius D H, Nelson T R & Jaffe J S (1992). 3-Dimensional ultrasound analysis based on Color-Flow Doppler and grey-scale image data: a preliminary report. Journal of Ultrasound in Medicine, 11, 225-232.

Pye S D, Wild S R, & McDicken W N (1988). Clinical trial of a new adaptive TGC system for ultrasound imaging. British Journal of Radiology, 61, 523-526.

Rabiner L R, McGonegal C A & Paul D (1976). FIR windowed filter design program - WINDOW. In Programs for Digital Signal Processing, (IEEE Press: New York), 5.2.1 - 5.2.19.

Raichelen J S, Trivedi S S, Herman G T, Sutton M G & Reichek N (1986). Dynamic three dimensional reconstruction of the left ventricle from two-dimensional echocardiograms. Journal Amer Coll of Cardiology, 8,2,364-370.

Rao V V B & Rao K S (1986). A new algorithm for real-time median filtering. IEEE Trans on Acoustics, Speech and Signal Processing, 34, 1674-1675.

Redman J D, Walton W P, Fleming J E & Hall A J (1969). Holographic display of data from ultrasonic scanning. Ultrasonics, 7, 26-29.

Reid J M & Spencer M P (1972). Ultrasonic Doppler technique for imaging blood vessels. Science, 176, 1235-1236.

Restori M & Wright J E (1977). C-scan ultrasonography in orbital diagnosis. Journal of Ophthalmology, 61, 735-740.

Restori M (1985). Real-time immersion B-scan and C-scan techniques in ophthalmic diagnosis. Ultrasound in Med & Biol, 11, 1, 185-192.

Restori M (1979). Ultrasound in orbital diagnosis. Transactions of the Ophthalmological Society of the United Kingdom, 99, 223-225.

Richards D S (1990). VLSI Median Filters. IEEE Trans Acoustics, Speech and Signal Processing, 38, 1, 145-153.

Ritenour E R, Nelson T R & Raff U (1984). Applications of the median filter to digital radiographic images. In Proceedings of 1984 International Conference on Acoustics, Speech and Signal Processing, San Diego, California, 23,.1.1-23.1.4.

Robinson D E & Knight P C (1981). Computer reconstruction techniques in compound pulse-echo imaging. Ultrasonic Imaging, 3, 217-234.

Robinson D E (1972). Display of three-dimensional ultrasonic data for medical diagnosis. The Journal of the Acoustical Society of America, 52, 673-687.

Sabella P (1988). A rendering algorithm for visualizing 3D scaler fields. Computer Graphics (proceedings of SIGGRAPH '88),22,4,51-58.

Sakas G, Schreyer & Grimm M (1995). Preprocessing and volume rendering of 3D ultrasonic data. IEEE Computer Graphics and Applications. Visulation case studies, 47-54

Schmitt R M, Meyer C R, Carson P L & Samuels B I (1984). Median and spatial low-pass filtering in ultrasonic computed tompgraphy. Med Phys, 11, 767-771.

Schuster E, Knoflach P, Huber K & Grabner G (1986). An interactive processing system for ultrasonic compound imaging, real-time image processing and texture analysis. Ultrasonic Imaging, 8, 131-150.

Shattuck D P & von Ramm O T (1982). Compound scanning with a phased array. Ultrasonic Imaging, 4, 93-107.

Shattuck D P, Weinshenker M D, Smith S W & von Ramm O T (1984). Explososcan: a parallel processing technique for high speed ultrasound imaging with linear phased arrays. Journal of the Acoustical Society of America, 75, 1273-1282.

Sherrier R H & Johnson G A (1987). Regionally adaptive histogram equalization of the chest. IEEE Trans on Medical Imaging, 6, 1-7.

Siu J, Li J & Luthi S (1993). A real-time 2-D median based filter for video signals. IEEE Trans on Consumer Electronics, 39, 2, 115-121.

Skorton D J, McNary C A, Child J S, Newton F C & Shah P M (1981). Digital image processing of two dimensional echocardiogram: identification of the endocardiium. Am J Cardiol, 48, 479-486.

Smith S W, Pavy S G & von Ramm O T (1991). High-speed ultrasound volumetric imaging system - Part 1: Transducer design and beam steering. IEEE Trans on Ultrasonics, Ferroelectrics and Frequency Control, 38,2, 100-108.

Sohn C, Grotepass J, Schneider E, Funk A & Sohn G (1988). Initial studies of three-dimensional imaging using ultrasound. Z Geburtshife Perinatol, 192, 241-248.

Somer J (1968). Electronic sector scanning for ultrasonic diagnosis. Ultrasonics, 6, 153-159.

Song J & Delp E J (1990). The analysis of morphological filters with multiple structuring elements. Computer Vision, Graphics and Image Processing, 50, 308-328.

Steen E & Olstad B (1994). Volume rendering of 3D medical ultrasound data using direct feature mapping. IEEE Trans on Medical Imaging, 13, 3, 517-525.

Steinke W & Hennerci M (1989). Three-dimensional ultrasound imaging of carotid artery plaques. J Cardiovasc Tech, 8, 15-22.

Stickels K R & Wann LS (1984). An analysis of three-dimensional reconstructive echocardiography. Ultrasound in Med & Biol, 10, 5, 575-580.

Stone M (1990). A three-dimensional model of tongue movement based on ultrasound and x-ray and microbeam data. The Journal of the Acoustical Society of America, 87, 2207-2217.

Taylor K J W & Strandness D F (eds) (1990). Dopplex Doppler Ultrasound. (Churchill Livingstone: New York).

Taylor K J W, Burns P N & Wells P N T (1988). Clinical applications of doppler ultrasound. (Raven Press: New York).

Thurstone F L, Kjosnes N L & McKinney W M (1965). Ultrasonic scanning and biologic tissue by a new technique. Science, 149, 302-303.

Toda K, Hayama M, Moriizumi T & Yasuda T (1982). A method of obtaining C-scan images by use of interdigital transducers. The Journal of the Acoustical Society of America, 71, 5, 1285-1286.

Tomtec: Tomographic Technology GmbH, Breslauer Strasse 123, Eching W8057, Munich, Germany

Toriwaki J & Yokoi S (1986). Basics of algorithms for processing three-dimensional digitized pictures. Syst Comput Jpn, 17, 73-82.

Upson C & Keeler M (1988). VBUFFER: Visible volume rendering. Computer Graphics (Proceedings of SIGGRAPH '88),22,4, 59-64.

Vainio O, Neuvo Y & Butner S E (1989). A signal processor for median-based algorithms. IEEE Trans Acoustics, Speech and Signal Processing, 37, 9, 1406-1413.

Venetsanopoulos A N & Cappellini V (1986). Real-time image processing. In multidimensional systems, Edited by S G Tzafestas (Marcel Dekker: New York), 345-399.

von Ramm O T, Smith S W & Pavy H G (1991). High speed ultrasound volumetric imaging system - Part II: Parallel processing and image display. IEEE Trans on Ultrasonics, Ferroelectrics and Frequency Control, 38,2,109-115.

von Ramm O T, Smith S W & Pavy H G (1991). High Speed Ultrasound Volumetric Imaging System - Part II: Parallel Processing and Image Display. IEEE Trans on Ultrasonics, Ferroelectrics and Frequency Control, 38,2,109-115.

Warren J M (1981). Inexpensive expansion of ultrasonic C-scan capabilities. Materials Evaluation, 40, 122-124.

Warren J M (1984). C-scan recordings: A tutorial. Materials Evaluation, 42, 983-985.

Watkin K L & Rubin J M (1989). Pseudo-three-dimensional reconstruction of ultrasonic images of the tongue. The Journal of the Acoustical Society of America, 85, 496-499.

WE DSP32 support software library user manual (1988).

WE DSP32 digital signal processor information manual (1988).

Wells P (1977). Biomedical ultrasonics. (Academic press: NewYork).

Wells P N T & Halliwell M (1981). Speckle in ultrasonic imaging. Ultrasonics, 19, 225-229.

Wells P N T (editor) (1987). The safety of diagnostic ultrasound. (British Institute of Radiology: London) Supplement 20.

Wendt P D, Coyle E J & Gallagher N C (1986). Some convergence properties of median filters. IEEE Trans on Circuits and Systems, 33, 276-286.

Weyman J P & Levine R A (1991). A new fully integrated system for three-dimensional echocardiographic reconstruction: validation for ventricular volume (abstract). Circulation, 48 (suppl 2): II -684.

White D N, Clark J M & White M N (1966). Studies in ultrasonic echoencephalography - VII: General principles of recording information in ultrasonic B- and C-scanning and the effects of scatter, reflection and refraction by cadaver skull on this information. Med. & Biol. Eng. 5, 3-14.

White D N, Clark J M, Chesebrough J N, White M N & Campbel J K (1968). Effect of the skull in degrading the display of echoencephalographic B and C scans. The Journal of the Acoustical Society of America, 44,1339-1345.

Whittingham T A (1982). Real-time depth scanning with phase arrays. Acoustical Imaging, 10, 1-16.

Wild J J & Reid J M (1952). Progress in the techniques of soft tissue examination by 15 MC pulsed ultrasound. In ultrasound in biology and medicine, ed E Kelly, 30-48 (American Institute of Biological Sciences, Washington D C).

Yli-Harja O, Astola J & Neuvo Y (1988). Analysis of the properties of median and weighted median filters using threshold logic and stack filter representation. Research

Report 10/1988, Department of Information Technology, Lappeenranta University of Technology, Finland.

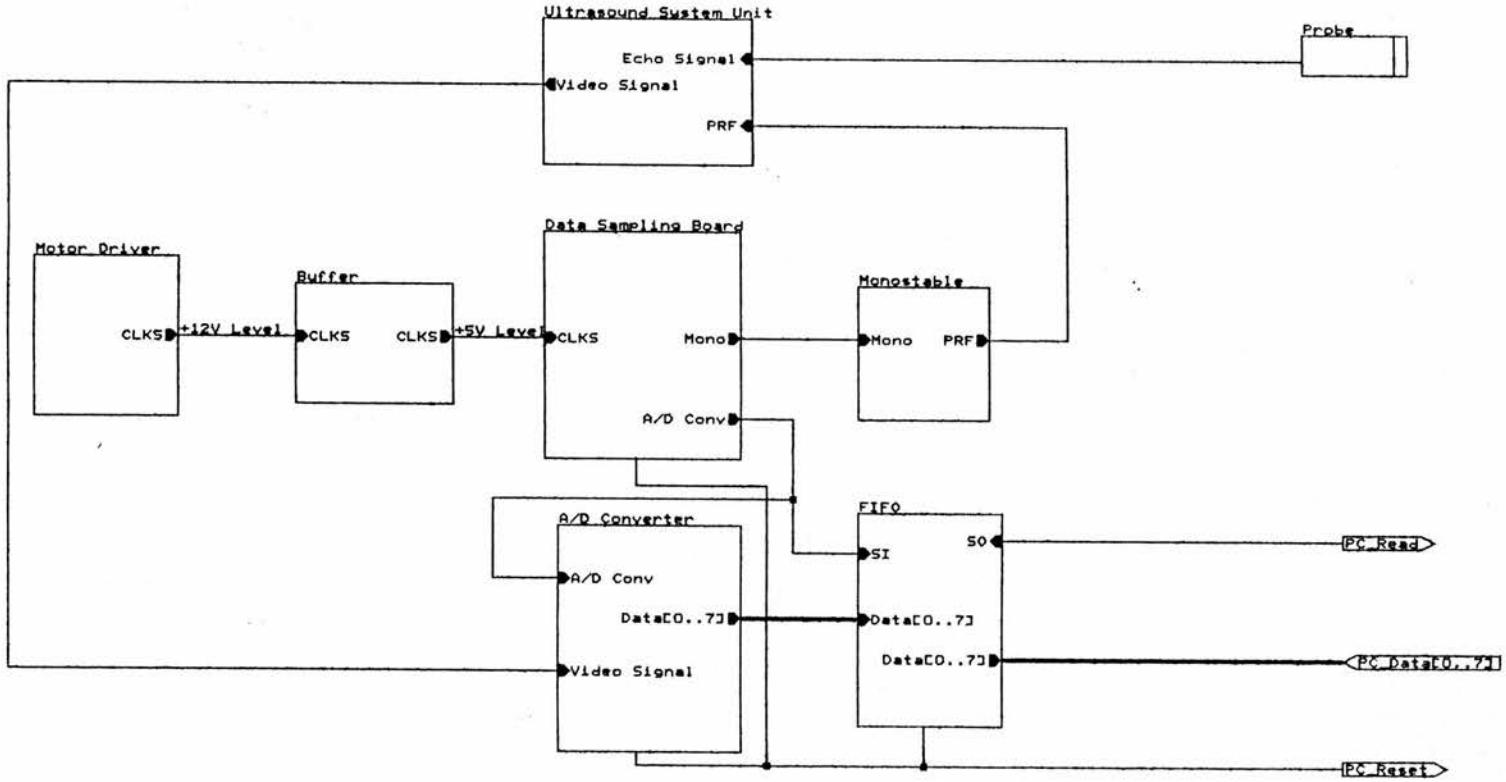
Ylitalo J, Alasaarela E, Tauriainen A, Tervola K & Koivukangas J (1986). Three-dimensional ultrasonic C-scan imaging using holographic reconstruction. IEEE Trans on Ultrasonics, Ferroeletrics and Frequency Control, 33, 6, 731-739.

Appendix A

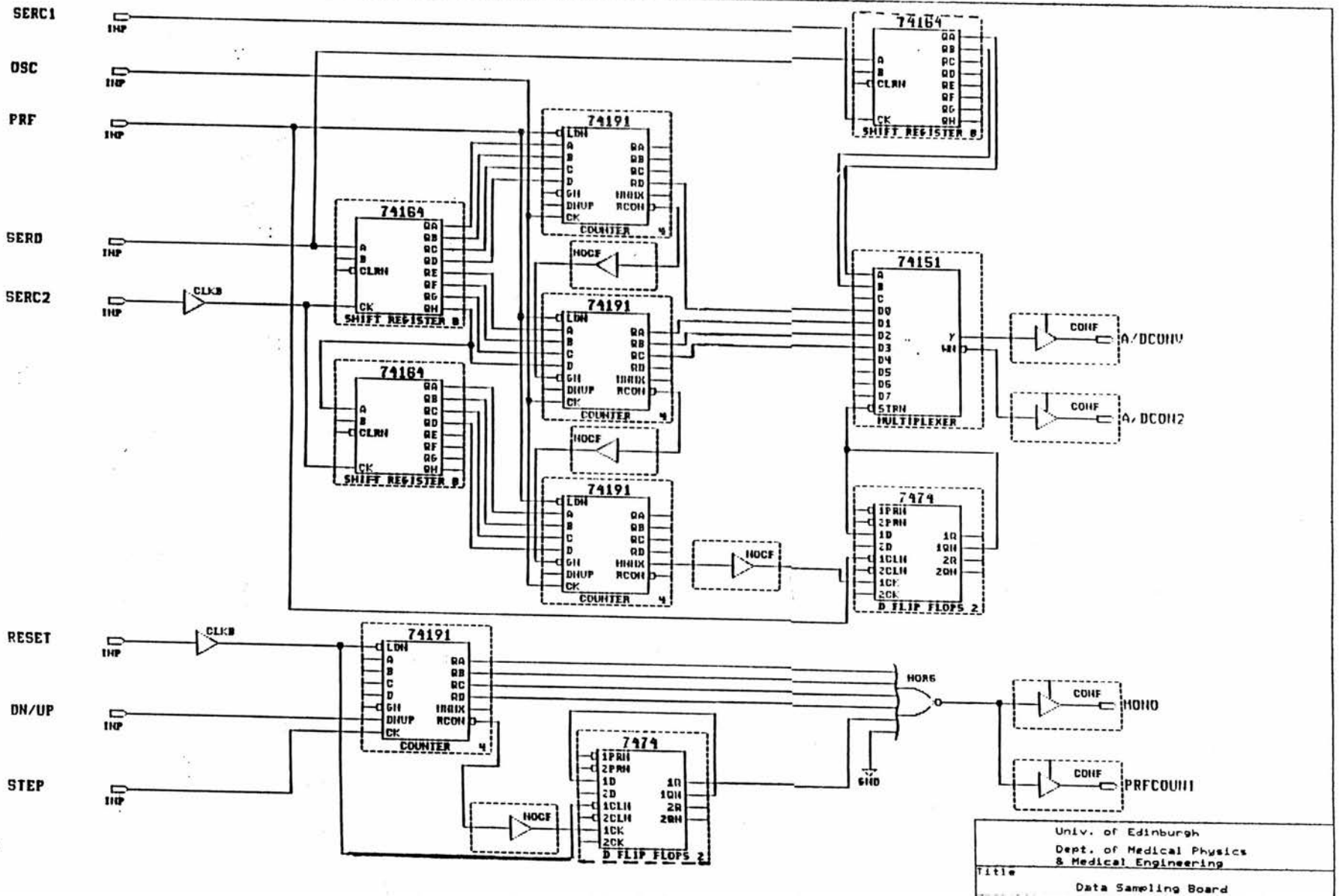
Circuit Diagrams

C-scan Ultrasound Imaging System: Sheets 1 and 2

3D Ultrasound Imaging System: Sheets 3 to 12

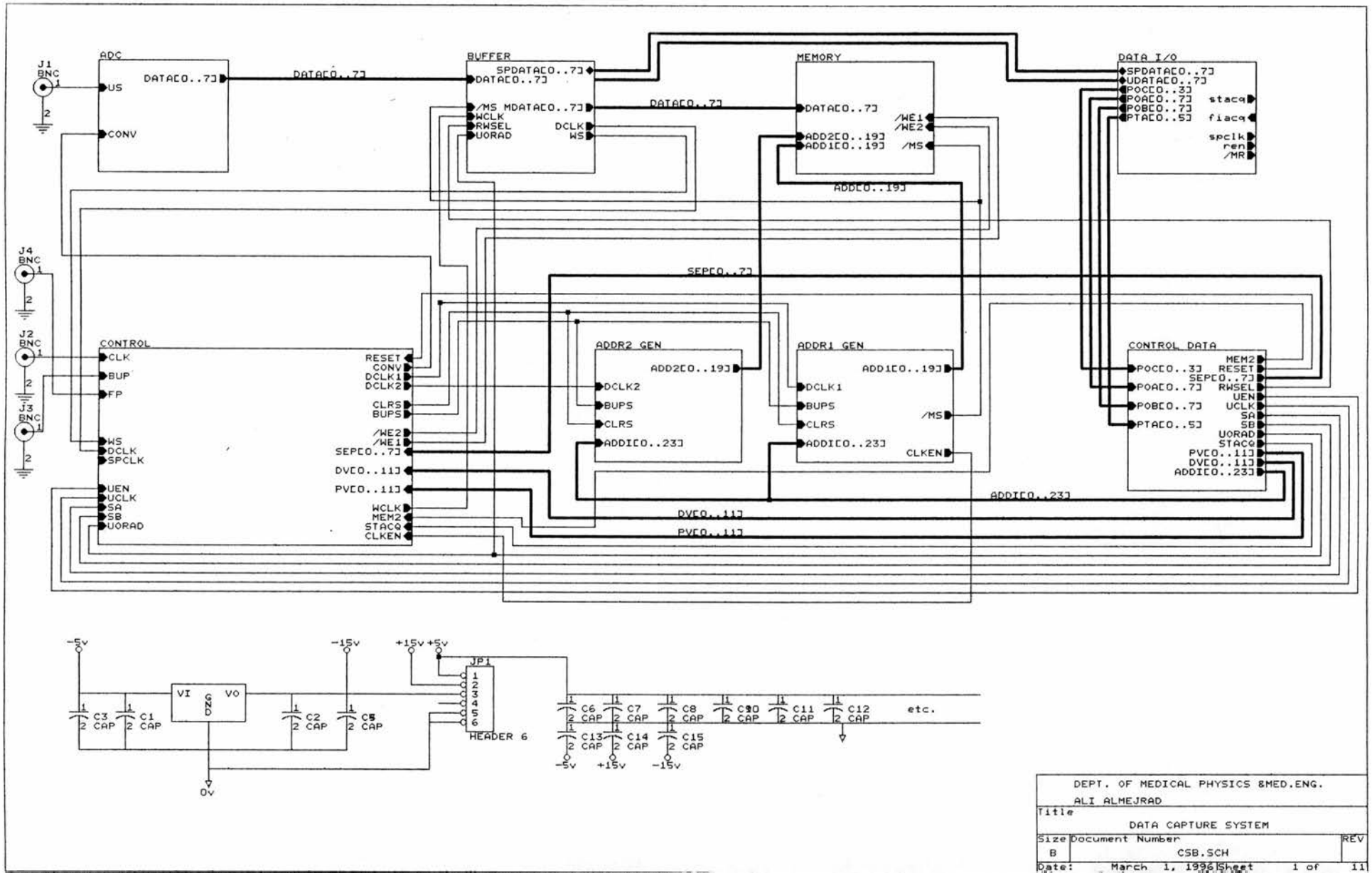


Univ. of Edinburgh Dept. of Medical Physics & Medical Engineering		
Title: C-Scan Ultrasound System		
Size: B	Document Number: C_Scan.sch	REV: A
Date: March 25, 1996	Sheet 1 of 2	



Univ. of Edinburgh
Dept. of Medical Physics
& Medical Engineering

Title
Data Sampling Board



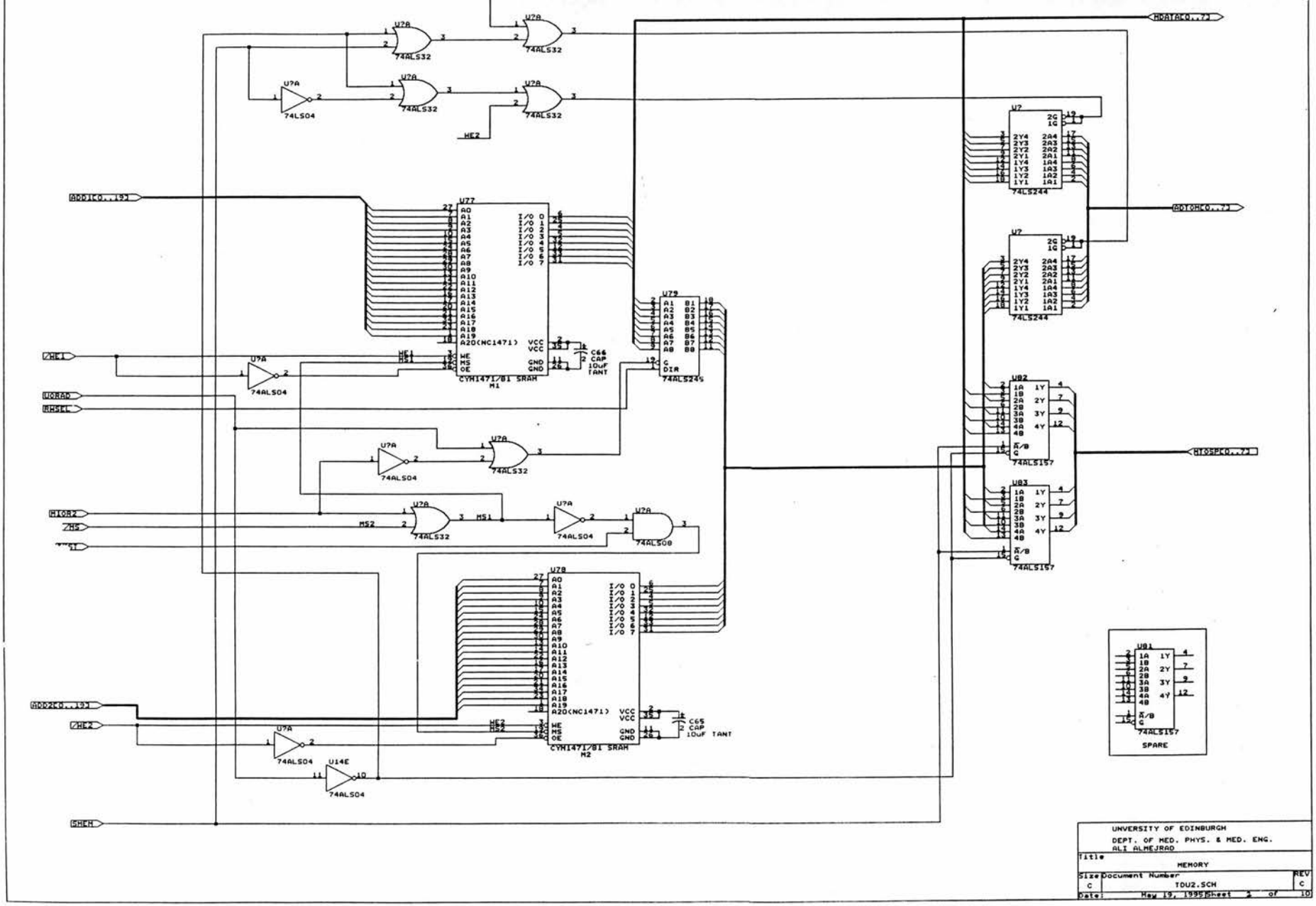
DEPT. OF MEDICAL PHYSICS &MED.ENG.

ALI ALMEJRAD

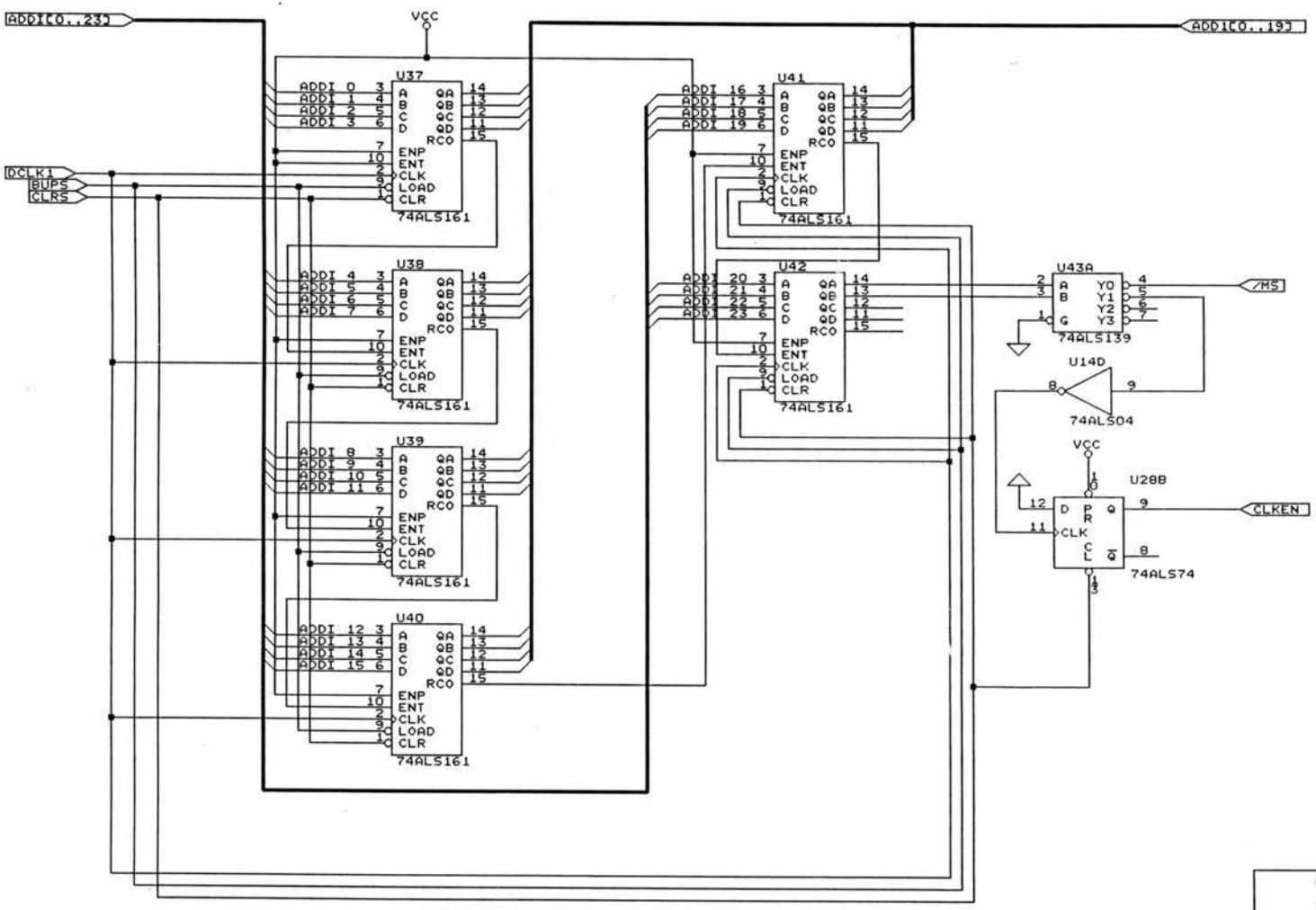
Title: DATA CAPTURE SYSTEM

Size	Document Number	REV
B	CSB.SCH	

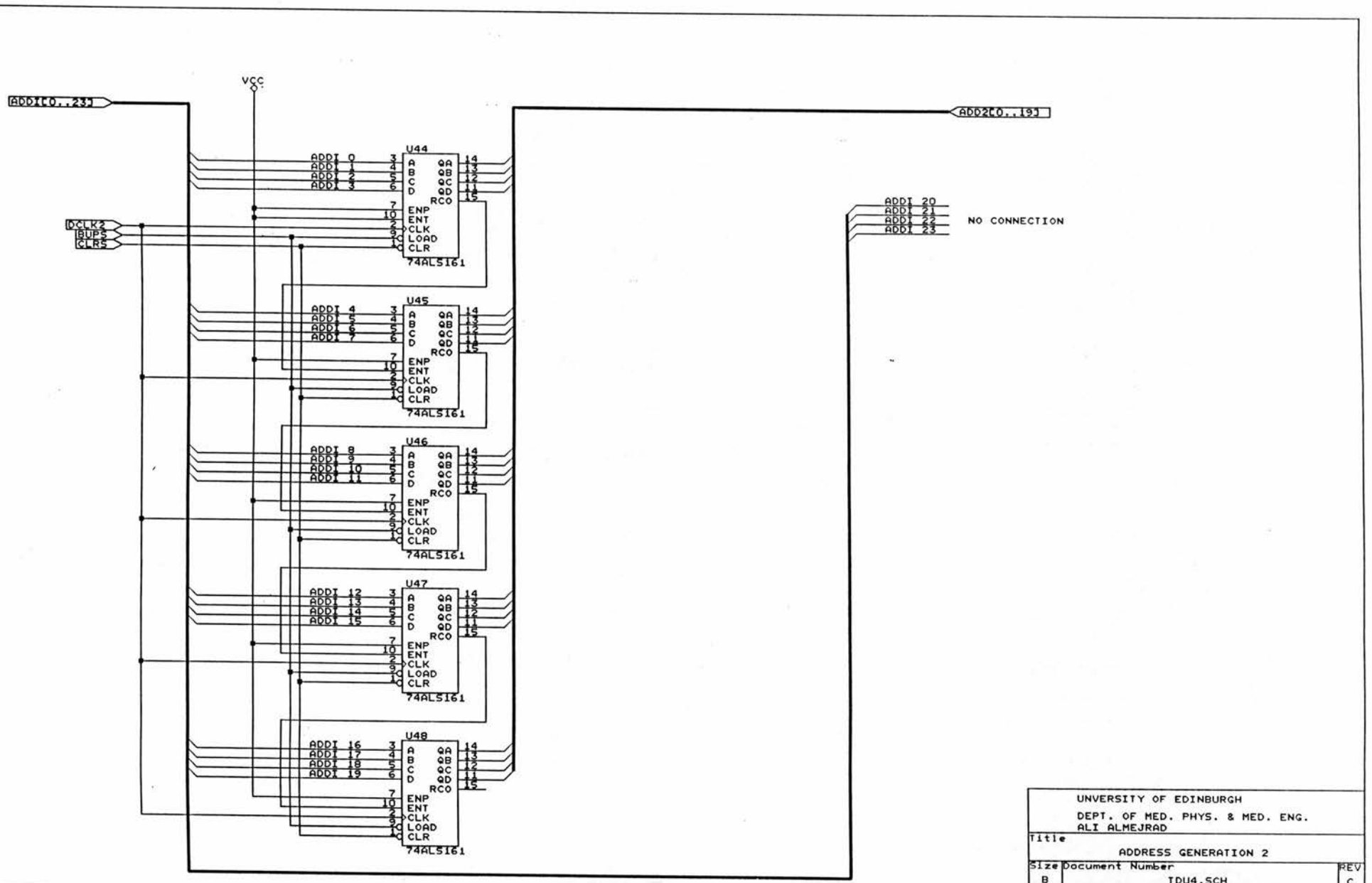
Date: March 1, 1996 Sheet 1 of 11

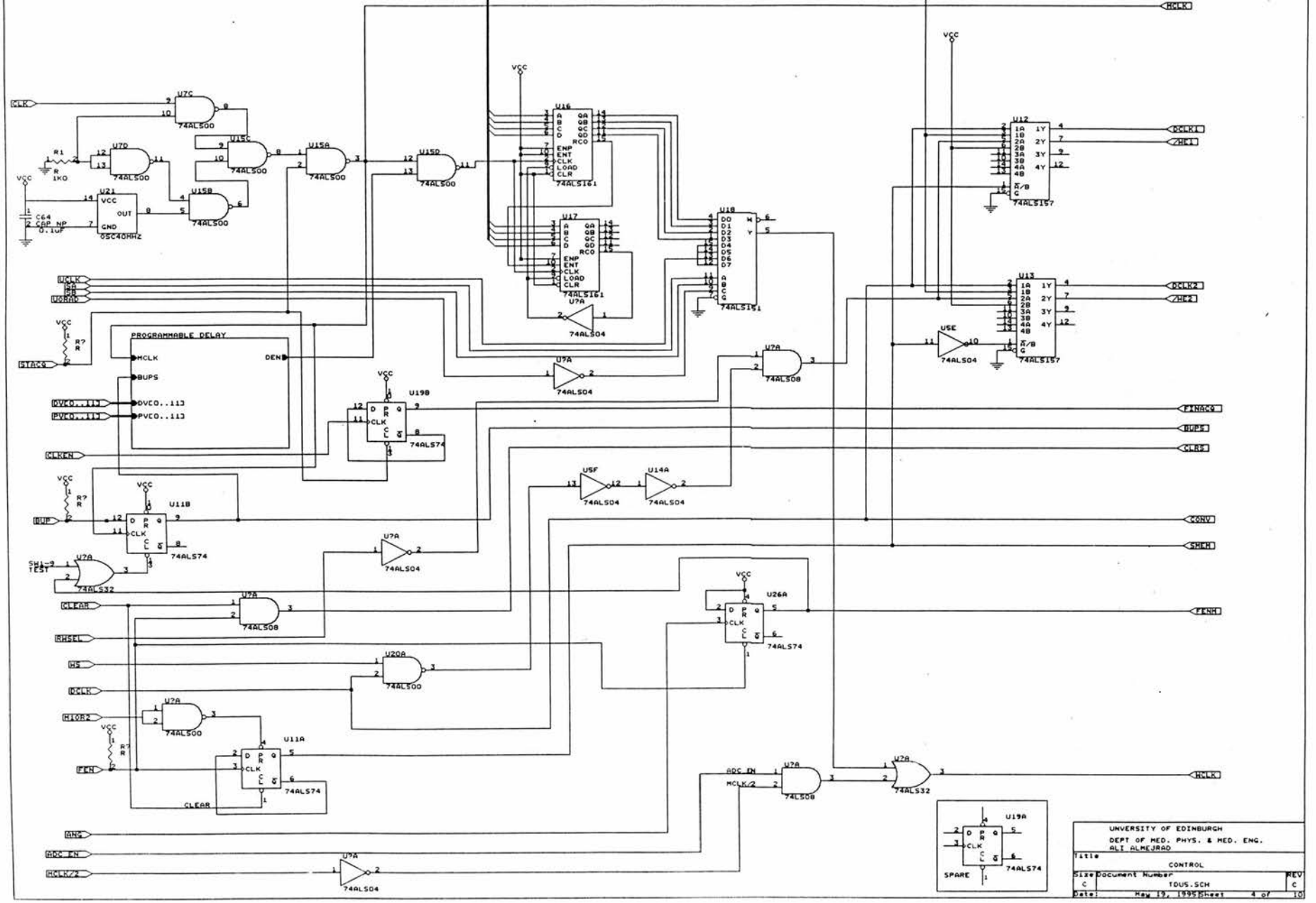


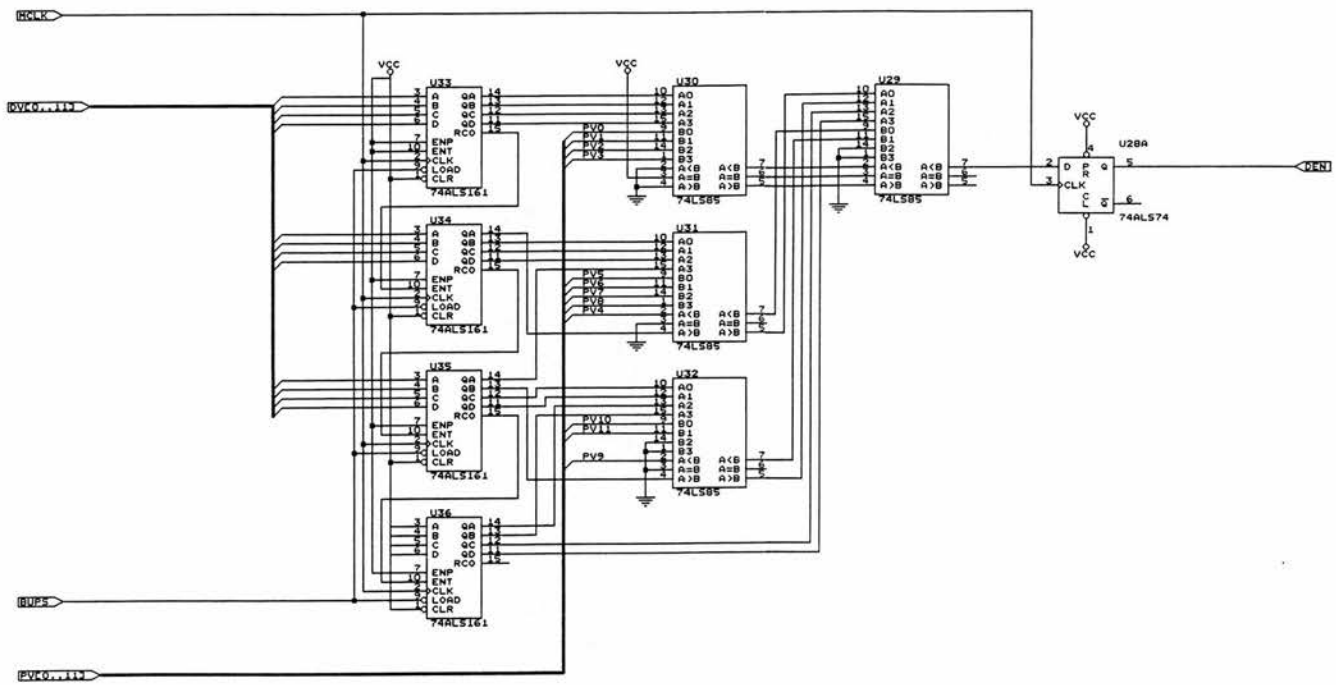
UNIVERSITY OF EDINBURGH		
DEPT. OF MED. PHYS. & MED. ENG.		
ALI ALMEJRAD		
Title: MEMORY		
Size: C	Document Number: TDU2.SCH	REV: C
Date: May 19, 1995	Sheet 2 of 10	



UNIVERSITY OF EDINBURGH		
DEPT. OF MED. PHYS. & MED. ENG.		
ALI AMEJRAD		
Title: ADDRESS GENERATION 1		
Size: B	Document Number: TDU3.SCH	REV: C
Date: August 22, 1994	Sheet 6 of 10	


ADDI[0..23]
VCC
DCLK2
BUPS
CLRS
ADDI 0
ADDI 1
ADDI 2
ADDI 3
U44
A
B
C
D
QA
QB
QC
QD
RCO
ENP
ENT
CLK
LOAD
CLR
74ALS161
ADDI 4
ADDI 5
ADDI 6
ADDI 7
U45
ADDI 8
ADDI 9
ADDI 10
ADDI 11
U46
ADDI 12
ADDI 13
ADDI 14
ADDI 15
U47
ADDI 16
ADDI 17
ADDI 18
ADDI 19
U48
ADD2[0..19]
ADDI 20
ADDI 21
ADDI 22
ADDI 23
NO CONNECTION
UNVERSITY OF EDINBURGH
DEPT. OF MED. PHYS. & MED. ENG.
ALI ALMEJRAD
Title
ADDRESS GENERATION 2
Size
B
Document Number
TDU4.SCH
REV
C
Date:
August 22, 1994
Sheet
7 of
10


CLK
R1
1KΩ
C64
CAP NP
0.1UF
OSC40MHz
74ALS00
74ALS161
74ALS151
74ALS157
74ALS04
74ALS08
74ALS74
74ALS32
PROGRAMMABLE DELAY
MCLK
BUPS
DVCO..113
PVCO..113
DEN
STACS
CLKEN
BUP
SW1-3
TEST
CLEAR
RWSEL
HS
MCLK
HIOR2
PEN
MNG
ADC EN
MCLK/2
SPARE
DCLK1
/HE1
DCLK2
/HE2
PINACQ
BUPS
CLRS
CONV
SHEN
PENM
HCLK
UNIVERSITY OF EDINBURGH
DEPT OF MED. PHYS. & MED. ENG.
ALI ALMEJRAD
Title
CONTROL
Size
C
Document Number
TDUS.SCH
REV
C
Date:
May 19, 1995
Sheet
4 of
10



UNIVERSITY OF EDINBURGH

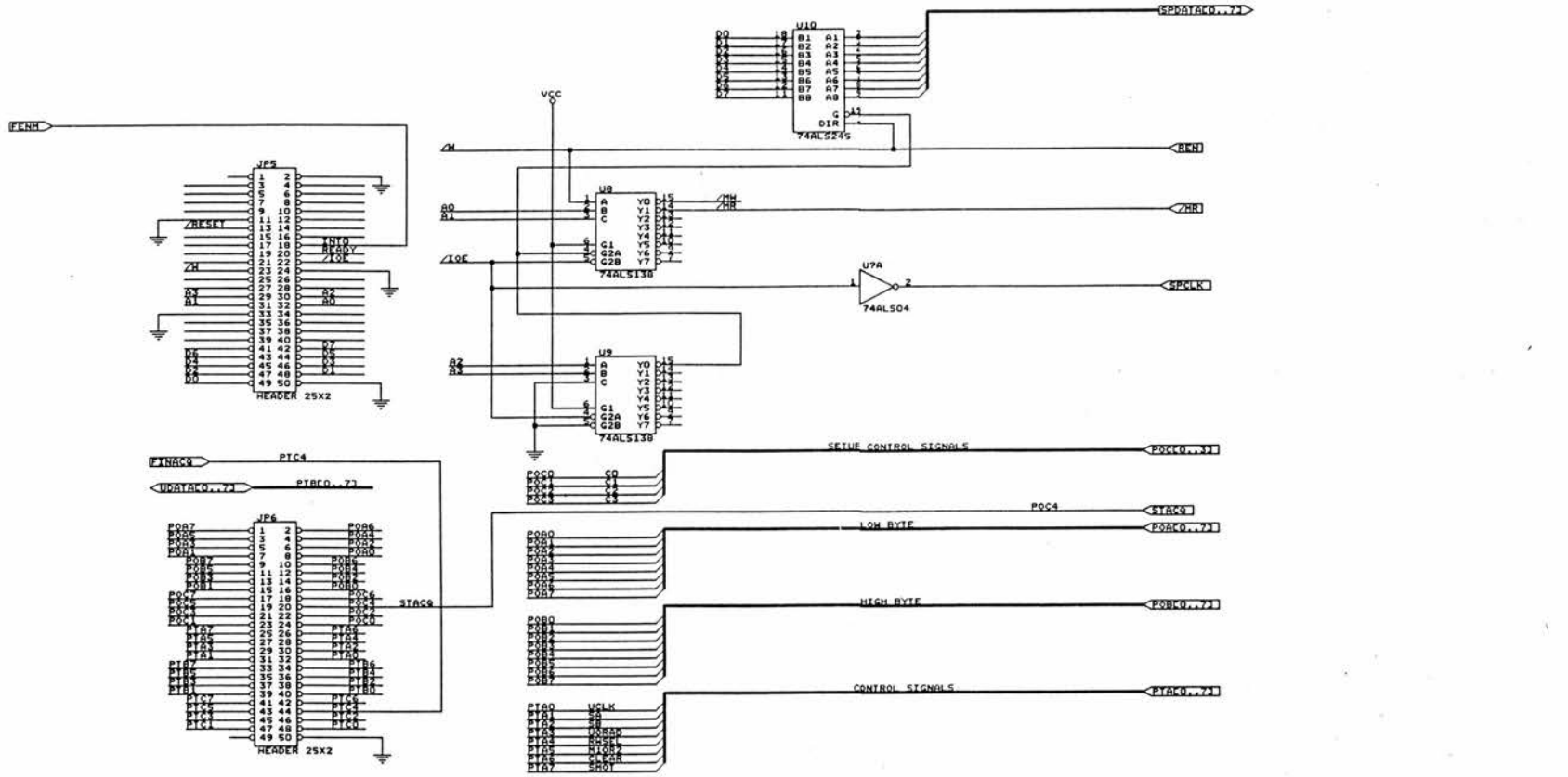
DEPT OF MED. PHYS. & MED. ENG.

ALI ALMEJRAD

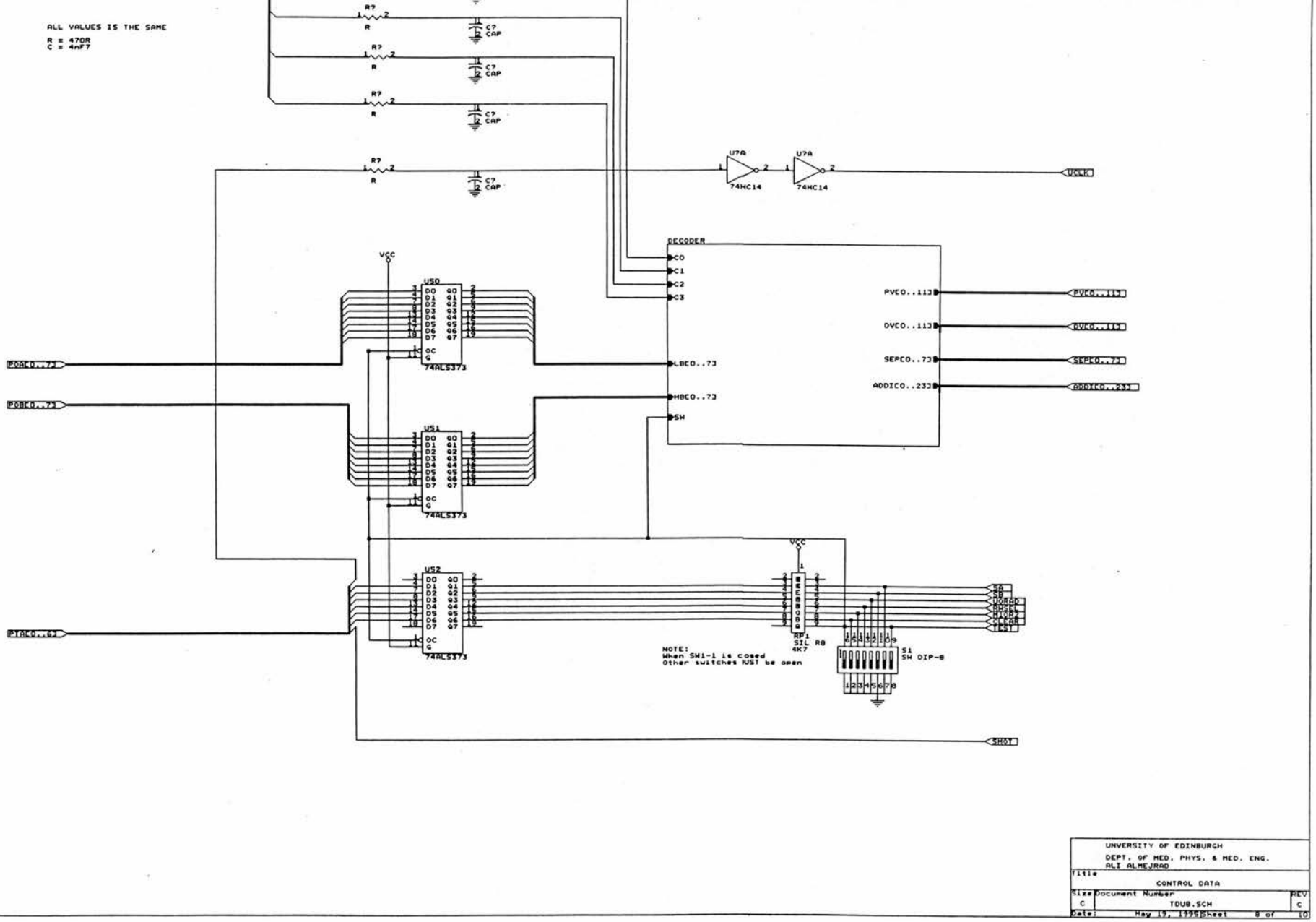
Title: PROGRAMMABLE DELAY

Size	Document Number	REV
C	TDU6.SCH	C

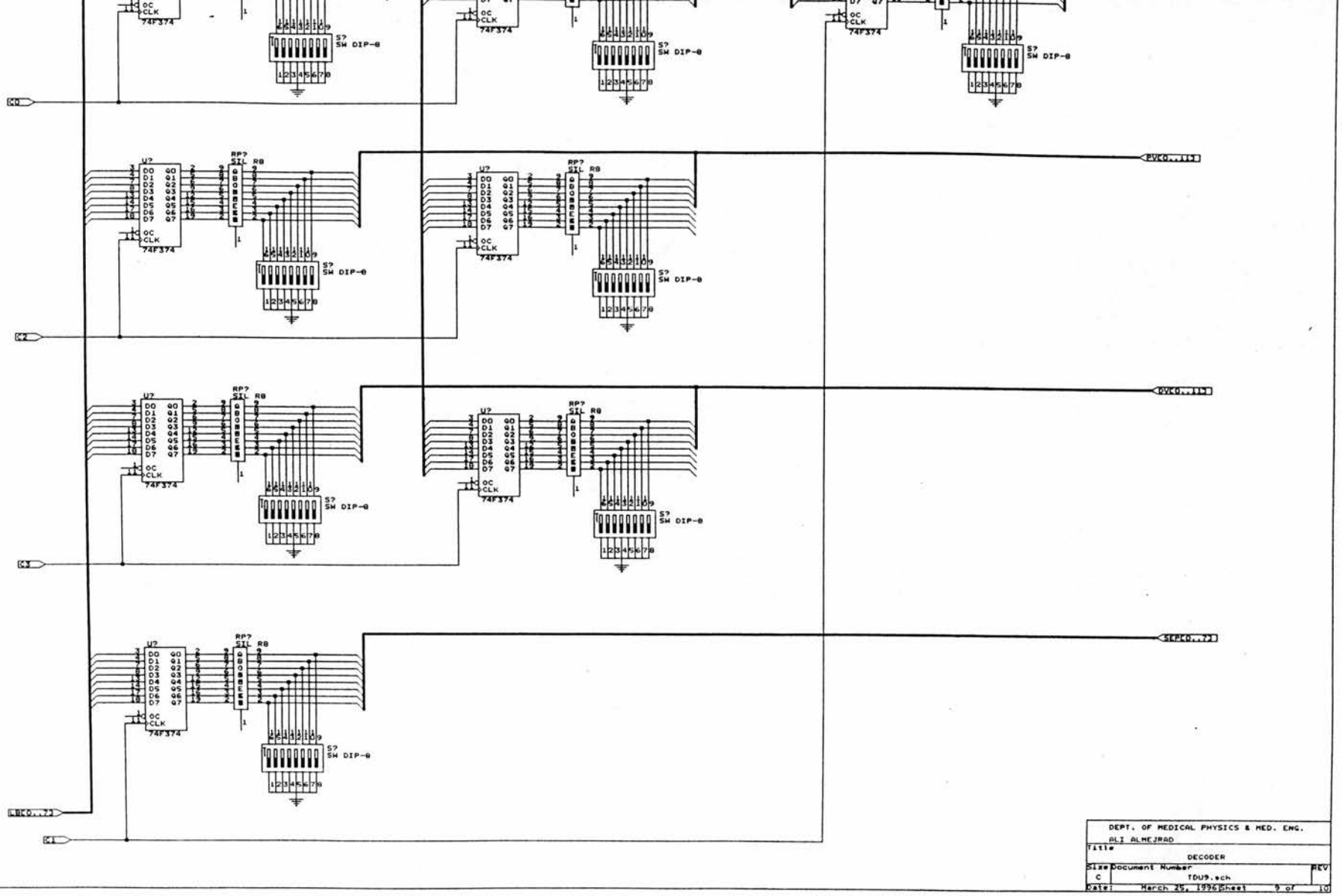
Date: April 26, 1995 Sheet 5 of 10

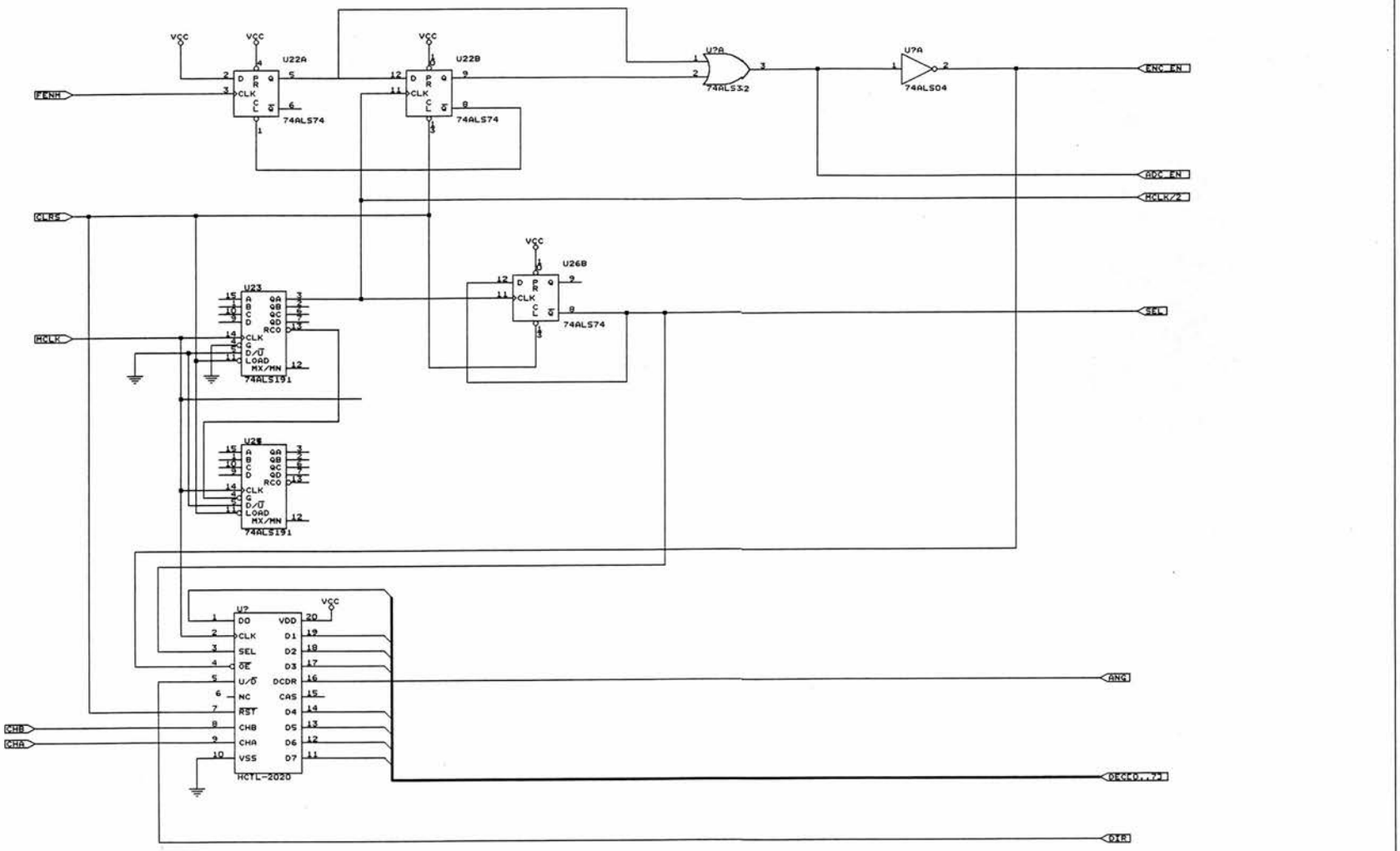


UNIVERSITY OF EDINBURGH		
DEPT OF MED. PHYS. & MED. ENG.		
ALI ALMEJRAD		
Title: I/O DATA		
Size: C	Document Number: TDU7.SCH	REV: C
Date: May 22, 1995	Sheet 3 of 10	



UNVERSITY OF EDINBURGH DEPT. OF MED. PHYS. & MED. ENG. ALI ALMEJRAD		
Title: CONTROL DATA		
Size: C	Document Number: TDUB.SCH	REV: C
Date: May 19, 1995	Sheet 8 of 10	


U?
74F374
RP?
SIL R8
S?
SW DIP-8
K0
K2
K3
K1
LBE0..7]
DVC0..11]
SEPC0..7]
DEPT. OF MEDICAL PHYSICS & MED. ENG.
ALI ALMEJRAD
Title
DECODER
Size
C
Document Number
TDU9.sch
REV
Date: March 25, 1996 Sheet 9 of 10


VCC
U22A
U22B
74ALS74
U7A
74ALS32
74ALS04
FENH
ENC_EN
ADC_EN
HCLK/2
CLRS
U26B
SEL
U23
U24
74ALS191
HCLK
U?
HCTL-2020
CHB
CHA
ANG
DECCO..7
DIR
U27A
U27B
spare
UNIVERSITY OF EDINBURGH
DEPT. OF MED. PHYS. & MED. ENG.
ALI ALMEJRAD
Title
DECODER CONTROL
Size
C
Document Number
TDU10.sch
REV
A
Date: May 22, 1995
Sheet 10 of 14

Appendix B

Program Listings

```
//*****************************************************************************************************************
//
//                                                  Main Program Code
//                                                   written in BC++
//                                                        by
//                                                   ALI  ALMEJRAD
//
//                                                   ver 3.0   date:10/6/93
//*****************************************************************************************************************

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <alloc.h>
#include <process.h>
#include <graphics.h>
#include <math.h>
#include <iostream.h>
#include <dos.h>
#include <ctype.h>
#include "C:\ali\misc.h"
#include "C:\ali\pcx.h"
#include "C:\ali\vga.h"
#include "C:\ali\imaged1.h"


/* Global Variables */
//extern int g_mode;
extern BYTE huge *buff;
static char choice;

/* External Functions */
extern  void Get_Msg(char *msg);
extern void Data_Acq(void);
extern void Image_Proc(void);
extern void Image_Disp(void);
extern void Image_Format(void);

//*****************************************************************************************************************
//                                                  Genral Functions
//*****************************************************************************************************************
int Image_Print(void)
{
 clrscr();
 textcolor(YELLOW);
 cout << "Image_Print is NO. : "  << choice << "\n" ;
 getch();
 return 0;
}
//*****************************************************************************************************************
int c_break(void)
{
 outportb(0x300,0);
 printf("\nHi ali from CSCAN program.Control-Break hit. Program aborting\n");
 fcloseall();
 farfree((BYTE far *)buff);
 return (0);
}

//*****************************************************************************************************************
//                                                  Main Function
//*****************************************************************************************************************

int main(void)
{
 int X1;
 char *str=" ";
 char sure;
 BOOL Finish=FALSE;

  ctrlbrk(c_break);
  clrscr();
  InitGraphic();    /* graphics screen */
  X1 = getmaxx()/2;
  setpalette(15,14);
```

```
  settextstyle(4,HORIZ_DIR,4);
   settextjustify(CENTER_TEXT,TOP_TEXT);
   outtextxy(X1,100,"C_SCAN SYSTEM");
  settextstyle(3,HORIZ_DIR,2);
   outtextxy(X1,170,"DEPT. OF MEDICAL PHYSICS");
   outtextxy(X1,220,"ALI ALMEJRAD");
   delay(1000);
   cleardevice();
   closegraph();
   do
   {
    Finish =0;
    InitGraphic();
    settextstyle(4,HORIZ_DIR,4);
    settextjustify(CENTER_TEXT,TOP_TEXT);
    outtextxy(X1,50,"C_SCAN  SYSTEM");
    setviewport(170,130,639,479,1);
    settextjustify(LEFT_TEXT,TOP_TEXT);
    do
    {
                clearviewport();
                settextstyle(3,HORIZ_DIR,3);
                outtextxy(0,0,  "  A  : Data Acquisition");
                outtextxy(0,30, "  B  : Image Display   ");
                outtextxy(0,60, "  C  : Image Print     ");
                outtextxy(0,90, "  D  : Image Processing");
                outtextxy(0,120,"  F  : Image Formating ");
                outtextxy(0,150,"  X  : Exit            ");
                flushall();
                outtextxy(0,180,"  Enter Your Choice :");
                do
                {
                  flushall();
                  choice = getch();
                  choice = toupper(choice);
                }
                while ((choice != 65) && (choice != 66) && (choice != 67) && (choice != 68) && (choice != 70) && (choice !=
88));
                strnset(str,choice,1);
                outtextxy(270,180,str);
                settextstyle(0,HORIZ_DIR,1);
                settextjustify(LEFT_TEXT,TOP_TEXT);
                outtextxy(0,270,"   Are you sure ? (y/n):  ");
                do
                {
                  flushall();
                  sure = getch();
                  sure = toupper(sure);
                }
                while((sure !=89) && (sure !=78));
                strnset(str,sure,1);
                outtextxy(200,210,str);
 }
 while (sure !=89);

 switch(choice)
 {
                        case  'X' : Finish=TRUE                  ; break;
                        case  'A' :closegraph(); Data_Acq()      ; break;
                        case  'B' :closegraph(); Image_Disp()    ; break;
                        case  'C' :closegraph(); Image_Print()   ; break;
                        case  'D' :closegraph(); Image_Proc()    ; break;
                        case  'F' :closegraph(); Image_Format()  ; break;
  } /* end of switch */
 restorecrtmode();
while (!Finish);
Get_Msg("\nEnd of Program.....Bye.");
closegraph();
return 0;
 //********************************** End of Main  Program **********************************************************
```

```
//*****************************************************************************************************************
//                                        Data Acquisition Code
//                                          written in BC++
//                                               by
//                                          ALI ALMEJRAD
//                                        ver:3.0  date: 16/6/1993
//*****************************************************************************************************************
#include "c:\ali\data_acq.h"
extern int c_break(void);
unsigned long offset;
int xmax;
int zmax = 15;
BYTE huge *buff_aux;
unsigned slices;

//*****************************************************************************************************************
                                          External Functions
//*****************************************************************************************************************
void exit_prog(char *msg)
{
 cout << msg ;
 textcolor(YELLOW);
 cprintf("\r\n\nPress any key to abort.....");
 getch();
 fcloseall();
 exit(1);
}
//*****************************************************************************************************************
void Get_Msg(char *msg)
{
 textcolor(YELLOW);
 cout << msg ;
 cprintf("\r\n\nPress any key to finish...");
 getch();
}
//*****************************************************************************************************************
void check_mem(void)
{
 unsigned long Nbytes=0;
 farfree((BYTE far*) buff);
 unsigned long Mem_left = (unsigned long) farcoreleft();
 printf("memory left = %lu bytes",Mem_left);
  getch();
 Nbytes = (unsigned long) sl*sp*slices;
 if (Nbytes > Mem_left)
 {
  printf("sorry ALI there is no enough memory!\n");
  printf("the memory available: %lu bytes",farcoreleft());
  exit_prog("");
 }
    //allocate buff to store the data
 buff = (BYTE huge *) farcalloc(Nbytes,(unsigned long) sizeof(BYTE));
 if(buff == NULL)
  printf("\n no enough memory in the far heap");
 printf("\nmemory left = %lu bytes",farcoreleft());
 getch();
 printf("\nThe difference = %lu",(Mem_left-farcoreleft()));
 getch();
}
//*****************************************************************************************************************
void check_desk()
{
 struct dfree free;
 long avail;
 int drive;
 char drv[3];
 unsigned long Nbytes=(unsigned long) sl*sp*slices;

 int length = strcspn(fname,":");
 if (length < strlen(fname))
 {
  strncpy(drv,fname,length);
  strupr(drv);
  drive = drv[0] - 'A' +1;
 }
 else
```

```
   drive = getdisk()+1;
   getdfree(drive, &free);
   if (free.df_sclus == 0xffff)
     exit_prog("\nError : Can not get the number of bytes available in the drive");
   avail = (long) free.df_avail * (long) free.df_bsec * (long) free.df_sclus;
   if (Nbytes > avail )
     exit_prog("\nSorry there is no enough space in disk");
   printf("\nDrive  %c: has %ld bytes available\n", drive+65-1 , avail);
   printf("This is the available memory.");
   textcolor(YELLOW);
   cprintf("\r\n\nPRESS ANY KEY TO START THE ACQUISITION...");
   getch();
}
//*****************************************************************************************************************************
//                                          General Functions
//*****************************************************************************************************************************
int FlipFlop (BYTE dat)
{
 byte_port_A = byte_port_A ^ dat; //send 0 to clock output
 outportb(DATA_A,byte_port_A) ;
 delay(1);
 byte_port_A = byte_port_A ^ dat; //send 1 to clock output
 outportb(DATA_A,byte_port_A) ;
 delay(1);
 //byte_port_A=byte_port_A ^ dat; //send 0 to clock output
 //outportb(DATA_A,byte_port_A) ;
 return 0;
}
//*****************************************************************************************************************************
int Pulse(void)
{
 byte_port_A = byte_port_A | 4;
 outportb(DATA_A,byte_port_A)  ;
 delay(1);
 FlipFlop(SERC1);
 byte_port_A = byte_port_A & 0;
 outportb(DATA_A,byte_port_A) ;
 delay(1);
 FlipFlop(SERC1);

 return 0;
}
//*****************************************************************************************************************************
int  Dec_to_Bin(int max_bits,int xposition,int yposition,int Number)
{
 int      Mask,Count1,Count2,A  ;
 BOOL     Bitset              ;
 char     Pattern[24]         ;
 Mask =1;
 Count1 =0;
 strcpy(Pattern," ");
 A =0;
 do
   {
    for (Count2=0 ;Count2 < 4 ; Count2++)
    {
            Bitset =(Number & Mask);
            Iarr[Count2+A]=Bitset;
            if (Bitset)  strcat(Pattern,"1") ;
                                  else strcat(Pattern,"0");
            Mask =Mask << 1;
            Count1 =Count1 +1;
    }
    A = A+4;
  strcat(Pattern," ");
   }
   while (Count1 < max_bits);
 gotoxy(xposition,yposition);
 printf("%s",Pattern);
 return 0;
}
//*****************************************************************************************************************************
int Pause(int speed)
{
 int k;
 for ( k =1 ; k<= speed ; k++)
```

```
 {
 ;
 }
 return 0;
}

//*****************************************************************************************************************
//***************************** 1-Scanner Position  ***************************************************************
int Change_DirX(void)
{
 byte_port_A = byte_port_A ^ DIR_X;
 outportb(DATA_A,byte_port_A);
 return 0;
}
//*****************************************************************************************************************
int Setup_X(void)
{
 clrscr();
 cout <<" Please wait ....." <<"\n";
 cout <<" Position Scanner ( X Motor ) in Progress.....";
 byte_port_A = byte_port_A | GO | LO;
 outportb(DATA_A, byte_port_A);
 do
 {; }
 while (!kbhit());
 getch();
 byte_port_A = byte_port_A ^ GO ;
 outportb(DATA_A,byte_port_A);
 Change_DirX();
 clrscr();
 return 0;
}
//*****************************************************************************************************************
int Setup_Y(void)
{
 int  m ;
 clrscr();
 cout <<"Please wait ....." <<"\n";
 cout <<"POSITION SCANNER  ( Y MOTOR ) IN PROGRESS.....";
 do
 {
  statC=statC ^ CLKY;
  for (m=0; m<10 ;m++)
  {
   statC =statC ^ CLKY;
   outportb(DATA_C,statC) ;
   Pause(400);
   statC=statC ^ CLKY;
   outportb(DATA_C,statC) ;
   Pause(400);
  }
 }
 while(!kbhit());
 delay(1);
 statC=statC ^ Y_DIR;
 outportb(DATA_C,statC);
 clrscr();
 return 0;
}
//*****************************************************************************************************************
int Setup(void)
{
 char   L  ;
 BOOL   quit ;

 textmode(LASTMODE);
 clrscr();
 do
 {
  textcolor(YELLOW);
  gotoxy(29,10);
  cout << "SCANNER SETUP  ";
  textcolor(WHITE);
  gotoxy(25,12);
  cout <<"PRESS X : Setup X_Motor ";
  gotoxy(25,14);
```

```
    cout <<"PRESS Y : Setup Y_Motor ";
    gotoxy(25,16);
    cout <<"PRESS E : Exit        ";

    quit =FALSE;
    L=getch();
    L=toupper(L);

    switch( L)
    {
              case  'X' : Setup_X();break;
              case  'Y' : Setup_Y();break;
              case  'E' : quit =TRUE;break;
    }
   }
   while( !quit);
   return 0;
 }
 //*************************************************************************************************************************
 //************************* 2-Scan Parameters ****************************************************************************
 int Load_Plane_Depth()
 {
 int i ;

   for (i =11; i>=0 ;i--)
   {
    if (Iarr[i] !=0) {
    byte_port_A = byte_port_A & 0x0;
    outportb(DATA_A,byte_port_A); }  //send data 0 to the input data pin

                          else {
                          byte_port_A = byte_port_A |4;
                          outportb(DATA_A,byte_port_A) ; }  //send 1
                          delay(1);
                          FlipFlop(SERC2);
                          delay(1);
   }
   return 0;
 }
 //*************************************************************************************************************************
 int Load_Planes_Interval(void)
 {
 int  i ;

   for (i=1 ;i>=0 ;i--)
   {
    if (Iarr[i] !=0 ) {
     byte_port_A = byte_port_A | 4;
     outportb(DATA_A,byte_port_A)   ;
     }

                         else {
                          byte_port_A = byte_port_A & 0;
                          outportb(DATA_A,byte_port_A) ;
                          }
    delay(1);
    FlipFlop(SERC1);
    delay(1);
   }
   return 0 ;
 }
 //*************************************************************************************************************************
 void  Enter_Scan_Parameters(void)
 {
  char resp[50],command[50];

  textmode(LASTMODE);
  clrscr();
  //Get time & date
  getdate(&Date);
  gettime(&Time);
  flushall();
  textcolor(15);
  cprintf("Enter No. of Sampling Points then Press RET.\r\n");
  scanf("%d",&sp);
  textcolor(14);
```

```
 cprintf("Enter No. of Scan Lines then Press RET. \r\n");
 scanf("%d",&sl);
 textcolor(15);
 cprintf("Enter No. of Slices then Press RET. \r\n");
 scanf("%d",&slices);
 textcolor(14);
 cprintf("Enter Distance between Each Raster Position in steps then Press RET.\r\n");
 scanf("%d",&steps);
 textcolor(15);
 do
 {
  cprintf("Input Depth for First Image Plane in mm then Press RET.\r\n");
  scanf("%d",&dp);
 }
 while  (dp <= 0 && dp >= 149);
 textcolor(15);
 do
 {
  cprintf("Input Plane Interval by Entering the number corresponding to\r\n");
  cprintf("the Required Interval Below:\r\n");
  cprintf("(0)=1.5 mm  (1)=3.0 mm   (2)=6.0 mm  (3)=0.75 mm\r\n") ;
  scanf("%d",&N);
 }
 while (N < 0 || N >3) ;
 strcpy(resp,"y");
 do
 {
 cprintf("Enter filename for data storage (MAX 8 characters) then Press RET.\r\n");
 scanf("%s",fname);
 if(strlen(fname) > 80)
 {
  cout << "\nSorry filname is too long (max. is 80 char.) ";
  strcpy(resp,"n");
 }
 else
  if(access(fname,00) == 0)
  {
   cout << "\n\nWarning : Existing file.....overwrite (y/n)? ";
   scanf("%s",resp);
   if(strcmp(resp,"Y") == 0 || strcmp(resp,"y") == 0)
    {
            strcpy(command,"del ");    //del filname
            strcat(command,fname);
            system(command);
    }
  }
 }
 while(strcmp(resp,"Y") !=0  && strcmp(resp,"y") !=0);
 flushall();
 textcolor(14);
 cprintf("Enter your comments:\r\n");
 flushall();
 gets(comments);
 flushall();
 getch();
 clrscr();
 adc =floor((double)(225-dp)/(double) DI[N]);
}
//*****************************************************************************************************************************
void Display_Scan_Parameters(void)
{
 BOOL  Done  ;
 char  ch ;
 Done=FALSE;

 textmode(LASTMODE);
 do
 {
  clrscr();
  textcolor(YELLOW);
  gotoxy(10,3);
  cprintf("  SCAN PARAMETERS DISPLAY:  \n");
  gotoxy(10,4);
  cout <<" ------------------------- ";
  textcolor(15);
  gotoxy(10,6);
```

```
    cprintf("  No. of Sampling Points            = %d ",sp);
    gotoxy(10,7);
    cprintf("  No. of Scan Lines                 = %d ",sl);
    gotoxy(10,8);
    cprintf("  No. of Slices                     = %d ",slices);
    gotoxy(10,9);
    cprintf("  The Depth of the First Image      = %d  mm",dp );
    gotoxy(10,10);
    cprintf("  The Distance b/w Each Raster Position= %d  steps",steps );
    gotoxy(10,11);
    cprintf("  The Interval between Planes       = %f   mm",DI[N]);
    gotoxy(10,12);
    cprintf("  No. of A/D Conversions (MAX.)     = %f    " ,adc);
    gotoxy(10,13);
    cprintf("  Data File Name                    = %s  ",fname);
    textcolor(YELLOW);
    gotoxy(10,16);
    cprintf("  Do you want to change the values? (Y/N)");

    do
    {
     ch=getch();
     ch=toupper(ch);
    }
    while (ch !='Y' && ch !='N');
    textcolor(WHITE);
    //ch=toupper(ch);
    switch (ch)
    {
     case 'Y' : Enter_Scan_Parameters();break;
     case 'N' :  Done =TRUE;break;
    }
   }
   while (!Done);
    gotoxy(10,22);
    cprintf("Press any key to load the values");
    getch();
}
//*****************************************************************************************************************************
void Load_Scan_Parameters(void)
{
   byte_port_A = 0;            //initiliaze port A
   outportb(DATA_A, byte_port_A);
   textmode(LASTMODE);
   clrscr();
   coint=N;
   depth_delay=floor((double)(2*dp*8E6)/(double)15.0E5);
   codel=(int)depth_delay;
   gotoxy(13,10);
   printf("Plane_Inteval in BINARY= ");
   Dec_to_Bin(4,55,10,coint);
   gotoxy(13,11);
   //printf("%d   %d",Iarr[1],Iarr[2]);      // choosen for example
   Load_Planes_Interval();
   gotoxy(13,12);
   printf("The Depth of the First Image in BINARY= ");
   Dec_to_Bin(12,55,12,codel);
   gotoxy(13,13);
   //printf("%d   %d   %d   %d    %d  %d   %d
%d",Iarr[1],Iarr[2],Iarr[3],Iarr[4],Iarr[5],Iarr[6],Iarr[7],Iarr[8]);
   Load_Plane_Depth();
   gotoxy(20,20);
   textcolor(YELLOW);
   cprintf("Press any key to return to the MainMenu");
   getch();
}
//*****************************************************************************************************************************
void open_hdrfile()
{
  char x2[50];
  char fname1[25];

  strcpy(fname1, fname);
  strcat(fname1,".hdr");
  strcpy(x2,"C:\\DATA\\HEADER\\");
  strcat(x2,fname1);
```

```
 if((hdrfile = fopen(x2,"wt")) == NULL)
 {
  cout << "\nSorry can not open this file.....";
  getch();
  exit(1);
 }
}
//*****************************************************************************************************************
void save_hdrfile(void)
{
 double step_mm;

 fprintf(hdrfile," \nVERSION ALI ALMEJRAD (1) ");
 fprintf(hdrfile," \nDate %d-%d-%d",Date.da_day, Date.da_mon, Date.da_year);
 fprintf(hdrfile," \nTime %d:%d:%d",Time.ti_hour,Time.ti_min,Time.ti_sec);
 fprintf(hdrfile," \nNo. of sample points: %d  ponts",sp);
 fprintf(hdrfile," \nNo. of sample lines : %d  lines",sl);
 fprintf(hdrfile," \nNo. of slices : %d slices",slices);
 step_mm = (double) steps*0.00577;
 fprintf(hdrfile," \nDistance between lines : %f  mm",step_mm);
 fprintf(hdrfile," \nDistance between samples : 0.75 mm");
 fprintf(hdrfile," \nInterval between planes : %f  mm",DI[N]);
 fprintf(hdrfile," \nThe depth of the first image plane from the face of the transducer : %d  mm",dp);
 //fprintf(hdrfile," \nNo. of conversions : %f ",adc);
 fprintf(hdrfile," \nComments : %s",comments);
 fclose(hdrfile);
}
//*****************************************************************************************************************
void Scan_Parameters(void)
{
 textmode(LASTMODE);
 Enter_Scan_Parameters();
 Display_Scan_Parameters();
 Load_Scan_Parameters();
 textmode(LASTMODE);
 clrscr();
 check_mem();
 check_desk();
 open_hdrfile();
 save_hdrfile();
}
//********************************** 3-Demo ***********************************************************************
void Demo(void)
{
  sl =60;
  sp =60;
  steps =100;
  dp =50;
  N =3;
  strcpy(fname,"demo");
  adc= floor((double)(225-dp)/(double)DI[N]);        //calculation of no. of ADCs
  Display_Scan_Parameters();
  Load_Scan_Parameters();
  textmode(LASTMODE);
  clrscr();
  check_mem();
  check_desk();
}
//*****************************************************************************************************************
/************************************ 4-Scan ***********************************************************************
int Read_FIFO()
{
BYTE  ADC_D;
BYTE s  ;
int i;

 for ( i=0 ; i < 10000 ; i++)
 {
 if ((inportb(DATA_C) & O_READY ) == 2)
    break;
    i=1;
 }
 s=0;
 for (z=0 ; z< slices ; z++)
 {
  s =s ^ SO;
```

```
    outportb(DATA_C ,s);
    s =s ^ SO;
    outportb(DATA_C,s);
    ADC_D = inportb(DATA_B);
    zmax = 15;
    xmax = sp;
    offset =(unsigned long)  ( z +(unsigned long)( x * zmax) +(unsigned long) y *(zmax * xmax));
      *(buff+(unsigned long) offset) = ADC_D;
    printf("\nOffset = %lu , y = %d ",offset,y);
    Dec_to_Bin(8,30,10,ADC_D);
   *buff_aux++ =  ADC_D;
    }
   clear FIFO
   while((inportb(DATA_C) & O_READY ) == 2)
   {
    s =s ^ SO;
    outportb(DATA_C ,s);
    s =s ^ SO;
    outportb(DATA_C,s);
    ADC_D = inportb(DATA_B);  //clear FIFO
    }
    return 0;
}
//*****************************************************************************************************************
int Test_FIFO(void)
{
 int  ADC_D;
 BYTE s  ;

 textmode(LASTMODE);
 clrscr();
  outportb(DATA_A,RESET);
  do
  {
    while ((inportb(DATA_C) & O_READY ) ==2);
    s=0;
    outportb(DATA_C,s);
    for (z=0 ; z<slices ; z++)
    {
      s =s ^ SO;
      outportb(DATA_C,s);
      delay(1000);
      s =s ^ SO;
      outportb(DATA_C,s);
      ADC_D = inportb(DATA_B);
      zmax = slices;
      xmax = sp;
      offset =(unsigned long)  ( z +(unsigned long)( x * zmax) +(unsigned long) y *(zmax * xmax));
       *(buff+(unsigned long) offset) = ADC_D;
      // printf("\nOffset = %lu , y = %d ",offset,y);
      Dec_to_Bin(8,30,10,ADC_D);
    }
  }
  while (!kbhit());
  return 0;
}
//*****************************************************************************************************************
int Step_Y(int steps, int speed)
{
 int m  ;

 statC=statC ^ CLKY;
 for (m=1 ; m<=steps ; m++)
 {
   outportb(DATA_C,statC);
   Pause(speed);
   statC =statC ^ CLKY;
   outportb(DATA_C,statC);
   Pause (speed);
  }
  return 0;
}
//*****************************************************************************************************************
int Scan(void)
{
 struct time t;
```

```
 unsigned long T1,T2,TDiff;

 textmode(LASTMODE);
 cout <<"SCANNING IN PROGRESS. Wait please.....";
 offset = 0L;
 byte_port_A = (byte_port_A ^ RESET) | HI;
 outportb(DATA_A,byte_port_A);
 buff_aux = buff;
 //get the starting time
 gettime(&t);
 T1 = ((unsigned long) (t.ti_min * 60 + t.ti_sec) * 100 ) + t.ti_hund;
 for (y=0 ; y< sl ; y++)
 {
  byte_port_A = byte_port_A | GO | RESET |HI;
  outportb(DATA_A,byte_port_A);
  for (x=0 ; x < sp ; x++)
  {
          Read_FIFO();
  }
  byte_port_A = byte_port_A ^ GO;
  outportb(DATA_A,byte_port_A);
  //delay(1);
  Change_DirX();
  //delay(1);
  Step_Y (steps,200);
  //delay(1);
 }
  byte_port_A = byte_port_A & STOP;
  outportb(DATA_A,byte_port_A);
  //get the end time
  gettime(&t);
  T2 = ((unsigned long) (t.ti_min * 60 + t.ti_sec) * 100 ) + t.ti_hund;
  TDiff = (unsigned long) (T2-T1);
  printf("The Scan Time is : %lu \n",TDiff);
  printf("Press any key to go back to the main screen");
  flushall();
  getch();
 return 0;
}
//*****************************************************************************************************************
int Sim_Data(void)
{
 int  z,x,y ;
 char x1[20];
 char fname2[50];
 textmode(LASTMODE);
 clrscr();
 printf("\nPlease Wait; Saving Data in Progress... ");
 strcpy(fname2,fname);
 strcat(fname2,".dat");
 strcpy(x1,"C:\\DATA\\");
 strcat(x1,fname2);
 MyStream=fopen(x1,"wb");
 for(y=0 ;y < sl ; y++)
 {
  for(x=0 ; x< sp ; x++)
  {
   for(z=0 ; z< slices ; z++)
   {
    zmax = slices;
    xmax = sp;
    offset =(unsigned long)  ( z +(unsigned long)( x * zmax) +(unsigned long) y *(zmax * xmax));
    if (x == y)
    *(buff+(unsigned long) offset) = z +50;
    else
    *(buff+(unsigned long) offset) = 0;
     fputc(*(buff+offset),MyStream);
   }
  }
  }
  fclose(MyStream);
  return 0;
}
//*****************************************************************************************************************
int   DummyData(void)
{
```

```
 int  z,x,y ;
 unsigned temp;
 char x1[20];
 char fname2[50];
 textmode(LASTMODE);
 clrscr();
 printf("\nPlease Wait; Saving Data in Progress... ");
 strcpy(fname2,fname);
 strcat(fname2,".dat");
 strcpy(x1,"C:\\DATA\\");
 strcat(x1,fname2);
 MyStream=fopen(x1,"wb");
 BYTE huge *Buff_aux;
 Buff_aux = buff;
 temp =0;
 for(y=0 ;y < sl ; y++)
 {
  for(x=0 ; x < sp ; x++)
  {
   for(z=0 ; z < slices ; z++)
   {
    if(floor(y/2)*2 ==y)
            {
            temp += x;
            *Buff_aux++ =  temp;
            }
    else
            {
            temp -=x;
            *Buff_aux++ =  temp;
            }
   }
  }
 }
 Buff_aux = buff;
 for(y=0 ;y < sl ; y++)
  for(x=0 ; x < sp ; x++)
   for(z=0 ; z < slices ; z++)
    fputc(*Buff_aux++,MyStream);

 fclose(MyStream);
 return 0;
}
//*****************************************************************************************************************
//************************** 5-Display Binary Data ***************************************************************
int Display_Data()
{
 textmode(LASTMODE);
 textcolor(11);
 for( y=0 ; y <sl ; y++)
 {
  for (x=0 ; x < sp ; x++)
  {
   for (z=0 ; z < slices ; z++)
   {
            zmax = slices;
            xmax = sp;
            offset =(unsigned long)  ( z +(unsigned long)( x * zmax) +(unsigned long) y *(zmax * xmax));
            printf("[%3d,%3d,%3d]= %d \n ",z,x,y, *(buff+offset));
   }
  }
 }

 while (!kbhit());

 return 0;
}
//*****************************************************************************************************************
//************************** 6-Save Data *************************************************************************
void Save_Data(void)
{
 int  z,x,y ;
 char x1[20];
 char fname2[50];

 restorecrtmode();
```

```
 textmode(LASTMODE);
 clrscr();
 printf("\nPlease Wait; Saving Data in Progress... ");
 strcpy(fname2,fname);
 strcat(fname2,".dat");
 strcpy(x1,"C:\\DATA\\");
 strcat(x1,fname2);
 MyStream=fopen(x1,"wb");
 offset = 0L;
 buff_aux = buff;
 for(y=0 ;y < sl ; y++)
  {
   for(x=0 ; x< sp ; x++)
   {
    for(z=0 ; z< slices ; z++)

    {
     fputc(*buff_aux++,MyStream);
    }
   }
  }
  fclose(MyStream);
}
//*************************************************************************************************************************
/****************************** 7-Debug ***********************************************************************************
int Test_Motor(void)
{
 int    loop  ;
 BYTE   state ;

 clrscr();
 state=0;
 do
 {
  state=GO ^ LO ;
  outportb(DATA_A,state);
  for (loop=1 ; loop<=500 ;loop++)
  {
   delay(5);
  }
  state=GO | LO;
  outportb(DATA_A,state) ;
  Change_DirX();
  sound(100);
  delay(60);
  nosound();
 }
 while (!kbhit());
 return 0;
}
//*************************************************************************************************************************
int ADC_Cal(void)
{
 int ADC_D ;

 clrscr();
 gotoxy(13,10);
 cout << "ADC_Data =";
 do
 {
  ADC_D =inportb(DATA_B);
  Dec_to_Bin(8,30,10,ADC_D);
  delay(500);
 }
 while (!kbhit()) ;
 return 0;
}
//*************************************************************************************************************************
void display_image(int slice_no,int LP,int LL)
{
 int FP = 0;
 int FL = 0;
 register unsigned col,row,color;
 unsigned ColSpan,RowSpan;
 unsigned long pos;
 int col1,col2,row1;
```

```
 ColSpan=LP-FP;
 RowSpan=LL-FL;
 pos = slice_no;
 col1 =0;
 col2 =ColSpan;
 row1 =0;
 for(row=0; row<RowSpan; row++)
 {
  for(col=0; col<ColSpan; col++)
  {
   color = *(buff+pos);
   color>>=2;
   if(floor(row/2) * 2 == row)
    /* even row */
     PutPixel256(col1+col,
                          row+row1,color);
   else
    /* odd row */
   PutPixel256(col2-col,
                          row+row1,color);
   pos = pos + slices;
  }
 }
 flushall();
 getch();
}
//*****************************************************************************************************************
void Save_ image(void)
{

 int no_of_slices;
 char d_image,s_raw;
 char string[25];
 char aux[30],aux1[30];
 no_of_slices = 0;
 do
 {
  // textmode(LASTMODE);
   strcpy(aux,"C:\\data\\");
   strcpy(aux1,fname);
   itoa(no_of_slices,string,10);
   strcat(aux1,string);
   strcat(aux,aux1);
   InitGraphics();
   Set256ColorMode();
   LoadGray64Palette();
   display_image(no_of_slices,sp,sl);
   flushall();
   getch();
   printf("\nDo you wish to see another slice ?");
   printf("\nPRESS Y to see and SPACEBAR to stop");
   flushall();
   d_image = getch();
   d_image = toupper(d_image);
   no_of_slices++;
   clrscr();
 }
 while((d_image == 'Y') && (no_of_slices < 15) && (d_image !=' '));
 printf("\nDo you wish to save raw data ? PRESS Y if YES");
 flushall();
 s_raw = getch();
 if( s_raw == 'Y' || s_raw =='y')
  SaveData();
 clrscr();
 getch();
 farfree((BYTE far *)buff);
 closegraph();
}
//*****************************************************************************************************************
void Real_t_scan(void)
{
  Scan();
  printf("\nPress any key to display slice by slice....");
  flushall();
  getch();
  image_display();
```

```
}
//************************************ I/O Board Initilization ***************************************************************
void Init_PC36AT(int i_o_par)
{
 outportb(CONTROL,i_o_par);
}
//*****************************************************************************************************************************
//************************************ Menu ***********************************************************************************
void MainMenu(void)
{
int    X1;//Y
int  choice,sure;
int    Quit1;
char   *str = " ";

outportb(DATA_A,0);
 clrscr();
 do
   {
            Quit1 = 0;
            InitGraphic();    /* graphics screen */
            X1 = getmaxx()/2;
            setpalette(15,11);
            settextstyle(4,HORIZ_DIR,3);
            settextjustify(CENTER_TEXT,TOP_TEXT);
            outtextxy(X1,25,"DATA ACQUISTION");
            settextstyle(3,HORIZ_DIR,2);
            setviewport(170,100,639,479,1);
            settextjustify(LEFT_TEXT,TOP_TEXT);
            do
            {
             clearviewport();
              outtextxy(0,0,"    A : Position Scanner");
              outtextxy(0,25,"   B : Scan Parameters");
              outtextxy(0,50,"   C : Scan");
              outtextxy(0,75,"   D : Dummy Data");
              outtextxy(0,100," R : Real Time Scan(DD)");
              outtextxy(0,125," S : Save Data");
              outtextxy(0,150," F : Demo");
              outtextxy(0,175," Q : Quit");
              flushall();
             outtextxy(0,220," Enter Your Choice :");
             do
             {
              flushall();
              choice = getch();
              choice = toupper(choice);
             }
             while ((choice != 65) && (choice != 66) && (choice != 67) && (choice != 68) && (choice != 82) && (choice != 83)
&& (choice != 70) && (choice != 81));
             strnset(str,choice,1);
             outtextxy(225,220,str);
             outtextxy(0,260," Are you sure ? (y/n):  ");
             do
             {
               flushall();
               sure = getch();
               sure = toupper(sure);
              }
              while((sure !=89) && (sure !=78));
              strnset(str,sure,1);
              outtextxy(240,260,str);
            }
            while (sure !=89);

            switch(choice)
            {
               case 'Q' : Quit1 = 1         ; break;
               case 'A' :closegraph(); Setup()          ; break;
               case 'B' :closegraph(); Scan_Parameters()  ; break;
               case 'C' :closegraph(); Scan()           ; break;
               case 'D' :closegraph(); Dummy_Data()       ; break;
               case 'R' :closegraph(); Real_t_scan()  ; break;
               case 'S' :closegraph(); Save_Data()        ; break;
               case 'F' :closegraph(); Demo()           ; break;
```

```
            } /* end of switch */

          }
          while (Quit1 == 0);
          closegraph();
}
//*****************************************************************************************************************
//****************************** Main Program ***********************************************************************
 int Data_Acq(void)
{
  Init_PC36AT(0x83);
  MainMenu();
  Init_PC36AT(0x80);
  outportb(DATA_A,0);
  outportb(DATA_A,0);
  outportb(DATA_A,0);
  fcloseall();
  farfree((BYTE far *)buff);
  Get_Msg("\nEnd of Data Acquistion....");
  return 0;
}
//****************************** End of Data Acquisition Program*****************************************************



//*****************************************************************************************************************
//
//                                    Image Processing Code
//                                      written in BC++
//                                           by
//                                     ALI AL-MEJRAD
//
//                                  Vers: 2.3  Last Update: 12/8/93   */
//*****************************************************************************************************************

#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <ctype.h>
#include <string.h>
#include <iostream.h>
#include <process.h>
#include <math.h>
#include <graphics.h>
#include "C:\ALI\others.h"
#include "C:\ALI\pcxfmt.h"
#include "C:\ALI\vga.h"
#include "C:\ALI\supfunc.h"
#include "C:\ALI\proces1.h"
#include "C:\ALI\geproc.h"
#include "C:\ALI\proces2.h"

extern void InitGraphic(void);
BYTE huge *TheImage;
BYTE huge *Buffer;
unsigned G_F ; /* controls generation of output files */
//*****************************************************************************************************************
//                                        Image Processing Functions
//*****************************************************************************************************************
 void IB(void)
 {
  unsigned scol,srow,sw,sh;
  short b_f;
  unsigned shft;
  char filename[50],InfileName[50],tempfn[50];
  char pfile;
  /* 1-Image Brightening */
  /* load the file into memory */
  clrscr();
  strcpy(filename,"C:\\data\\" );
  printf("\n\nPlease enter your filename with the extension only not the path");
 printf("\nFilename-> ");
 scanf("%s",InfileName);
 strcat(filename,InfileName);
 strcpy(tempfn,filename);
```

```
 printf("Do you wish to save %s after processing ?(Y/N)",filename);
 pfile = getch();
 pfile = toupper(pfile);
 if (pfile == 'Y')
    G_F = T;
 else
    G_F = F;
 do
 {
 clrscr();
 printf("\nthe file name --> %s",filename);
 printf("\n\nPlease enter the following:\n");
 printf("the first source col (<319)     --> ");
 scanf("%d",&scol);
 printf("the first source row  (<199)   --> ");
 scanf("%d",&srow);
 printf("the source width      (<320)   --> ");
 scanf("%d",&sw);
 printf("the source height     (<200)   --> ");
 scanf("%d",&sh);
 printf("the brightness factor (e.g:+17)  --> ");
 scanf("%hd",&b_f);
 }
 while((scol > 319) || (srow > 199) || (sw >320) || (sh > 200));
 if (ReadPCXFileToBuf(filename,&TheImage) != NoError)
  exit(1);
 //prepare the name of the origional slice with/without histogram
  strcat(filename,"H");
 /* display the image pointed at by TheImage */
 DisplayImageInBuf(TheImage,INIT_VGA,WAIT_KEY);
 DisplayHist(TheImage,scol,srow,sw,sh,185,0);
 if (G_F)
   Wr_File(filename,8,320,200,1,320);
 if((sh+100) > 199)
 {
  AdjImageBrightness(TheImage,b_f,scol,srow,sw,sh,0,0);
  shft = 0;
 }
 else
 {
 AdjImageBrightness(TheImage,b_f,scol,srow,sw,sh,0,100);
 shft = 100;
 }
 DisplayImageInBuf(TheImage,NOVGAINIT,WAIT_KEY);
 DisplayHist(TheImage,scol,srow+shft,sw,sh,185,100);
 if(G_F)
 {
 //prepare the name of the processed slice
 strcat(filename,"B");
 Wr_File(filename,8,320,200,1,320);
 }
 farfree((BYTE far *)TheImage);
 if (G_F)
 {
 if (ReadPCXFileToBuf(tempfn,&TheImage) != NoError)
    exit(1);
 strcat(tempfn,"B");
 AdjImageBrightness(TheImage,b_f,scol,srow,sw,sh,0,0);
 DisplayImageInBuf(TheImage,INIT_VGA,WAIT_KEY);
 Wr_File(tempfn,8,320,200,1,320);
 }
 farfree((BYTE far *)TheImage);
}
//*****************************************************************************************************************************
void IT(void)
{
 unsigned th_v,scol,srow,sw,sh;
 char filename[50],InfileName[50],tempfn[50];
 char pfile;

 /*  3 - Image Thresholding */
 /* load the PCX file into memory */
 clrscr();
 strcpy(filename,"C:\\data\\" );
 printf("\n\nPlease enter your filename with the extension only not the path");
 printf("\nFilename-> ");
```

```
 scanf("%s",InfileName);
 strcat(filename,InfileName);
 strcpy(tempfn,filename);
 printf("Do you wish to save %s after processing ?(Y/N)",filename);
 pfile = getch();
 pfile = toupper(pfile);
 if (pfile == 'Y')
   G_F = T;
 else
   G_F = F;
 do
 {
   clrscr();
   printf("\nthe file name --> %s",filename);
   printf("\n\nPlease enter the following:\n");
   printf("the first source col (<319)     --> ");
   scanf("%u",&scol);
   printf("the first source row  (<199)   --> ");
   scanf("%u",&srow);
   printf("the source width      (<320)   --> ");
   scanf("%u",&sw);
   printf("the source height     (<200)   --> ");
   scanf("%u",&sh);
    printf("the threshold value             --> ");
    scanf("%u",&th_v);
 }
 while((scol > 319) || (srow > 199) || (sw >320) || (sh > 200));
  if (ReadPCXFileToBuf (filename,&TheImage) != NoError)
     exit(1);

  DisplayImageInBuf(TheImage,INIT_VGA,WAIT_KEY);
 if((sh+100) > 199)
   ThresholdImage(TheImage,th_v,scol,srow,sw,sh,0,0);
 else
   ThresholdImage(TheImage,th_v,scol,srow,sw,sh,0,100);
 DisplayImageInBuf(TheImage,NOVGAINIT,WAIT_KEY);
 if(G_F)
 {
   //prepare the name of the processed slice
   strcat(filename,"T");
    Wr_File(filename,8,320,200,1,320);
 }
 farfree((BYTE far *)TheImage);
 if (G_F)
 {
  if (ReadPCXFileToBuf(tempfn,&TheImage) != NoError)
    exit(1);
 strcat(tempfn,"T");
 ThresholdImage(TheImage,th_v,scol,srow,sw,sh,0,0);
 DisplayImageInBuf(TheImage,INIT_VGA,WAIT_KEY);
 Wr_File(tempfn,8,320,200,1,320);
 }
 farfree((BYTE far *)TheImage);
}
//*****************************************************************************************************************************
 void CS(void)
 {
 unsigned th_v,scol,srow,sw,sh;
 unsigned shft;
 char filename[50],InfileName[50],tempfn[50];
 char pfile;
 /*  4 - Contrast Stretching */
  clrscr();
 //create directory
 strcpy(filename,"C:\\data\\" );
 printf("\n\nPlease enter your filename with the extension only not the path");
 printf("\nFilename-> ");
 scanf("%s",InfileName);
 strcat(filename,InfileName);
 strcpy(tempfn,filename);
 printf("Do you wish to save %s after processing ?(Y/N)",filename);
 pfile = getch();
 pfile = toupper(pfile);
 if (pfile == 'Y')
   G_F = T;
 else
```

```
   G_F = F;
 do
 {
  clrscr();
  printf("\nthe file name --> %s",filename);
  printf("\n\nPlease enter the following:\n");
  printf("the first source col (<319)     --> ");
 scanf("%u",&scol);
  printf("the first source row  (<199)   --> ");
 scanf("%u",&srow);
 printf("the source width      (<320)   --> ");
scanf("%u",&sw);
printf("the source height     (<200)   --> ");
scanf("%u",&sh);
printf("the threshold value             --> ");
scanf("%u",&th_v);
}
while((scol > 319) || (srow > 199) || (sw >320) || (sh > 200));
if (ReadPCXFileToBuf (filename,&TheImage) != NoError)
  exit(1);
//prepare the name of the processed slice
strcat(filename,"H");
/* display the image pointed at by TheImage */
DisplayImageInBuf(TheImage,INIT_VGA,WAIT_KEY);
DisplayHist(TheImage,scol,srow,sw,sh,185,0);
if (G_F)
  Wr_File(filename,8,320,200,1,320);
  //StretchImageContrast(TheImage,Histogram,th_v,scol,srow,sw,sh);
  if((sh+100) > 199)
  {
    StretchImageContrast(TheImage,Histogram,th_v,scol,srow,sw,sh,0,0);
   shft = 0;
  }
  else
  {
    StretchImageContrast(TheImage,Histogram,th_v,scol,srow,sw,sh,0,100);
   shft = 100;
  }
DisplayImageInBuf(TheImage,NOVGAINIT,WAIT_KEY);
DisplayHist(TheImage,scol,srow+shft,sw,sh,185,100);
if(G_F)
{
 //prepare the name of the processed slice
  strcat(filename,"S");
  Wr_File(filename,8,320,200,1,320);
}
farfree((BYTE far *)TheImage);
if (G_F)
{
 if (ReadPCXFileToBuf(tempfn,&TheImage) != NoError)
   exit(1);
strcat(tempfn,"S");
DisplayImageInBuf(TheImage,INIT_VGA,WAIT_KEY);
DisplayHist(TheImage,scol,srow,sw,sh,185,0);
StretchImageContrast(TheImage,Histogram,th_v,scol,srow,sw,sh,0,0);
DisplayImageInBuf(TheImage,NOVGAINIT,WAIT_KEY);
Wr_File(tempfn,8,320,200,1,320);
}
farfree((BYTE far *)TheImage);
}
//*****************************************************************************************************************************
void IM(void)
{
 char filename[50],InfileName[50],tempfn[50];
 unsigned scol,srow,sw,sh,dcol,drow;//,interpolate,dw,dh;
 float hsc,vsc;
 unsigned row,col;
 unsigned long off;
  char pfile;
 /* 6-Image Magnification */
 /* allocate far memory buffer for images */
 Buffer = (BYTE huge *)farcalloc(RASTERSIZE,(long)sizeof(BYTE));
 if (Buffer == NULL)
 {
  restorecrtmode();
  printf("Error Not enough memory for geometric operation\n");
```

```
    exit(ENoMemory);
  }
  clrscr();
  //create directory
  strcpy(filename,"C:\\data\\" );
  //read_filename(char *filename)
   printf("\n\nPlease enter your filename with the extension only not the path");
   printf("\nFilename-> ");
   scanf("%s",InfileName);
   printf("\nEnter ImageWidth :");
   scanf("%d",&i_w);
   printf("Enter ImageLength:");
   scanf("%d",&i_l);
   strcat(filename,InfileName);
   strcpy(tempfn,filename);
   printf("Do you wish to save %s after processing ?(Y/N)",filename);
   pfile = getch();
   pfile = toupper(pfile);
   if (pfile == 'Y')
     G_F = T;
   else
     G_F = F;
   clrscr();
   do
   {
    clrscr();
     printf("\nthe file name --> %s",filename);
     printf("\n\nPlease enter the following:\n");
     printf("the first source col (<319)     --> ");
    scanf("%u",&scol);
    printf("the first source row   (<199)   --> ");
    scanf("%u",&srow);
    printf("the source width       (<320)   --> ");
    scanf("%u",&sw);
    printf("the source height      (<200)   --> ");
   scanf("%u",&sh);
    printf("the horizontal scale            --> ");
   scanf("%f",&hsc);
   printf("the vertical scale               --> ");
   scanf("%f",&vsc);
   printf("the first destination col (<319) --> ");
   scanf("%u",&dcol);
   printf("the first destination row (<199) --> ");
   scanf("%u",&drow);
   //scanf("%d",&interpolate);
   if((scol >319) || (srow >199) || ((scol+sw) > 320) || ((srow+sh) >200)
              ||(dcol >319) || (drow >199) || ((dcol+(hsc*sw)) >320) ||((drow+(vsc*sh)) >200))
              {
              printf("\nOut of range error");
              printf("\nYou must satisfy the following relations:");
              printf("\n1- scol + swidth  < 320");
              printf("\n2- srow + shieght < 200");
              printf("\n3- dcol + (hscale*swidth)  < 320");
              printf("\n4- drow + (vscale*shieght) < 200");
              printf("\nPlease press any key to start entering the values again");
              getch();
              }
  }
  while((scol >319) || (srow >199) || ((scol+sw) > 320) || ((srow+sh) >200)
              ||(dcol >319) || (drow >199) || ((dcol+(hsc*sw)) >320) ||((drow+(vsc*sh)) >200));

if (ReadPCXFileToBuf (filename,&TheImage) != NoError)
  exit(ENoMemory);
DisplayImageInBuf(TheImage,INIT_VGA,WAIT_KEY);
/* Clear the output buffer to Blk. */
Clr_Image(Buffer,sw,MINROWS,320-sw,200,BLK);
Sc_Image(TheImage,scol,srow,sw,sh,hsc,vsc,Buffer,dcol,drow,T);
for ( row=drow ; row < 200 ; row++)
    for( col=dcol ; col < 320 ; col++)
   {
      off = row ;
      off *=320 ;
      off +=col ;
      PutPixel256(col,row,Buffer[off]);
   }
   getch();
```

```
   if(G_F)
   {
    strcat(filename,"M");
    Wr_File(filename,8,320,200,1,320);
   }
   /* Clear the output buffer to Blk. */
    Clr_Image(Buffer,sw,MINROWS,320-sw,200,BLK);
    Sc_Image(TheImage,scol,srow,sw,sh,hsc,vsc,Buffer,dcol,drow,T);
    for ( row=0 ; row < 200 ; row++)
     for( col=dcol ; col < 320 ; col++)
     {
      off = row ;
      off *=320 ;
      off +=col ;
      PutPixel256(col,row,Buffer[off]);
     }
    getch();
    if(G_F)
    {
     //prepare the name of the processed slice
    strcat(filename,"I");
    Wr_File(filename,8,320,200,1,320);
    }
     farfree((BYTE far *)TheImage);
     farfree((BYTE far *)Buffer);
    if (G_F)
    {
      /* allocate far memory buffer for images */
     Buffer = (BYTE huge *)farcalloc(RASTERSIZE,(long)sizeof(BYTE));
     if (Buffer == NULL)
     {
        restorecrtmode();
        printf("Error Not enough memory for geometric operation\n");
       exit(ENoMemory);
     }
     if (ReadPCXFileToBuf(tempfn,&TheImage) != NoError)
       exit(1);
    strcat(tempfn,"IM");
     Sc_Image(TheImage,scol,srow,sw,sh,hsc,vsc,Buffer,0,0,T);
     DisplayImageInBuf(Buffer,INIT_VGA,WAIT_KEY);
     Wr_File(tempfn,8,320,200,1,320);
     }
     farfree((BYTE far *)TheImage); /* free memory */
     farfree((BYTE far *)Buffer);
 }
//*****************************************************************************************************************
//                                  Convolution
//*****************************************************************************************************************
 void artest(void)
 {
 struct time t;
 unsigned long T1,T2,TDiff;
static double LP1[]=
{ 0.11111111, 0.11111111, 0.11111111,
  0.11111111, 0.11111111, 0.11111111,
  0.11111111, 0.11111111, 0.11111111 };

static short HP1[]=
{  -1, -1, -1,
           -1,  9, -1,
           -1, -1, -1 };

static short LAP2[]=
{  -1, -1, -1,
   -1,  8, -1,
   -1, -1, -1 };

static short BLUR[]=
{   1,  1,  1,  1,  1,
            1,  1,  1,  1,  1,
            1,  1,  1,  1,  1,
            1,  1,  1,  1,  1,
            1,  1,  1,  1,  1 };

BYTE huge *TheImage;
BYTE huge *TheOutImage;
```

```
unsigned scol,srow,sw,sh;
char filename[50],InfileName[50],tempfn[50];
char temp1[50],temp2[50],temp3[50],temp4[50],temp5[50];
char pfile;
clrscr();
//create directory
strcpy(filename,"C:\\data\\" );
printf("\n\nPlease enter your filename with the extension only not the path");
printf("\nFilename-> ");
scanf("%s",InfileName);
strcat(filename,InfileName);
strcpy(tempfn,filename);
printf("Do you wish to save %s after processing ?(Y/N)",filename);
pfile = getch();
pfile = toupper(pfile);
if (pfile == 'Y')
   G_F = T;
 else
   G_F = F;
  do
  {
  clrscr();
  printf("\nthe file name --> %s",filename);
  printf("\n\nPlease enter the following:\n");
  printf("the first source col (<319)     --> ");
  scanf("%d",&scol);
  printf("the first source row  (<199)   --> ");
  scanf("%d",&srow);
  printf("the source width      (<320)   --> ");
  scanf("%d",&sw);
  printf("the source height     (<200)   --> ");
  scanf("%d",&sh);

  }
  while((scol > 319) || (srow > 199) || (sw >320) || (sh > 200));
if (ReadPCXFileToBuf (filename,&TheImage) != NoError)
  exit(1);
DisplayImageInBuf(TheImage,INIT_VGA,WAIT_KEY);
if (RealConvolution(TheImage,scol,srow,sw,sh,
            LP1,3,3,0,F,&TheOutImage) == NoError)
{
  DisplayImageInBuf(TheOutImage,NOVGAINIT,WAIT_KEY);
  if (G_F)
  {
   strcpy(temp1,tempfn);
   strcat(temp1,"flp");
   Wr_File(temp1,8,320,200,1,320);
  }
farfree((BYTE far *)TheOutImage);
}
getch();
//*****************************************************************************************************************
//                          /* 2 - Sobel Edge Detection */
//*****************************************************************************************************************
getch();
DisplayImageInBuf(TheImage,NOVGAINIT,WAIT_KEY);
if (SobelEdgeDet(TheImage,scol,srow,sw,sh,
                          10,F,&TheOutImage) == NoError)
{
  DisplayImageInBuf(TheOutImage,NOVGAINIT,WAIT_KEY);
  if (G_F)
  {
    strcpy(temp4,tempfn);
    strcat(temp4,"ed");
    Wr_File(temp4,8,320,200,1,320);
  }
   farfree((BYTE far *)TheOutImage);
  }
  /* Do detection again this time with overlaying */
  DisplayImageInBuf(TheImage,NOVGAINIT,WAIT_KEY);
  if (SobelEdgeDet(TheImage,scol,srow,sw,sh,
            10,T,&TheOutImage) == NoError)
{
 DisplayImageInBuf(TheOutImage,NOVGAINIT,WAIT_KEY);
 if (G_F)
 {
```

```
  strcat(temp4,"o");
  Wr_File(temp4,8,320,200,1,320);
}
farfree((BYTE far *)TheOutImage);
}
//*****************************************************************************************************************
//                          /* 3 - Median Filtering */
//*****************************************************************************************************************
getch();
DisplayImageInBuf(TheImage,NOVGAINIT,WAIT_KEY);
gettime(&t);
T1 = ((unsigned long)  (t.ti_min * 60 + t.ti_sec)* 100) + t.ti_hund;
if (MedianFilter(TheImage,scol,srow,sw,sh,
                          3,3,&TheOutImage) == NoError)
{
gettime(&t);
T2 = ((unsigned long)  (t.ti_min * 60 + t.ti_sec)* 100) + t.ti_hund;
DisplayImageInBuf(TheOutImage,NOVGAINIT,WAIT_KEY);
if (G_F)
{
 strcpy(temp5,tempfn);
 strcat(temp5,"fm");
 Wr_File(temp5,8,320,200,1,320);
}
C_Image(TheOutImage,TheImage);
farfree((BYTE far *)TheOutImage);
}
 getch();
 farfree((BYTE far *)TheImage);
 restorecrtmode();
 TDiff = (unsigned long) (T2 - T1);
 printf("The Processing Time is : %lu \n",TDiff);
 printf("Press any key to go back to the main screen");
 flushall();
 getch();
}
//*****************************************************************************************************************
//                                    Main Function
//*****************************************************************************************************************
void Image_Proc(void)
{
 int FINISH;
 char choice,sure;
 char *str =" ";
 int X1;
  clrscr();
  do
  {
     FINISH = 0;
     InitGraphic();    /* graphics screen */
     X1 = getmaxx()/2;
     setpalette(15,12);
     settextstyle(4,HORIZ_DIR,3);
     settextjustify(CENTER_TEXT,TOP_TEXT);
      outtextxy(X1,25,"IMAGE PROCESSING");
     settextstyle(3,HORIZ_DIR,2);
     setviewport(170,100,639,479,1);
     settextjustify(LEFT_TEXT,TOP_TEXT);
     do
     {
      clearviewport();
       outtextxy(0,0,  " A : Image Brightness");
       outtextxy(0,50, " C : Image Thresholding");
       outtextxy(0,75, " D : Contrast Stretching");
       outtextxy(0,100," E : Filters");
       outtextxy(0,125," F : Image Magnification");
       outtextxy(0,150," Q : Quit");
       flushall();
       outtextxy(0,200," Enter Your Choice :");
       do
       {
         flushall();
        choice = getch();
        choice = toupper(choice);
        }
```

```
        while ((choice != 65) && (choice != 66) && (choice != 67) && (choice != 68) && (choice != 69) && (choice != 70) &&
(choice != 81)) ;
        //choice = toupper(choice);
       strnset(str,choice,1);
       outtextxy(225,200,str);
       outtextxy(0,240,"Are you sure ? (y/n):  ");
       do
       {
         flushall();
        sure = getch();
        sure = toupper(sure);
       }
       while((sure !=89) && (sure !=78));
      strnset(str,sure,1);
      outtextxy(240,240,str);
      }
      while (sure !=89);
     switch(choice)
     {
           case  'Q' : FINISH = 1              ; break;
           case  'A' :closegraph(); IB()       ; break;
          case  'C' :closegraph(); IT()        ; break;
          case  'D' :closegraph(); CS()        ; break;
          case  'E' :closegraph(); artest()    ; break;
          case  'F' :closegraph(); IM()        ; break;
    } /* end of switch */
    restorecrtmode();
    }
    while (FINISH == 0);
    farfree((BYTE far *)Buffer);
   farfree((BYTE far *)TheImage);
   closegraph();
}
//************************************ End of Image Processing Programs ********************************************


//*****************************************************************************************************************
//
//                                          Display Program Code
//                                            written in BC++
//                                                 by
//                                           ALI S ALMEJRAD
//                                        version:2       date:15/7/1993
//*****************************************************************************************************************
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <alloc.h>
#include <process.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#include "C:\ali\others.h"
#include "C:\ali\pcxfmt.h"
#include "C:\ali\vga.h"
#include "C:\ali\image_di.h"

/* Global Variables */
BYTE huge *Image;
unsigned FL,FP,LL,LP;
extern unsigned slices;

/* Functions */
extern void exit_prog(char *);

void DisplayData(BYTE huge *PictData,unsigned LP, unsigned LL,unsigned slice_no)
{
 int FP = 0;
 int FL = 0;
 register unsigned col,row,color;
 unsigned ColSpan,RowSpan;
 unsigned  long pos;
 int col1,col2,row1;
 ColSpan=LP-FP;
 RowSpan=LL-FL;
```

```
 pos =  66 * 160;
 col1 =0;
 col2 =ColSpan-1;
 row1 =0;
 for(col=0; col<ColSpan; col++)
 {
   for(row=0; row<RowSpan; row++)
   {
     color = *(PictData + 1 +row+pos);     //+(unsigned long) pos);
     color>>=2;
     if(floor(row/2) * 2 == row)
     /* even row */
      PutPixel256(col1+col,
              row+row1,color);
     else
     /* odd row */
      //PutPixel256(col2-col,
                          row+row1,color);
     pos = pos * slices;
   }
 }
 flushall();
 getch();
 }

//*****************************************************************************************************************************
//                                Main Program
//*****************************************************************************************************************************
void Image_Disp()
{
 unsigned long RasterSize;
 unsigned i_w,i_l;
 char InfileName[50],InFileName[50],filename[50];
 char string[25];
 char aux[25],aux1[25];
 char d_image;
 unsigned slice_no = 0;

 textmode(LASTMODE);
 clrscr();
 //create directory
 strcpy(filename,"C:\\data\\" );
 //read_filename(char *filename)
 printf("\n\nPlease enter your filename ONLY ");
 printf("\nFilename-> ");
 scanf("%s",InfileName);
 strcpy(InFileName,InfileName);
 strcat(InFileName,".dat");
 printf("\nEnter ImageWidth :");
 scanf("%d",&i_w);
 printf("Enter ImageLength:");
 scanf("%d",&i_l);
 printf("Enter No of Slices:");
 scanf("%d",&slices);
 strcat(filename,InFileName);
 RasterSize=i_w * i_l * slices;
 unsigned long Nbytes=0;
 unsigned long Mem_left = (unsigned long) farcoreleft();
 printf("\nThe difference between the highest allocated block in the far");
 printf("\nheap and the top of the far heap is : %lu bytes\n", Mem_left);
 getch();
 Nbytes = (unsigned long) RasterSize;
 if (Nbytes > Mem_left)
 {
  printf("sorry ALI there is no enough memory!\n");
  printf("the memory available: %lu bytes",farcoreleft());
 exit_prog("");
 }
 //allocate buff to store the data
 Image = (BYTE huge *) farcalloc( RasterSize,(unsigned long) sizeof(BYTE));
 printf("Allocating Image Buffer- RasterSize is % lu bytes \n\n", RasterSize);
 InitGraphics();
 printf("Reading The Image File (Raw Data) into memory \n");
 if(ReadRawImageFileToBuf(filename,i_w,i_l,slices,&Image) !=NoError)
  exit(1);
InitGraphics();
```

```
 Set256ColorMode();
 LoadGray64Palette();
do
{
 clrscr();
 printf("\nDo you wish to see slice no. %d ?",slice_no+1);
  printf("\nPRESS Y to see , N to see the others , SPACEBAR to stop and S to save");
  flushall();
  d_image = getch();
  d_image = toupper(d_image);
  slice_no++;
  if(d_image == 'N')
    continue;
  if(d_image == 'S')
  {
   InitGraphics();
   Set256ColorMode();
   LoadGray64Palette();
   DisplayData(Image,i_w,i_l,slice_no);
   strcpy(aux,"C:\\data\\");
   strcpy(aux1,InfileName);
   itoa(slice_no,string,10);
   strcat(aux1,string);
   strcat(aux,aux1);
   WritePCXFile(aux,8,320,200,1,320);
   closegraph();
  }
  else
  {
   InitGraphics();
   Set256ColorMode();
   LoadGray64Palette();
   DisplayData(Image,i_w,i_l,slice_no);
   closegraph();
  }
 }
 while(((d_image == 'Y') || (d_image == 'N') || (d_image == 'S')) && (slice_no < slices-1) && (d_image !=' '));
 restorecrtmode();
 printf("\n\nPress any key to terminate image display \n\n");
 flushall();
 getch();
 clrscr();
 farfree((BYTE far *)Image);
 closegraph();
}
/************************************ End of Display Program ***************************************************


//*************************************************************************************************************
//
//                                        Image Comparison Program Code
//                                                written in BC++
//                                                     by
//                                              ALI S ALMEJRAD
//                                        version:1.0       date:24/8/1993
//*************************************************************************************************************
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <alloc.h>
#include <process.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#include "C:\ali\others.h"
#include "C:\ali\pcxfmt.h"
#include "C:\ali\vga.h"
#include "C:\ali\supfunc.h"

/* Global Variables */

BYTE huge *Image;
BYTE huge *OutImage;
//*************************************************************************************************************
//                                              General Functions
```

```
//*****************************************************************************************************************
void exit_prog(char *msg)
{
 printf("\n%s", msg);
 textcolor(YELLOW);
 cprintf("\r\n\nPress any key to abort.....");
 getch();
 fcloseall();
 exit(1);
}

/*****************************************************************************************************************
//                                                Main Program
//*****************************************************************************************************************
void main()
{
 unsigned long RasterSize,off;
 unsigned scol,srow,sw,sh,dcol,drow;
 unsigned row,col;
 char InfileName[30];
 char InFileName[30];
 char OutFileName[30];
 char filename[30];
 char outfilename[30];
 textmode(LASTMODE);
 clrscr();
 strcpy(filename,"C:\\data\\" );
 strcpy(outfilename,"C:\\data\\");
 printf("\nPlease enter your filename ONLY ");
 printf("\nFilename of origional image-> ");
 scanf("%s",InfileName);
 printf("\nFilename of processed image-> ");
 scanf("%s",OutFileName);
 strcpy(InFileName,InfileName);
 printf("\nPlease enter the following:\n");
 printf("the first source col (<319)     --> ");
 scanf("%d",&scol);
 printf("the first source row  (<199)   --> ");
 scanf("%d",&srow);
 printf("the source width      (<320)   --> ");
 scanf("%d",&sw);
 printf("the source height     (<200)   --> ");
 scanf("%d",&sh);
 printf("the first dest col    (<319)   --> ");
 scanf("%d",&dcol);
 printf("the first dest row    (<199)   --> ");
 scanf("%d",&drow);
 strcat(filename,InFileName);
 strcat(outfilename,OutFileName);
 RasterSize=sw*sh;
 unsigned long Nbytes=0;
 unsigned long Mem_left = (unsigned long) farcoreleft();
 Nbytes =  (unsigned long) RasterSize;
 if (Nbytes > Mem_left)
 {
  printf("sorry ALI there is no enough memory!\n");
  printf("the memory available: %lu bytes",farcoreleft());
  exit_prog("");
 }
 //allocate buff to store the data
 Image = (BYTE huge *) farcalloc( RasterSize,(unsigned long) sizeof(BYTE));
 OutImage = (BYTE huge *) farcalloc( RasterSize,(unsigned long) sizeof(BYTE));
 printf("\n\nPress any kye to display the two images\n");
 printf("After display press any key to terminate.\n");
 getch();
 if( ReadPCXFileToBuf(filename,&Image) !=NoError)
  exit(1);
 if( ReadPCXFileToBuf(outfilename,&OutImage) !=NoError)
  exit(1);
 InitGraphics();
 Set256ColorMode();
 InstallPCXFilePalette();
 for ( row=0 ; row < 128 ; row++)
   for( col=0 ; col < 128 ; col++)
   {
    off = row ;
```

```
    off *=320 ;
    off +=col ;
    PutPixel256(col,row,Image[off]);
   }
  for ( row=0 ; row < 128 ; row++)
    for( col=0 ; col < 128 ; col++)
    {
    off = row ;
    off *=320 ;
    off +=col ;
    PutPixel256(col+dcol,row+drow,OutImage[off]);
   }
  flushall();
  getch();
  clrscr();
  restorecrtmode();
  printf("GOOD BYE ALI !\n\n");
  farfree((BYTE far *)Image);
  farfree((BYTE far *)OutImage);
  closegraph();
}
/********************************** End of Image Comparison Program ***********************************************

//*****************************************************************************************************************
//
//                                          Format  Program Code
//                                             written in BC++
//                                                   by
//                                            ALI S ALMEJRAD
//                                        version:2.0       date:8/12/1994
//*****************************************************************************************************************
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <alloc.h>
#include <process.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#include "C:\ali\others.h"
#include "C:\ali\pcxfmt.h"
#include "C:\ali\vga.h"
#include "C:\ali\supfunc.h"


/* Global Variables */

BYTE huge *Image1;
unsigned FL1,FP1,LL1,LP1;
extern unsigned slices;

void FormatData(BYTE huge *PictData,unsigned LP1, unsigned LL1,unsigned slice_no,unsigned f)
{
 int FP1 = 0;
 int FL1 = 0;
 register unsigned col,row,color;
 unsigned ColSpan,RowSpan;
 unsigned  long pos;
 int col1,col2,row1;
 ColSpan=LP1-FP1;
 RowSpan=LL1-FL1;
 pos =  slice_no;
 col1 =0;
 col2 =ColSpan-1;
 row1 =0;
 for(row=0; row<RowSpan; row++)
 {
            for(col=0; col<ColSpan; col++)
            {
             color = *(PictData + 1 +(unsigned long) pos );
             if(floor(row/2) * 2 == row)
              /* even row */
              {
            if(floor(row/f) * f == row)
                        PutPixel256(col1+col,row+row1,color);
```

```
                    else
                    PutPixel256(col1+col,row+row1,0);
            }
            else
             /* odd row */
            {
                    if(floor(row/f) * f == row)
                    PutPixel256(col2-col,row+row1,color);
                    else
                    PutPixel256(col2-col,row+row1,0);
            }
            pos = pos + slices;
            }
}
flushall();
getch();
}

/*****************************************************************************************************************************
//                                                  Main Program
//*****************************************************************************************************************************
void Image_Format(void)
{
 unsigned long RasterSize;
 unsigned i_w,i_l,f;
 char InfileName[50],InFileName[50],filename[50];
 char string[25];
 char aux[25],aux1[25];
 char d_image;
 unsigned slice_no = 0;
 textmode(LASTMODE);
 clrscr();
 strcpy(filename,"C:\\data\\" );
 printf("\nEnter No of lines to neglect-> ");
 scanf("%d",&f);
 printf("\n\nPlease enter your filename ONLY ");
 printf("\nFilename-> ");
 scanf("%s",InfileName);
 strcpy(InFileName,InfileName);
 strcat(InFileName,".dat");
 printf("\nEnter ImageWidth :");
 scanf("%d",&i_w);
 printf("Enter ImageLength:");
 scanf("%d",&i_l);
 printf("Enter No of Slices:");
 scanf("%d",&slices);
 strcat(filename,InFileName);
 RasterSize=i_w * i_l * slices;
 unsigned long Nbytes=0;
 unsigned long Mem_left = (unsigned long) farcoreleft();
 printf("\nThe difference between the highest allocated block in the far");
 printf("\nheap and the top of the far heap is : %lu bytes\n", Mem_left);
 getch();
 Nbytes = (unsigned long) RasterSize;
 if (Nbytes > Mem_left)
 {
            printf("sorry ALI there is no enough memory!\n");
            printf("the memory available: %lu bytes",farcoreleft());
            exit_prog("");
 }
 //allocate buff to store the data
 Image1 = (BYTE huge *) farcalloc( RasterSize,(unsigned long) sizeof(BYTE));
 printf("Allocating Image Buffer- RasterSize is % lu bytes \n\n", RasterSize);
 InitGraphics();
 printf("Reading The Image File (Raw Data) into memory \n");
 if(ReadRawImageFileToBuf(filename,i_w,i_l,slices,&Image1) !=NoError)
 exit(1);
 do
 {
            FormatData(Image1,i_w,i_l,slice_no,f);
            strcpy(aux,"C:\\data\\");
            strcpy(aux1,InfileName);
            strcat(aux1,"f");
            itoa(f,string,10);
            strcat(aux1,string);
            itoa(slice_no,string,10);
```

```
            strcat(aux1,string);
            strcat(aux,aux1);
  }
  while((slice_no < slices-1));
  restorecrtmode();
  printf("\n\nPress any key to terminate image display \n\n");
  flushall();
  getch();
  clrscr();
  farfree((BYTE far *)Image1);
}

/********************************** End of IFormat Program ***********************************************************

/* ********************************************************************************************************************/
/*    TDU_DSP2.s                                                                                                      */
/*    VER:4.0                                                                                                         */
/*    DATE:8/7/95                                                                                                     */
/*    PROGRAMMER:ALI ALMEJRAD                                                                                         */
/*    COMMENTS:This program uses INT1 and DSPLINK                                                                     */
/*          to collect 3D Ultrasound Images in                                                                        */
/*          conjunction with the other parts of                                                                       */
/*          TDU System                                                                                                */
/********************************************************************************************************************/
#define   LINKADDR    0x700000
#define   SRAMADDR    0x080000
#define   WAITCNT     0x000054
#define   PCFLAG      0x000058
#define   DSPFLAG     0x00005c
#define   SAMPCNT     0x000060
#define   FRAMECNT    0x000064
#define   CURRSRAM    0x000068
#define   LINE        0x00006C
#define   NLINES      0x000070

.=0x000000                  /* Reset Vector       */
                    r22e = 0x000010       /* IVTP at 0x000010    */
                    goto START            /* Jump to program     */
                    nop                   /* Latent Instruction  */


.=0x000010                  /* INT1 vector location */
                    goto ISR              /* Jump to ISR         */
                    nop                   /* Latent Instruction  */


.=0x000080
START:              r1e = 0xf80000      /*Internal sync for int2*/
                     r2 = 0
                     *r1 = r2
                     r3e = 0xfa0000
                     r1 = 0x70
                     *r3 = r1
                     r1e = 0                 /* Clear r1 */
                     *PCFLAG = r1e       /* Clear all flags */
                     *DSPFLAG = r1e
                      *WAITCNT = r1e
                     r9e = 0x0c0000
                     r4e = 0x080000
REPEAT:              r2e = 32766
                      r3 = 0x0000
CLRD:                *r4++ = r3
                     if(r2-- >= 0) goto CLRD  /* r2 must <= fffe */
                        nop
                     r9e - r4
                     if ( ne ) goto REPEAT
                       nop
                      r3e = 0
                      r1 = 0x800d         /* Int1 */
                      r9 = 1             /* Counter for wait */
                      r2e = 0
                      r18e = 0
                      pcw = r1
                      r17e = SRAMADDR
                      *CURRSRAM = r17e
```

```
                    r11e = 160   /* no of lines */
                    *NLINES = r11e
                    r10e = 2560  /* 160 lines * 16 frames  */
                      nop
                    r10e = r10 - 2    /* Fram counter - 2 */
                    *LINE = r10e
                    r2e = 0
WAIT:               r11e = *NLINES
                     nop
                    r11e - 0
                    if ( ne ) goto WAIT
                     nop
                     r3e = *PCFLAG       /* Set PCFLAG to 0 */
                     r11e = 160          /* nop */
                     nop
                    *NLINES = r11e
                    r16e = 0x7ffffe
                    r2e = r2 + 1         /* Increment wait counter */
                    r16e - r2
                    if ( ne ) goto MODIFY
                      nop
                     r2e = 0

MODIFY:             r3e - 0
                    if( ne ) goto IDLE
                      nop
JUMP:               goto WAIT
                      nop

IDLE:                r1 = 0x000d
                    pcw = r1                    /* End of data collection */
                    *DSPFLAG = r3e
                    r17e = *CURRSRAM                /* Set ACKFLAG from DSP to PC*/
                    nop
                    *CURRSRAM = r17e     /* Send the last addr to PC*/
                    r2e = *LINE
                    nop           /* only for checking */
                    *WAITCNT = r2e
                    goto IDLE         /* Loop for ever */
                    nop
/*****************************************************************************************************************/
/*                The  the Interrupt Service Routine                                                             */
/*****************************************************************************************************************/
ISR:
                    *R1LOC = r1e           /* Push registers onto software stack*/
                    *R8LOC = r8e
                    *R5LOC = r5e
                    *R6LOC = r6e
                    *R10LOC = r10e
                    *R18LOC = r18e

                    /* Registers must be set to values */
                    /* required by ISR */
                     r1e = 64    /*  *SAMPCNT  @21/9/95 */
                     r6e = *CURRSRAM
                     r10e = *LINE
                     r18e = *NLINES
                     r1e = r1 + 2          /* Read 2 bytes from encoder */
                     r8e = LINKADDR

CONTINUE:           r5 = *r8                /* read data from ADC */
                      nop
                     *r6++ = r5l /* store data */
                     r1e = r1 - 1
                     if ( ne ) goto CONTINUE
                      nop
                     if(r10-- >= 0) goto EndSub     /* Decrement Framecnt every frame */
                      nop
                     /* Does the main program wants to stop the data  acquisition?*/
                     r6e = SRAMADDR
                     r10e = 2558    /* [(160 L. * 16 Fr.) - 2]  */
                     nop

EndSub:             r18e = r18 - 1
                    *CURRSRAM = r6e
                    *NLINES = r18e
```

```
                    *LINE = r10e
                    r1e = *R1LOC          /* Pull registers from software stack*/
                    r8e = *R8LOC
                    r5e = *R5LOC
                    r6e = *R6LOC
                    r10e = *R10LOC
                    r18e = *R18LOC
                    ireturn             /* Return from ISR    */
                     nop
/*****************************************************************************************************************/
/*                              Software Stacks                                                                  */
/*****************************************************************************************************************/
 R1LOC:   2*nop
 R8LOC:   2*nop
 R5LOC:   2*nop
 R6LOC:   2*nop
 R10LOC:   2*nop
 R18LOC:   2*nop
 R17LOC:   2*nop
/********************************** End of the DSP Code ************************************************************/
```