Design Specification Document for the Ultrasound Library

Document Number: DRM124
Rev. 0, 9/2011

Contents

Chapter 1
Introduction

1.1 Introduction

This document explains the main structure and design of the ultrasound library. Each component can be used independently. You can use this feature for developing new software products.

The modules or components are listed and therefore can easily be identified.

Chapter 2
Decomposition

2.1 Decomposition Description

The ultrasound library is created from a single signal called beamforming, and an ultrasound 2D image in a gray scale. The process that the signal goes through is shown in Figure 2-1.

This is the algorithm flow chart.

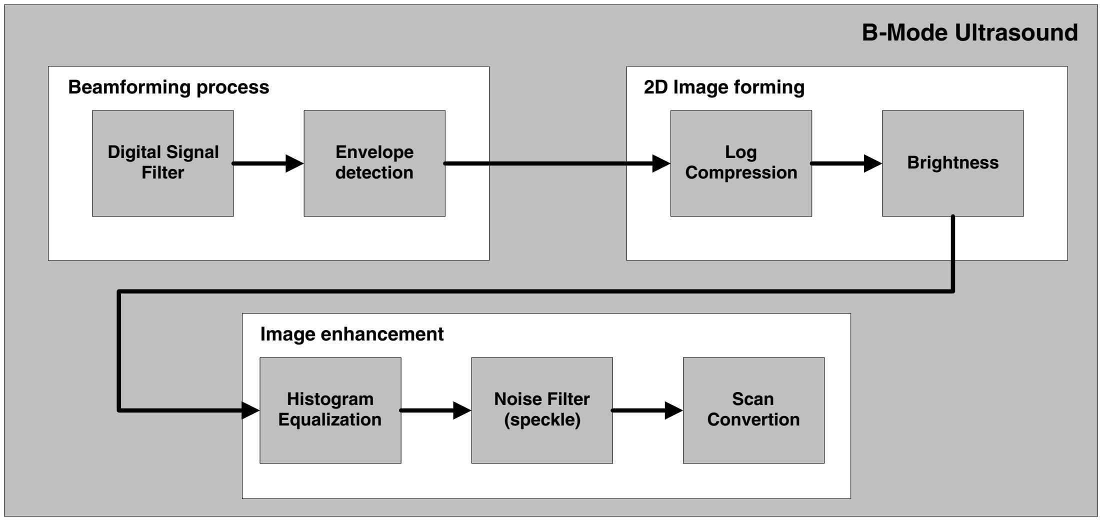


Figure 2-1. Flow chart for the main ultrasound library algorithm

2.1.1 Module decomposition

The ultrasound software library is divided into three main modules; each one represents a high level step in the principal algorithm.

The modules are:

- Beamforming process

- 2D image forming
- Image enhancement

The main-module decomposition for the ultrasound software library is displayed in Figure 2-2 . The principal modules can be easily identified. The purpose for breaking down the algorithm into these parts was to identify the independent parts and to encourage the reuse of the modules. Moreover, in case of improvement or library tuning, you can modify each of the modules independently.

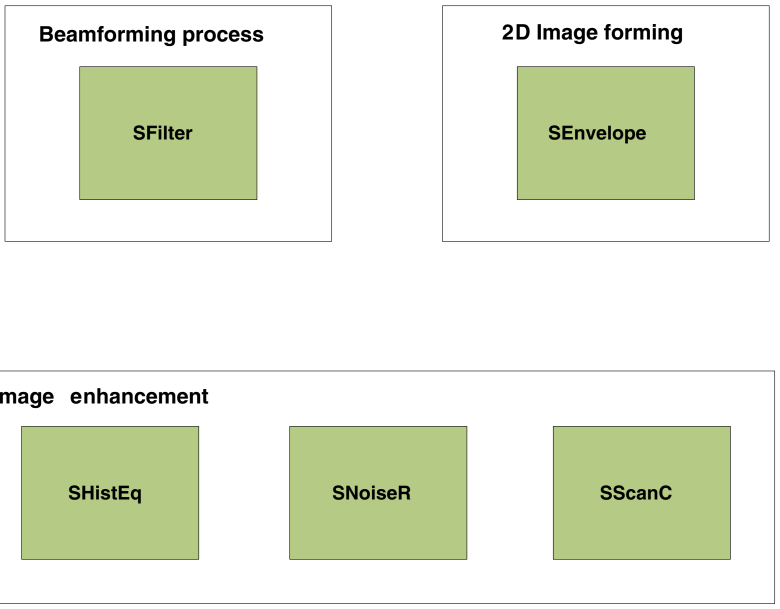


Figure 2-2. Ultrasound software library modules

A brief description of each is presented. For further details go to the userguide titled *Ultrasound Software Library Userguide* (document MEDIMGLIBUG)

2.1.1.1 Module 1— Beamforming Process

Use this module to process an input signal. You will get a filtered and demodulated signal to create an output gray scale image using the 2D image forming module.

2.1.1.2 Module 2—2D Image forming

Use the 2D image forming module after the beamforming process module to generate a demodulate image from the input signal. This module performs the signal demodulation, brightness representation, and log compression routines. The output will be a raw gray scale image.

2.1.1.3 Module 3—Image enhancement

This module is divided into three major components, SHistEq, SNoiseR, and SScanC, each one performs a simple tasks that transforms the input gray-scale raw image into an output processed gray-scale image with histogram equalization, noise reduction, and scan conversion algorithms applied.

2.2 Data decomposition

All the specialized data types are identified with the prefix “S”, this means that the data type is a structure and is composed of a collection of datum.

2.2.1 Data entity 1— SFilter

Purpose—To represent a filtered signal

Function—Instantiate a Filter (FIR, Hilbert) passed to the functions of the Beamforming Process module to filter the signal.

Subordinates—This data does not present subordinates.

2.2.2 Data entity 2— SEnvelope

Purpose—To represent a demodulated signal

Function—Used to save the datum after signal demodulation, this data is passed to the functions of the 2D image forming module to demodulate the signal

Subordinates—This data does not present subordinates.

2.2.3 Data entity 3— SHistEq

Purpose—To represent the histogram equalization algorithm and its parameters

Function—This structure contains the information for performing the histogram equalization algorithm, if the histogram equalization algorithm needs fine tuning check for the established parameters in this data type.

Subordinates—This data does not present subordinates.

2.2.4 Data entity 4— SNoiseR

Purpose—To represent the noise reduction algorithm and its parameters

Function—This structure contains the information for performing the noise reduction algorithm; in this case, the parameters are set for median filter noise reduction. If you want to change the established parameters check this data structure.

Subordinates—This data does not present subordinates.

2.2.5 Data entity 5— SScanC

Purpose—To represent the scan conversion algorithm and its parameters

Function—This structure contains the information for performing the scan conversion algorithm. The implemented algorithm is the bilinear scan conversion algorithm, it is set according to parameters in Figure 3. If you need to make some changes, review this structure.

Subordinates—This data does not present subordinates.

2.2.6 Data entity 6— SImage

Purpose—To represent output image

Function—This structure contains the information for memory allocation, size, and color depth. The structure is yielded with data access and processing methods.

Subordinates—This data does not present subordinates.

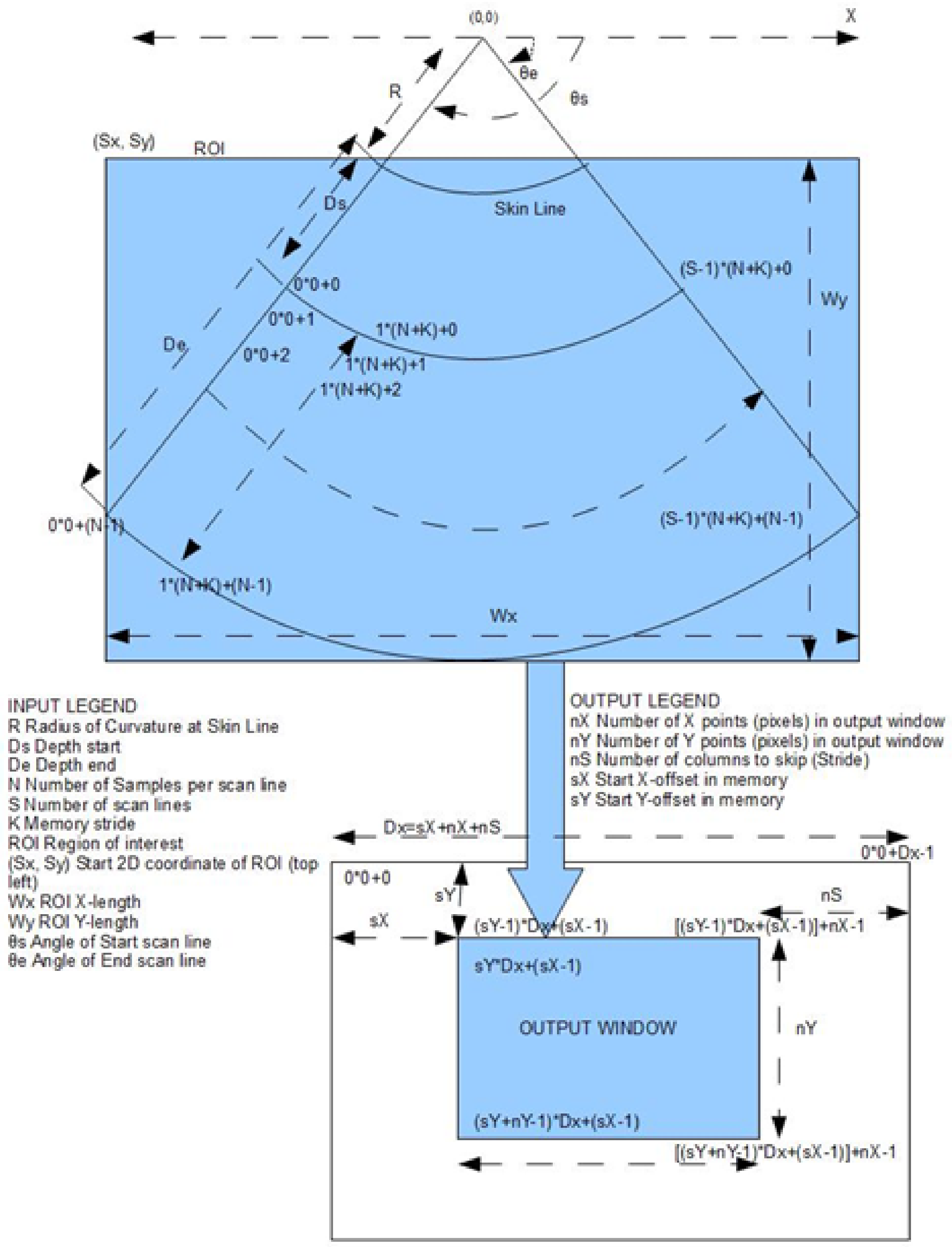


Figure 2-3. Scan conversion data-structure parameters

2.3 Concurrent process decomposition

The ultrasound software library uses the six cores of the MSC8156 board. First, an initialization phase is launched to set all the algorithm requirements. Second, a hard computing phase starts; its duty is to loop in the generation of gray-scale images going through the complete algorithm.

2.3.1 Process 1— initUltrasoundLibrary

Purpose—To initialize the ultrasound library, allocate all the required memory, and pre-calculate all the values for the algorithms.

Function—This process is called by the main function in the pre-computing phase. Its purpose is to calculate all the common data for the algorithms, therefore there is no need to re-calculate them when the hard work load has begun.

Subordinates—runUltrasoundLibraryOtherCore and runUltrasoundLibraryMasterCore

2.3.2 Process 2 — runUltrasoundLibraryOtherCore

Purpose—To execute the complete algorithm, from a beamforming signal, follow the complete algorithm and obtain a gray-scale image.

Function—This process is repeated for each input signal. It is loaded and executed in all the cores except the master.

Subordinates—This process does not present subordinates.

2.3.3 Process 3 — runUltrasoundLibraryMasterCore

Purpose—To synchronize the cores to execute the complete algorithm

Function—This process is repeated for each input signal. It is loaded and executed only in the master core.

Subordinates—This process does not present subordinates.

Chapter 3
Dependency

3.1 Dependency Description

The components can be used independently. Generally, each has its own data hence they do not present high coupling. The only one shared structure is called SImage, this is just used in the 2D Image forming and image enhancement modules.

3.1.1 Dependencies

The dependencies among the modules are illustrated in Figure 3-1. It displays the only one dependency given by the SImage structure which is filled with raw data in the 2D image processing module and then processed by the image enhanced module to generate the enhanced gray-scale output image.

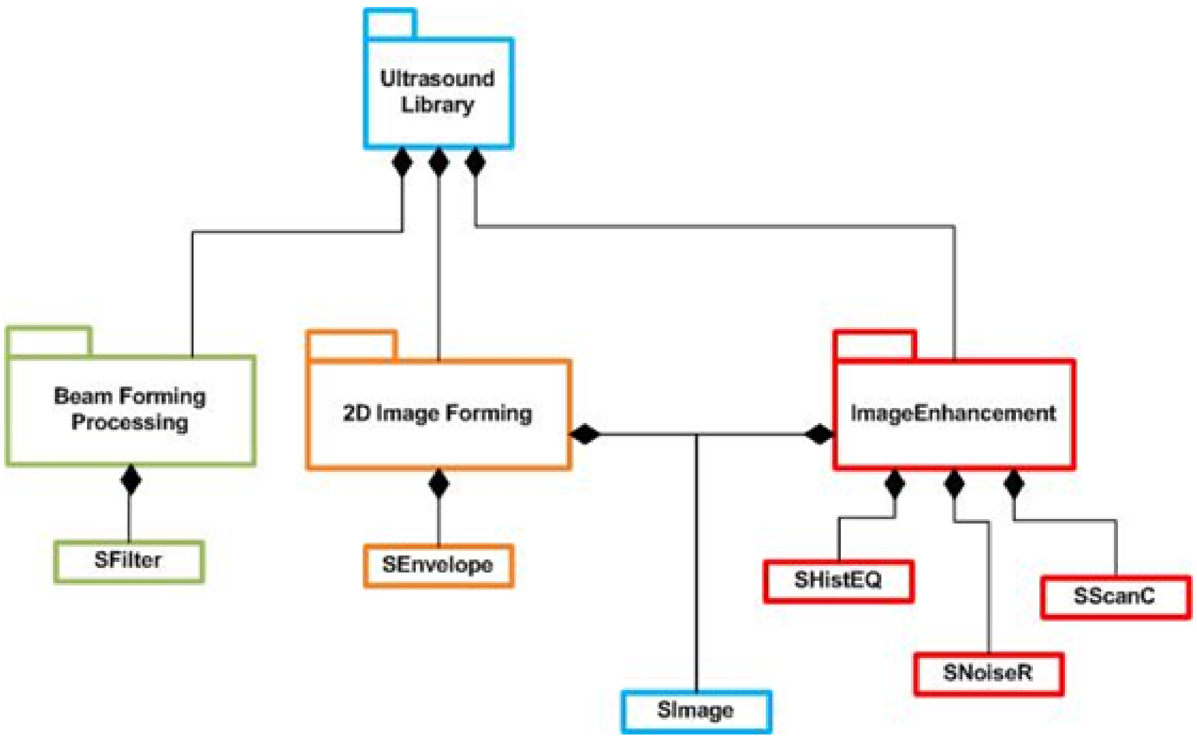


Figure 3-1. Modules dependencies

3.1.1.1 Inter-module dependencies

The modules do not have dependencies, both of them work with just one data structure. The image enhancement module, works with three data structures; however, they can work individually. They are a data type dependency (check that section).

3.1.1.2 Inter-process dependencies

If present, this section must describe the dependencies between various threads or contexts of execution.

3.1.1.3 Data Dependencies

The three elements in the image enhancement module need the common data type SImage. The data type is used for reading and storing the resulting datum.

3.2 Resources

Resources are elements used by the entity that are external to the actual design, but are needed by the entity to perform its specified function.

3.2.1 Hardware resources

The only requirement is an MSC8156 board to use or test the ultrasound library.

The following hardware resources on the board are necessary:

- Free access to all 6 six cores
- Around free 20 Mb in DDR0 memory
- Approximately 512 Kb in M3 memory for process initialization

3.2.2 Software resources

It is necessary to have the following software tools:

- CW version 10.1.5 or greater.
- It is mandatory to use SMART OS when using your library because of the message passing among the cores.
- FIR Freescale Kernel

For more details on specific software requirements, please see the Ultrasound Software Manual.

Chapter 4
Interface

4.1 Interface Description

Refer to the userguide Ultrasound Software Library Userguide (document MEDIMGLIBUG).

Chapter 5
Detail

5.1 Detail Description

This description contains the details needed by programmers prior to implementation and for producing unit test plans.

To know the detail description and how to use each module separately refer to the use case documents.

5.1.1 Design considerations

The design goal was to develop independent units that could be reused in the design and development of software systems that are related to ultrasound imaging.

The components are slightly coupled and do not depend in other elements, just their own data.

5.1.1.1 Assumptions

All the components have an initialization phase, refer to the userguide Ultrasound Software Library Userguide.

5.1.1.2 Development methods

All the functions are highly coupled to the hardware design of the MSC8156 board.

The implemented methods are described:

5.1.1.2.1 Fir Filter

A FIR Filter was implemented. A given desired ideal frequency response, say D(w), being periodic in w with period 2pi, need to only be specified over one complete Nyquist interval -pi<w<pi. The corresponding impulse response say d(k), is related to D(w) by the DFT and the inverse DFT relationships:

eqn1

$$D(\omega)=\sum_{k=-\infty}^{\infty} d(k)e^{-j\omega k} \leftrightarrow d(k)=\int_{-\pi}^{\pi} D(\omega)e^{j\omega k}\frac{d\omega}{2\pi}$$

So for example if you want a lowpass filter, this is

eqn2

$$D(\omega = \begin{cases} 1, \text{if} -\omega_c \leq \omega \leq \omega_c \\ 0, \text{if} -\pi \leq \omega < -\omega \text{or} \omega_c < \omega_c \leq \pi \end{cases}$$

After applying the DFT:

eqn3

$$d(k)=\int_{-\omega_c}^{\omega_c} 1 * e^{j\omega k}\frac{d\omega}{2\pi} = \frac{e^{j\omega_c k} - e^{-j\omega_c k}}{2\pi jk}$$

Rewriting the previous expression

eqn4

$$d\ k) = \frac{\sin(\omega_c k}{\pi k}, -\infty < k < \infty$$

For computational purposes the value of k=0 in the previous equation will be taken as

eqn5

$$d\ (0) = \frac{\omega_c}{\pi}$$

In a similar way there is an expression for the band pass filter (Orfanidis, 1996).

eqn6

$$d(k) = \frac{\sin(\omega_b k)}{\pi k}, -\infty < k < \infty$$

To implement a band pass filter in the time domain the frequencies wa and wb is needed, and the size of the filter. The size is important because there is a compromise between the precession of the filter and the computational cost of applying it.

5.1.1.2.2 Log compression

After the ultrasound image from the RF signal is obtained, you must filter it and apply the envelope technique. After this, it can be plotted. It would look like Figure 5-1[1].

1. This Image is courtesy of "Fundamentals of medical Imaging", Paul Suetens

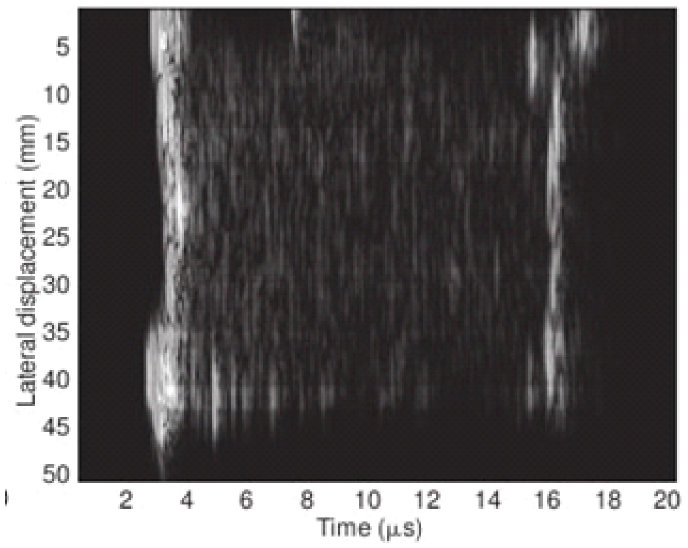


Figure 5-1. Image example

The reason for this kind of image, is the difference in amplitude between the specular and the scatter reflection, yielding a large dynamic range. To overcome this problem, a suitable gray level transform can be applied. Typically, a logarithmic function is used. The log-compressed version of the previous image can be seen in the following image. The scatter or speckle can now easily be perceived (Suetens, 2002).

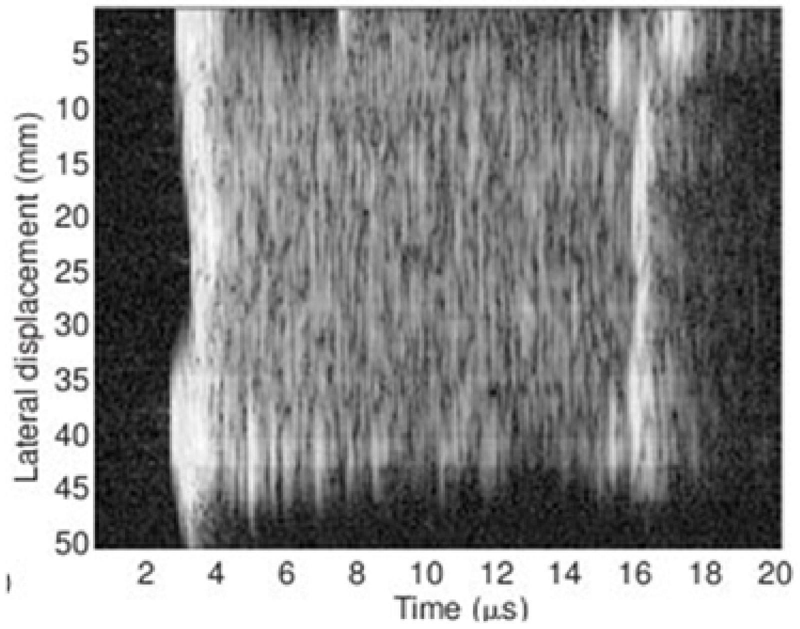


Figure 5-2. Image example

The signal log compression is obtained by applying the next equation to the signal S.

$$S' = D * \log(S) + Gain$$

The parameters D and Gain are estimated to have a dynamic range that fits in the maximum dynamic range of the human eye 30 dB, and is usually used to adjust the brightness.

5.1.1.2.3 Histogram equalization

One way for improving image contrast is the histogram equalization, which makes use of the image's histogram. The mathematical definition for a histogram is a function mi that counts the number of observations that fall into each of the disjoint categories, in this case, the gray scale used for the images. The technique used is the cumulative histogram equalization. In this section the general idea for the cumulative histogram equalization is described.

Here are the steps to implement this algorithm.

- Get histogram for the image
- Calculate the cumulative distribution function histogram
- Get the new values through the general histogram equalization formula
- Assign new values for each gray value in the image

The following block diagram shows the implementation of the algorithm mentioned above, using MATLAB.

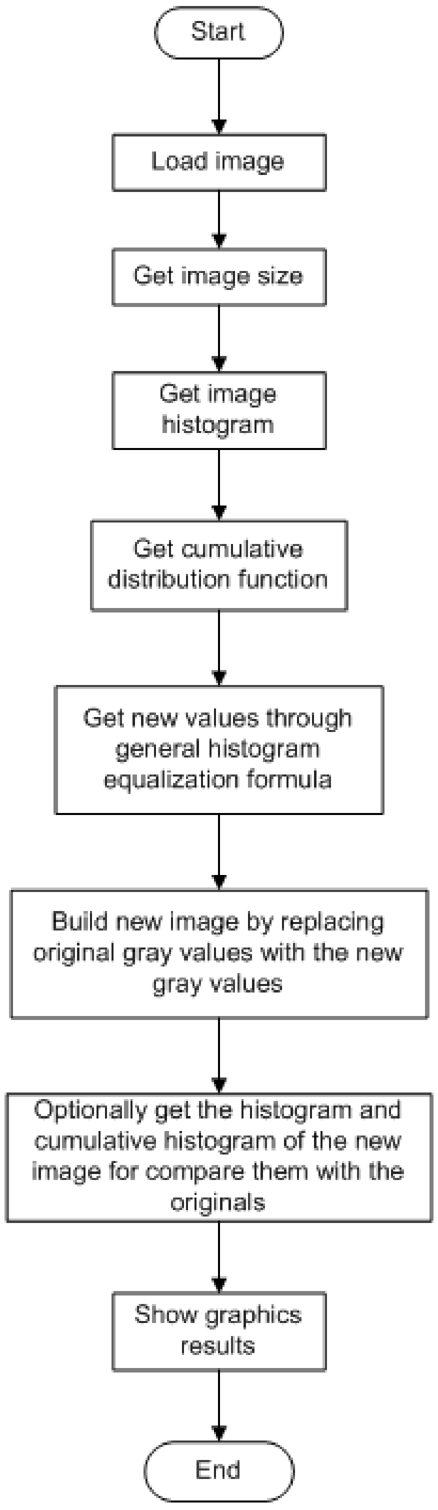

Start
Load image
Get image size
Get image histogram
Get cumulative distribution function
Get new values through general histogram equalization formula
Build new image by replacing original gray values with the new gray values
Optionally get the histogram and cumulative histogram of the new image for compare them with the originals
Show graphics results
End

Figure 5-3. Block diagram

The cumulative distribution function is described in the next formula

eqn7

$$\mathrm{cdf}(x) = \sum_{x_i \le x} h(x_i)$$

Where x is a gray value and h is the image histogram. The cumulative distribution function for each gray tone is calculated by the next code.

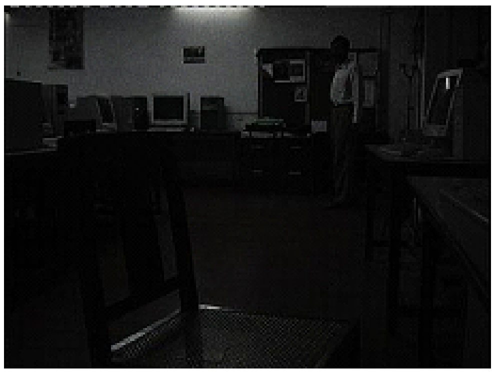
The general histogram equalization formula is:

eqn8

$$\mathrm{eh}(i) = \mathrm{round}\left(\frac{\mathrm{cdf}(i) - \mathrm{cdf}_{\min}}{(\mathrm{MxN}) - \mathrm{cdf}_{\min}} x(L-1)\right)$$

Where cdfmin is the minimum value of the cumulative distribution function, M x N are the image’s number of columns and rows and L is the number of gray levels used (in most cases 256).

The next figures are the result of applying this algorithm, in figure 8, the image is darker and its histogram is shown. Figure 9 shows the same image processed with histogram equalization.



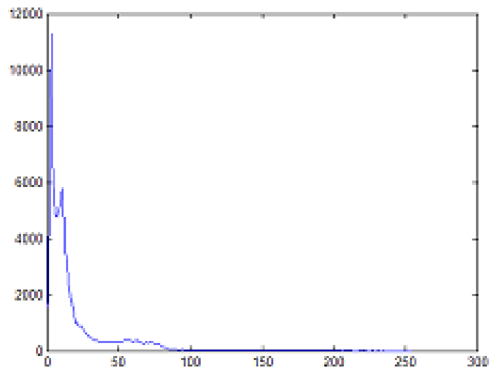


Figure 5-4. Dark image and its histogram

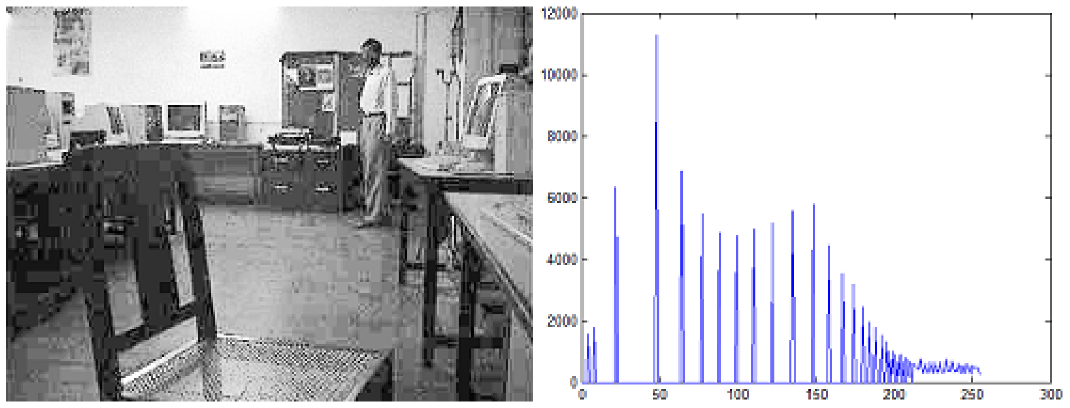


Figure 5-5. Image processed and its new histogram

5.1.1.2.4 Median filter for noise reduction

The median filter is a sliding-window spatial filter. It replaces the center value in the window with the median of all the pixel values in the window. The median filter is an effective method that can suppress isolated noise without blurring sharp edges. Specifically, the median filter replaces a pixel by the median of all pixels in the area, (expressed below).

eqn 9

$$y(m,n) = \text{median}\{x(i,j),(i,j)\} \in W$$

Where w represents a neighborhood centered around location (m,n) in the image.

5.1.1.2.5 Bilinear scan conversion

The scan conversion algorithm is divided in two principal parts; the first, is the transformation of Cartesian coordinates to polar and afterwards the bilinear interpolation is executed.

To estimate the coordinate transformation, the following equations are used.

eqn 10

$$\theta = a\tan2\left(\frac{y}{x}\right)$$

eqn11

$$r = \sqrt{x^2 + y^2}$$

Where:

θ— Pixel angle in polar coordinates

r—Pixel length in polar coordinates

x and y—Pixel position in Cartesian coordinates

After coordinates have been transformed to the interpolar form, the algorithm looks for the nearest angle and the length of each pixel of the image to perform the bilinear interpolation

Bilinear interpolation is executed using four data points from adjacent scan lines to compute the value for the needed pixel. This method does not require over-sampled scan lines. But it does require more computation than the linear interpolation method. The next figure shows the relationship between the four data points and the pixel. To obtain the pixel value, three linear interpolations must be performed.

Two data points are interpolated between S1 and S2 and between S3 and S4. Then the pixel value is calculated by interpolating between the two new data points.

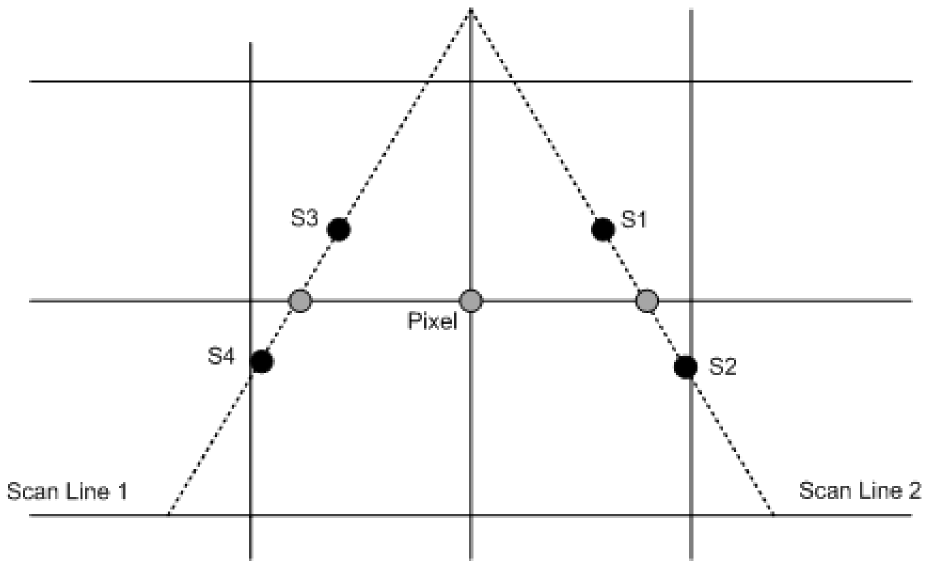


Figure 5-6. Example of Scan Conversion

5.1.1.2.6 Envelope detection

There are two approaches, the first one uses the Hilbert transform and the other uses a complex rotator.

5.1.1.2.6.1 Envelope detection using the Hilbert transform

First the algorithm is explained. Second, the two ways of estimating the Hilbert transform are also explained.

The algorithm is explained in three steps:

1. Given a signal S1 with a carrier frequency
2. Estimate the signal Hilbert transform of S1, storing it in S2. There are now two signals, S1 and S2.
3. Estimate the norm of the complex signal S1 + iS2, this is H = sqrt(S12 + S22)

To estimate the Hilbert transform and have S1 and S2 there are two possible ways.

This can be seen in the following diagram:

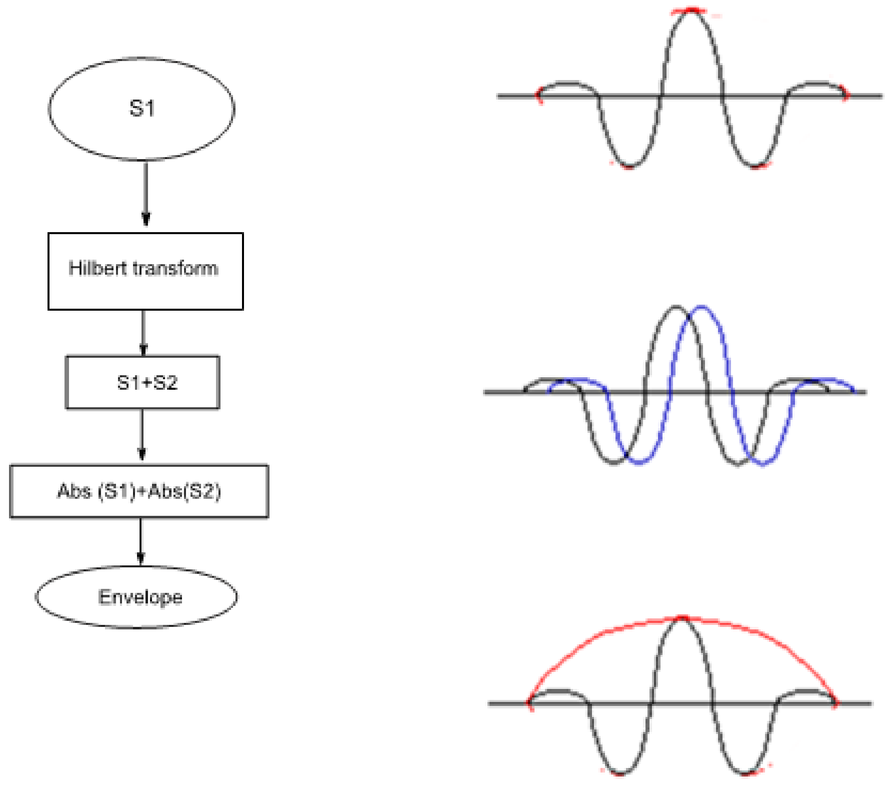


Figure 5-7. Envelope detection

5.1.1.2.6.2 Hilbert transform

The Hilbert transform can be understood as a 90 shift of the signal in the frequency domain, this can be executed in the frequency domain and in the time domain.

In the frequency domain:

1. It calculates the FFT of the input sequence, storing the result in a vector x.

2. It creates a vector h whose elements h(i) have the values:
   - 1 for i = 1, (n/2)+1
   - 2 for i = 2, 3, ... , (n/2)
   - 0 for i = (n/2)+2, ... , n
3. It calculates the Element-Wise product of x and h.
4. It calculates the inverse FFT of the sequence obtained in step 3 and returns the first n elements of the result.

The resulting signal is a complex signal that its real part will play the paper of S1 and the complex part will be S2.

Hilbert transform in the time domain can be executed as follows:

1. Estimate the following filter to the original signal

   eqn12

$$H\left(n\right)=\begin{cases}\frac{2}{n^{*}\pi}\sin^{2}\frac{n\pi}{s}\,\mathrm{if}\,n\neq 0\\ 0\,\mathrm{if}\,n=0\end{cases}$$

2. Apply the filter, convoluting the signal with the filter

Envelope detection using a complex rotator

There are ways to find the envelope signal, a complex rotator in the baseband followed by a low pass filter is one of them, this method requires knowing the operating center frequency and possibly tracked for changes. The advantage is that it is easier to implement.

5.1.1.3 Performance aspects

You can develop new methods and add them to the ultrasound library, pay close attention to the functions interface. It must remain in the same order to be reusable. Using techniques as function overloading can be helpful for the scalable features of the ultrasound library.

5.1.2 Module Description

The following paragraphs describe each of the modules used in this library.

5.1.2.1 Module 1—Beamforming Processing

Use this module to process an input signal, you will obtain a filtered and demodulated signal to create an output gray scale image using the 2D image forming module.

5.1.2.2 Module 2— 2D Image Forming

Use the 2D image forming module after the beamforming process module to generate an image from the input signal. This module performs the signal demodulation, brightness representation, and log compression routines. The output is a raw gray scale image.

5.1.2.3 Module 3— Image Enhancement

This module is divided into three major components, SHistEq, SNoiseR, and SScanC, each one is in charge of performing tasks to transform the input gray-scale raw image into an output processed gray-scale image with histogram equalization, noise reduction, and scan conversion algorithms applied.

5.1.3 Processing description

You can look for the description of the module process in the case use document.

5.1.4 Data description

The data description can be found in userguide Ultrasound Software Library Userguide (document MEDIMGLIBUG).

5.1.5 Component and unit test

Refer to the Test Plan Document.

5.1.6 Glossary

Design entity—A design element (component) that is structural and functional from other elements, and that is separately named and referenced.

Design entities result from a decomposition of the software component requirements. The objective is to divide the component into separate subcomponents that can be considered, implemented, changed, and tested with minimal effect on other entities.

Entities can exist as a component, subcomponents, data, modules, programs, and processes.

Entity attribute—A named characteristic or property of a design entity. It provides a statement of fact about the entity.

Design entity type—An attribute that describes the kind, or nature, of a design entity.

Design entity purpose—An attribute that describes the rationale for the creation of a design entity (like functional or performance requirements that led to the creation of that entity).

Design entity function—An attribute that describes what a design entity does.

Design entity subordinates—An attribute that identifies parent or child relationships between a design entity and its composing entities.

Software component—A unit of composition with a contractually specified interface and explicit context dependencies.

Appendix A References

This section provides references to the resources used to help write this document or relate to it.

Reference Item	Revision	Description
IEEE Std 1016-1998	23 Sep1998	IEEE Recommended Practice for Software Design Descriptions
IEEE Std 610.12-1990	28 Sep1990	IEEE Standard Glossary of Software Engineering Terminology

Appendix B
Version Tracking

Date	Revision Level	Description	Author
26 September 2011	0	Initial Release	Jose Fernandez

Document Number: DRM124
Rev. 0, 9/2011