Joseph Wakooli

Designing an Analysis Tool for Digital Signal Processing

Helsinki Metropolia University of Applied Sciences
Bachelor of Engineering
Information Technology
Thesis
30 May 2012

Metropolia

Abstract

Author(s) Title Number of Pages Date	Joseph Wakooli Designing an analysis tool for digital signal processing 43 pages + 5 appendices 5 May 2012
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Embedded Engineering
Instructor(s)	Dr. Antti Piironen, Head of Department, Information Technology

The goal of the project was to design an innovative analysis tool for digital signal processing that acquires different types of signals from different types of hardware as well as analyzes these signals. The project was carried out by developing software using the Matlab programming environment. A small user friendly graphical user interface was designed.

The software captures samples from the required hardware devices as well as analyzes the samples from either files or through the stream. The types of signals that this software analyzes include data stored in files, images, audio samples and video frames. It analyzes the samples using the Fast Fourier Transform approach. The software displays both linearity and the logarithm of the spectrum for all signal types, and the spectrogram for data and audio samples. This software also finds the normalized and phase-only correlation for any two signals. In addition, the software determines the distance that the sound pulse travels to the far end of the obstacle.

The results obtained from the project were convincing and have been applied in many fields of electronics that are used today. The project can be improved with the help of better programming tools and methods.

Keywords	Digital signal processing, Fast Fourier Transform, spectrum, spectrogram, correlation, data, audio, image, video, stream

Contents

1 Introduction

The goal of the project was to design an innovative analysis tool for Digital Signal Processing (DSP). The tool is mostly for acquiring different types of signals from the required hardware devices as well as analyzing signals. This project was proposed to me and to an exchange student called Hayashi Kazuya from the Tokyo National College of Technology, Department of Electronic Engineering, Japan, by Dr Antti Piironen and an exchange lecturer, Professor Hiroyuki Aoki from the Tokyo National College of Technology, Department of Electronic Engineering. The tool was meant to be a small graphical user interface that captures audio samples through the microphone, image files and video frames via the webcam.

DSP is a wide concept which can be broken down into small pieces. The term digital means dealing with the binary values 0's and 1's while the term signal refers to any time-varying or spatial-varying quantity. An example of a time varying quantity is speech whereas a photograph is an example of a space-varying quantity. An example of signal processing is analyzing a recorded voice to determine its pitch or manipulating a photograph by adjusting its colours.

Signal processing may involve operation on or analysis of signals in either discrete or continuous time. Signals of interest can include sound, images, time varying measurement values and sensor data. These signals are either analog or digital electrical representations of time-varying or spatial-varying physical quantities. This DSP analysis tool provides a good understanding of the applications of the DSP in the real world.

The report presents detailed explanation about the theoretical background of key concepts in the field of DSP and definitions on which the project was based. The materials and tools that were used to develop the project as well as the detailed procedure in which the project was conducted are explained in the methods and materials section. The outcomes of the project are then presented in the results section and the interpretations of the project are presented in the discussion section and thereafter conclusions are drawn in the last section.

2 Theoretical background

Signals play an important role in the day today life since they carry information which can be conveyed, displayed, or manipulated. Examples of such signals include raw data from files, sound, speech, video frames, images and radar signals. Majority of the signals people encounter in their daily life are analog which means that they vary continuously with time. DSP signals are often derived from sampling analog signals at regular intervals. (Ashok 2007, 1.) Therefore, DSP results in the digital representation of signals with the help of processors to analyze, modify or extract data from the analog signals. However, digital signals are often processed to remove interference from a signal, obtain a spectrum of data or to transform the signal into a more suitable form.

DSP is a technique used to analyze various signals and it involves electronic processing of signals. It finds its application in various areas ranging from broadcasting to medicine. Examples of areas where DSP is applied include broadcasting, instrumentation, military, control, telecommunication, automatic control, navigation and biomedical applications. The main purpose for DSP is to measure, filter and compress the continuous real world analog signals. DSP usually begins with converting the analog signal to digital form by sampling using a device known as analog to digital converter. (Emmanuel et al. 2002, 2.)

DSP has been one of the fastest growing fields in the area of electronics that is used today to carry information in digital form. Key applications areas on how DSP is applied include:

- pattern recognition
- image enhancement
- spectrum analysis
- noise reduction
- speech recognition
- text to speech
- sonar processing
- radar processing
- video conferencing

- echo cancellation
- patient monitoring
- servo control
- scanners
- digital television
- digital cameras
- digital cellular mobile phones

DSP is used in any area where information is handled in digital form. (Emmanuel et al. 2002, 1.) DSP is performed based on different operations and stages. Key DSP operations include convolution, correlation, filtering, transformation and modulation. The digital processing of an analog signal requires the use of an analog-to-digital converter (ADC) for sampling the analog signal prior to processing and a digital-to-analog converter (DAC) to convert the processed signal back to analog form. (Ashok 2007, 3.)

2.1 Analysis of still data

The term data refers to any distinct pieces of information usually formatted in an organized way. Data can exist in a variety of forms, for example as numbers, text on pieces of paper, bits and bytes stored in electronic memory, and facts stored in a person's mind. The major purpose for storing data is for future use. Data can be stored in electronic memory in places such as files, and databases.

These storage places are usually located on devices such as computers and mobile phones. The data stored in electronic memory can be acquired from devices or be manually derived by a person before it is stored. Both data acquired directly from a person or located in a storage place can be analyzed using special tools for special purposes.

2.2 Analysis of audio samples

An audio signal is a sound within the acoustic range hearable by humans. It is usually in the form of an electrical signal. An audio signal is an example of a one-dimensional signal where the independent variable is time (Sanjit 2006, 1). This section involves

discussing how audio data can be presented in different views and what this information shows as well as the practical use of this information once it has been obtained and understood. Audio analysis therefore refers to the extraction of information and meaning from audio signals for analysis.

In computer systems, an audio signal is a sound system that comes with or can be added to a computer. An audio signal can be stored as a file and this file contains a record of captured sound which can be played back in the future time. These audio samples or files are in the form of a vector array of values organized in a particular way. Examples of formats in which audio files can be stored include Waveform Audio (.wav), Windows Media Audio (.wma) and Sun Audio (.au).

Sound is a sequence of naturally occurring analog signals also known as waves of pressure that propagate through any media such as solid, liquid and gas. It is converted into digital samples by an audio card using a module known as ADC. The signal to the ADC is continuously sampled at a certain rate known as sampling rate, and the ADC presents a new sample to the DSP at this rate (Kenton 2009, 25).

Audio samples are discrete values or numbers which represent the amplitude of an audio signal taken at different points in a period of time. A continuous signal can be sampled at a certain sampling rate meaning that the signal can be converted into a digital representation using the acquired samples. Digital audio samples can be converted back into an analog audio signal, which may or may not be identical to the original signal. These digital signals are converted back into analog signals using a module known as DAC when the sound is played. This generates the varied sounds waves. This process of converting the digital audio signal to analog audio signal is referred to as reconstruction. (Ashok 2007, 3.)

2.3 Digital image analysis

Digital image analysis is a key factor in solving computer image problems. It does not produce pictorial results but it rather produces numerical or graphical information. It involves manipulating the data to determine exactly the information on the computer imaging system. Images contain numerous amounts of data typically on the order of hundreds of kilobytes or even megabytes. (Umbaugh 2005, 67.) An image is in form of

an array, or a matrix of square pixels arranged in columns and rows. An image signal such as a photograph is an example of a two-dimensional signal where the two independent variables are the two spatial variables (Sanjit 2006, 1).

An image may also be defined as a three-dimensional array of values specifying the colour of each rectangular area. These matrices provide a means of storing large quantities of information in such a way that each piece of information can easily be identified and manipulated (Croft et al. 2001, 236). In a grey image each picture element has an assigned intensity that ranges from 0 to 255. The grey scale image is usually what is referred to as black and white image although the name emphasizes the variety of shades of grey. A normal grey scale image has 8 bit colour depth equivalent to 256 grey scales. A true colour image has the colour depth of 24 implying that each colour is equivalent to 8 bits. (Usher 2012.)

There are two general groups of images, vector graphics which is also known as line art and bitmaps which are either pixel based or represent images. Vector graphics is the use of geometrical primitives to represent an image in computer graphics. Examples of these geometrical shapes include curves and lines. A bitmap is an image file format where a map of bits is used to store an image which also implies one bit per pixel. (Usher 2012.) Image files can be stored in a variety of formats. Some of the most common file formats are:

- Joint Photographic Expert Group (JPEG) which is an efficient destructively compressed 24 bit bitmap format that is widely used especially for the web and internet (Usher 2012).

- Tagged Image File Format (TIFF) which is a standard 24 bit publication bitmap format compressing non-destructively (Usher 2012).

- Portable Network Graphics (PNG)

- Graphics Interchange Format (GIF) which is an 8 bit non-destructively compressed bitmap format mostly used for the web. It has severe sub-standards one of which is the animated GIF (Usher 2012).

- Photoshop document (PSD) which is a dedicated Photoshop format keeping all the information in an image including all the layers (Usher 2012).

There exist two colour models in the science communication, RGB and CMYK. The RGB colour model relates to the way colour is perceived with the initial letter R standing for red, G standing for green and B standing for blue. RGB uses additive colour mixing and is the basic colour model used in television or any other medium that projects colour with light. It is the basic colour model used for computers and web graphics though it cannot be used in the printing production. The secondary colours of the RGB are cyan, magenta and yellow which are obtained by combining two of the primary colours. Red and green combined together make yellow, yellow and blue make cyan, and finally blue and red make magenta. However, the combination of the three primary colours in their full intensity make white. The three primary colours are mixed together with the help of the additive colour mixing model. (Usher 2012.)

The CMYK colour model comprises of four colours and is commonly used in the printing production by laying down the overlapping layers of varying percentages of the transparent cyan, magenta and yellow inks. The CMYK colour model relates to the way colour is perceived with the initial letter C standing for cyan, M standing for magenta, Y standing for yellow and K for black. The primary colours are obtained by mixing two of the secondary colours. A combination of magenta and cyan make blue, magenta and yellow make red while cyan and yellow make green. A combination of the three secondary colours makes black (K). The black ink is additionally added to make a complete CMYK colour model. However, the CMYK colour model uses the subtractive colour model. (Usher 2012.)

In some situations, processing an image requires converting the image to a grey scale which is a two dimensional (2-D) signal. A single RGB image is made up of three matrices which are identical in size but not in values with each matrix representing a single colour from the RGB colour scale. An RGB image can be converted to gray scale. A gray scale contains a single matrix meaning that all the three matrices are combined to form a single matrix. An image that is in the CMYK colour model is made up of four matrices and it may be converted into the RGB or gray scale format if it has to be analyzed. (Usher 2012.) The reason for converting CMYK to either RGB or gray scale format is that most of the Matlab functions used to analyze the images support at least one of the two formats.

2.4 Analysis of video frames

A video is an ordered array of several video frames forming up one complete video file. A video frame may be obtained from an image before it is added to a video file. These video frames are usually captured at a certain time interval known as sampling rate. This interval determines the quality of a video that includes high definition (HD). Each frame of a black-and-white video frame is a 2-D image signal that is a function of two discrete spatial variables with each frame occurring sequentially at a discrete instant of time. Hence, a black-and-white video signal is an example of a three-dimensional (3-D) signal where the three independent variables are the two spatial variables and time. A colour video signal is a three-channel signal composed of three 3-D signals representing time and the three primary colours red, green and blue. (Sanjit 2006, 1.)

A video can be stored in a variety of video formats. Examples of these formats include Audio Video Interleave (AVI), Motion Picture Experts Group (MPEG), Windows Media-Video (WMV) and Audio Video Interleave (AVI). Video frames can be processed or analyzed exactly like image files since one can easily be converted to the other.

2.5 Fast Fourier Transform

Fast Fourier Transform (FFT) is an efficient algorithm used to compute the Discrete Fourier Transform (DFT) and its inverse. DFT is one of the algorithms which refer to a means of examining a sampled signal in the frequency domain. It involves mapping one ordered set of numbers known as time domain to a different ordered set known as frequency domain information (Bateman 2002, 402). A non-periodic signal has a non-periodic analog spectrum described by its Fourier Transform whereas if the signal is periodic and discrete, its spectrum is also periodic and discrete.

A variety of FFT algorithms exist in mathematics from simple to complex. While the FFT only requires a few lines of code, it is not the most complicated algorithm in the area of DSP. In simple terms the FFT means that the signal that is in the time domain is converted into the frequency domain to make the analysis easier. (Stephen 2011.)

The time and frequency domains each contain one signal made up of a number N of complex points in the complex notation. Each of these complex points is composed of two numbers, the real part and the imaginary part. For example, when talking about a

complex sample X[42], the sample refers to the combination of ReX[42] and ImX[42]. In other words, each complex variable holds two numbers, the real and imaginary part. When two complex variables are multiplied, the four individual components must be combined to form the two components of the product. The FFT operates by decomposing an N point time domain signal into N time domain signals each composed of a single point. The second step is to calculate the N frequency spectra corresponding to these N time domain signals. Lastly, the N spectra are synthesized into a single frequency spectrum. Figure 1 shows an example of the time domain decomposition used in the FFT for a 16-point signal. (Stephen 2011.)

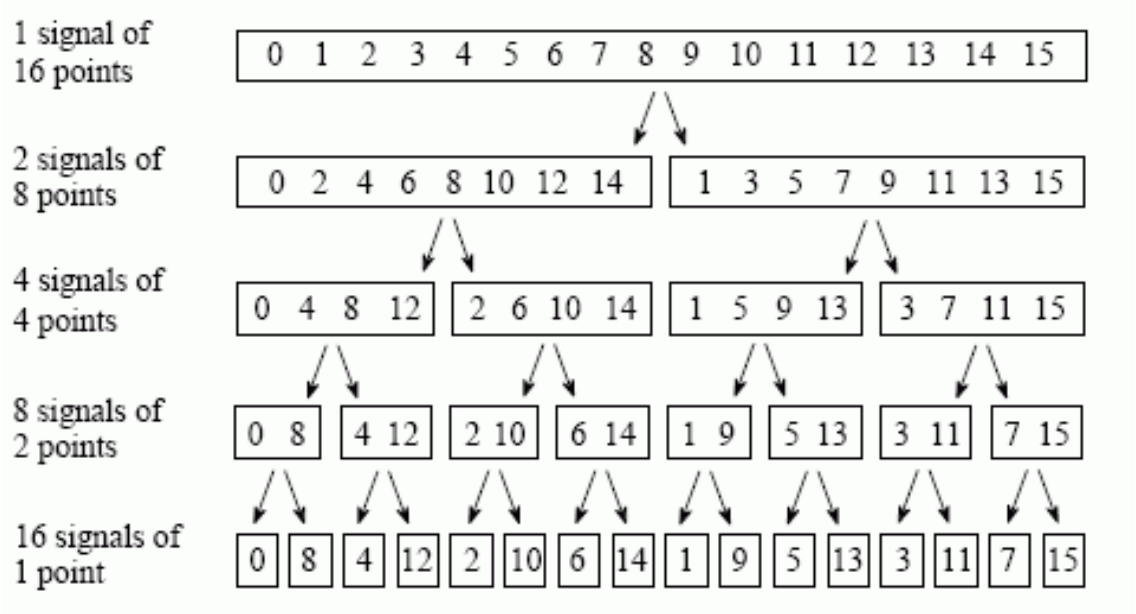


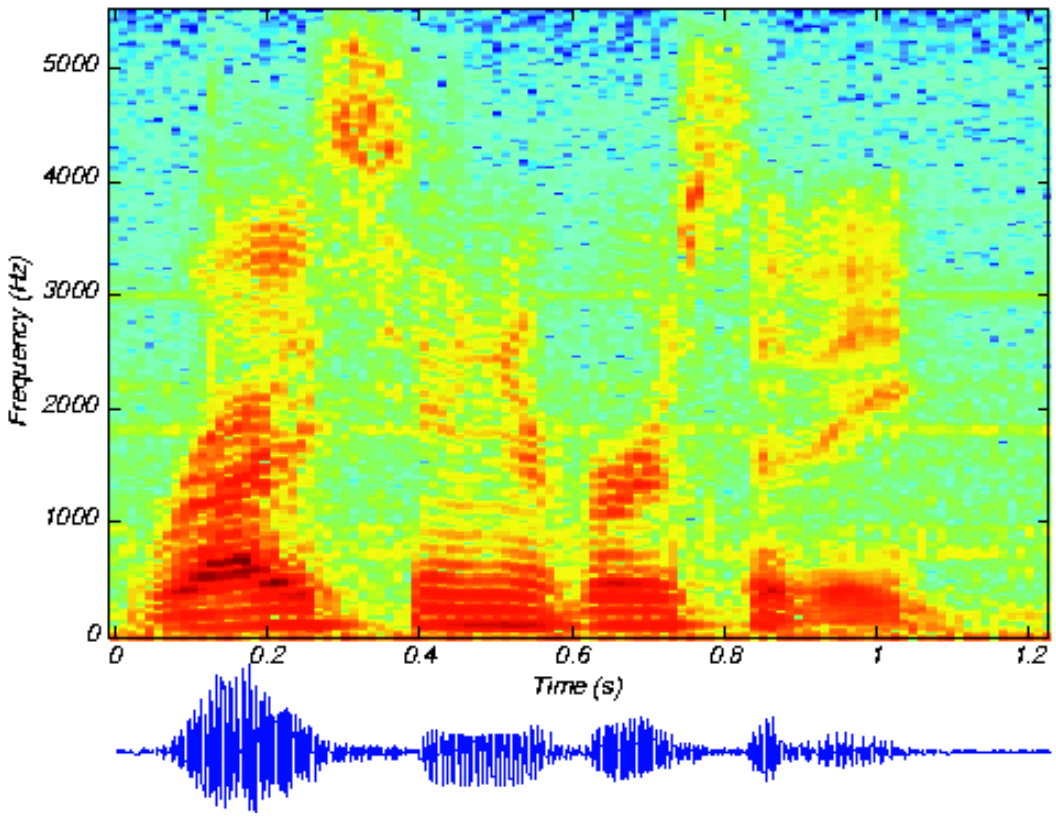
Figure 1. FFT decomposition of a signal (Stephen 2011)

A 16-point signal is decomposed through four separate stages as seen in figure 1. The first stage breaks the 16-point signal into two eight-point signals. The second signal then decomposes each eight-point signal into two four-point signals. The four-point signals become four in number. The pattern continues until there are N signals composed of a single point. In this case, the N signal of a single point implies 16 signals of a single point. Each of the stages uses an interlace decomposition that separates the even and odd numbered samples. (Stephen 2011.)

2.6 Spectrogram

Audio signals can be analyzed to give the sound pressure or amplitude of the signal versus time. The parameters such as the average level, beginning and end of speech segments and pauses can be computed from the recorded signal. Other questions can be answered more easily if the signal is transformed into the frequency domain. The information about the frequencies and levels of the tones a signal is composed of can be obtained from the spectrum of the signal. Majority of the signals including speech are not stationary but change over time making it insufficient to compute the spectrum for a complete signal. (Johnson 2009.)

This spectrum may be displayed as a three-dimensional diagram, the first axis for time, the second for frequency and the third for the signal level. The spectrum represented with a three-dimensional diagram is known as spectrogram. It displays the level of the signal at a given time and frequency. A spectrogram shows how the spectral density of a signal varies with time. Figure 2 shows an example of a speech spectrogram. (Johnson 2009.)



Figire 2. Speech spectrogram (Johnson 2009)

Time is always displayed along the horizontal axis whereas the frequency is displayed along the vertical axis. The intensity of each frequency is represented with a colour, rather than a graph, and the coloured lines scroll downwards over time, giving a visual representation of where the frequency intensities were over the last few seconds. (Johnson 2009.)

A spectrogram may also be known as a spectral waterfall, sonogram, voiceprint or voice gram. However, a spectrogram shows exactly the same information as spectrum analyzers even though it is represented in an entirely different way. (Johnson 2009.)

2.7 Correlation

Correlation is a technique used in quantitative comparison of two functions and the correlation of signals shows how similar the two signals are. A correlation of 1 means that the two signals are identical, a correlation of 0 means they are not related to each other, and a correlation of -1 means that one of the signals is the inverse of the other. Correlation takes only two forms, auto-correlation and cross-correlation. Auto-correlation involves only one signal and it provides information about the structure of the signal or its behavior in the time domain. The main purpose for determining auto-correlation is to identify hidden periodicities. Auto-correlation compares the values of the samples at one time to the values of the samples at another time on the same signal (Emmanuel et al. 2002, 8.) It has been useful in calculation of energy spectral density and energy content of waveforms.

Correlation that involves two different signals is referred to as cross-correlation. It is a measure of similarities or shared properties between two signals as a function of a time-lag applied on one of them (Emmanuel et al. 2002, 7). It measures the dependence of the values of one signal on another signal. Key areas where cross-correlation is applied include analysis and detection or recovery of signals that are mixed with noise, estimation of periodic signal in noise, pattern recognition and delay measurements. The signal buried in noise can be estimated by cross-correlating it with an adjustable template signal. In most cases the template is adjusted with trial and error and guided by any fore knowledge until the cross-correlation function has been maximized. The template with a maximum value is the estimate of the signal. (Emmanuel et al. 2002, 257.)

Cross-correlation is also used in determining the signal-to-noise ratio for any periodic noisy signal. Both the signal-to-noise ratio and signal powers may be determined by measuring the correlation coefficients of a noisy periodic signal. Another application of correlation is the correlation detection implementation of the matched filter. (Emmanuel et al. 2002, 257.)

Cross-correlation takes two forms, normalized and phase-only cross-correlation. Normalized cross-correlation refers to a process used to find indices or patterns within an image (MathWorks 2012). It has been widely used to locate faces with various poses, illumination and clutter. It is used to find small templates in the image which matches a template. On the other hand, phase-only correlation is a method of registration whereby the shift between two images in the spatial-domain is reflected as a phase change in the frequency domain (Huang et al. 2005).

2.8 Sonar processing

Sonar is any system that uses acoustic means to detect, localize, track, or classify objects (Kongsberg 2009). The detection and range part of the system is accomplished first by timing the delay between the transmission pulse of the sound pulse and its subsequent return pulse as seen in figure 3.

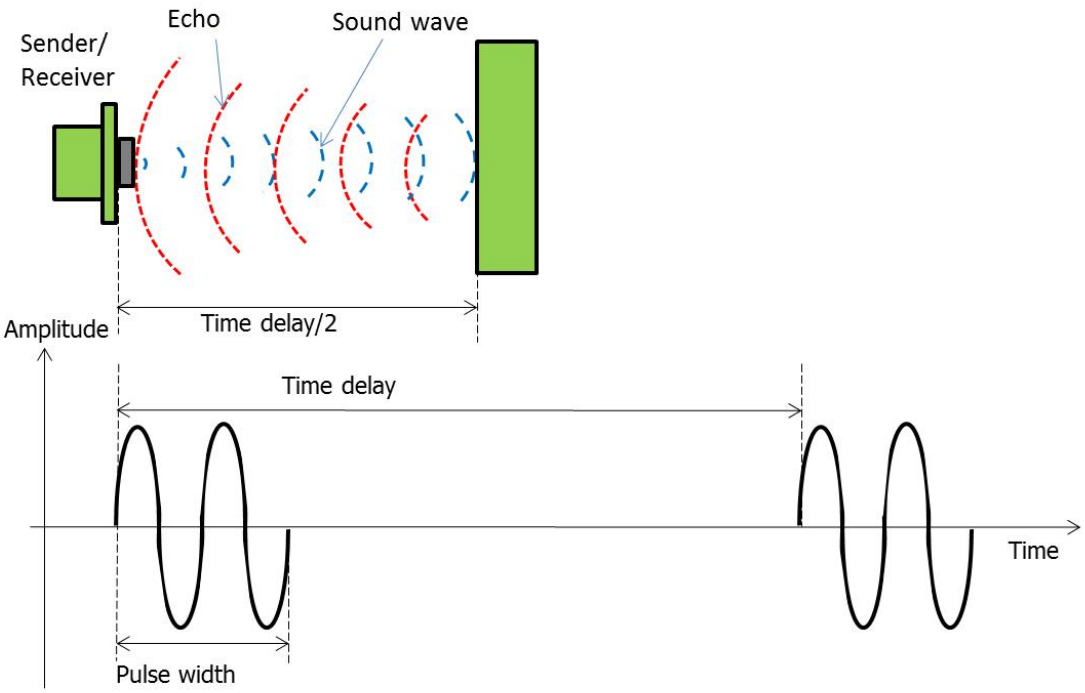


Figure 3. Detection of an echo pulse

The range is then obtained from the product of the time delay shown in figure 3 and the speed of sound divided by two. The speed of sound varies depending on the temperature and pressure. The speed of sound at the sea level is 340.29 m/s at which all sound waves travel. The factor of two comes from the fact that the sound pulse travels to the target device and back before detection of the subsequent pulse occurs.

The primary role of sonar is the detection and tracking of echoes from devices such as submarines and, to a lesser extent, surface ships operating in the world's oceans. Submarines are highly capable weapons platforms that are difficult to detect when submerged. Because sound propagates relatively well in the ocean, the Navy has relied heavily on the use of acoustic detection systems for finding submarines and water depth. The navy uses an echo sounder that is attached to the bottom of a ship which sends an outgoing sound pulse downwards into the water. The sound energy travels through the water to the bottom of the ocean where it is reflected back towards the source where it is received and recorded as seen in figure 4. (Robert 2012.)

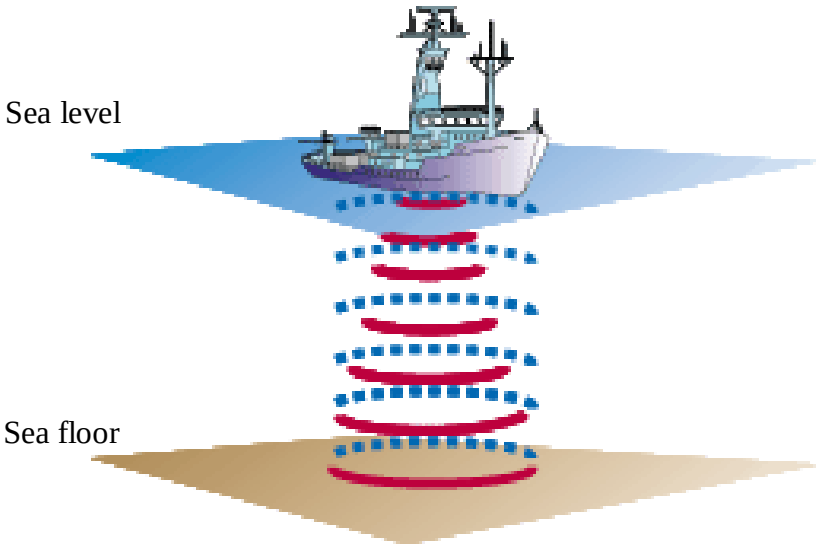


Figure 4. Echo sounder

The time that it takes for the sound pulse to make the round trip to the seafloor and back can accurately be measured as seen in figure 4. Water depth is determined from the travel time and the average speed of sound in water.

Sonar systems take two forms, the passive and active. Passive sonar systems detect sound radiated by a target of interest while active sonar systems launch pulses of acoustic energy and detect echoes from targets. (Kongsberg 2009.) The primary purpose of the passive sonar system is to provide the best possible image of the surface and underwater environment by processing data from existing sonar arrays, providing high-quality displays for the sonar operators for surveillance, and for tactical and safety purposes (Robert 2012).

3 Methods and materials

The DSP analysis tool was developed using the Matlab programming environment. Matlab is a programming environment that is used for developing algorithms, data analysis, visualization, and numerical computations. It helps to solve technical computing problems faster than with traditional programming languages such as C and C++. Different Matlab toolboxes were required to complete the whole project. Examples of toolboxes required include Image Processing Toolbox, Image Acquisition Toolbox, Data Acquisition Toolbox, and Signal Processing Toolbox. The project was divided into four phases that included planning, designing, implementation, and testing. During the planning phase, the requirements were analyzed with the help of a documented system as shown in figure 5.

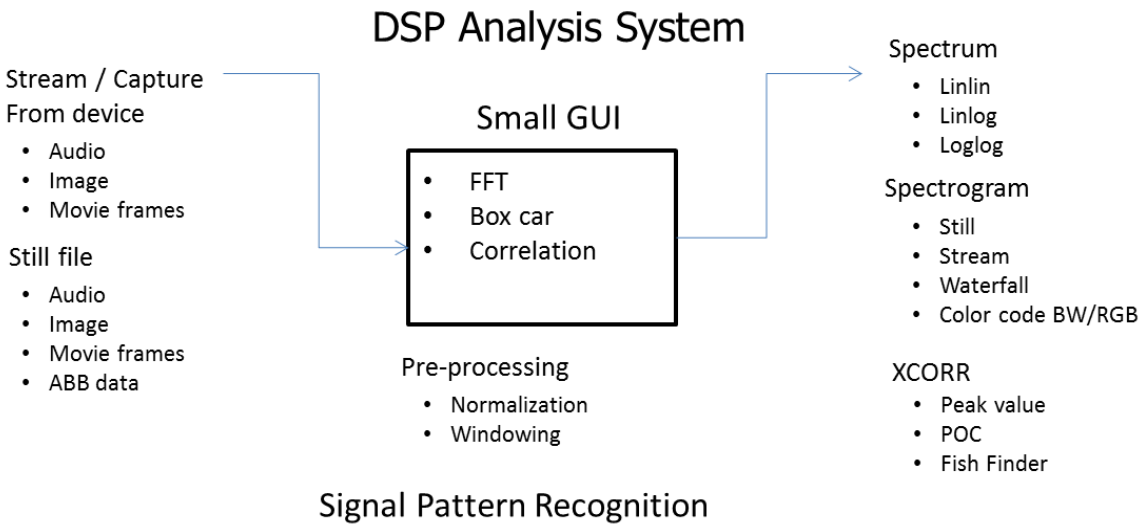


Figure 5. Requirements of the DSP analysis tool

The inputs of the required tool are clearly seen on the left hand side while the outputs are seen on the right hand side of figure 5. Pre-processing the samples is performed before the output is determined. Designing the DSP analysis tool leads to signal pattern recognition which provides a reasonable answer for any possible input.

3.1 Design

The design phase involved designing some Universal Modeling Language (UML) diagrams that include usecase diagram as shown in figure 6. The user inputs signals either from still files or stream and the signals inputted are either data stored in files, audio samples, images or video frames. Files are inputted by selecting still files stored in the memory of computer devices while stream data are obtained by capturing samples directly from the hardware devices.

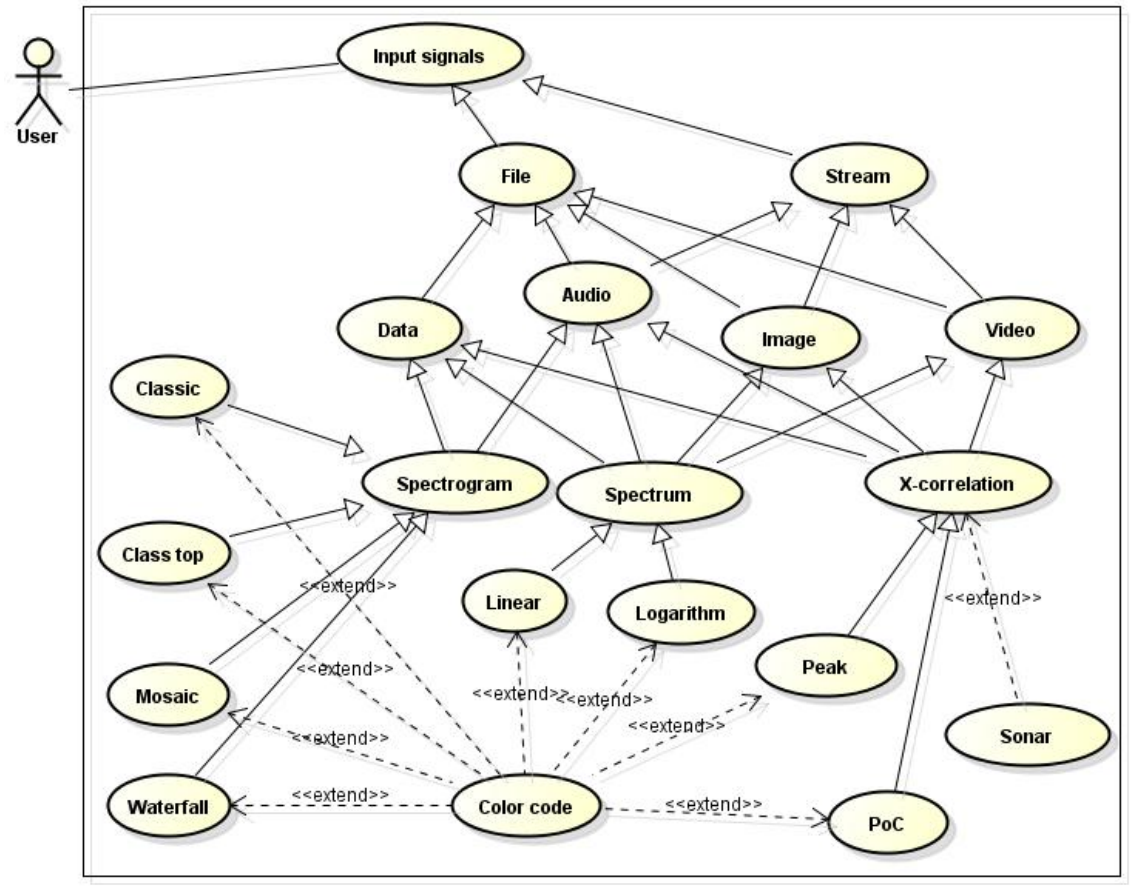


Figure 6. Usecase diagram of the DSP analysis tool

The relationship between the different elements of the DSP analysis tool is clearly seen in figure 6. The type of analysis method was chosen for the respective signal inputted. The spectrogram can only be chosen for either the data stored in files or for the audio samples as shown in figure 6. Other analysis methods that can be chosen are spectrum and correlation although only one analysis method can be chosen at a time. Each of these analysis methods takes different forms, the linear form and logarithmic form for

the spectrum among others. Specific graphs of the analysis methods with various colour codes are chosen depending on the user's choice.

3.2 Implementation

The design phase was then followed with the implementation phase. This phase started with designing the graphical user interface (GUI) the aim of which was to suit the user's interests. A number of things were put into consideration while designing the GUI. These include clarity, conciseness, preciseness and consistence. A tab menu was chosen as the best design since the system had a lot of information. The tab menu required resizing the GUI depending on the size of monitors. Different components were placed at different coordinates while some were visually deactivated on different tabs depending on the functionality of the tab, and this helped in achieving simplicity as seen in figure 7.

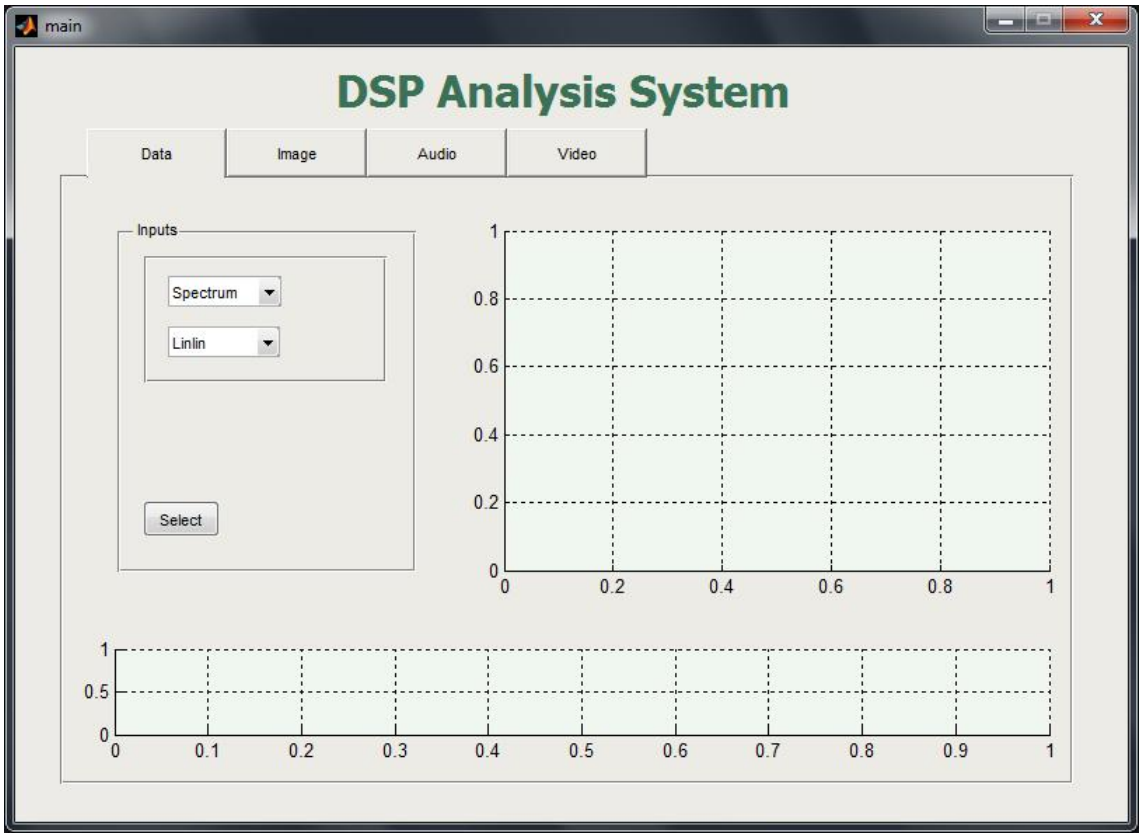


Figure 7. Default view of the DSP Analysis system tool

The drag and drop method was used while designing the GUI shown in figure 7. The same components were positioned accordingly on different tab panels. The best features for each component were chosen as part of the graphical user interface. The

GUI was followed with carrying out more research about the functionality of the system. This GUI was then combined with the small samples of the engine that were developed separately from the main project. The reason for this was to avoid messing up the code.

The implementation phase involved checking for the type of computer architecture on which the Matlab programming environment is installed. The reason for this was that some Matlab functions are used depending the Matlab architecture programming environment that is installed on a computer. Examples of such Matlab functions include *analoginput* and *analogoutput* which only work with the 32-bit Matlab architecture programming environment while the *audiorecorder* function only works on 64-bit Matlab architecture programming environment. This was followed by getting the monitor dimensions to cater for different sizes of the monitors on which this software application can run.

3.3 Test

The testing phase was done concurrently with the implementation phase. Each new feature was tested before moving on to develop another feature. This helped to maintain consistency during the process of software development. This phase also involved testing the software application on different computers, monitors among others.

The software application was designed in a way that all the samples capture from the hardware devices are saved as files. Images were stored as JPEG, audio samples as WAV and video frames as AVI files. Saving the samples to files was done automatically after capturing the samples. The user is not aware how the samples are saved. Each file saved on a computer has a unique file name and this is achieved by storing an integer value in a generic data file. Each time a new file is created, the program reads and gets the integer value in the generic data file, increments the value by one, concatenates the incremented value to a unique string and then saves the incremented value back to the generic data file. The unique string which is concatenated with the incremented value is then used as the name of the new file.

The processing method used to generate majority of the results is sequential programming while in some cases, parallel processing was used to generate better results. The spectrum and spectrogram of the still data files and audio files were archived using parallel processing while the rest of the results were archived using sequential processing.

3.4 Audio acquisition

The acquisition of audio samples on the 32-bit Matlab programming environment was done separately from that of the 64-bit Matlab programming environment. The 32-bit Matlab programming environment uses the *analoginput* and *analogoutput* Matlab functions to acquire audio samples from the microphone and output sound through the soundcard of the computer respectively. This means that these two functions can even perform their task without giving any error even though a microphone or speaker is not connected to the computer.

However, these functions are not implemented in the 64-bit Matlab programming environment. Different Matlab functions known as *audiorecoder* and *sound* are provided to acquire audio samples from the microphone and to output sound through the speaker respectively. The *audiorecorder* and *sound* Matlab functions show an error in case the microphone or a speaker are not plugged onto the computer that is installed with a 64-bit Matlab programming environment respectively. The *audiorecorder* and *sound* Matlab functions do not exist in the 32-bit Matlab programming environment. Therefore, checking for the processor architecture helped to cater for the acquisition of audio samples and output sound on both the 32-bit and the 64-bit Matlab programming environment.

3.5 Spectrum

The spectrum of all signals was achieved by converting the signal from its time domain form to frequency domain form using Matlab functions *fft* and *fft2* for audio samples and image files respectively. The FFT was obtained for only signals that are in 2-dimensional form. Even though some signals are of three-dimensional (3-D) form, they are converted into 2-D form before transformation from the time domain signal to the frequency domain takes place. Examples of signals that are three-dimensional in form

include colour images and video frames. The 3-D images may be in the form of RGB and indexed format. A three-dimensional image may become a two-dimensional image if it has been converted to gray scale as shown in figure 8.

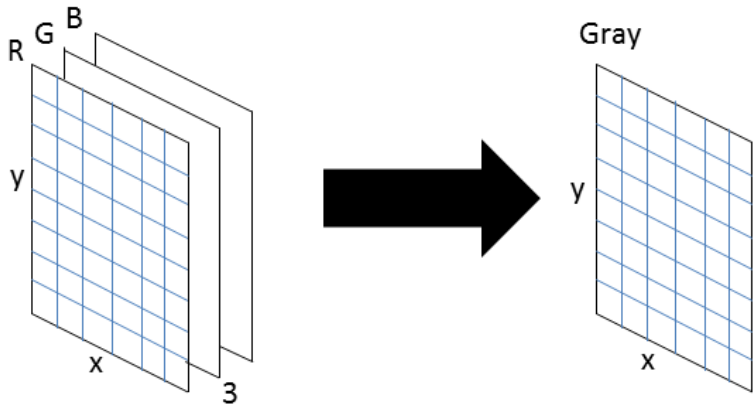


Figure 8. Converting an RGB image to gray scale.

The RGB image is converted to gray scale by eliminating the hue and the saturation information while retaining the luminance. The Matlab programming environment uses the *rgb2gray* Matlab function to convert RGB values to gray scale values by forming a weighed sum of the R, G, and B components shown in equation 1.

$$\text{Gray} = 0.2989 * \text{R} + 0.5870 * \text{G} + 0.1140 * \text{B} \qquad \text{equation (1)}$$

This makes it possible to find the FFT of the gray scale image. Images with other colour modes are converted to gray scale since different webcams output images and video frames of different colour modes. The results after taking the Fast Fourier Transform is then presented either in its linear or logarithmic form on a graph.

3.6 Cross-correlation

The cross-correlation between any two signals was only obtained if the two digital signals were of the same size. In situations where the two signals are not of the same size, the remaining part of the signal with a smaller size is filled with zeros. This was done by first getting the size of both images where image one was donated with sides x1 and x2 and image two with sides y1 and y2 as shown in figure 9.

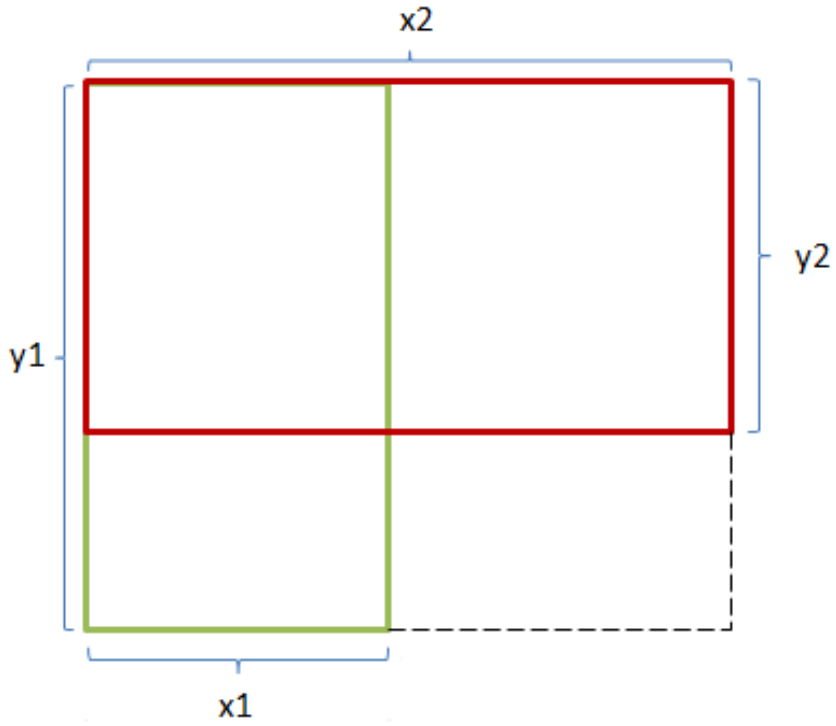


Figure 9. Resizing the two images with zeros

The horizontal sides of both images shown in figure 9 are compared to determine which of the two is greater than the other. If the sides of the two images are not the same, the difference between the sides of the two images is obtained and the image with a shorter side is filled with zeros. The image with sides x1 and y1 is increased to become size x2 and y1. The same process is also done for the vertical sides of the two images. The size of the two images now becomes x2 and y1. The cross-correlation of the two images is now obtained as a normalized correlation or phase-only correlation.

3.7 Sonar

The sonar uses the method of cross-correlation to determine the relationship between the original sound pulse and the echo pulse. The distance is calculated by obtaining the time taken for the echo to travel to the extreme end and back multiplied by the speed of sound divided by two. This is sometimes referred to as sonar or radar processing.

The process involves outputting a sound through the speaker and then immediately listening to the echo. The timer is started immediately when the sound pulse is

outputted and stopped when there is the highest relationship between the original sound pulse with its echo pulse. The result is then outputted as a spike onto the graph.

3.8 Working environment

The software created in the project described in this thesis was developed at the Metropolia University of Applied Sciences at the Leppävaara campus. The working environment at the Metropolia University of Applied Sciences consists of various rooms all over the school premises ranging from laboratories, library, and other rooms where the Matlab programming environment was installed on the computers for the project. The computers were well furnished with an updated version of the environment which was needed to develop the software tool.

The work itself was rewarding in that the developers had a chance to learn new development techniques and work with new people. Several meetings with the project supervisor Dr. Antti Piironen and Professor Hiroyuki Aoki were held. The two members were able to learn how to network and share information amongst themselves. The working hours were flexible as long as the required amounts of work were done each week.

The Matlab programming environment was installed on a good number of computers which were used to develop the software application described in this thesis. Most of the necessary Matlab toolboxes that were required to develop the software existed in the Matlab programming environment that was installed on the school computers. Examples of toolboxes required included Image Processing Toolbox and Signal Processing Toolbox.

Personal computers and laptops were also used to develop this software application especially during hours when the school was closed. This helped to improve the quality of the software application.

3.9 Software development

An incremental and interactive method of software development was used to develop the software in a relatively small group made up of two members. Roles and rules were predefined amongst the members although they were not followed. The process can be characterized as good co-operation among colleagues, which helped obtaining good results. Both members managed to adapt to the new technologies that were used during the development of the software.

However, both members were inquisitive to learn and develop high quality features of this piece of software. Developing this software was quite flexible since the members were able to work on the project even at their home premises.

4 Results

A graphical user interface was designed using the Matlab programing language and the best design was chosen from all the designs which were initially made. A simple and easier tab menu design was taken into consideration while choosing a design. The design chosen was simple and could cater for most of the characteristics of a good user interface. The software application is able to analyze different signals that include data from text files, images, audio samples and video frames. These signals may either be from still files or acquired directly through the stream from the required hardware. The software only acquires the signals through the stream from the webcam and sound card or microphone. Either the images or the video frames are acquired from the webcam while audio samples are acquired from the microphone.

A lot of programming logic was used while designing the software application. Each time a loophole was detected in the software, various solutions were suggested and there after the best solutions were chosen from the list of the suggested items.

The parts discussed in this chapter include the images, audio samples and the video frames. Data samples are analyzed exactly like the audio samples and will not be discussed in this section. The only difference between data and audio signals is that the data samples are stored in the text files while the audio samples are stored in the audio files. The analysis of video frames is not discussed in this chapter since they are analyzed exactly like that of an image. The only difference is that a video frame is first converted into an image before it is analyzed.

New directories are created the first time the software is run on a computer installed with the Matlab programming environment. These directories are found in the Documents folder of a computer on which this software has been run. The directories are used as references to where all other files that this software creates are saved. Sample files are also copied to these respective directories the first time the software in run on a computer installed with the Matlab programming environment. The sample files consist of at least two sample files for each signal type.

4.1 Spectrum of an image

The spectrum of an image involved obtaining the linear and logarithmic forms. The user interacts with the system with the help of buttons, radio buttons and popup menus. A still image file located on the lower left hand side was selected using the select button, and other parameters comprising of radio buttons and popup menus were also selected as seen in figure 10. The parameters selected include the still file, spectrum, linearity of the FFT spectrum and jet as a colour code. The show button is only activated if an image is selected. The show button was then clicked on to provide the output shown on the right hand side of figure 10.

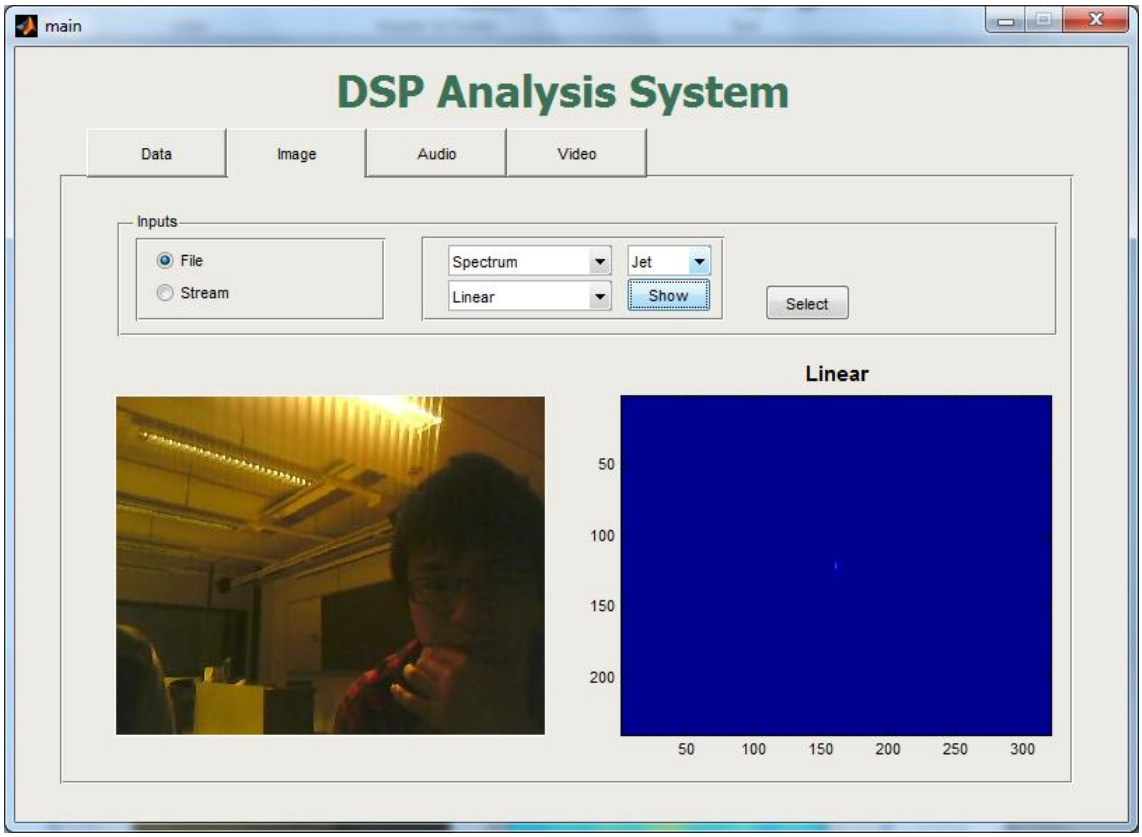


Figure 10. Linear FFT spectrum for a still image

The linear FFT spectrum of the image shown on the right hand side of figure 10 was obtained by first determining the number of dimensions of the image and any image that was not in a 2 dimensional format was converted to grey scale using the a Matlab function *rgb2gray*. The FFT is now performed using the Matlab function *fft2* and thereafter the FFT shift is obtained from the result which places the FFT spectrum impulse in the center using the Matlab function *fftshift*.

The same still image file located on the lower left side was selected using the select button, and other parameters comprising of radio buttons and popup menus were also selected as shown in figure 11. The parameters selected include the still file, spectrum, logarithm of the FFT spectrum and jet as a colour code. The show button is only activated if an image is selected. The reason for this is to make sure that the software does not go through the unnecessary conditions. The show button was then clicked to provide the output shown on the right hand side of figure 11.

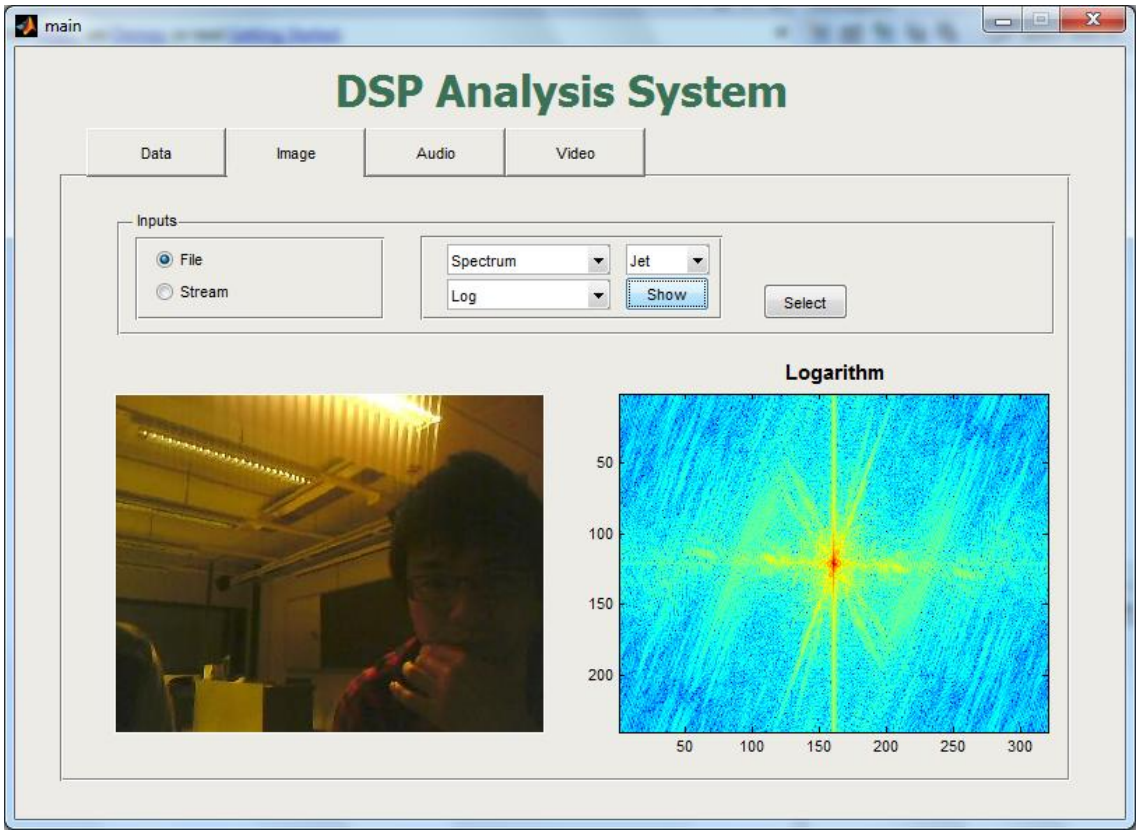


Figure 11. Logarithm of the FFT spectrum for a still image

The logarithm of the FFT spectrum displayed in the two dimensional form on the right hand side of figure 11 was obtained using the Matlab function *log2*. The logarithm of the shifted FFT spectrum was taken and thereafter displayed onto the graph. The logarithm helps to bring out details of the FFT spectrum in regions where the amplitude is smaller. This 2-D graph was obtained with the help of the Matlab function *imagesc*.

The spectrum of images or video frames was also presented in a graph in a three-dimensional format. This helped to see the amplitudes of the spectrum clearly. The three-dimensional graph was presented for both the linear and logarithmic spectrum. Figure 12 shows the three-dimensional graph for the logarithm of the spectrum.

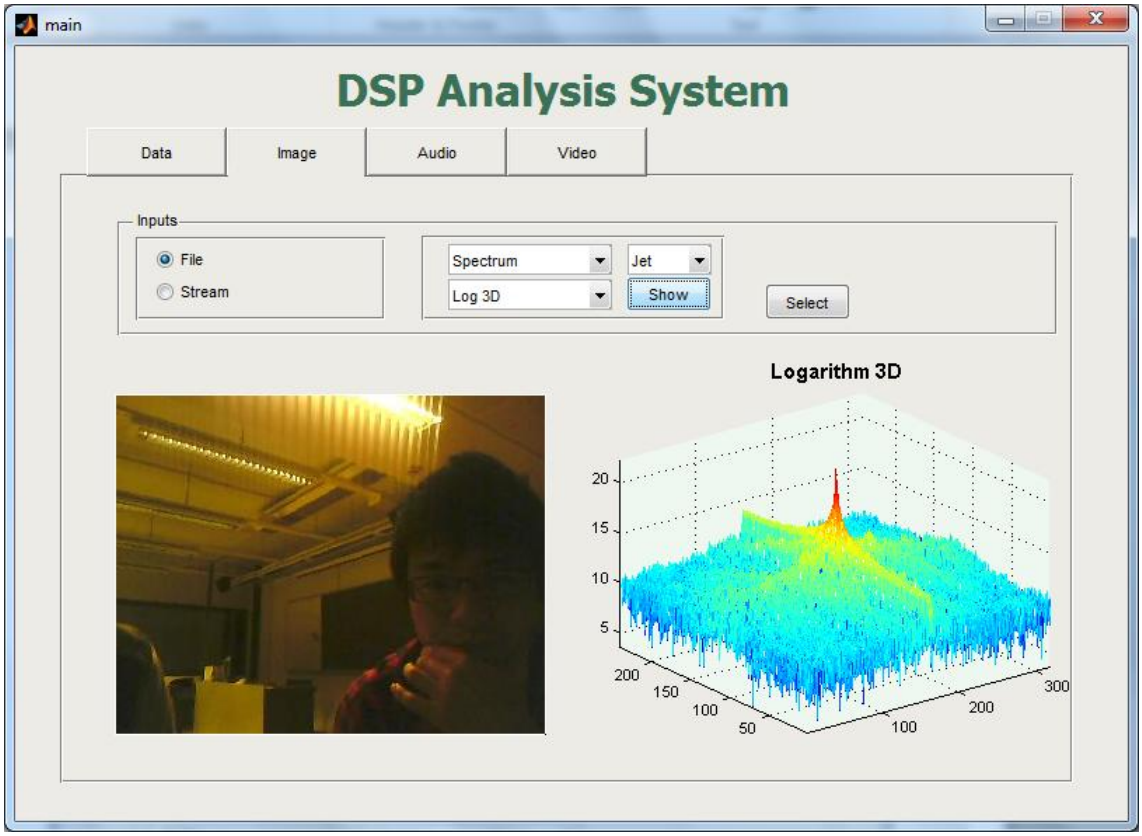


Figure 12. Logarithm of the FFT spectrum for a still image presented as 3D

The logarithmic 3-D spectrum shown in figure 12 was obtained in the same way as the two dimensional logarithmic spectrum. The only difference was in the way the result was presented in the graph. The 3-D graph was obtained with the help of the Matlab function *mesh* that produces wireframe surfaces to define a surface.

4.2 Spectrum of an audio signal

The linlin FFT spectrum implies that both the horizontal axis and the vertical axis are presented in their linear form. The linlin FFT spectrum of the audio signal was obtained by taking the FFT of a few samples of an audio signal. A still file was selected and thereafter the play button was clicked on, which initialized the play of the audio file from the beginning to the end. Small samples at different time intervals were taken from the audio signal and their FFT spectrum was obtained and displayed as seen in the graph found at the right hand side of figure 13. The Matlab programing environment uses the Matlab function *fft* to generate the FFT spectrum.

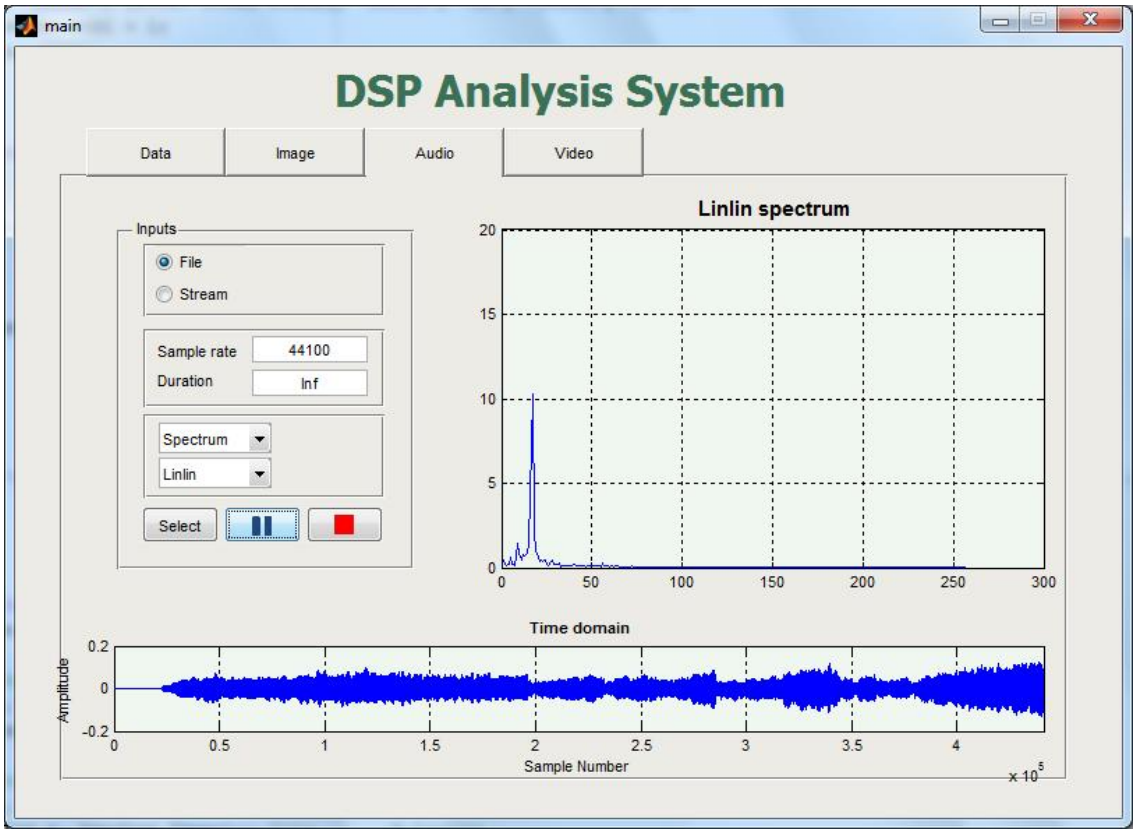


Figure 13. Linlin FFT spectrum for samples taken from an audio signal

The FFT graph located on the right hand side of figure 13 shows the average frequency on the horizontal axis and the amplitude of this spectrum on the vertical axis of the samples. The amplitude indicates how strong the average frequency is with a value of about eleven at a frequency of about 20 Hertz.

The same audio signal was used to generate the linlog FFT spectrum shown in figure 14. Linlog implies that horizontal values are presented in their linear form while the vertical values are presented in their logarithmic form. The linlog FFT spectrum was obtained from the FFT spectrum of the samples that are taken from an audio signal. The logarithm of the amplitude values of these samples which is also known as the values on the vertical axis was then calculated, and the results are presented on a graph located on the right hand side of figure 14.

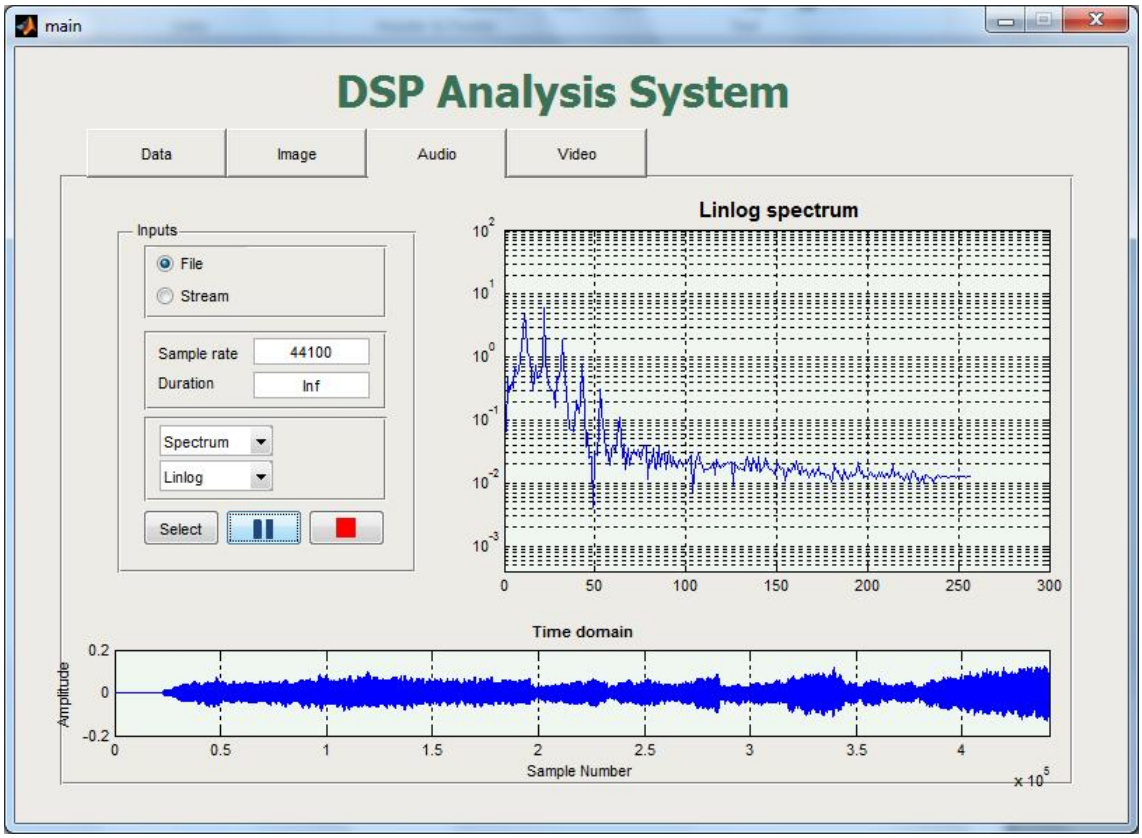


Figure 14. Linlog FFT spectrum for samples taken from an audio signal

The linlog FFT spectrum was acquired for both audio samples obtained from a still file and those acquired through the stream. The Matlab programming language uses the Matlab function *log* to obtain the logarithm of a value. In this case the corresponding logarithm of all values was obtained.

The loglog implies that both the horizontal and the vertical values are presented in their logarithmic forms. The loglog FFT spectrum shown in the graph that is located at the top right hand side of the figure 15 was generated by first obtaining the FFT spectrum of the audio signal. The samples used to obtain the spectrum were obtained from one of the still files.

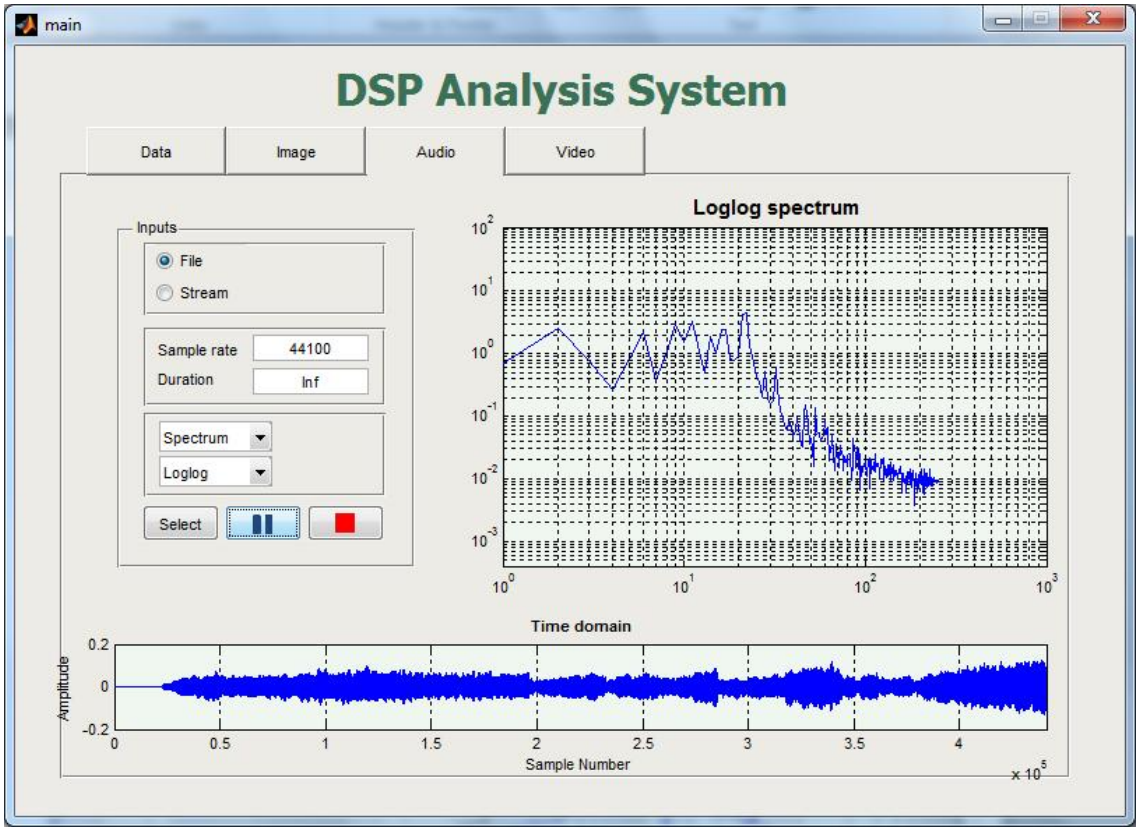


Figure 15. Loglog FFT spectrum of samples taken form an audio signal

The loglog FFT spectrum was obtained from the FFT spectrum of the samples that are taken from an audio signal. The logarithm of both the amplitude and frequency values of each samples was then calculated using the Matlab function *log* and thereafter the results are presented in the graph found on the right hand side of figure 15.

4.3 Cross-correlation of image files

The cross-correlation of images and video frames was obtained by taking their normalized and phase-only correlation for any two signals. The normalized cross-correlation of images and video frames was obtained for any two images or video frames or a combination of an image and a video frame. Any image that was found to

have more than one dimension was first converted to gray with the help of the Matlab function rgb2gray.

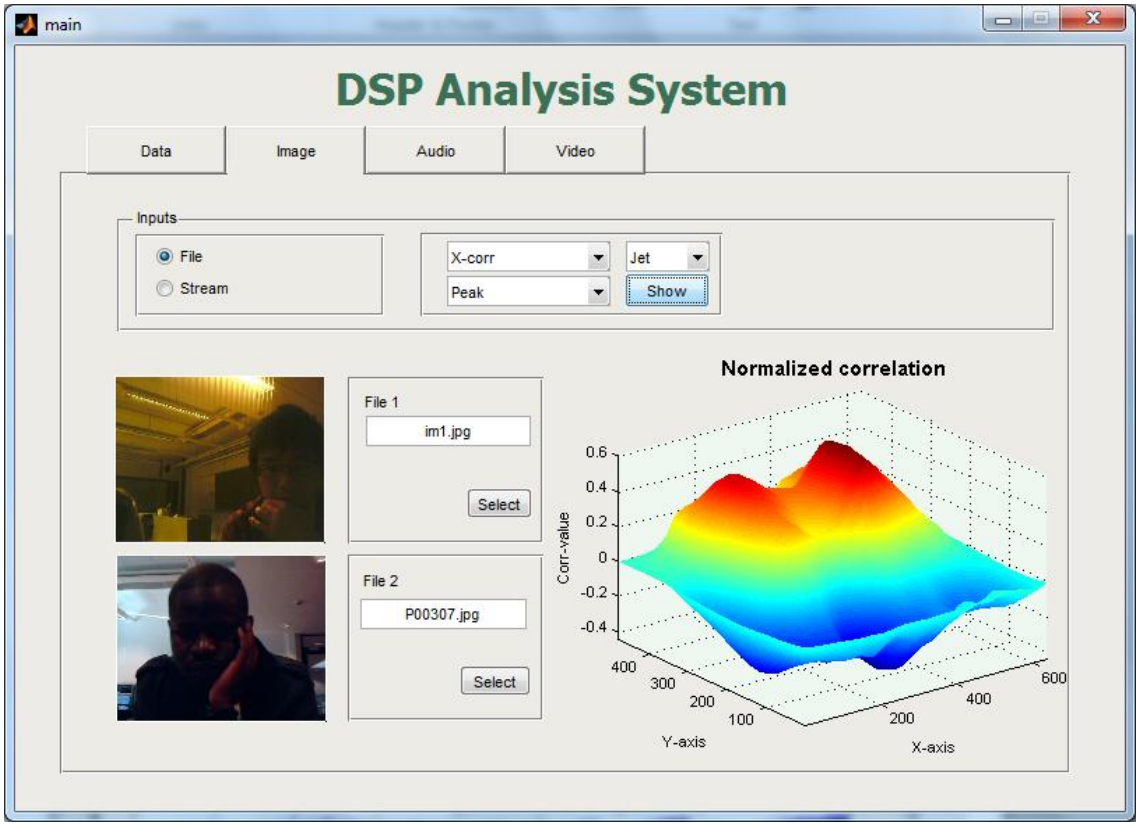


Figure 16. Normalized cross-correlation of two images.

The size of both images was first determined and both images were then made to be the same size by filling the smaller image with zeros. The FFT spectrum of both images was obtained by using the Matlab function *fft2* and the values of the first FFT spectrum were multiplied with the conjugate values of the second FFT spectrum. The inverse FFT of the outcome was performed using the Matlab function *ifft2* to obtain the normalized correlation. The Matlab function *fftshift* was used to shift the peak values to the center while the peak value and its location was then obtained from the result using the Matlab function *find* with two output parameters. The result was then displayed in the graph shown on the right hand side of figure 16. The same procedure was done to graph the correlation of video frames with the exception that the video frames required first converting them to an image.

The phase-only correlation of images and video frames was also obtained for any two images or video frames or a combination of an image and a video frame. Any image that was found to have more than one dimension was first converted to gray scale by using the Matlab function *rgb2gray* before being correlated with another image.

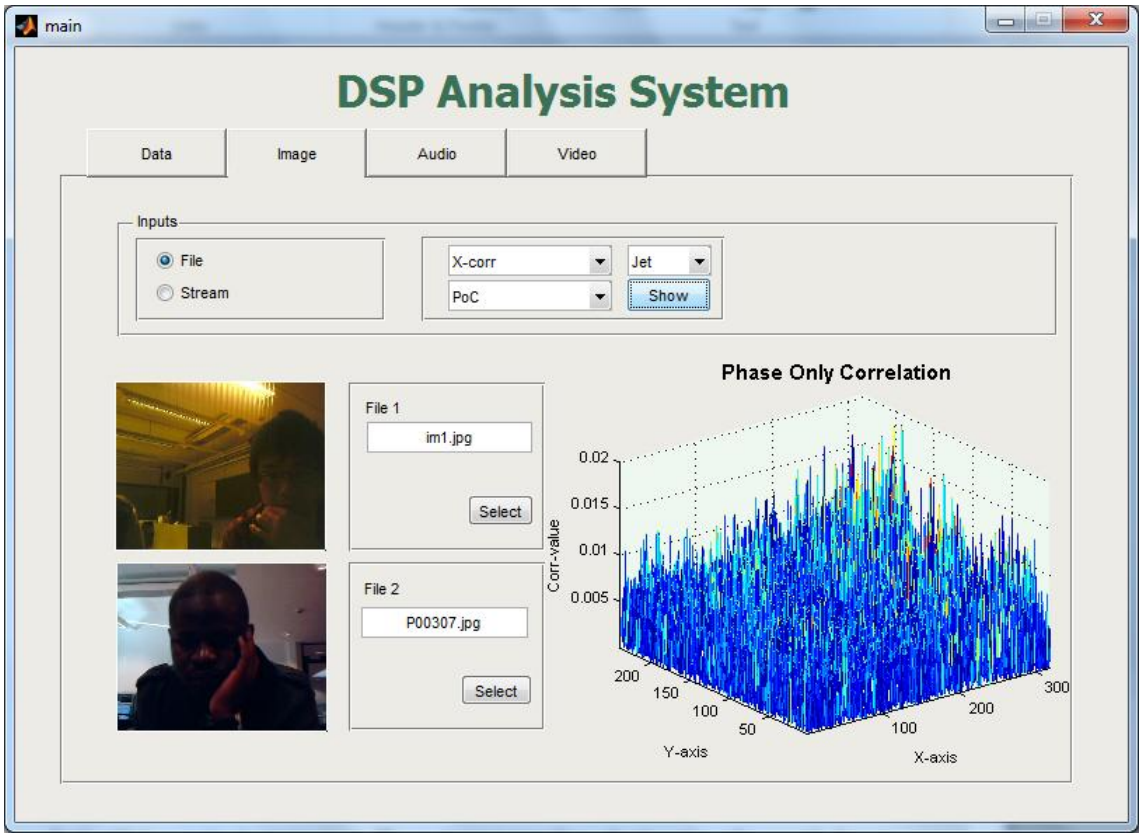


Figure 17. Phase-only correlation of two images

Each size of both images was first determined using the Matlab function *size* and both images were then made the same size by filling the smaller image with zeros. The FFT spectrum of both images was then obtained by using the Matlab function *fft2* and the values of the first FFT spectrum were multiplied with the conjugate values of the second FFT spectrum. The inverse FFT of the outcome was performed using the Matlab function *ifft2* and there after the Matlab function *fftshift* was used to center the peak values. The Matlab function *max* was applied twice on the result to obtain phase-only correlation. The peak value and its location is now obtained from the output and thereafter the result presented in the graph shown at the right hand side of figure 17. The same procedure was done to graph the correlation of video frames with the exception that the video frames required first converting them to an image.

4.4 Cross-correlation of audio signals

The cross-correlation was generated for any two sets of audio samples either acquired from the still file or through the stream from the hardware device. Sample signals that were acquired from still files and thereafter the normalized cross-correlation was obtained by correlating the first signal against the second signal as shown in figure 18.

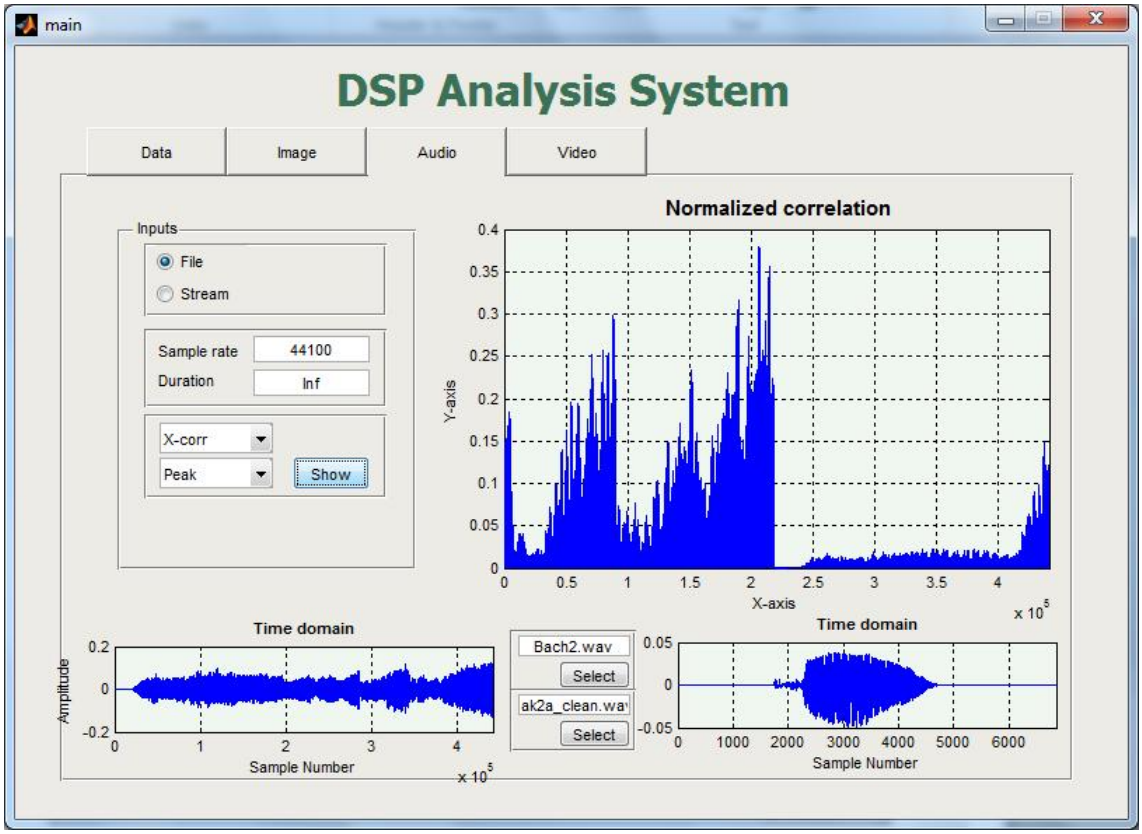


Figure 18. Normalized cross-correlation between two audio signals

The normalized cross-correlation of the audio signals shown at the top right hand side in figure 18 was obtained by first selecting a single channel from each audio sample. The length of each audio signal was then obtained to find which of the two signals contains fewer samples in relation to the other signal. The signal with more samples was then reduced to ensure that both signals had the same amount of samples. The FFT spectrum of the individual signals was obtained and the values of the FFT spectrum in the first signal were multiplied with the respective values for the conjugate FFT spectrum in the second signal. The inverse FFT of the result was then performed using the Matlab function *ifft* to obtain the normalized correlation. The Matlab function *fftshift* was then used to shift the peak values to the center and the peak value with its

location was obtained from the output. The results were then presented in a graph as seen in figure 15.

The phase-only correlation was obtained for any two sets of audio samples acquired either from the still file or through the stream as shown in the graph located on the top right hand side of figure 19. The two audio signals shown at the bottom of figure 19 were acquired from still files and the two signals were correlated with one another. A graph with a single spike located in the center with less noise at the bottom indicates high similarity in the two signals and the vice versa is also true.

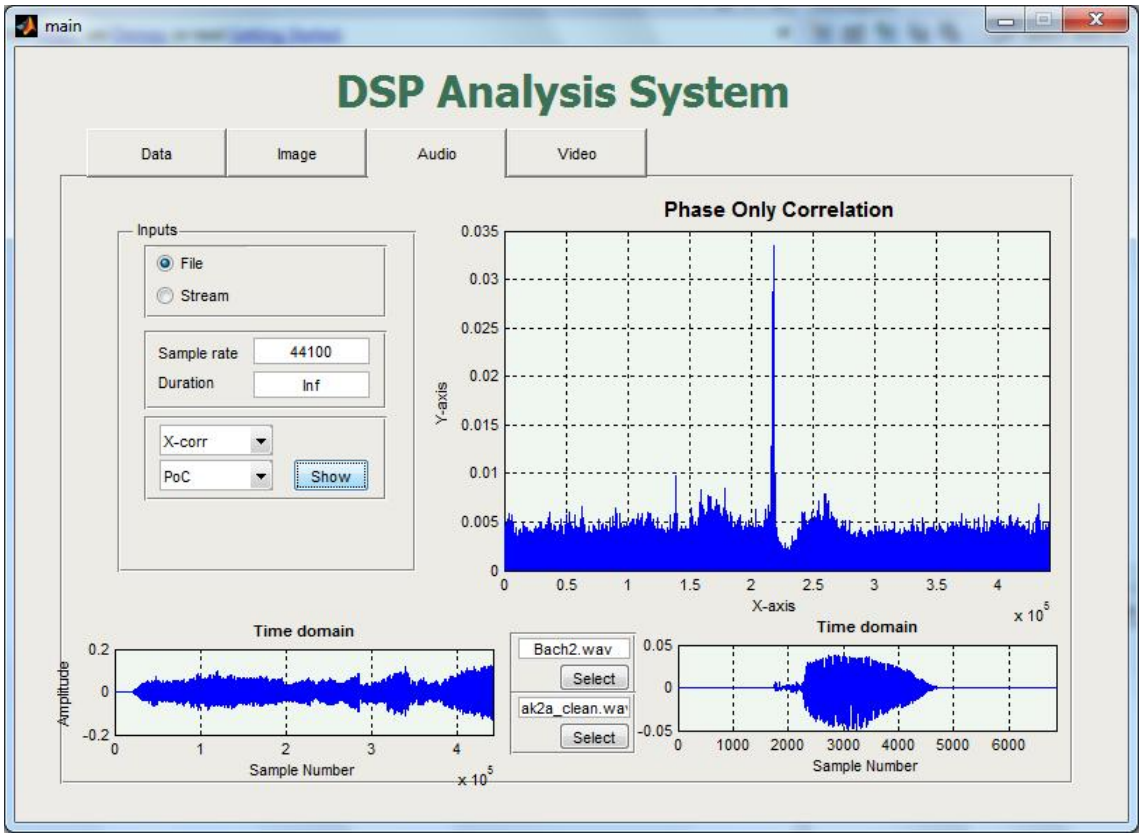


Figure 19. Phase-only correlation for two audio signals

Even though the spike is located at the center of the phase-only correlation in figure 19, there is no relationship between the two signals since there is a lot of noise at the bottom. The phase-only correlation was obtained by first carrying out the same steps for normalized correlation from selecting a single channel from each audio signal throughout to multiplying values of the FFT spectrum in the first signal with the respective values for the conjugate FFT spectrum in the second signal. The result was

then divided with the absolute values of the same results, the peaks values centered in the middle of the graph and the result then graphed on scale.

4.5 Cross-correlation of video frames

The cross-correlation of two video signals was obtained by correlating one video frame of the first video signal against the respective video frame for the second video signal. In situations where the number of frames for both videos were not the same, the correlation of frames for any of the two videos with the smallest number of frames was made. Figure 20 shows the normalized cross-correlation between two video frames.

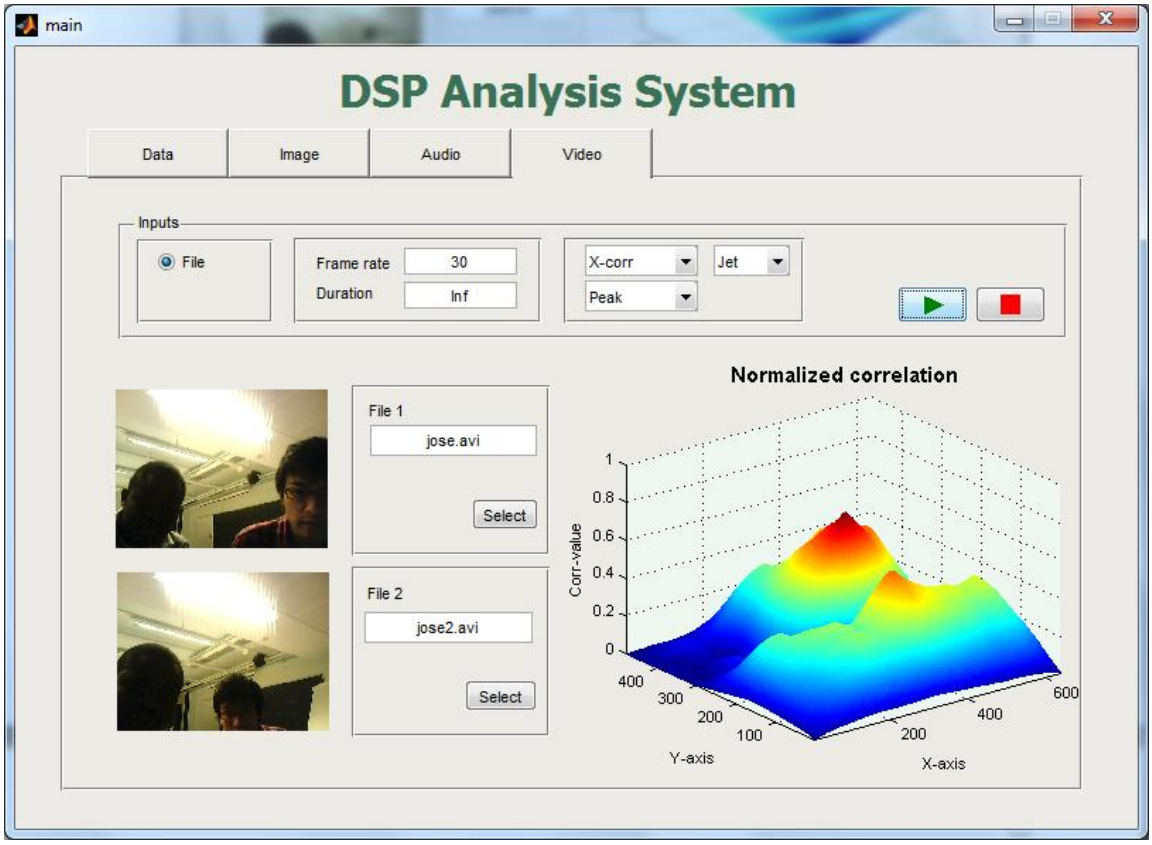


Figure 20. Normalized cross-correlation of two video frames

The correlation of an image against all the video frames of a single video file was also obtained. Images were also correlated with video frames acquired directly through the stream. The correlation between two frames acquired directly from the stream was also obtained to help detecting sudden changes in time. The current video frame was always correlated with the previous video frame and a sudden difference in any two adjacent video frames was indicated in the phase-only correlation graph as noisy.

4.6 Spectrogram of an audio signal

The spectrogram of both the audio signal and data samples was also obtained with the help of the Matlab function *spectrogram*. The spectrogram in figure 21 was obtained by representing various spectra on the same graph. It displays the level of the signal at a given time and the frequency as a colour or in gray scale. The spectrogram shows how the spectral density of a signal varies with time.

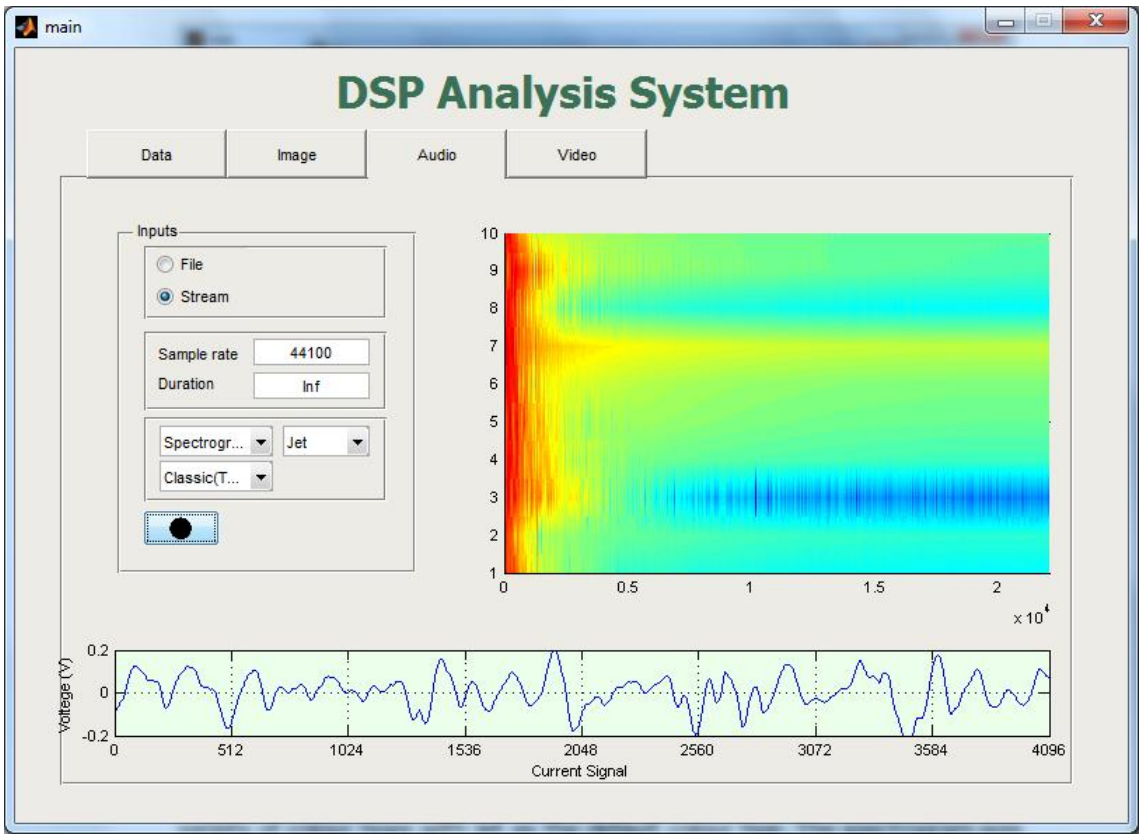


Figure 21. Top view of a spectrogram for an audio signal

The time is displayed on the vertical axis and the frequencies along the horizontal axis of the graph that is located on the top right hand side of figure 21. A variety of colour hues with jet as the default colour hue were used. The spectrogram for both still files and samples acquired through the stream was obtained. Figure 21 shows the top view of the spectrogram that was obtained from the samples acquired through the stream.

The spectrogram was also displayed in the mosaic view form by grouping small pixel samples to form a big rectangular area. Each rectangular area matches with the average color value of that area as presented in the graph shown on the top right hand side of figure 22.

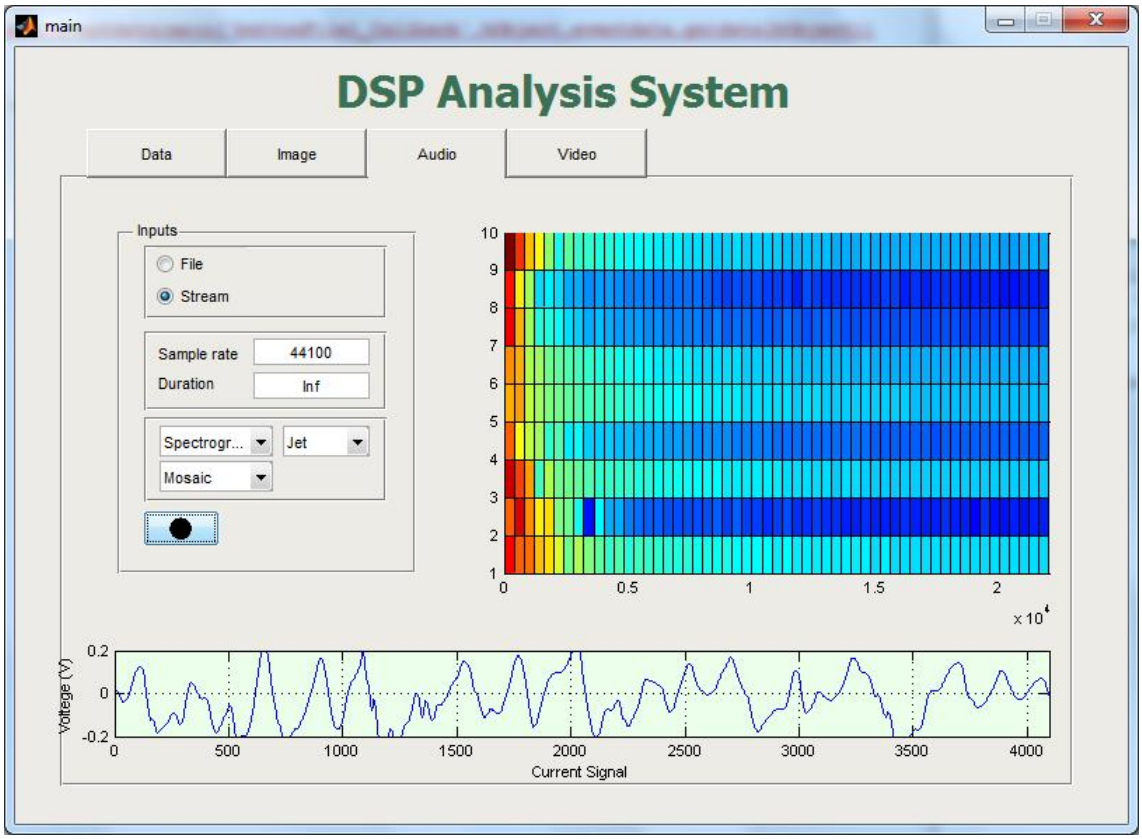


Figure 22. Top view mosaic of a spectrogram for audio samples

The spectrogram was also displayed on the graph in several other forms. The examples include a waterfall, rotation and as a 3-D format.

4.7 Sonar

The sonar was accomplished by using the same concept that a sonar or radar system uses. The system was accomplished for only 32-bit Matlab programing environment. A sound pulse was first released, then a timer started and detecting of the return pulses started. The timer was stopped immediately when the pulse that resembles the original sound pulse was detected. This gave the time delay that the pulse had to travel to and from the far end. The time delay was obtained by taking the difference between the time when the original pulse was released and the echo pulse was detected. The time delay was then displayed with a spike in the graph as seen in figure 23.

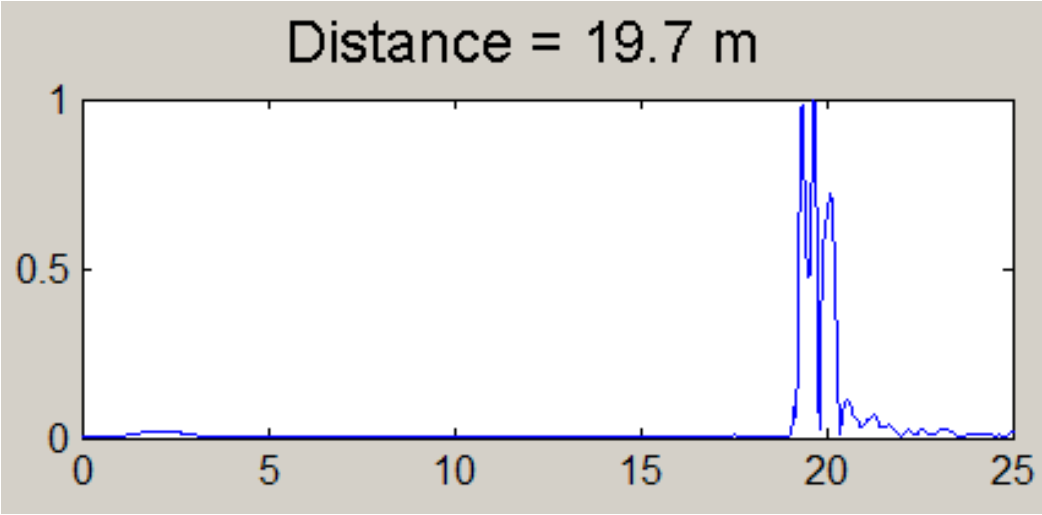


Figure 23. Time difference between original sound pulse and return pulse

The display unit takes a variety of forms but in general it is designed to present the received information. The most basic display that takes the form of time delay versus amplitude was used as seen in figure 23. The vertical axis shows the strength of the return pulse while the horizontal axis shows the delay time. This form provides no information about the direction of the target. The range was obtained by multiplying the time delay by the speed of sound divided by two. The factor of two indicates that the pulse travels to and from the far end. One of the example applications of the sonar is a fish finder.

5 Discussion

Developing software using Matlab programming language is faster once one has mastered the language well. It is a good tool that one can use to develop prototypes for any piece of software. Many computers found at the Metropolia University of Applied Sciences had the Matlab programming environment installed on them which made the work easier.

Audio samples were acquired through the stream from either the sound card or the microphone. The algorithm for acquiring audio samples from the 32-bit Matlab programming environment was done separately from that of the 64-bit Matlab programing environment. Since the current computer systems available are of either 32-bit or 64-bit processor architecture, there was need to cater for the users with either of the two computer processor architectures. This doubled the amount of work since there was need to cater for both the 32-bit and 64-bit Matlab programming environment.

The implementation of the algorithm for acquisition and analysis of audio samples at the same time from the 32-bit Matlab programming environment was much easier as compared to that of the 64-bit Matlab programming environment. The 32-bit Matlab programming environment uses the *analoginput* and *analogoutput* Matlab functions that come with the data acquisition toolbox while the 64-bit Matlab programming environment uses the *audiorecorder* Matlab function to acquire audio samples through the stream. However, the *analoginput* and *analogoutput* Matlab functions do not exist in the 64-bit Matlab programming environment. Once the two functions have been started, samples are collected continuously, analyzed and graphed in a desired format on real time without stopping the functions. The two functions are only stopped if the user stops the process of analyzing the signal.

The 64-bit Matlab programming environment that uses the *audiorecorder* Matlab function to acquire data from the microphone. The data from the *audiorecoder* variable requires stopping the *audiorecorder* function before data is analyzed and graphed. This makes it difficult and complicated to acquire as well as analyze data at the same time.

The data acquired from the sound card or microphone is stored as a vector array in the Matlab programming environment. The technique that was used to implement the algorithm that arranges the small signals containing data on the 64-bit Matlab programming environment during analysis was to collect the vector arrays of data in the form of an array. The array containing these vector arrays was re-arranged to form one vector array at the end when the analysis of audio signals in real time was stopped. The small vector arrays were concatenated to form one vector array and this was saved on the computer as a single WAV file.

The programming method used mostly was sequential programing. In some situations, the software application took a lot of time to generate the results. The algorithms for spectrum, spectrogram and the cross-correlation were implemented using sequential programming. Since the Matlab functions responsible for generating these results were a little slow, there was need to use parallel programming to improve the quality of the product. However, in some cases parallel programming was used.

Sub-functions were used to allow reusability of the components which helped to reduce the amount of code for the whole software system. The same sub-function was used to generate the spectrum of an image and a video frame. However, object oriented programming styles were not used since it took some time for the researchers to learn how to use the Matlab programing language. The type of programming style that was used was not object oriented. This might be taken into consideration while improving this prototype in the near future.

The results were positive and convincing enough for the future developments. A simple and easy to use GUI was designed to make the user's life easy. The designed GUI may not be the best design but it was the best possible end-product to be created. The researchers gained a lot in the project since the majority of the things they dealt with to make project were new to them. There were high interests in finding ways of improving the quality of the product that was designed. The results encouraged the researchers to continue working on the product as well as acquiring more knowledge in the field of DSP. Based on the results, I believe that the product can be developed further using better programming tools and concepts. Programming languages that include assembly, C and C++ among others, can be used to improve the quality of the

tool. The GUI may also be redesigned to make it look better as well as to reduce its size.

The strength of this project is the researcher's experiences to deal with a variety of new concepts in the field of Digital Signal Processing. The idea of the project might be similar to that of other researchers` although the experiences encountered in this project are unique. The thesis was organized with clear theories. The software was tested and I believe that a better product will be developed in the future. With the experience and knowledge that is obtained from the project, more research will continue in the field of Digital Signal processing.

6 Conclusion

Digital Signal Processing involves processing digital signals that are obtained by sampling analog signals and the subsequent reconstruction of signals from their samples. Digital signal processing finds its applications in many areas especially where the electronic devices are used ranging from home use to industrial use. DSP involves analyzing different signals in various ways that include converting the time domain signal to the frequency domain. Examples of the signals that are analyzed include images, audio samples and video frames.

The goal of the project was to design an innovative analysis tool for Digital Signal Processing (DSP) mostly for acquiring different types of signals from the required hardware devices as well as analyzing signals. The tool is a small graphical user interface that captures audio samples through the microphone or sound card, images files and video frames via the webcam. In addition, the tool analyzes these signals by displaying the spectra, spectrogram, and correlation of any two signal types. This tool displays the spectrum both in their linear and logarithmic form. It displays the spectrogram of both data and audio samples. The audio samples are either acquired from the audio files or directly through the stream. It analyzes the samples using the Fast Fourier Transform approach.

A wide range of signals exist in the day-to-day living. These signals can be analyzed by converting the time domain signal to the frequency domain to obtain a spectrum. It turns out that viewing a signal in the frequency domain is usually a lot more useful than viewing it in the time domain. The frequency domain shows the strength of a signal. The Fast Fourier Transform turns out to be the best way to achieve the spectrum of any signal. The spectrum can be manipulated in different ways to achieve spectrograms. Any two signals can also be compared to find the similarity by finding the correlation between them.

References

1. Ashok A. Digital signal processing a modern introduction. Toronto, Canada: Chris Carson; 2007.

2. Bateman A, Iain PS. The DSP handbook algorithms, applications and design Techniques. Edinburgh Gate, Harlow: Pearson Education Limited; 2002.

3. Croft A, Davison R, Hargreaves M. Engineering mathematics. Tottenham Court Road, London: Pearson Education Limited; 2001.

4. Emmanuel CI, Barrie WJ. Digital signal processing, a practical approach, second edition. Edinburgh Gate: Pearson Education Limited; 2002.

5. Huang J, Tan T, Ma L. Phase correlation based iris image registration model, volume 20 [online]. Beijing, China: National Laboratory of Pattern Recognition. May 2005.
   URL: http://ranger.uta.edu/~huang/papers/JCST05.pdf. Accessed 5 April 2012.

6. Johnson D. Spectrograms [online]. IBC Plaza. Houston; Connexions; 8 August 2009.
   URL: http://cnx.org/content/m0505/latest/. Accessed 20 April 2012.

7. Kenton W. Digital signal processing world class designs. Burlington, MA: Elsevier; 2009.

8. Kongsberg Defence and Aerospace AS. Submarine sonar processing suit [online]. Norway: Gruppen; January 2009.
   URL:http://www.kongsberg.com/en/kds/products/navalsystems/submarinesystems/~/media/KDS/Files/Products/Naval%20Systems/Submarine%20SPS_1.ashx. Accessed 14 April 2012.

9. MathWorks. Image processing toolbox [online]. Natick, Massachusetts, United States of America; 2012.
URL: http://www.mathworks.se/help/toolbox/images/ref/rgb2gray.html. Accessed 20 April 2012.

10. MathWorks. Image processing toolbox. [online]. Natick, Massachusetts, United States of America; 2012.
URL:http://www.mathworks.se/products/demos/image/cross_correlation/imreg.html. Accessed 20 April 2012.

11. Robert EZ. A fresh look at 'broadband' passive sonar processing [online]. Arlington VA, United States of America:
URL:http://traktoria.org/files/sonar/signal_processing/A_Fresh_Look_At_%27Broadband%27_Passive_Sonar_Processing__Zarnich_ASAP99.pdf. Accessed 05 April 2012.

12. Sanjit KM. Digital Signal Processing, A computer-based approach Third Edition. New York: McGraw-Hill; 2006.

13. Steven WS. Digital signal processing, A practical guide for engineers and Scientists. United States of America: Elsevier; 2003.

14. Stephen WS. The scientists and engineers guide to digital signal processing [online]. San Diego: California Technical Publishing; 2011.
URL: http://www.dspguide.com/ch12/2.htm. Accessed 10 April 2011.

15. Symes P. Digital video compression. New York, NY; McGraw-Hill; 2004.

16. Umbaugh ES. Computer imaging, digital image analysis and processing. Florida, Boca Raton: CRC press, 2000; N.W. Corporate Blvd; 2005.

17. Usher O. Introduction to image processing [online]. Germany: Hubble; 2012.
URL:http://www.spacetelescope.org/static/projects/fits_liberator/image_processing.pdf. Accessed 27 March 2012.

A function that finds the spectrum of audio and data samples

```
% Calculate the FFT of the data.
function [f, flog, mag, lin] = fftcalc(d, fs, preview)
% Calculate the FFT of the data.
xFFT = fft(d);
xFFT = abs(xFFT);

% Avoid taking the log of 0.
index = xFFT == 0;
xFFT(index) = 1e-17;

lin = xFFT(1:preview/2);

mag = 20*log10(xFFT);
mag = mag(1:preview/2);

f = (0:length(mag)-1)*fs/preview;
f = f(:);

flog = log(f);
end
```

A function that finds the spectrum of images and video frames

```
% --- FFT calclation for Image and Video
function [ffts, fftsLog] = imagefft(img)
% --- Get the image and perform FFT
dim = ndims(img);
if dim == 3
    img = rgb2gray(img);
end
fftimg = fft2(img);             % Perform FFT
ffts = fftshift(fftimg);        % FFT shift
fftsLog = log2(ffts);                   % Perform logarithm
end
```

A function that finds the normalized and phase-only correlation of two audio or data signals

```
% --- Correlation Function for Audio
function [resulta, resultb, ra, xa, rb, xb ] = calcaudiocorr(a1,
a2, handles)
a1 = a1(:, 1);                                  % Choose only 1-
channel
a2 = a2(:, 1);
samplesa1 = length(a1);                         % Find the
number of samples
samplesa2 = length(a2);

% If these files have different number of samples, padding zero.
if samplesa1 > samplesa2
    a2(samplesa1, 1) = 0;
elseif samplesa1 < samplesa2
    a1(samplesa2, 1) = 0;
end

R = fft(a1).* conj(fft(a2));    % Calculate 'R'

resulta = ifft(R);              % This is only for Normalized
Corr.
resultb = ifft(R./ abs(R));     % This is only for Phase Only
Corr.

resulta = fftshift(resulta);    % Shift the peak to center
ra = max(max(resulta));         % Find peak value (It shows
similarities between both of files.)
xa = find(resulta == ra);       % Find peak location (It shows
shift length between both of files.)

resultb = fftshift(resultb);    % Shift the peak to center
rb = max(max(resultb));         % Find peak value (It shows
similarities between both of files.)
xb = find(resultb == rb);       % Find peak location (It shows
shift length between both of files.)
end
```

A function that finds the normalized correlation of two images or video frames

```
% --- Normalized Correlation for Image and video
function [result, r, x, y] = corrcalc(img1, img2)
dim1 = ndims(img1);
if dim1 == 3
    img1 = rgb2gray(img1);
end
dim2 = ndims(img2);
if dim2 == 3
    img2 = rgb2gray(img2);
end
R = fft2(img1).* conj(fft2(img2));
result = ifft2(R);
result = fftshift(result);
r = max(max(result))          % Peak value
[y x] = find(result == r)     % Find peak location
end
```

A function that finds the phase-only correlation of two images or video frames

```
% --- Correlation Function for Audio
function [resulta, resultb, ra, xa, rb, xb ] = calcaudiocorr(a1,
a2, handles)
a1 = a1(:, 1);                                  % Choose only 1-
channel
a2 = a2(:, 1);
samplesa1 = length(a1);                         % Find the
number of samples
samplesa2 = length(a2);

% If these files have different number of samples, padding zero.
if samplesa1 > samplesa2
    a2(samplesa1, 1) = 0;
elseif samplesa1 < samplesa2
    a1(samplesa2, 1) = 0;
end

R = fft(a1).* conj(fft(a2));    % Calculate 'R'

resulta = ifft(R);              % This is only for Normalized
Corr.
resultb = ifft(R./ abs(R));     % This is only for Phase Only
Corr.

resulta = fftshift(resulta);    % Shift the peak to center
ra = max(max(resulta));         % Find peak value (It shows
similarities between both of files.)
xa = find(resulta == ra);       % Find peak location (It shows
shift length between both of files.)

resultb = fftshift(resultb);    % Shift the peak to center
rb = max(max(resultb));         % Find peak value (It shows
similarities between both of files.)
xb = find(resultb == rb);       % Find peak location (It shows
shift length between both of files.)
end
```