A COMPUTATIONAL MODELLING FRAMEWORK FOR TIME-FREQUENCY SIGNAL REPRESENTATIONS

By

Cesar A. Aceros Moreno

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS

December, 2010

Approved by:

Nestor Rodriguez, Ph.D. Member, Graduate Committee	Date
Kejie Lu, Ph.D. Member, Graduate Committee	Date
Domingo Rodriguez, Ph.D. President, Graduate Committee	Date
Darrell Hajek, Ph.D. Representative of Graduate Studies	Date
Erick E. Aponte, Ph.D. Associate Director of the ECE Department	Date

Abstract of Dissertation Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Science

A COMPUTATIONAL MODELLING FRAMEWORK FOR TIME-FREQUENCY SIGNAL REPRESENTATIONS

By

Cesar A. Aceros Moreno

December 2010

Chair: Domingo Rodriguez
Major Department: Electrical and Computer Engineering

This thesis presents an open source computational tool framework for the visualization and analysis of signals with time-dependent spectral content. SIRLAB (SIgnal Representation LABoratory) is the name given to this tool framework written in C-language for a Linux environment and using the OpenCV (Open Source Computer Vision) platform, a software library of programming functions for near-real-time computer vision application development. SIRLAB was initially developed as an application tool kit for environmental surveillance operations pertaining to acoustic monitoring of birds, amphibians, and aquatic animals. In this setting, it receives acoustic raw signal-data and it produces ordered sets of spectrogram frames which may be presented in a streaming video format due to its fast computation. Computer speed ups of more than 30 times have been reached when compared with MATLAB implementations utilizing the same computational resources and algorithm formulations. This allows to produce streaming video with a frame rate of 30 frames per second, for some applications, reaching the ATSC digital television frame rate standard.

Resumen de Disertación Presentado a Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de Maestría en Ciencias

UN MARCO DE MODELAMIENTO COMPUTACIONAL PARA REPRESENTACIONES DE SEÑALES TIEMPO-FRECUENCIA

Por

Cesar A. Aceros Moreno

Diciembre 2010

Consejero: Domingo Rodriguez
Departamento: Ingeniería Eléctrica y Computadoras

Esta tesis presenta un marco computacional de código abierto para la visualización y analisis de señales con contenido espectral dependiente del tiempo. SIRLAB (Laboratorio de representacion de señales) es el nombre dado a esta marco computacional escrito en lenguage C para ambiente Linux y usando la plataforma de OpenCV, una librería de software para desarrollo de aplicaciones en visión de computadoras para procesamiento cercano a tiempo real. SIRLAB fue inicialmente desarrollado como un juego de herramientas para operaciones de monitoreo ambiental asociado al monitoreo acústico de aves, anfibios, y animales acuaticos. En esta aplicaciones, SIRLAB recibe señales acústicas en datos crudos y produce conjuntos ordenados de espectrogramas los cuales pueden ser presentados en un formato de video debido a su rápida computación. Mejoras en la computación de hasta 30 veces con respecto a las implementaciones de Matlab han sido alcanzadas utilizando los mismos recursos computacionales y formulaciones de algoritmos. Esto permite producir video a una tasa de 30 cuadros por segundo, para algunas aplicaciones, alcanzando estándares de televisión digital (ATSC).

To my wife, daughter, and my parents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

This thesis deals with the development of a computational modelling framework (CMF) named SIgnal Representation LABoratory (SIRLAB). This CMF is intended to time-frequency signal representation. The computational framework is intended to provide a open source library with a set of computational tools for signal processing. Before to introduce the computational framework we must lay some foundations of signal processing. A generalized signal processing framework was presented in the MsC Thesis of Yuji Yunes [1] where physical signals are sampled and pass thru appropiate time observation windows, in order to produce one dimensional finite discrete signals evaluable for further treatment, see Figure 1–1.

The group of signals of interest exist in the space of continuous physical signals $L(\mathbb{R})$ (see Figure 1–1). Prior to use these signals in digital signal processing we must sample them and then use windowing techniques to obtain a discrete finite sequence. In other words, the signal passes from $L(\mathbb{R})$ to $l(\mathbb{Z})$ through a sampling process. The sampled version then is cut from $l(\mathbb{Z})$ to $l^2(\mathbb{Z}_N)$ when a windowing operator is applied. The new signal is in $l^2(\mathbb{Z}_N)$. This is the space of discrete finite signals with finite energy. Finally, through the use of time-frequency tools it is possible to map signals in the space $l^2(\mathbb{Z}_N)$ to the space $l^2(\mathbb{Z}_N \times \mathbb{Z}_N)$. The increased dimensionality of the signal space reveals more information about the original signal. However more information introduces an increment in the computational power required for

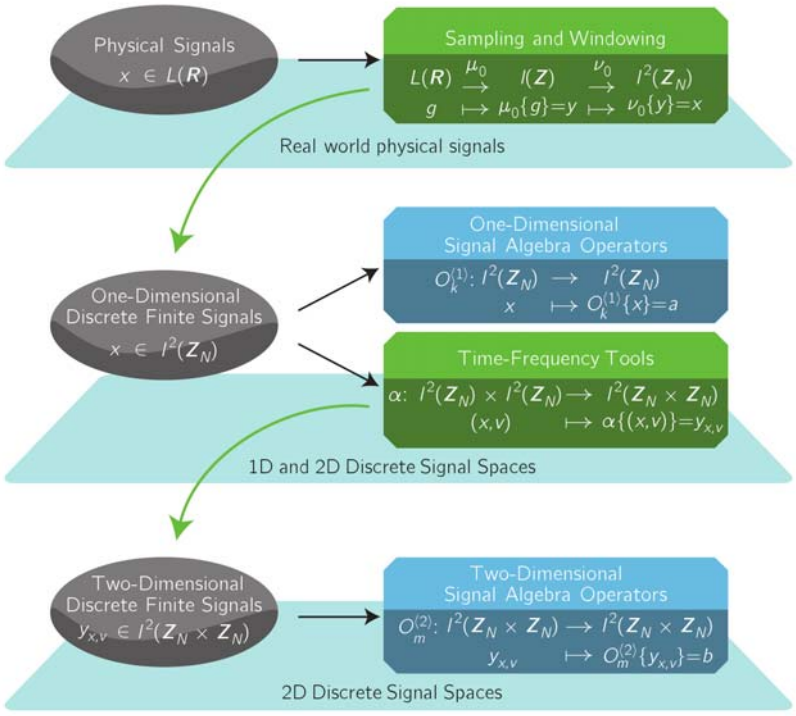


Figure 1–1: A general approach for acoustic signal analysis

the signal processing analysis.

For the computation of signals, we start at the signal space $l^2(\mathbb{Z}_N)$ where N is the length of the signal acquired. The CMF proposed will provide primitives for the development of time-frequency tools to pass from the one-dimensional space to a higher dimensional spaces. The short time fourier transform (STFT) and the discrete chirp fourier transform (DCFT) are only two in a large group of signal operators.

The CMF is expected to be a useful tool to process signals and perform information extraction from signals acquired from the physical world. In the case of this project we are specially interested in bioacoustical signals acquired by hydrophones and acoustic data from anurans and birds. The bioacoustic input data is usually

stored in an uncompressed format as waveform audio file format (WAV or WAVE) files. The CMF output data will be a series of frames or images built from data provided using the signal operator (STFT or DCFT in this project) selected, or a video created based on the images previously mentioned.

We envision the computational framework as a signal processing environment that takes the signals acquired and, using signal operators, delivers the output data in a standard output format.

The increase in data acquisition of real-world signals is becoming a major problem due to the exorbitant amount of data available. This rise is linked to a high storage capacity and high computing capacity requirements of the collected data. The high computing capacity is most important because of its "adequate" processing depends on the quality of information that can be extracted. The development of strategies to select in a signal the useful information and discard that which is not important has been and is a hard problem in the field of signal processing. The pertinence of this thesis is shown in (based in a concept by Stephen M. Griffin) Figure 1–2 that depicts the stages to reach knowledge understanding.

A useful and well known strategy in signal processing is to make a mapping that increases the dimensionality of the signal space. In this project, we map signals from 1-D signal spaces to 2-D signal spaces through the STFT and/or DCFT. This mapping process deals with images as an end product. Is a fact that we are living in a world where images and videos are becoming important not only in the scientific world but also in diverse fields such as military, biology, communications, video-games, etc.

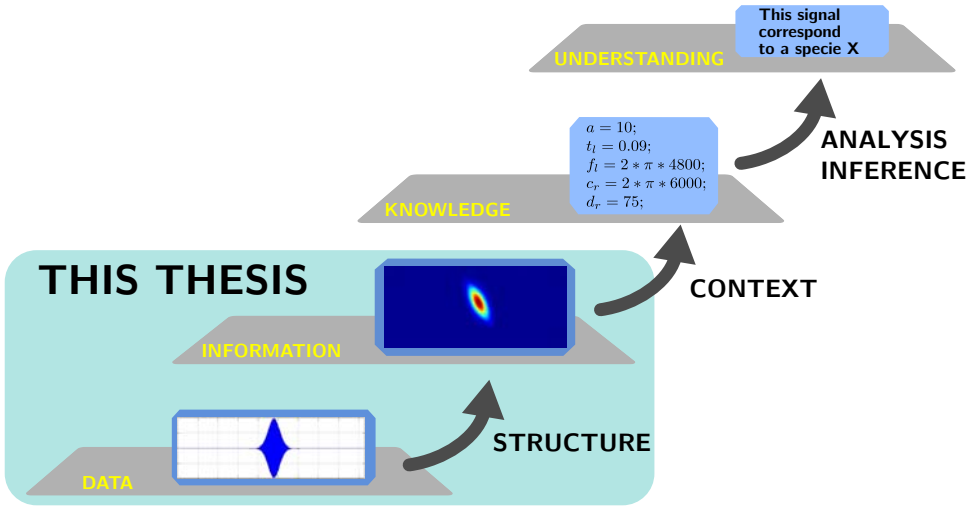


Figure 1–2: Evolution of Understanding

The 21st century is the century of biology. A new kind of biology mostly focused in sophisticated techniques based in mathematics, physics, and chemistry. Many of the tools to procure new discoveries in biology are and will be established under novel tools for signal processing. These new instruments are strongly based in computers, mathematics, and combined with an improved understanding of how to model the phenomena involved in the source, channel and receiver relation. The modelling of biological systems is a hard problem. With this work we are in the development of appropriate tools that support ways to differentiate things that are useful from those are not. The time-frequency analysis is a way to do this separation.

The information extraction from large real world data have been traditionally a demandant process in terms of man's job, computational processing and expertise of the persons involved. This project pays special attention to the development of a really automated information processing system that goes from signal to images or movies. These outputs might contain valuable information that could be relevant to the decision making process.

1.1 The Concept of Time-Frequency Signal Analysis

Time frequency representation is a modality for representing a signal in a 2-D format where the horizontal axis, or abscissa, usually represents the independent variable of time and the vertical axis, or ordinate, usually represents the independent variable of frequency. The value of the representation is then given by the third axis (See Figure 1–3).

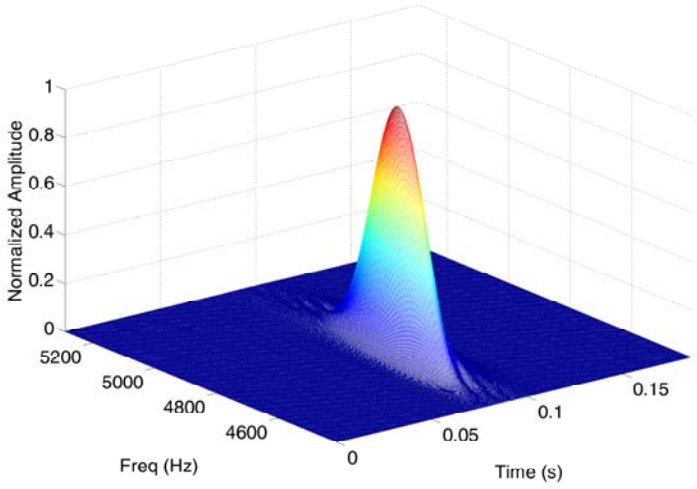


Figure 1–3: Chirplet 3D Representation

Another representation format commonly used is based on a color scheme where the color value describes the magnitude or intensity of the representation (see Figure 1–4).

There exist many signals whose frequency or spectral content changes with time. These signals are best analyzed using time-frequency signal analysis tools. Examples of these tools are the short time fourier transform (STFT), the Wigner distribution (WD), the ambiguity function (AF), and the discrete chirp fourier transform (DCFT). The main objective of this thesis is the formulation of a computational modelling framework (CMF) where these time-frequency tools will be implemented

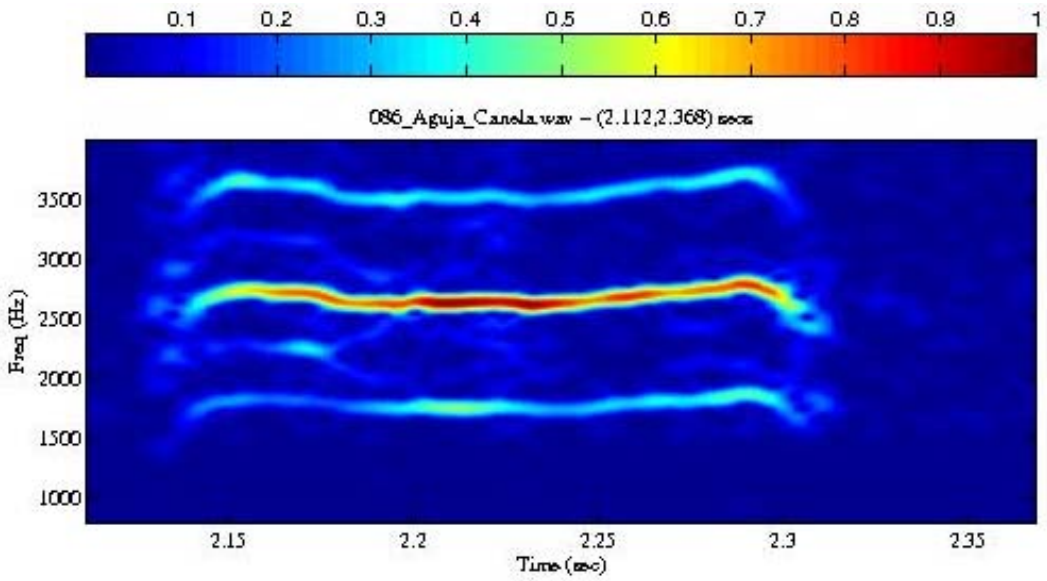


Figure 1–4: Aguja Canela STFT Representation using Matlab

in an integrated manner with an appropriate user interface. The CMF provides additional features such as the reception of acoustic raw signal-data and the production of ordered sets of spectrogram frames which may be presented in a video format. It is envisioned that this CMF will be endowed with the following attributes:

1. Signal processing primitives built-in.
2. Data acquisition using standard sound input.
3. Open-source code available for signal processing applications.
4. Stochastic analysis tools implemented.
5. Change detection capabilities.

Figure 1–5 shows two signals whose spectral contents change with time. Each signal is composed of two time segments. Each time segment corresponds to a different sinusoidal frequency. For example, in the first signal the first segment has a sinusoidal frequency of 300Hz and the second segment has a sinusoidal frequency of 100Hz.

The second signal has as a first segment a sinusoidal frequency of 100 Hz and a second segment with a sinusoidal frequency of 300 Hz. It can be noticed that the Fourier transform of both of these signals is exactly the same, as depicted in Figure 1–5. From this point of view it can be determine that the Fourier transform cannot be used as a discrimination tool to distinguish these two signals. Here is where the concept of time-frequency signal analysis becomes instrumental. Time-frequency signal analysis tools are computational tools designed to discriminate time-frequency signals. A commonly used time-frequency tool is the short-time Fourier transform (STFT). The STFT is the magnitude of a windowed Fourier transform of a given signal. The window is usually smaller than the signal and is displaced or shifted throughout the entire signal.

Using the STFT in the particular example presented in Figure 1–5, it is possible to apply a time shift window to discriminate the frequency content present in the signal at different time shifts. Figure 1–6 shows the STFT output as an image for each of the two signals mentioned in Figure 1–5. We can see that the STFT is a tool that can be used to discriminate time-frequency signals. Time-frequency tools, in general, are designed to possess diverse computational processing techniques to extract individual attributes, features, or characteristics associated with a signal.

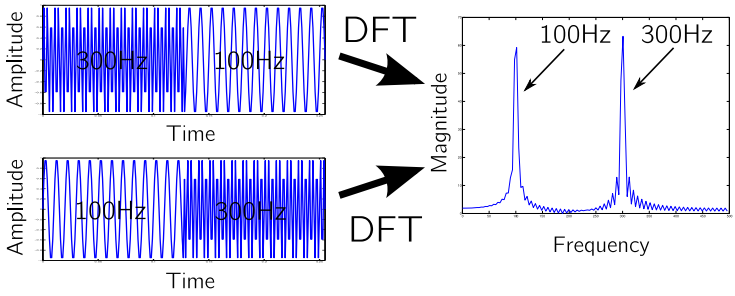


Figure 1–5: Two Signals DFT Transform

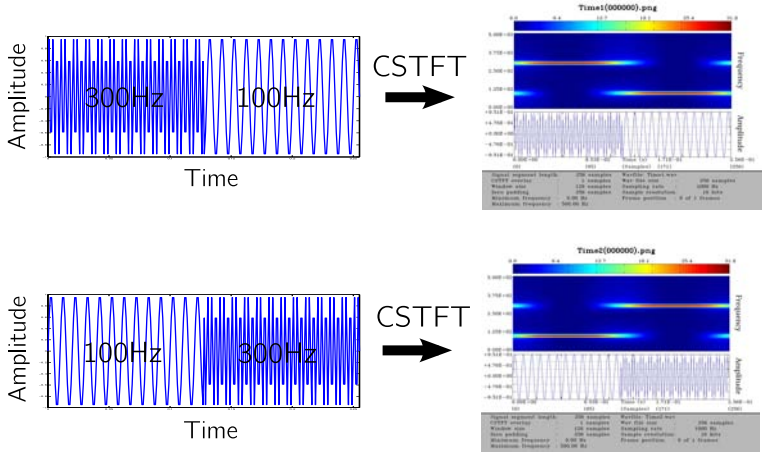


Figure 1–6: Two Signals STFT Transform

CHAPTER 2
BACKGROUND RESEARCH AND PREVIOUS WORKS

Prior to enumerate the works mentioned, we consider important to introduce the concept signal-channel-targets interaction. This concept appears in a formulation developed as the scattering channel model (see Figure 2–1) by Prof. Domingo Rodriguez. This model takes input signals from the multicomponent polynomial phase signal subspace. These signals interact with the scattering media producing a new multi-component polynomial phase signals. The channel attributes are encoded in the polynomial phase signal parameters and a parameter estimator may be used to extract these parameters. The parameter array space can be treated to extract further intelligence. In this thesis we build a computational framework that enables the implementation of estimators for extraction of elements in the parameter array subspace.

Many works have been made to process signals using multiple signal transformations in order to extract parameters and information. The proposed computational framework gives a first step toward formulation of signal processing operators as mathematical entities. These tools will allow formulate the signal-channel-targets interaction in a novel form. As a consequence of this the formulation of channel models, information extraction, and decision making using mathematical operators

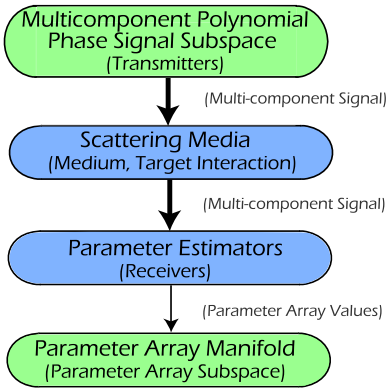


Figure 2–1: Scattering Channel Model

is possible.

Time-frequency analysis is a powerful tool that has been widely studied, but continues providing new ways to handle the signals and extract their information. The Ervin Sejdic et al. [2] work is focused on the study of the concentration of energy to perform feature extraction and classification. The authors first give an overview of feature extraction techniques based in the Cohen's Class to perform time-frequency analysis and obtain the classification using the concept of energy concentration in the TF domain. The paper is a very detailed summary of the work in the time-frequency field in the last years.

The work of LJubisa Stankovic [3] proposes methods and algorithms suitable to be implemented in this thesis. Her work uses the Wigner distribution based in time-varying filtering and produces an improved way to detect multi-component and mono-component signals in noisy environments.

The noisy environment is a condition very common in EEG and this leads us to the paper of J. Zygierewicz et al. [4], where they develop a high resolution method to

produce time-frequency images, this method is suitable for implementation in the CMF of this thesis.

Antonio Costa and Stephan Hengstler [5] propose a method for using auto-regressive analysis for time-frequency images and thereby improve visualization and facilitate the extraction of parameters.

The research of Kathleen A. Lindlan [6] proposes a solution of how integrate database analysis with high performance scientific computing work to achieve the high requirements needed. This kind of work is of special importance especially when you are dealing with near real-time processing methods.

The work of Emmanuel J. Candes et al. [7], is centered on the problem of detecting one-dimensional signals from noisy measurements. The signal model is defined as $y_i = \alpha S_i + z_i$ with $i = \{1, \ldots, N\}$, where (S_i) are sampled values $S_i = S(t_i)$ of a signal of interest and each $S(t)$, $t \in [0, 1]$. (z_i) is a zero-mean stochastic vector. The authors use a set of chirplets to test the highest local approximation with the unknown input signal. With each approximation the path of the chirplet is detected and a chirplet graph is generated.

Another work is written by Xiang-Gen Xia [8]. This paper describes the discrete chirp Fourier transform (DCFT) a powerful but computationally demanding signal operator defined for a signal $x[n] \in \mathbf{Z}_N$ as

$$X_c[k, l] = \left(\frac{1}{\sqrt{N}}\right) \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi}{N}(ln^2+kn)} \tag{2.1}$$

The DCFT is particularly useful for a prime number signal length because in this case the estimation of parameters of multicomponent polynomial chirp signals is relatively easy. The parameters extracted are the information present in the chirp signals. The information allows us to infer characteristics and behaviors of the medium where the signal was propagated. The author also remarks the DCFT as a fractional fourier transform where the rotation angle is related to the variable l.

A huge and seminal paper in the area was written by Leon Cohen [9]. In this work the author summarizes fundamental ideas and methods of joint time-frequency distributions. Special attention is paid to Cohen's class formulation; given by

$$P(t,w) = \frac{1}{4\pi^2} \int\int\int e^{j(\theta u - \theta t - \tau w)} \phi(\theta,\tau) s(u + \frac{\tau}{2}) \cdot s^*(u - \frac{\tau}{2}) du \; d\theta \; d\tau \quad (2.2)$$

where the function $\phi(\theta,\tau)$ is the kernel function in the ambiguity domain and s is the signal. The Wigner-Ville, Choi-Williams, and Page distributions are some examples of distributions that belongs to Cohen class.

An interesting and recent work is made by Nicholas N. Bennett and Naoki Saito [10]. The authors develop a methodology to extract parameters of chirp signals using the Hough transform [11]. The authors present a collection of new algorithms which employ a local Fourier basis and variants of the randomized Hough transform to compute estimates for parameters of multiparameter chirps. The Hough transform is widely used as a feature extraction technique. It is also used in image analysis, computer vision, and digital image processing. The use of this transform has been evolved from the classical form oriented to line identification in images to a most sophisticated uses such as identifying positions of objects such as circles or ellipses.

OpenCV has implemented Hough transform tools for detection of ellipses. In conclusion this paper emphasizes on the usefulness of edge information in lowering the variance of parameter estimates and in decreasing the amount of spurious cross talk observed in the traditional Hough processing techniques.

Another work from Roberto E. González and Nelson D. Padilla [12] is oriented to detection of filaments in astronomy. In order to detect a filament, the authors construct a backbone linking two nodes, this provides a skeleton-like path connecting the highest local dark matter (DM) density traced by non-node haloes. The estimation of the characteristic DM density between two skeleton-candidate haloes use two approximations, the Voronoi tessellation density and a proxy of the minimum DM density between the two haloes assuming (NavarroFrenkWhite) NFW profiles. This application is akin to the time-frequency representation of signals. We can use similar methods to determine filaments in the images of the t-f operators. Figure 2–2 shows the results of four filaments detected using the methodology formulated in [12].

Along the same line is other work of Ery Arias-Castro et al. [13]. In this work the authors show the importance of the well known connect-the-dots (CTD) problem. The authors use Hölder Class and particularized Lipschitz graphs to develop a geometric approach to the problem of the CTD. An example of the results is shown in figure 2–3 where Lipschitz graphs are immersed in uniformly distributed points in the range $[0, 1]^2$. The basis for these results are in [14] and [15]. The most recent work of Arias-Castro is [16] where the authors expands the formulations of the papers [13, 14] to grayscale images and use beamlets networks in 2D and 3D to characterize

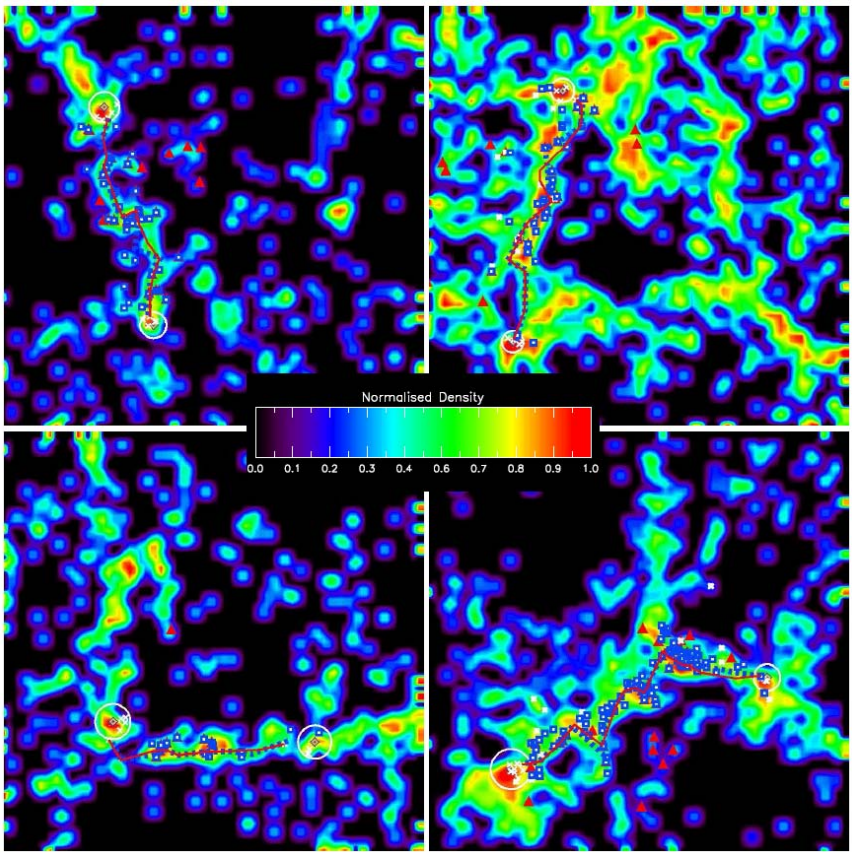


Figure 2–2: Four examples of detected filaments. The red solid lines show filament skeletons, the blue dashed lines show the re-centred skeleton.

filamentary structures in 3-D datasets and make analyzes of point clouds.

Another approach for information extraction is related with data mining in Cook and Holder [17]; in this effort the authors suggest the discovery and identification of substructures in structural data. The data mining considered uses two techniques: unsupervised pattern discovery and supervised concept learning from examples. The applications mentioned in this paper are CAD diagrams and chemical structures. The data mining techniques lack of scalability especially when problems involve large numbers of data and features.

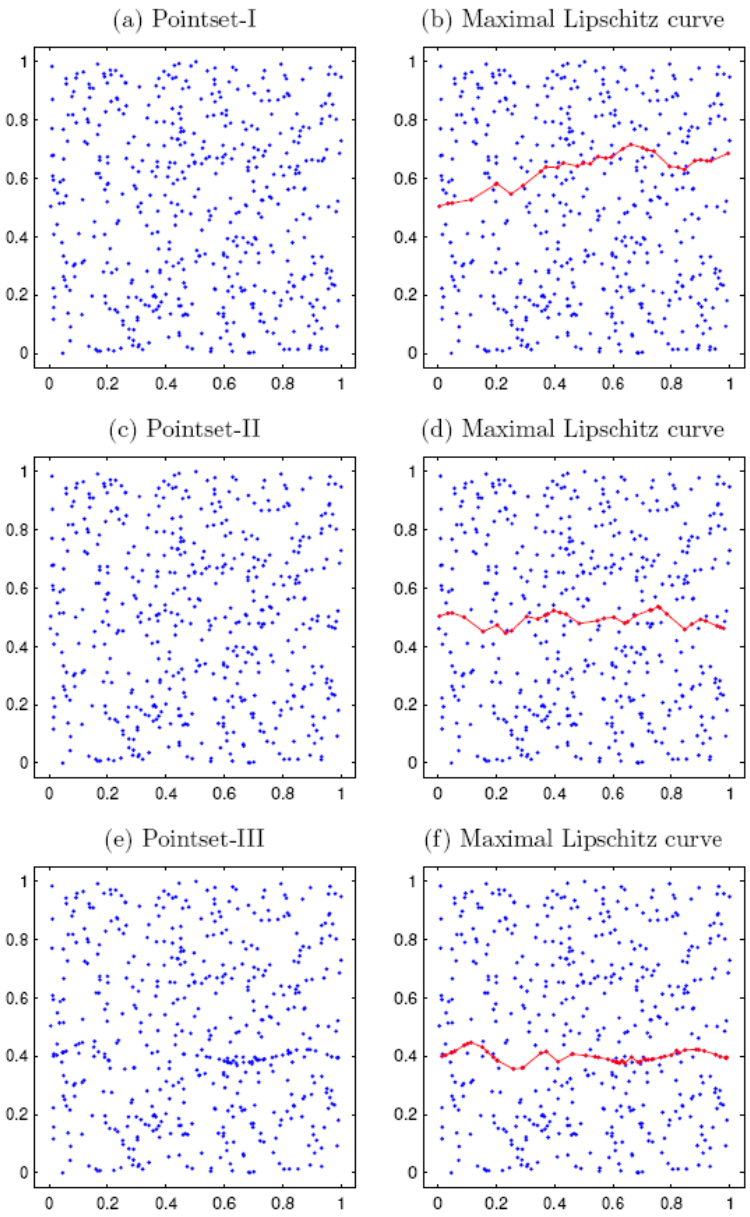


Figure 2–3: Examples for Connect-The-Dots. Three examples of scattered points in [0; 1], and the corresponding CTD solution. In each case the class Γ of curves is the set of Lipschitz graphs. Panels (a),(c), and (e) show the pointsets only, while panels (b), (d), and (f) show the maximal Lipschitz curves, with 31; 35; 54 points respectively-out of $n = 500$ points total. Panel (a) shows a uniformly distributed random pointset. Panels (c) and (e) show clouds with a small number of points on a Lipschitz curve, in addition to uniform random points. It seems unlikely that visual inspection would detect non-uniform structure in (c) (compare (a)); however, a statistical test based on CTD counts can reliably establish its presence.

A German laboratory, MMER, of the Technische Universität München, in the paper [18] develops a computational framework for video applications. The authors base the CF on the same open source libraries that we are planning to use in this project. The main purpose of the paper is the generic assembly of processing chains to shows applications on the area of video analysis and pattern recognition. They focus the development to the use of multi-core CPU architectures as a way to improve the performance of the applications.

The work in [19] is oriented toward the design of ultrahigh strength, high toughness steels. The authors create a computational framework for metallurgical application. The CF was developed to help in the design of super steels. This work is a good example of a computational framework for an specific purpose.

The Prof Adam W. Bojańczyk of Cornell Univesity proposed an interesting project [20, 21] oriented to the parallelization of space-time adaptive processing (STAP) structures. He which refers to adaptive radar processing algorithms that take the signals from both multiple sensors to cancel interferences and detect a target. The Algorithmic Library for Parallel STAP (ALPS) takes advantage of the arrival of parallel processing as a way to overcome the computational complexity of the time-frequency operations. Unfortunately the project ALPS was not longer developed and these references are the last known about it.

CHAPTER 3
SIRLAB THEORETICAL FRAMEWORK

This chapter presents the theoretical formulations associated with the SIRLAB framework. A mathematical signal processing framework must be implemented along the basis of the concept of a signal linear algebra. Concepts of signal algebra definitions have been treated in the thesis of Cesar Aceros of 2005 [22]. In addition, this type of framework requires as part of its background concepts pertaining to signal classification and linear vector spaces which can be found in the works of Yuji Yunes [1] and Dilia Rueda-Serrano [30]. The chapter continues describing the different types of mathematical tools used to treat these signals in order to extract the relevant information important to an information user.

3.1 Time Frequency Signal Operator Formulation

3.1.1 Cyclic Ambiguity Function

The ambiguity function is useful for time delay and Doppler estimation as well as for modeling point target response functions and it is the map

$$\begin{aligned} A: \quad l^2(\mathbb{Z}_N) \times l^2(\mathbb{Z}_N) &\rightarrow l^2(\mathbb{Z}_N \times \mathbb{Z}_N) \\ (f, g) &\mapsto A_{\{f,g\}}[m, k] \end{aligned}$$

where,

$$A_{\{f,g\}}[m, k] = \sum_{n \in \mathbb{Z}_N} f[n] g^*[\langle n + m \rangle_N] W_N^{kn}$$

and $W_N^{kn} = e^{-\frac{j2\pi}{N}(kn)}$, $m \in \mathbb{Z}_N$ and $k \in \mathbb{Z}_N$.

3.1.2 Wigner Distribution

The discrete Wigner transform of a signal f is the map

$$\begin{array}{rccc} W: & l^2(\mathbb{Z}_N) \times l^2(\mathbb{Z}_N) & \rightarrow & l^2(Z/2N \times Z/2N) \\ & (f, f) & \mapsto & W_{\{f\}} \end{array}$$

where,

$$\begin{aligned} W_{\{f\}}[n,k] &= \frac{1}{N} \sum_{\tau \in \mathbb{Z}_N} \sum_{\nu \in \mathbb{Z}_N} f(\tau, \nu) W_N^{n\nu + k\tau} \\ &= \frac{1}{N} \sum_{\tau \in \mathbb{Z}_N} \sum_{\nu \in \mathbb{Z}_N} \sum_{l \in \mathbb{Z}_N} W_N^{n\nu + k\tau} \times \rho_N W_N^{-\nu l} x(\langle l + \tau \rangle_N) y^*(l) \end{aligned}$$

and, $W_N^{n\nu+k\tau} = e^{-\frac{j2\pi}{N}(n\nu+k\tau)}$, $n \in \mathbb{Z}_N$ and $k \in \mathbb{Z}_N$.

3.1.3 Discrete Chirp Fourier Transform

The DCFT of a discrete signal $x[n]$, $n \in \mathbb{Z}_N$, is defined as follows:

$$\begin{array}{rccc} Xc: & l^2(\mathbb{Z}_N) & \rightarrow & l^2(\mathbb{Z}_N \times \mathbb{Z}_N) \\ & (x) & \mapsto & Xc_{\{x\}} \end{array}$$

$$Xc_{\{x\}}[k,l] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x[n] W_N^{kn+ln^2}$$

Here, $W_N = e^{-j\frac{2\pi}{N}}$, and $k \in \mathbb{Z}_N$ and $l \in \mathbb{Z}_N$.

3.1.4 Cyclic Short Time Fourier Transform

The analysis window $v_z \in l^2(\mathbb{Z}_M)$ needs to be mapped to the same space as the signal space, i.e. $l^2(\mathbb{Z}_N)$. This is done by appending $N - M$ zeros to v, where this new mapped signal is denoted by $v \in l^2(\mathbb{Z}_N)$. The *cyclic shift operator* and the *Cyclic Short-Time Fourier Transform* are defined as follow.

Cyclic Shift Operator:

Let $x \in l^2(\mathbb{Z}_N)$ be an arbitrary signal, the cyclic shift operator S_N^k acting over x is

defined as:

$$S_N^k : l^2(\mathbb{Z}_N) \longrightarrow l^2(\mathbb{Z}_N)$$
$$x \longmapsto S_N^k\{x\} = y, \qquad \text{where}$$

$$(S_N^k\{x\})[n] = y[n] = x[\langle n-k\rangle_N], \qquad k \in \mathbb{Z}_N$$

where $\langle p\rangle_N$ denotes the modulo operation, i.e., $\langle p\rangle_N = remainder(\frac{p}{N})$

Cyclic Short-Time Fourier Transform:

$$CSTFT : l^2(\mathbb{Z}_N) \times l^2(\mathbb{Z}_N) \longrightarrow l^2(\mathbb{Z}_N \times \mathbb{Z}_N)$$
$$(x, v) \longmapsto CSTFT\{(x, v)\} = S_{x,v},$$

where,

$$CSTFT_{\{x,v\}}[n, k] = S_{x,v}[n, k] = \sum_{m \in \mathbb{Z}_N} x[m]v[\langle n-m\rangle_N]W_N^{km}, \tag{3.1}$$

and $k, n, m \in \mathbb{Z}_N$

If we define $x_n[m] = x[m]v[\langle n-m\rangle_N]$ as an ensemble of signals parametrized in n and substitute it in equation 3.1 then, the CSTFT can be simplified to:

$$S_{x,v}[n, k] = \sum_{m \in \mathbb{Z}_N} x_n[m]W_N^{km},$$
$$S_{x,v}[n, k] = \mathbb{F}_N\{x_n[m]\}$$
$$S_{x,v}[n, k] = X_n[k]$$

3.2 Operator Approach to Acoustic Signal Analysis

Recalling figure 1–1, we can now envision the power of the mathematical tools introduced in this chapter. The time-frequency signal operators allow us to map signals from one signal space to another. This allows the extraction of useful information

for the user. On the other hand, the signal algebra operators map signals in the same signal space. This allows us to extract information and prepare signals to be used by time-frequency signal operators.

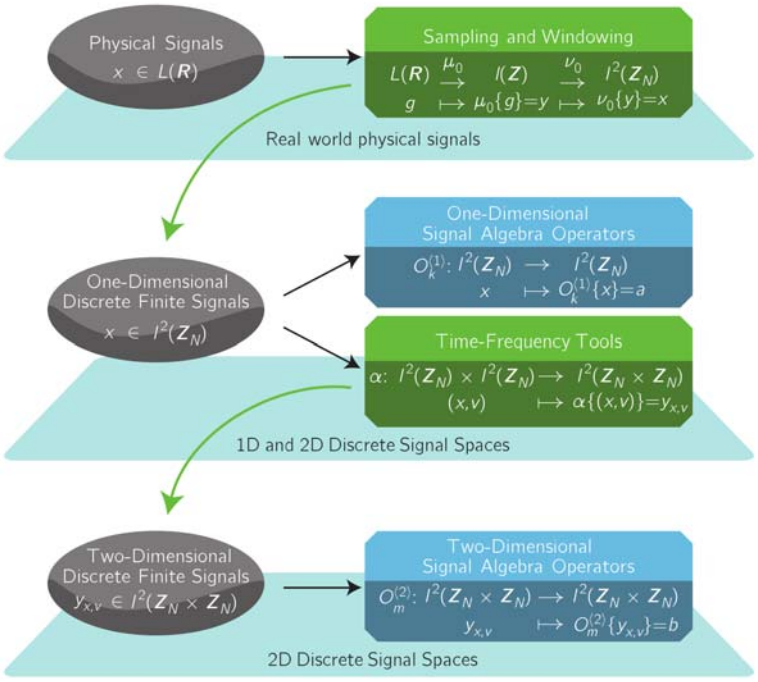


Figure 3–1: A general approach for acoustic signal analysis

CHAPTER 4
MODELLING AND SIMULATION ENVIRONMENTS

This chapter presents a brief description of two modelling and simulation environments, "Fast Multidimensional Convolutions and SAR Image Formation Simulations in a Matlab Environment" [30] and "Web-Based Data Processing for Environmental Surveillance Monitoring Applications" [35]. These works instantiate the operator signal algebra mathematical framework on which SIRLAB is based. The SIRLAB Framework is a design evolution of these previous modelling and simulation frameworks.

4.1 SARCSPE - Image Formation Simulations in Matlab

This work presents the SARCSPE, that is an environment for modelling and simulation of synthetic-aperture radar (SAR) using Matlab. Figures 4–1 and 4–2 show the environment.

4.2 JCID - Java Web-Based Data Processing Environment

This work presents the JCID, that is an environment for signal operators. Figures 4–3 and 4–4 show the environment.

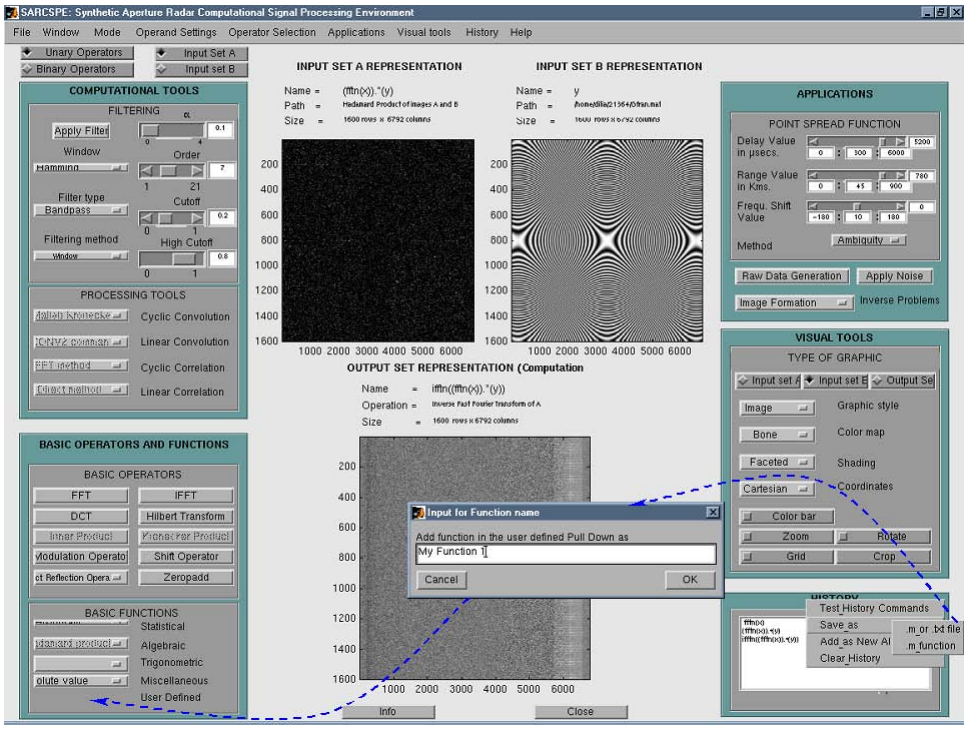


Figure 4–1: SARCSPE - Screenshot 1

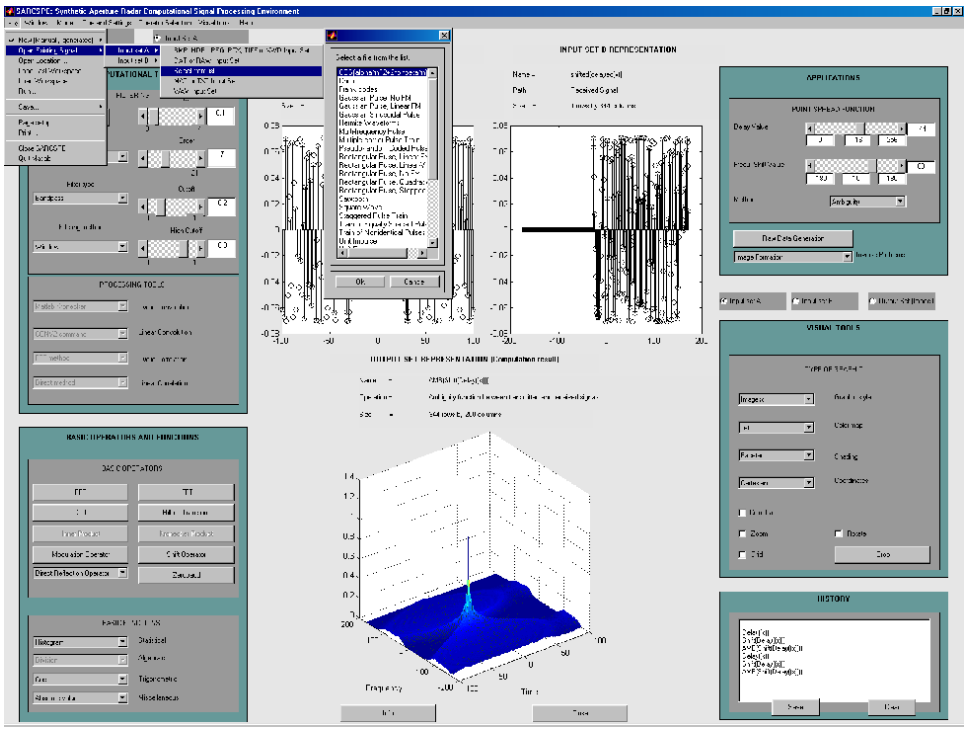


Figure 4–2: SARCSPE - Screenshot 2

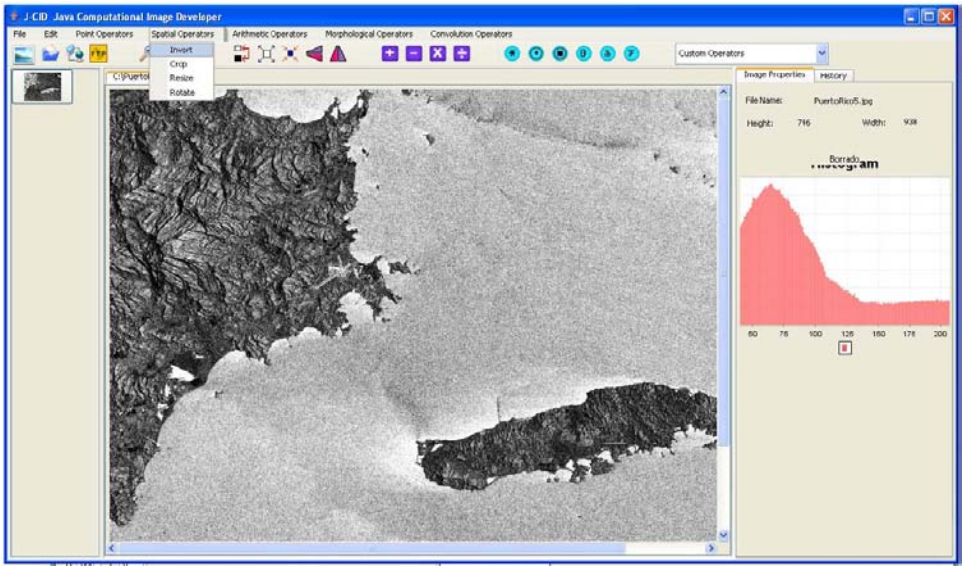


Figure 4–3: JCID - Screenshot 1

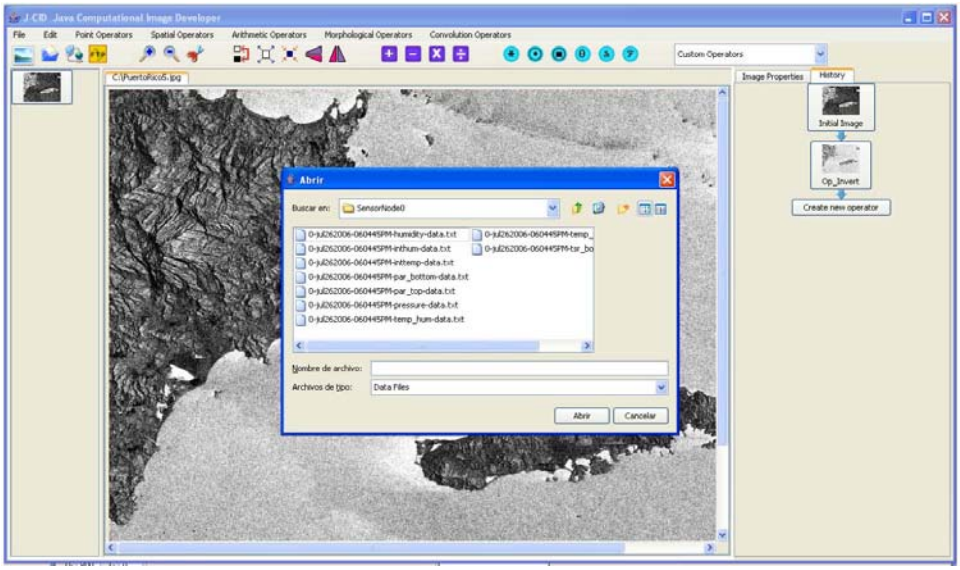


Figure 4–4: JCID - Screenshot 2

Prior to the development of the CMF a lot of work had been made in Matlab. The CMF formulated in this thesis is a response to several issues with the Matlab implementations of the time frequency operators. Here we enumerate some of the issues.

- Matlab is an excelent environment for testing of algorithm implementations. But, the work at the laboratory is conducted in digital signal processors, gumstyx (ARM processors), and recently with Alix 2D2. The testing of Matlab algorithm implementations in these computers is not feasible. For this reason the option of a framework that allows go from workstations up to small processors is a good idea.
- The implementation of signal processing algorithms in Matlab is very easy. But, these implementations runs too slow. We require a fast and easy implementation environment.
- Although this is not exactly an issue, the licensing of Matlab is expensive. For this reason the open source tools are a good idea. The problem of the open source tools is the time required to develop an implementation for signal processing. With this CMF we expect to make faster and easier the implementation of new time-frequency signal operators.

The AIPLAB has been working with Matlab for a long time. Matlab provided the basis and the inspiration for the development of the SIRLAB Framework. The students at AIPLAB frequently has to test in Matlab the algorithms before to implement in other computational structures. This work is not an exception. The implementation of time-frequency operators have been our work during the last 5 years. Clusters or parallel processing computers were the target computational structures. As a result several time-frequency operators have been implemented using Matlab.

Here we will explain some of the implementations using Matlab and their associated results.

4.3 Discrete Chirp Fourier Transform

The DCFT of a signal x is defined as

$$X_{C\{x\}}[k,l] = \frac{1}{\sqrt{N}} \sum_{n \in \mathbb{Z}_N} x[n] W_N^{kn+ln^2}$$

where, $W_N^{kn+ln^2} = e^{-\frac{j2\pi}{N}(kn+ln^2)}$, $k \in \mathbb{Z}_N$ and $l \in \mathbb{Z}_N$. Here we present some results of implementations using Matlab. The figures 4–5, 4–7, 4–6, 4–5, and 4–6 show results obtained using Matlab as algorithm implementation platform. The figures 4–5 and 4–6 show how the DCFT is capable to detect in peaks the parameters of a chirp signal of 6 components. In addition, increasing of the number of samples improves the signal to noise ratio for the detection.

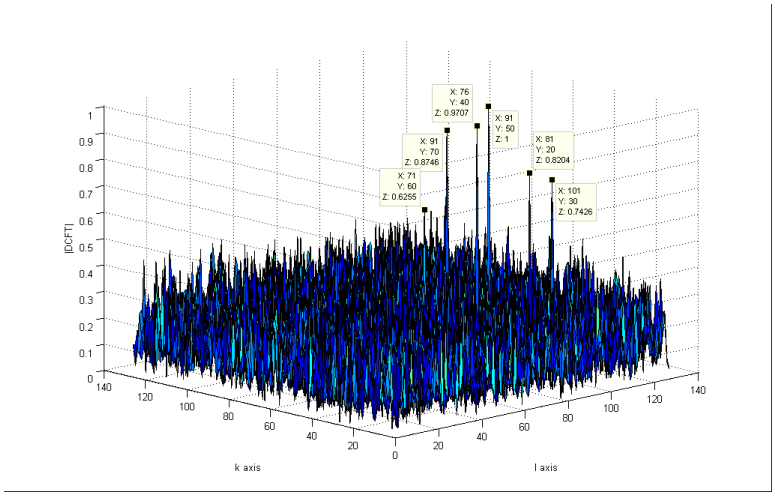


Figure 4–5: DCFT for a 6 Component Chirp Signal, N=127

The figure 4–7 shows three video frames of six moving targets. The targets are represented by the points.

Figures 4–8 and 4–9 show a DCFT output for a one component chirp signal and the DCFT output for a two components chirp signal.

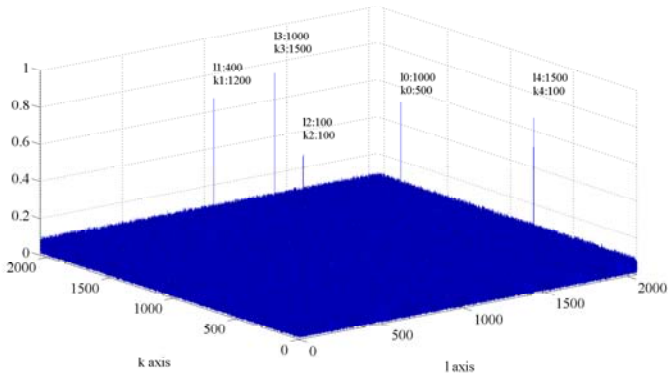


Figure 4–6: DCFT for a 6 Component Chirp Signal, N=1023

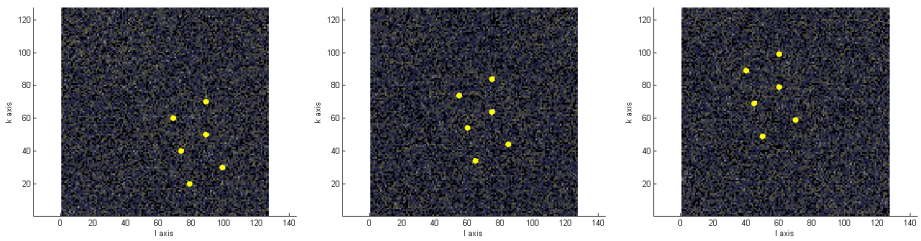


Figure 4–7: Three Frames Showing the Evolution of 6 Targets Using the DCFT

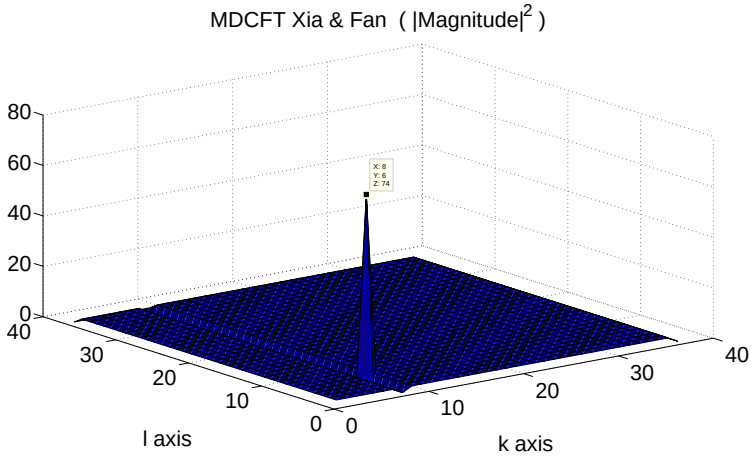


Figure 4–8: Modified DCFT for 1 Component Chirp Signal

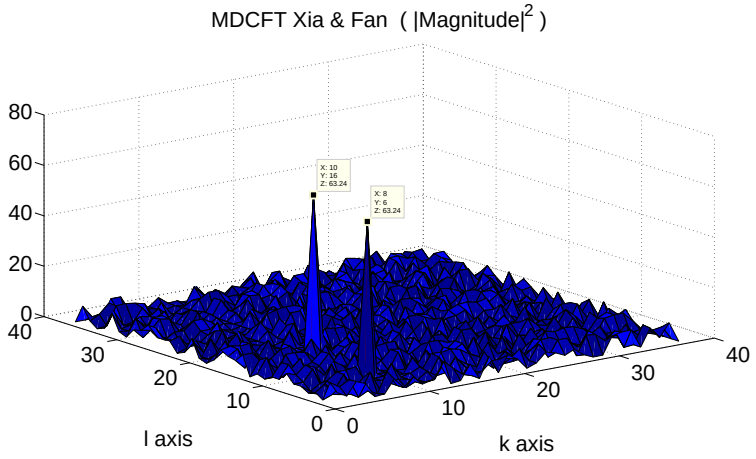


Figure 4–9: Modified DCFT for 2 Component Chirp Signal

4.4 Cyclic Short Time Transform

The CSTFT of a signal x is defined as

$$S_{\{x,v\}}[m,k] = \sum_{n \in \mathbb{Z}_N} x[n] v[\langle m-n \rangle_N] W_N^{kn}$$

where, $W_N^{kn} = e^{-\frac{i2\pi}{N}(kn)}$, $m \in \mathbb{Z}_N$ and $k \in \mathbb{Z}_N$. The figures 4–10 and 4–11 are two examples of the evolution in the design of the frame. The figure 4–12 is the most recent frame design.

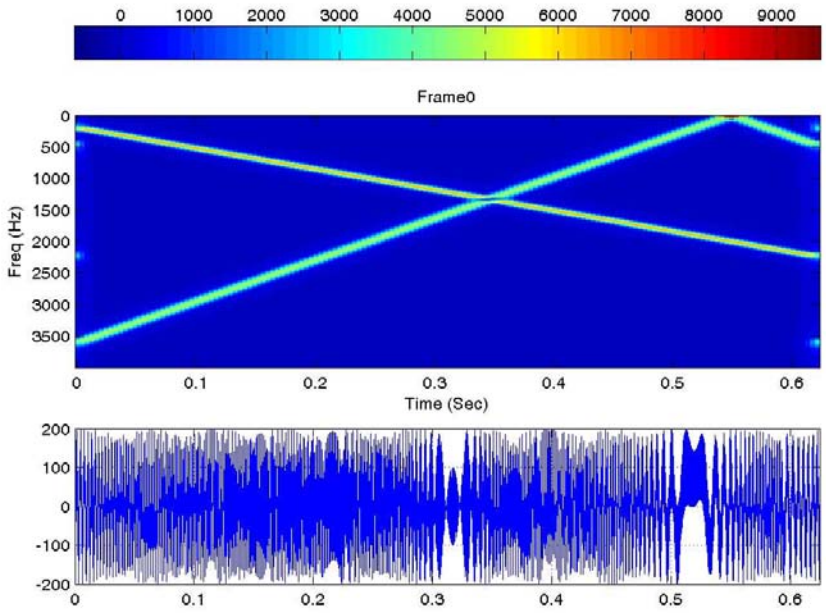


Figure 4–10: CSTFT of two chirp signals

4.5 Cyclic Ambiguity Function

The CAF of a pair of signals f and g is defined as

$$A_{\{f,g\}}[m,k] = \sum_{n \in \mathbb{Z}_N} f[n] g^*[\langle n+m \rangle_N] W_N^{kn}$$

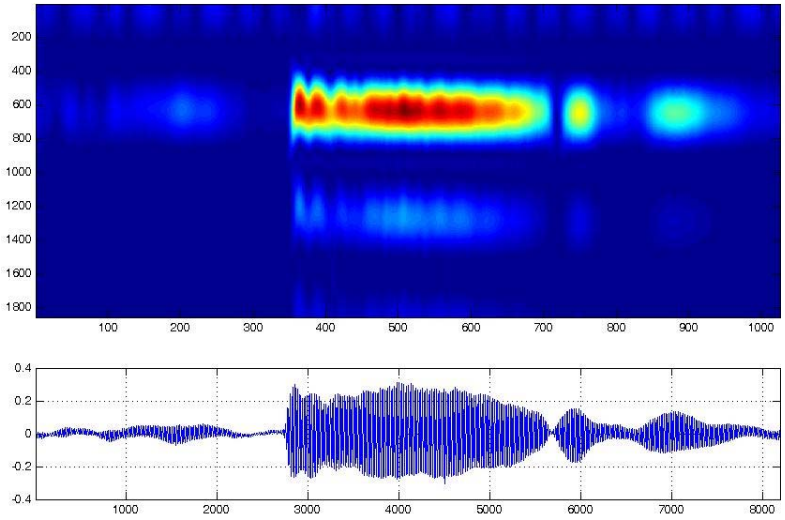


Figure 4–11: CSTFT of Anuran (Eleuterodactylus cooki)

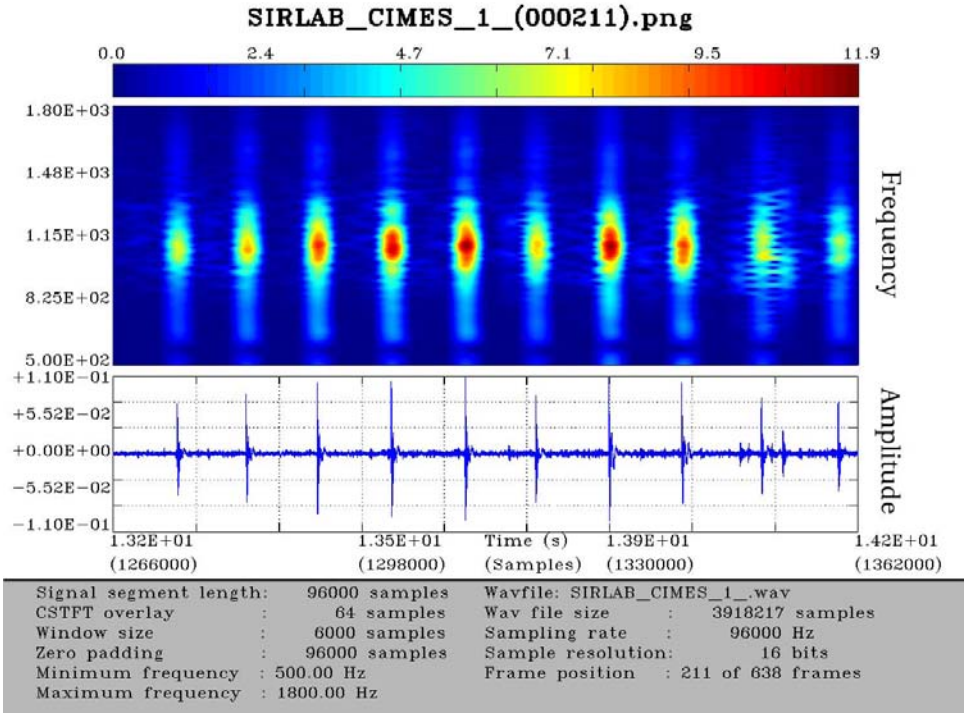


Figure 4–12: Current Frame Appearance for the CSTFT operator (hydrophone sound)

where, $W_N^{kn} = e^{-\frac{j2\pi}{N}(kn)}$, $m \in \mathbb{Z}_N$ and $k \in \mathbb{Z}_N$. The figures 4–13 and 4–14 shown the computation of the CAF for two chirp signals delayed t_d.

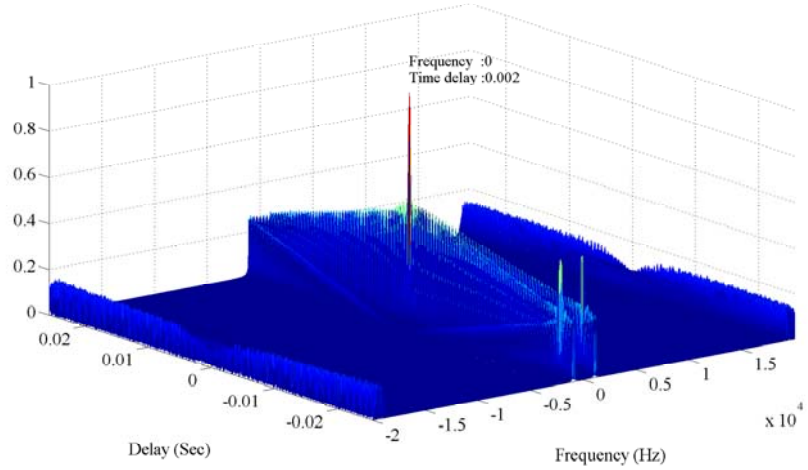


Figure 4–13: Ambiguity Function of Two Chirp Signals with $t_d = 0.002s$

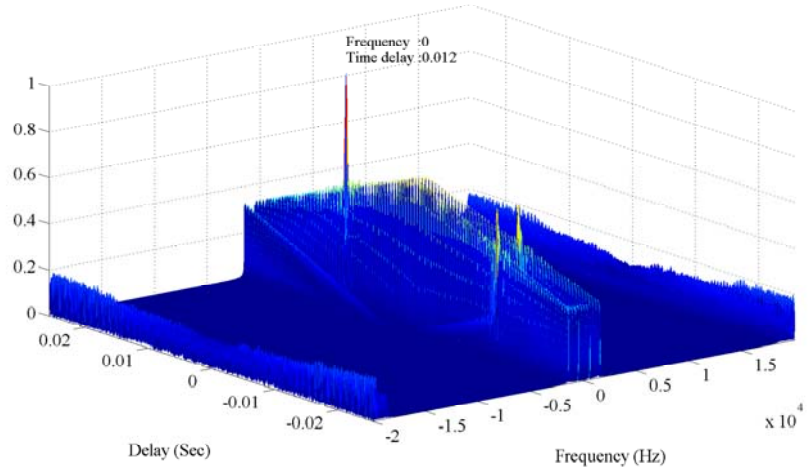


Figure 4–14: Ambiguity Function of Two Chirp Signals with $t_d = 0.012s$

4.6 Wigner Distribution

The WD of a signal f is defined as

$$W_{\{f\}}[n,k] = \frac{1}{N}\sum_{\tau\in\mathbb{Z}_N}\sum_{\nu\in\mathbb{Z}_N} f(\tau,\nu)W_N^{n\nu+k\tau} \tag{4.1}$$

$$= \frac{1}{N}\sum_{\tau\in\mathbb{Z}_N}\sum_{\nu\in\mathbb{Z}_N}\sum_{l\in\mathbb{Z}_N} W_N^{n\nu+k\tau}\times\rho_N W_N^{-\nu l}x(\langle l+\tau\rangle_N)y^*(l) \tag{4.2}$$

where, $W_N^{n\nu+k\tau} = e^{-\frac{j2\pi}{N}(n\nu+k\tau)}$, $n\in\mathbb{Z}_N$ and $k\in\mathbb{Z}_N$. The figure 4–15 shows two signals which will be processed by the WD operator. The outputs are presented in figures 4–16 and 4–17.

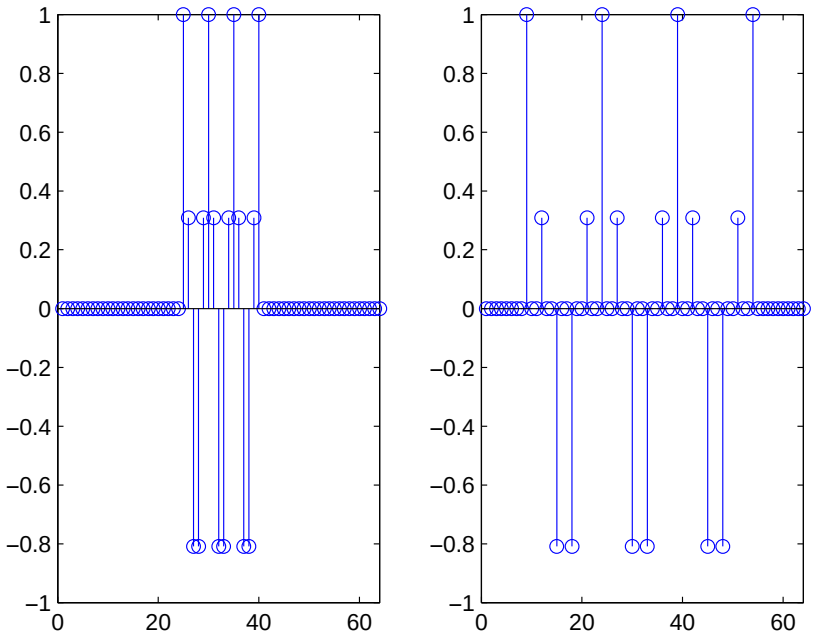


Figure 4–15: Left is the real part $x_1(n) = [zeros(1;24); exp^{(j*2*pi/N)*[0:15]}; zeros(1;24)]$ and right is the dilation/compression of $x_1(n)$ by 3

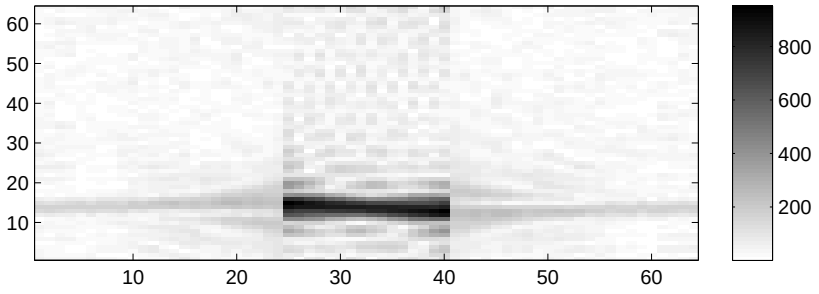


Figure 4–16: WD for Left Signal of Figure 4–15

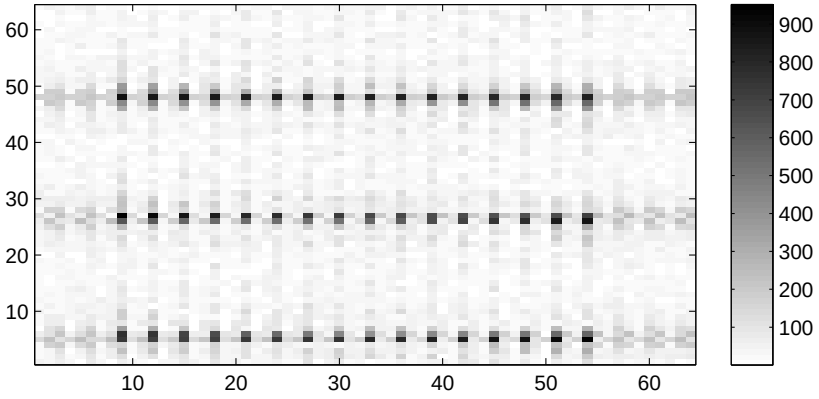


Figure 4–17: WD for Right Signal of Figure 4–15

CHAPTER 5
SIRLAB TARGET APPLICATION ENVIRONMENT

This chapter introduces a first application environment for the SIRLAB framework. The BESN (Bioacoustics Environmental Surveillance Network) constitutes a perfect application where the SIRLAB can be used. The BESN will give to SIRLAB the networking capabilities to integrate a cognitive wireless sensor network.

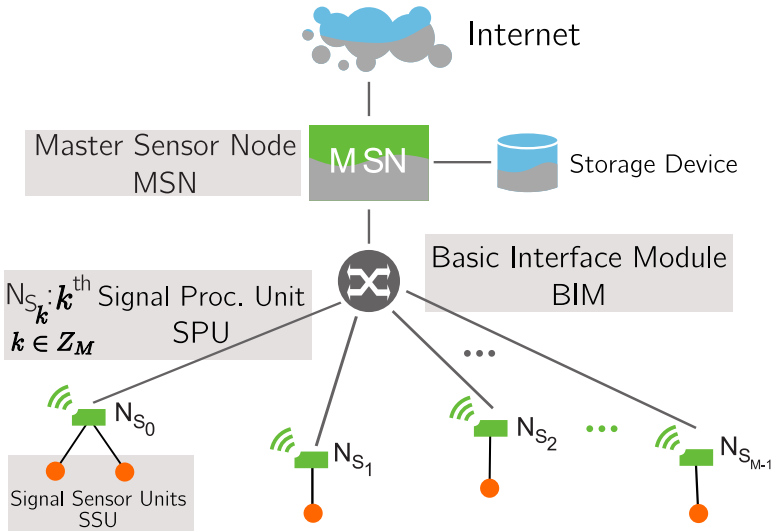


Figure 5–1: The SAP Concept

The computational structure where the CMF proposed is integrated in the sensor array processing (SAP) model depicted in Fig. 5–1 corresponds to the conceptual representation of our system. The sensor signal processing (SSP) nodes are a set of wireless, low-cost acoustic signal acquisition, low storage, and processing nodes which use Linux-based miniature single board computers. These nodes are combined and treated as a sensor array unit without a prescribed topology and their signal-based

acoustic information is aggregated to a Linux based high-performance embedded computing unit, called a master sensor node (MSN), for further raw data processing, algorithm implementation and information representation. The master sensor node (MSN) unit selected is an embedded PC in mini-ITX form factor from AOpenTM, model number i945GMt-FSA. The motherboard is equipped with one CPU: 2.00GHz Intel R Core 2 Duo T7200; Memory: Transcend 2GB So-DIMM DDRII 667MHz; Hard disk: Hitachi Travelstar 200GB 7200RPM SATA; OS: Linux Fedora core 12 (Red Hat).

The name of the computational framework is SIRLAB (SIgnal Representation LABoratory), and one of the purposes of it is to produce easy to use programming environment to be used by students and scientists. The SIRLAB is based completely in open source tools such as FFTW and OpenCV. The code developed in the project is C/C++ compatible. This work is focused in the processing of acoustic signals to analyze underwater acoustic signals collected by hydrophone and to monitor the behavior of amphibians such as the endemic Puerto Rican Crested Toad (Peltophryne [Bufo] lemur). Particular interest is placed in identifying the information present in the signals acquired by the physical sensors and extraction of information useful to the user with a reasonable computational effort. The Fig. 5–2 shows the NetSig/SIRLAB integration with an application with underwater and aerial microphones to acquire field data, and mobile devices as user interface for the CMF.

The SIRLAB makes the extraction of information easier, but at the expense of greater computational complexity. This is achieved through a conversion of the acquired signal to another signal which is more suitable for information extraction and change detection. To achieve the information extraction we build implementations of some of the transforms of Cohen's class. The Cohen class maps signals acquired in one dimension and projects into an image where time and frequency are represented. A

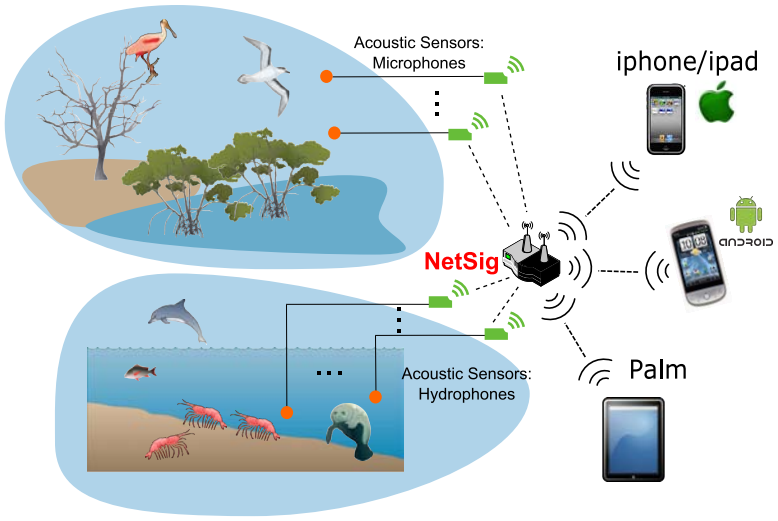


Figure 5–2: NETSIG and SIRLAB Integration Concept

time-frequency representation allows us to observe details in the signal that would not be noticeable otherwise.

The figure 5–3 depicts a NetSig node in a BESN environment. SIRLAB runs in every MSN and NetSig node of a BESN. The proposed computational framework includes a visualization tool to be used in a network integrated environment and signal processing shown in Fig. 5–3 as (NetSig) [31–33], and more specifically to be integrated into the SAP (see Fig. 5–1) which is a sub-system consisting of four parts: L signal sensor units (SSU), M signal processing units (SPU), one basic interface module (BIM) and one master sensor node (MSN) which specifications was mentioned above. Because of power considerations, and due to limited computational power of the SPUs the computational framework will not work at this level. Instead it will run at MSN level, where the output of the processing will be available to the Internet from the storage device.

The WALSAIP project has been developing a testbed at the Jobos Bay National Estuarine Research Reserve (JBNERR) (See Fig. 5–5), situated in the southeastern part of island of Puerto Rico. JBNERR is administered by the National Oceanic

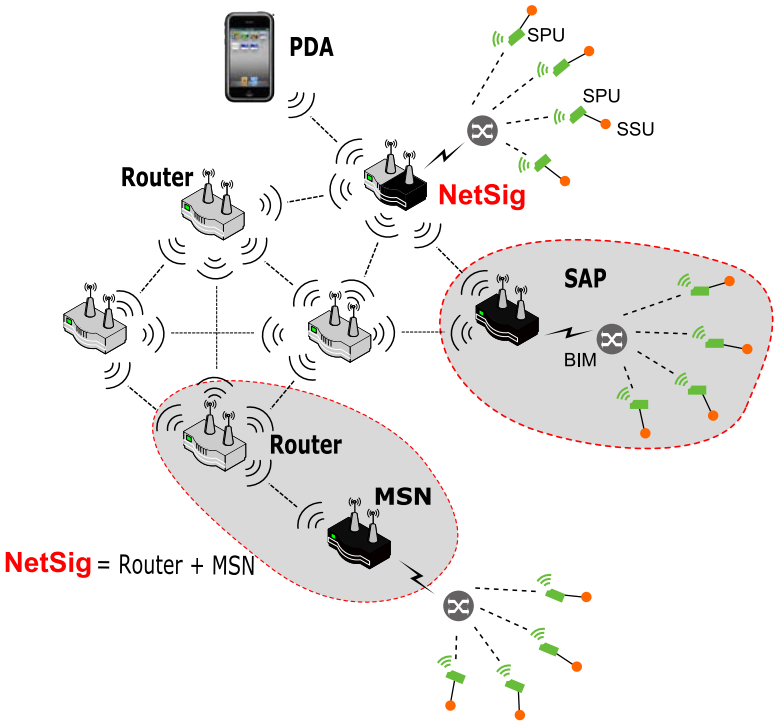


Figure 5–3: VESO Mesh Cognitive Wireless Sensor Network

and Atmospheric Administration (NOAA) as well as the Puerto Rico Department of Natural and Environmental Resources (DRNA). The Jobos Bay Reserve is the second largest estuarine area in Puerto Rico. It encompasses a chain of about 15 tear shape mangrove islets, known as Cayos Caribe, and the Mar Negro area in western Jobos Bay. The reserve is home to a number of species, including the endangered brown pelican, peregrine falcon, hawksbill sea turtle, and West Indian manatee. Several versions of testbeds have been developed and tested at JBNERR, Fig. 5–4 shows one of them installed in the mangrove area. Recently a hardware configuration is planned to be installed at a buoy seen in Fig 5–6, the location of the buoy can be seen at Fig 5–5.

Figure 5–4: Testbed Installed in the Mangrove Area, Wireless Antenna, Solar Panel, and MSN Pelican Box are visible

Figure 5–5: Jobos Reserve with JBNERR Office and Buoy Localizations

Figure 5–6: Buoy Intended as Testbed Place

CHAPTER 6
SIRLAB SOFTWARE DESIGN SPECIFICATION

6.1 Document Description

6.1.1 Introduction

SIRLAB is a computational modelling framework (CMF) for time frequency signal operators. The class of signals of interest are bioacoustical signals acquired by microphones and/or hydrophones, but the concept can be expanded to other kind of signals.

6.1.2 Intended Audience

This document is written to be useful for programmers and end-users, allowing them to understand the limits of the CMF proposed. Programmers will be interested because they will know how to add new features and improve the performance of the CMF. The end-users will understand what they can/can't do with the CMF.

6.1.3 Hardware and Software Resources

6.1.4 Version Management

There is no formal version management for the SIRLAB. Each signal processing operator must be operated independently from other signal operators. The system must be able to support many signal processing operators as was defined in chapter 5.

6.1.5 Definition, Acronyms and Abbreviations

In this section we will summarize and define some terms important to understand the SIRLAB CMF.

Term	Equivalent
AIPLAB	Automated Information Processing LABoratory
API	Application Programming Interface
ARM	Advanced RISC Machine
CAF	Cyclic Ambiguity Function
CMF	Computational Modelling Framework
CSTFT	Cyclic Short Time Fourier Transform
DCFT	Discrete Chirp Fourier Transform
DSP	Digital Signal Processor
FFTW	Fast Fourier Transform of the West
GUI	Graphic User Interface
OpenCV	Open Source Computer Vision
RIFF	Resource Interchange File Format
SIRLAB	SIgnal Representation LABoratory
WAV	WAVeform audio file format
T-F	Time Frequency

Table 6–1: Acronyms Table

Term	Definition
Output frame	Is an image resulting of the computation of a signal using the SIRLAB CMF.
RIFF	Is a generic file container format for storing data in tagged chunks.

Table 6–2: Definitions Table

6.1.6 Remarks

This chapter establishes some requirements for the SIRLAB CMF design. In addition this chapter provides an overview of the computational modelling framework.

6.2 Design Considerations

The concept of cyclic signal processing operators (such as the CSTFT) has been used recurrently in works of the AIPLAB. Several MsC projects have been developed based in the concept of cyclic signal processing operators such as works by Yuji Yunes [1] and Ivan Rivera [34]. They allow map signals from a signal space $l^2(\mathbb{Z}_N$ to an standard output signal space such as $l^2(\mathbb{Z}_N \times \mathbb{Z}_N)$ (see chapter 3). The cyclic signal operators allows to consider the creation of output frames in the standard output signal space. The concatenation of output frames produces video files showing the

evolution of the signals through time.

The development of a cyclic signal operator requires a computational algorithm implementation in a computational structure to be useful. Matlab is one alternative to make tests of the computational algorithm implementations, but it is not the only way. Some works at the AIPLAB (Lola Bautista [35] and Ana Ramirez [36] MsC theses) were developed with the idea to facilitate signal processing algorithm implementation.

The works presented above have been implemented in Matlab, Java, digital signal processors (DSP), and ARM processors. A lot of experience was collected in terms of hardware and user-interfaces as we will explain with two examples. First the Lola Bautista's thesis presents a Java implementation of signal operators that is a user interface for signal operators running behind. The second is by Yuji Yunes where he develop an implementation for the STFT operator in an ARM processor(Gumstix) used at AIPLAB. All the works previously mentioned are user-interface or hardware-software oriented. We need a complete system that takes signals from the real world, process the signals using a signal processing operator, and finally provides the output to the user.

In SIRLAB we consider an input -the signal or signals acquired from the real world- and an output as the set of output frames or the video files. The CMF must integrate the input/output representation to show both elements in the same frame.

6.2.1 Assumptions and Dependencies

Is very usual in signal processing the work with Fourier transforms. The FFTW is a well known implementation that has been widely used in science.

Usually the outputs of time-frequency signal operators need a graphical representation of the output in order to show the results to the user. For this reason it's necessary to consider a graphic management library that allows the mapping of the output of the time frequency operators to a standard picture format such as png, avi, etc. The tool considered to perform this is the OpenCV that is a powerful API built in C/C++ that allow image manipulations.

The selection of the operating system must consider open source projects such as mentioned above. Although FFTW and OpenCV runs in Windows, the option of GNU/Linux is preferred because it runs well in hardware that other groups at the AIPLAB are using to collect data. We expect the CMF defined here will be used by other groups to perform data analysis over the collected data.

It's expected that the CMF proposed here extend the capabilities to better use of hardware, taking advantages of new processors with multicore capabilities. Another improvement will be the development of the platform in other computer languages that allow more high level tools and the development of web applications to allow scientists make use of the CMF via mobile phones.

6.2.2 General Constraints

Hardware or software environment

As was mentioned in section 6.2.1 the system will be developed under GNU/Linux. We expect SIRLAB to run in the following computational resources. The **aip-srv01** is a Dual Quad Core Processor W55803.20GHz, 8M, 6.4GT/s, Dell Precision T7500 with 48GB (12X4GB) SDRAM Memory and a 1066MHz, 450GB, 3Gbps SAS, 15K RPM Hard Drive. It runs the operating system CentOS Linux Kernel 2.6.18-194.3.1.el5.centos.plus. It uses the C compiler gcc (GCC) 4.1.2 20080704 (Red Hat 4.1.2-48). The fast Fourier transform library utilized to perform all discrete Fourier transform computations is the FFTW Version 3.2.2. This computer platform is running OpenCV version 2.1. The **Desktop-PDC-Lab01** platform is an Intel(R) Pentium(R) 4, 3.80GHz CPU, with 3GB SDRAM, and 250GB, 7.2K, WD-HAWK Hard Drive. Operating system Fedora 12 Linux Kernel 2.6.32.19-163.fc12.i686.PAE. It uses the C compiler gcc (GCC) 4.4.4 20100630 (Red Hat 4.4.4-10). As the previous computer platform, this platform uses the FFTW Version 3.2.2 and the OpenCV version 2.1.

End-user environment

The SIRLAB CMF will not provides a graphic user interface (GUI) for end users. The end user at this stage will be a programmer or a CMF tool trained scientific. We plan the system at the end of this thesis will have a low level user interface. For this reason the end user will program the commands to the CMF through OS shell command scripts and text files will modify the parameters of the signal operator implementation. The signal operators that we will develop in this document are in table 6–3. These examples are only a few, many other T-F signal operators are suitable to be implemented reusing the code or programming new functions to the libraries.

Signal Operator	Mathematical Expression
CSTFT	$S_{\{x,v\}}[m,k] = \sum_{n\in\mathbb{Z}_N} x[n]v[\langle m-n\rangle_N]W_N^{kn}$ where, $W_N^{kn} = e^{-\frac{j2\pi}{N}(kn)}$, $m \in \mathbb{Z}_N$ and $k \in \mathbb{Z}_N$
CAF	$A_{\{f,g\}}[m,k] = \sum_{n\in\mathbb{Z}_N} f[n]g^*[\langle n+m\rangle_N]W_N^{kn}$ where, $W_N^{kn} = e^{-\frac{j2\pi}{N}(kn)}$, $m \in \mathbb{Z}_N$ and $k \in \mathbb{Z}_N$
DCFT	$X_{C\{x\}}[k,l] = \frac{1}{\sqrt{N}} \sum_{n\in\mathbb{Z}_N} x[n]W_N^{kn+ln^2}$ where, $W_N^{kn+ln^2} = e^{-\frac{j2\pi}{N}(kn+ln^2)}$, $k \in \mathbb{Z}_N$ and $l \in \mathbb{Z}_N$
WD	$W_{\{f\}}[n,k] = \sum_{\tau\in\mathbb{Z}_N}\sum_{\nu\in\mathbb{Z}_N} f(\tau,\nu)W_N^{n\nu+k\tau}$ where, $W_N^{n\nu+k\tau} = e^{-\frac{j2\pi}{N}(n\nu+k\tau)}$, $n \in \mathbb{Z}_N$ and $k \in \mathbb{Z}_N$.

Table 6–3: Signal Operators to be implemented

Data repository and distribution requirements

The CMF will be available as an open source application to be downloaded, utilized, and modified freely according to the user application. The CMF will be distributed under GNU General Public License(GPL) because the FFTW is GPL and OpenCV is BSD.

6.3 Verification and validation requirements (testing)

To validate the SIRLAB operator implementation we will use equivalent Matlab implementations of the same signal operator. During the past years at the AIPLAB have been developed implementations (mainly for Matlab) of several signal operators, that can be useful for testing and validation of the SIRLAB.

6.3.1 Goals and Guidelines

Parameters Specification

The development of a robust CMF capable to analyze WAV files according to a set of parameters given by user. A requirement for computation of T-F operators is a set of parameters which define the output characteristics. The parameters define the way as the operator is processed and have implication in the use of the computational resources such as memory use, computation time, and quality of the visualization.

Standard Output

The CMF must have the capability to place the results in defined standard directories. This is of particular interest because if we want to develop a web interface or a shell script, the application needs to know where the input WAV files must be placed and where the output files will be available after processing.

Spectrograms

The CMF must produce detailed spectrograms or T-F representations. The quality of the spectrograms are directly related with the parameters defined by the user.

Computation Time

The CMF is expected to improve the computation time in comparison to the Matlab implementations of the past. The Matlab implementations developed are very slow.

User Interface

Although is possible the development of the CMF to receive commands using hotkeys or inputs from the user. This scenario is not desired because it will restricts the usability of the CMF. The CMF is conceived to work at command line level. We think that a good user interface will be a web interface. Changes in the CMF processing are introduced in the web application through modifications to the parameter specification files.

6.4 System Architecture

The figure 6–1 is a summary of the concepts mentioned in this chapter. The figure shows to the right the two types of input files that SIRLAB needs for processing. The first are the WAV files that content the data acquired by physical sensors and the second are the parameter files. The parameter files are text files that contents a programmer defined set of values that specify the parameters for the implemented signal processing operator. The SIRLAB has a core that interface with the libraries built in SIRLAB. The SIRLAB libraries use the FFTW and the OpenCV routines to perform an output of the signal processing operator. The output of the SIRLAB consist of a collection of output ordered frames based in the WAV and the parameter files.

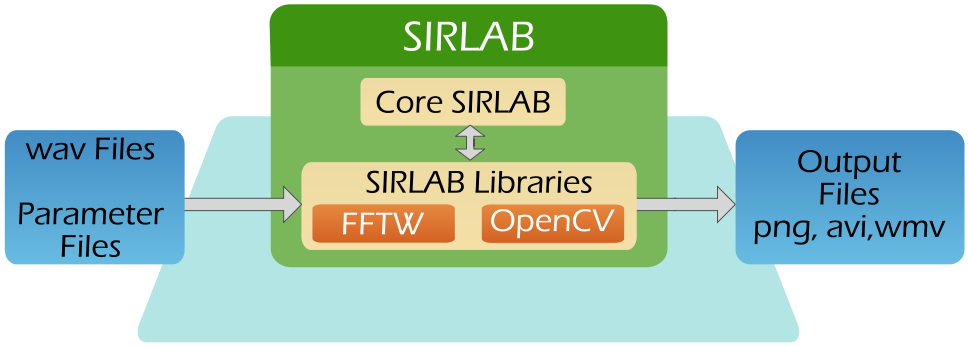


Figure 6–1: SIRLAB System Architecture

6.4.1 Architectural Strategies

As we mention in section 6.4, SIRLAB is a collection of functions. The functions in SIRLAB are divided in 4 categories: input, output, processing and mathematical libraries. The input library is the set of functions oriented to read the values in the input files (WAV or parameter file), e.g., a function that reads in the WAV file the number of samples. The output library contains functions that perform the adjustments necessary to produce output files, e.g., place the title to the frame or place the colorbar in the output image. The processing and mathematical libraries

are responsible of the execution of the operations required for the signal operator implementation, e.g., take two arrays and return the haddamard product or compute the FFT of an array. In chapter 7 we will explain each one of the categories and the functions implemented.

CHAPTER 7
SIRLAB DETAILED SYSTEM DESIGN

7.1 Introduction

In this chapter we introduce the SIRLAB, that is the name given to this computational tool framework written in C/C++ language for a Linux environment and using the FFTW and OpenCV APIs. The SIRLAB is a software composed by a set of libraries of programming functions designed for implementation of Cohen class signal processing operators. This chapter explains how to use the framework. We define each one of the functions in the SIRLAB libraries explaining the purposes of the function and the input/output parameters for the function.

7.1.1 Intended Audience

This document is written to be useful for programmers and advanced users interested in the development of new functionality to SIRLAB. We encourage them to build new functions for the libraries and operators. Other audience that could find interesting this chapter is the scientific community (i.e. biologist, chemists, physicists, etc) which can learn how to adjust parameters files for the signal processing operator implemented.

7.1.2 Version Management

Several signal operators have been developed in SIRLAB. In general we have defined a directory with the name <*name of the signal operator*><*version*> to identify what operator is implemented. For example stft4.0 is the most developed version for the short time Fourier transform operator of stft1.0, stft2.0, stft3.0 and stft4.0.

AmbFunc1.0 is the first version for the ambiguity function signal operator. Notice that doing a change from version x.0 to y.0 involves a major revision, and doing a change from version 4.x to 4.y involves minor revisions.

7.1.3 APIs used in SIRLAB

The SIRLAB is based in two well known open source APIs, the OpenCV and the FFTW. The links to both projects are:

- FFTW (Fast Fourier Transform of the West) - http://www.fftw.org
- OpenCV (Open Computer Vision) - http://opencv.willowgarage.com/wiki/

The OpenCV API is responsible for the graphic operations to generate output frames or movies. A lot of documentation are available about OpenCV, if you are planning to program over SIRLAB and you are not familiarized with this tool you should read the following book: Learning OpenCV: Computer Vision with the OpenCV Library [37]. The signal processing algorithms usually requires Fourier transforms to be implemented. We have chosen the FFTW (Version 3.2.2) because it provides a fast implementation of the Fourier transforms for several formats of the transforms (real-to-complex, complex-to-complex), several precision formats (double or floating point), and directions (forward or reverse). In addition FFTW allows multidimensional transforms. Useful documentation of the FFTW can be encountered at the website and the paper *The Design and Implementation of FFTW3* [38].

7.1.4 Remarks

This document pretends to be a guide to learn how to use and work the SIRLAB CMF. At the end of the reading of this chapter you must be familiarized with the platform and be able to use any signal processing operator implemented in the SIRLAB and/or develop new applications for signal processing operators.

7.2 Assumptions and Dependencies

In this section assumptions and dependencies show the type of systems in which the SIRLAB has run successfully, it is expected that similar systems can run SIRLAB without significant modifications. The hardware where SIRLAB were run is in the following table.

Computer Name	Specifications
aipsrv01[1]	Dual Quad Core Processor W5580 3.20GHz, 8M, 6.4GT/s, Dell Precision T7500 with 48GB (12X4GB) SDRAM Memory and 1066MHz, and 450GB 3Gbps SAS, 15K RPM Hard Drive.
Desktop PDC Lab[2]	Intel(R) Pentium(R) 4 CPU 3.80GHz, with 3GB SDRAM, and Hard Drive, 250GB, 7.2K, WD-HAWK.

Table 7–1: Hardware where SIRLAB were Tested

Software specifications table:

Related software	FFTW 3.2.2[1,2] OpenCV 2.1[1,2]
Operating systems	Centos Linux kernel 2.6.18 (Red Hat 4.1.2-48)[1] Fedora 12 Linux Kernel 2.6.32 (Red Hat 4.4.4-10)[2]
gcc compiler	gcc (GCC) 4.1.2 20080704[1] gcc (GCC) 4.4.4 20100630[2]
End-user characteristics (Input)	*wav* input files.[1,2] Parameters for processing in a text file.[1,2]
End-user characteristics (Output)	*png* or *avi* output file frame format[1,2]
	[1]aipsrv01 [2]Desktop PDC Lab

Table 7–2: SIRLAB Assumptions and Dependencies

7.3 SIRLAB Detailed System Design

7.3.1 System Architecture

The figure 7–1 shows a detailed system architecture from the presented at figure 6–1. This new figure presents the four libraries developed to perform the implementation of the time frequency operator. Later each one of the functions in the four libraries

will be explained in detail. The first library is *inputlib*, this library reads the parameters in the input parameter file and WAV file for use in the T-F operator. The *processlib* and the *aritmetlib* are the libraries that perform all the operations required to obtain an output. Finally *outputlib* takes the output of the operator and transform it to an image to be saved in the corresponding output directory.

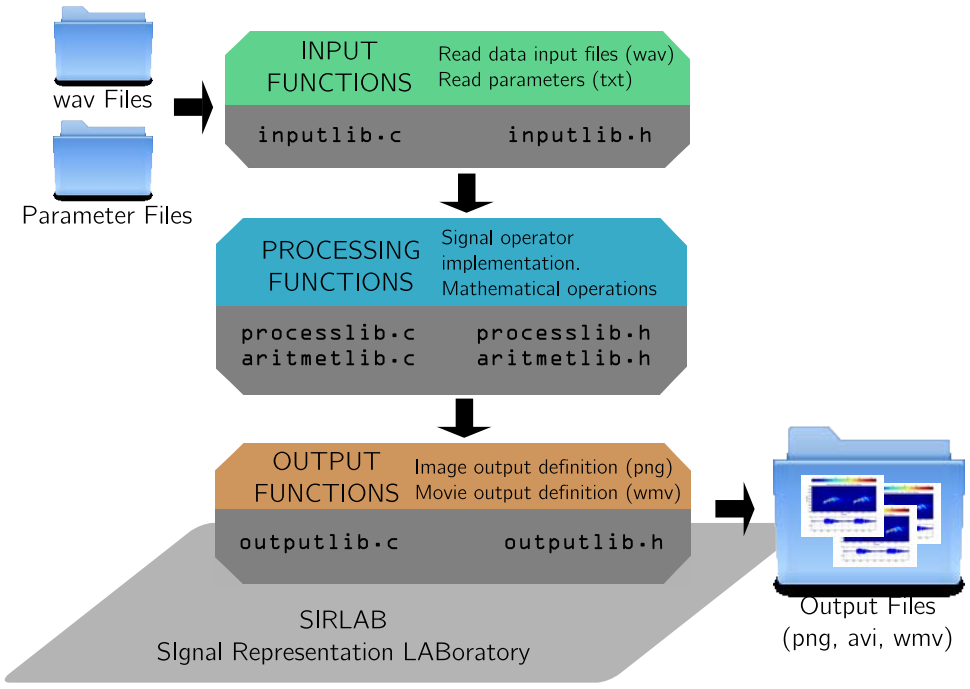


Figure 7–1: Detailed SIRLAB System Architecture

The list of the directories in tree-like format for the implementation of the CSTFT T-F operator (the name of the version is *stft4.0*) is shown in figure 7–2. The *wavs* directory content the input WAV files. The *parameter_file* contents text files with the parameters to process the CSTFT in different manners. The *models* directory contents the empty frames or templates to be used to place the output T-F image. The *colormaps* directory contents text files with the RGB specification of the color-bars. later we will explain in detail the format of the colorbar text files. And finally the *outfiles* directory. This directory contains subdirectories whose names are the

WAV file name with the prefix *D_*. Inside each subdirectory there are the png, avi or wmv output files.

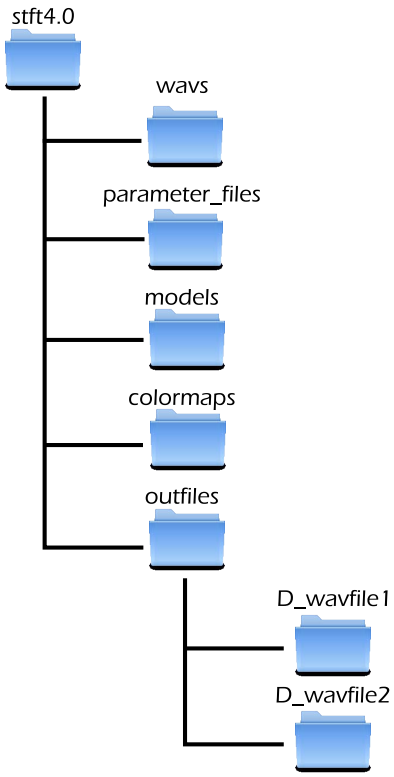


Figure 7–2: Tree Structure for the SIRLAB

7.4 SIRLAB Libraries and Function Description

7.4.1 Inputlib

To perform the time-frequency signal operator function is required a set of parameters that define how the output will be processed. In this library there are the functions that read the parameters needed for the processing of the time frequency signal operator. There exist two parameter sources for the SIRLAB processing schema. The first source are the parameters into the WAV file given in the metadata of the WAV file. The other source is a text file (input parameter file) modified by the user with the values for each parameter. The two functions are explained in this section.

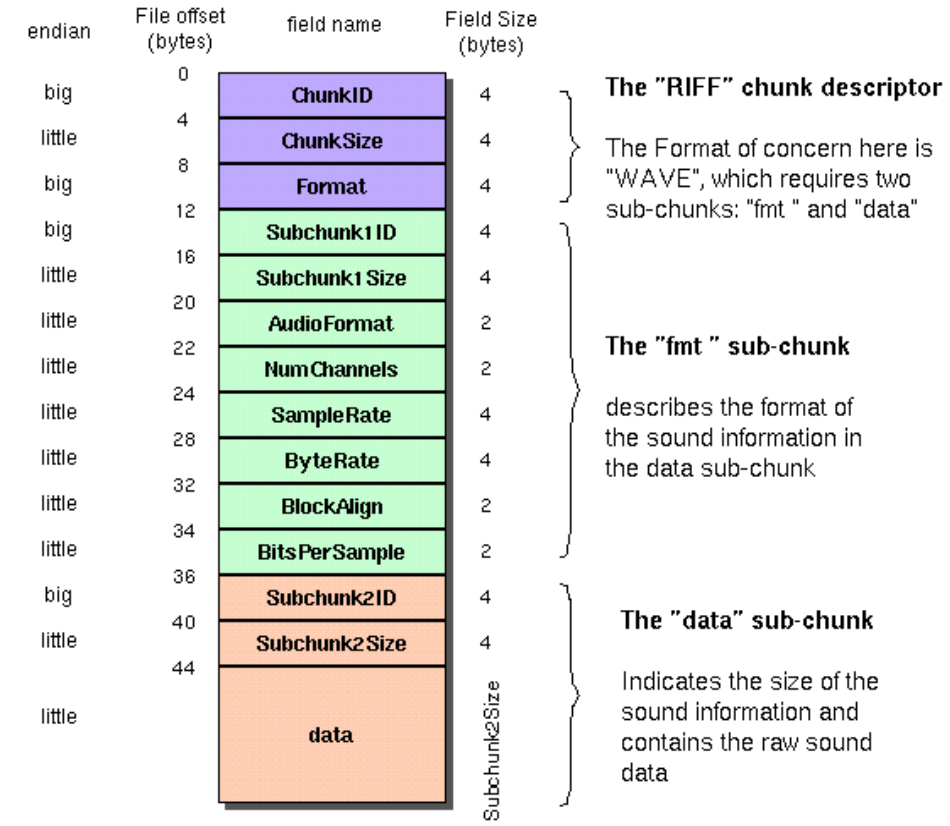
Reading WAV file parameters

To understand what this function does, we will first consider the structure of a WAV (or WAVE) file (see figure 7–3). The WAVE file format is a subset of Microsoft's RIFF specification for the storage of multimedia files. A RIFF (Resource Interchange File Format) file starts out with a file header followed by a sequence of data chunks. A WAVE file is often just a RIFF file with a single "WAVE" chunk which consists of two sub-chunks – a "fmt " chunk specifying the data format and a "data" chunk containing the actual sample data:
(https://ccrma.stanford.edu/courses/422/projects/WaveFormat/).
This function looks into the WAV and extracts the parameters that are needed. This function reads the parameters in the header of the WAV file. The place where the WAV files must be located in SIRLAB is the directory: *./wavs*. The unique input parameter of the function is *infilename* that is the name of the WAV file. The return parameters are the number of bytes per sample (*bytespersample*), the sampling rate (*samplerate*), the byte rate (*byterate*), the bits per sample (*bitspersam*), the number of channels in the WAV file (*wavnumchan*), and finally the number of all data samples in the WAV file (*numsamples*).

```
int read_wav_parameters(char              *infilename,
                        double            *bytespersample,
                        unsigned int      *samplerate,
                        unsigned int      *byterate,
                        unsigned int      *bitspersam,
                        short int         *wannumchan,
                        unsigned int      *numsamples);
```

Reading Parameters from Input Parameter File

An example of a input parameter file is in the figure 7–4. We can observe that each parameter has a number parameter identification with format "(XX)", where the number XX corresponds to the number of the parameter. After the parameter number follows a short description of the parameter, this part does not allow spaces.



(a)

```
00000000  52 49 46 46 3C AE DC 01 57 41 56 45 66 6D 74 20  RIFF<...WAVEfmt
00000010  10 00 00 00 01 00 01 00 00 EE 02 00 00 DC 05 00  ................
00000020  02 00 10 00 64 61 74 61 18 AE DC 01 03 00 07 00  ....data........
00000030  02 00 06 00 FA FF F8 FF F7 FF 06 00 07 00 07 00  ................
00000040  00 00 FE FF FF FF 08 00 03 00 06 00 05 00 07 00  ................
```

(b)

Figure 7–3: Standard WAVE Format 7–3(a) Table and 7–3(b) The WAV header using bvi (an hexadecimal binary file editor)

For this reason the _ is the separator of words. In the next line the value of the parameter is written. You can write up to 99 (enumerated from 01-99) parameters for use in a T-F operator implementation.

```
(01)Samples_of_the_CSTFT
2048
(02)Sample_overlay_for_CSTFT
16
(03)Samples_of_the_window
128
(04)Zeropadding
8192
(05)freqmin
700
(06)freqmax
6000
(07)Frame_Overlay
256
(08)Start_Percent_of_wav
0
(09)End_Percent_of_wav
100
(10)Outfile_directory
./outfiles/
```

Figure 7–4: *cstft_parameters.txt* Input Parameter File

```
int  read_parameters(   char            *infilename,
                        unsigned int    *numsamplesframe,
                        unsigned int    *jumpsamples,
                        unsigned int    *windowwidth,
                        unsigned int    *zeropadding,
                        double          *freqmin,
                        double          *freqmax,
                        unsigned int    *frameoverlay,
                        double          *startpercent,
                        double          *endpercent,
                        char            wavdir[100]);
```

This function reads values (parameters) in the parameter file. These values will be used to specify the behavior of the computation for the signal processing operator. The place where the parameter files must be located in SIRLAB is the directory: *./parameter_files*. The unique input parameter of the function is *infilename* that is the name of the input parameter file. The following are the return parameters for the CSTFT operator. The first is the number of samples needed to compute one frame (*numsamplesframe*). The second is the displacement factor of samples for the computation of a decimated CSTFT (*jumpsamples*). The third is the window width specification in number of samples (*windowwidth*). The fourth is the zero padding applied to the FFTs. This parameter increases the spectral resolution of

the output (*zeropadding*). The fifth and sixth (*freqmin, freqmax*) are the minimum and maximum frequencies that we want to see in the output frame. The seventh (*frameoverlay*) is the frame overlay, it specifies the number of samples that the CMF displaces in the WAV file to extract another set of samples (of size *numsample-frame*)to compute another output image. Is normal that the WAV files are very long, or the user is interested in an interval of the WAV file. With the eighth and ninth (*startpercent, endpercent*) parameters you specify the range where the CMF will work. The units of *startpercent, endpercent* are the percentage of the whole WAV file. And, finally, if the user wants to use another directory to store the output files, he may use the tenth parameter (*wavdir[100]*). The default is the *./outfiles* directory.

7.4.2 Processlib

The processlib is related with signal management functions in the SIRLAB Framework. The operations focus in the colormap selection, the reading of data segments in the WAV file, the creation of the window signal, the zeropadding signal operation, and mathematical adjustments to the data.

Read Colormap

This function reads and converts a colormap text file into a RGB matrix (colormap[256][3]) of 256 colors. The colormap text file is generated with help of Matlab. All the colormap files must be placed at directory *./colormaps*.

```
void readcmap(  int     colormap[256][3],
                char    filename[100]);
```

The figure 7–5 shows a colormap text file for the well know jet colormap of Matlab. The figure also shows three sets of values corresponding to the amount of red, green, and blue. The values are in the range $[0, 1]$, where 0 indicate no color and 1 indicate

full color. Each set of values is composed of 256 values.

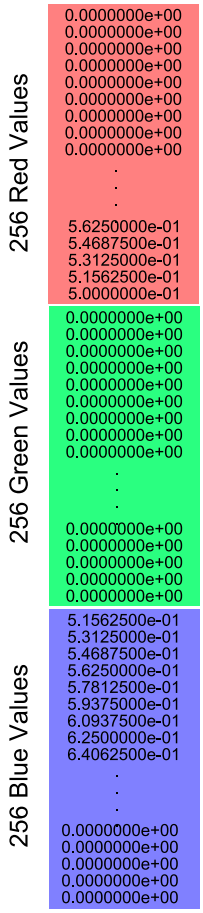


Figure 7–5: Values for the RGB in a SIRLAB Colormap File

Read Frame Data from WAV File

```
int read_frame_wav( fftw_complex     *datain,
                    int              framepos,
                    int              framelength,
                    int              bitspersam,
                    short int        wavnumchan,
                    double           *max,
                    char             *infilename);
```

The purpose of this function is the extraction a segment of data from the WAV file that is being processed. The size of the segment is specified in the input parameter file with the *numsamplesframe* value. The parameters given and returned by this

function are: **datain* is the pointer to the data and is an array of type `fftw_complex`, which is by default a double[2] composed of the real (in[i][0]) and imaginary (in[i][1]) parts of a complex number. The *framepos* specifies the sample at which the segment of data begins. The *framelength* specifies the number of samples that will be read, usually this value is *numsamplesframe*. The *bitspersam* is the number of bits for each sample and takes the value of $\{8, 16, 24, or 32\}$. The *wavnumchan* specify the number of channels that will be read. This value is one when data is real, because the value for the imaginary part of *datain* is zero. When *wavnumchan* is two the data is complex, indicating that channel 1 for real part and channel 2 for the imaginary part. The parameter *max* returns the maximum value read in the segment. And finally the *infilename* that is the name of the WAV file.

Create Window Data Signal

```
void create_window( fftw_complex    *window,
                    int             Nw,
                    int             numsamplesframe,
                    int             pos);
```

This function creates a zero-padded and shifted window function needed for the CSTFT computation. The window function is zero-valued outside of the region of support ($[0, N-1]$). An example of three window functions are in figure 7–6.

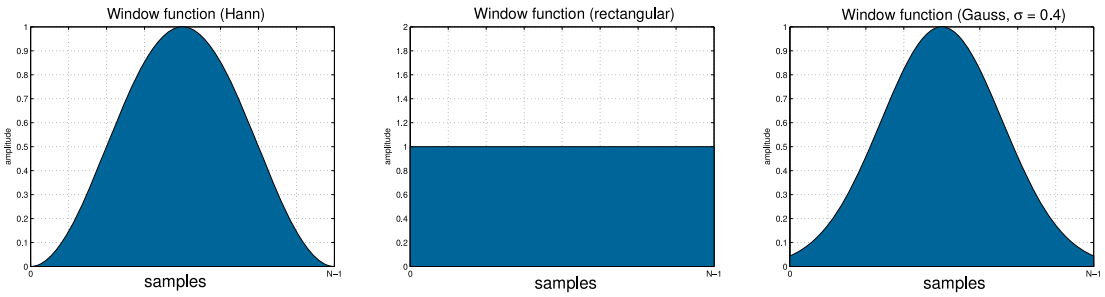


Figure 7–6: Three windows Hann, Rectangular and Gauss($\sigma = 0.4$) defined in interval $[0, N-1]$.

To create the zero-padded and shifted window function the following parameters must be given. The **window* is a pointer to the window data that is an array of type `fftw_complex` of size *numsamplesframe*. The *Nw* is the N size of the non-zero values of the window data. The parameter *pos* specifies where the non-zero windows values are centered. Is important to notice that the window data is cyclic. This means that for a window data of K samples and window size N, and window position P the non-zero values will be in the range $[\langle P - N/2 \rangle_K, \langle P - 1 + N/2 \rangle_K]$, where $\langle\rangle$ is the modulo operation. The figure 7–7 illustrates the concept of a displaced window data.

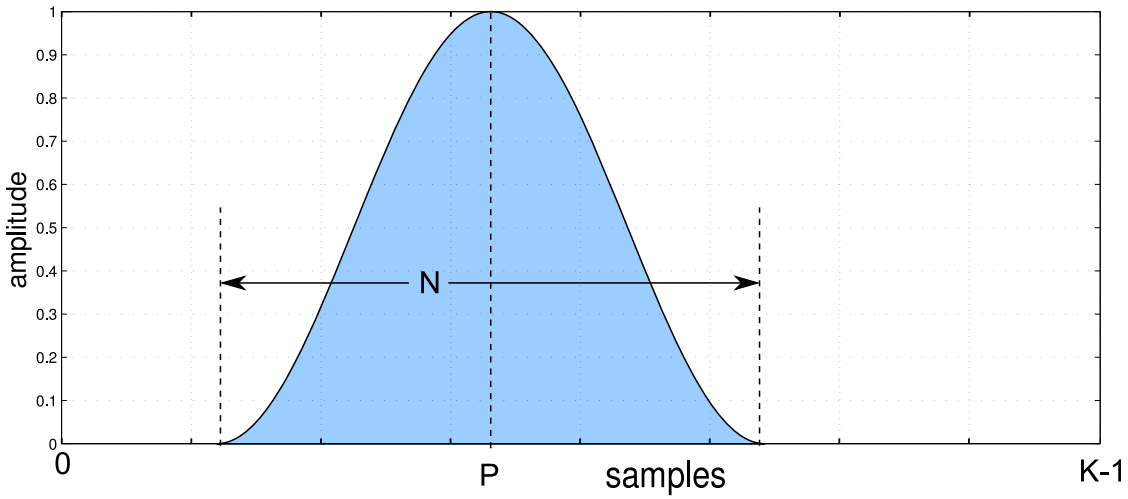


Figure 7–7: Example of a window data of size K samples, window size N, and window position P.

Zeropadding

Zero padding is an operation that consists of extending a signal (or spectrum) with zeros. It maps a length N signal to a length $M > N$ signal. The function *zeropaddingf* performs an extension using zero-padding to the input signal in *datanopad* of size N. The output is in the *datapad* of size $Npad$. Notice that $Npad \geq N$.

```
void zeropaddingf( fftw_complex    *datapad,
                   int             N,
                   fftw_complex    *datanopad,
                   int             Npad);
```

Write to the CSTFT Matrix

For computing a T-F operator, we must perform a set of operations several times and save the intermediate results in somewhere. The intermediate operations results are saved in a matrix equivalent data structure. OpenCV provides the IplImage, that is a very useful data structure that allows intermediate storage of the output for the T-F operator in double-precision. The IplImage data structure in essence is a CvMat but with some extra goodies buried in it to make the matrix interpretable as an image. This structure was originally defined as part of Intel Image Processing Library (IPL). An example of how a matrix is defined using IplImage is shown below:

```
IplImage *matstft = cvCreateImage(cvSize(px,py),
                                  IPL_DEPTH_64F,1);
```

This declaration create an image (matstft) of size px $\times$ py pixels, with 64-bit floating-point double-precision (IPL_DEPTH_64F).

```
void write2matstft( int            sam,
                    fftw_complex   *locdatain,
                    double         scale,
                    int            kmin,
                    int            kmax,
                    IplImage*      matstft,
                    int            col,
                    double         *maximo);
```

This function takes the data of size *sam* in the *locdatain* array and put in the corresponding column *col* in the image *matstft*. The locdatain is in complex data format and the matstft is composed of double-precision values. This function converts the complex data to a corresponding magnitude value using the following equation:

$$norma = \frac{\sqrt{(locdatain[i][0]^2 + locdatain[i][1]^2)}}{scale} \tag{7.1}$$

The values *kmin* y *kmax* specify the segment of the *locdatain* array that will be copied to *matstft*. The figure 7–8 shows graphically the process. In addition, the parameter *scale* allows to scale the data in the *matstft* array as shown in equation 7.1. The parameter *maximo* contains the maximum value of the *matstft* array.

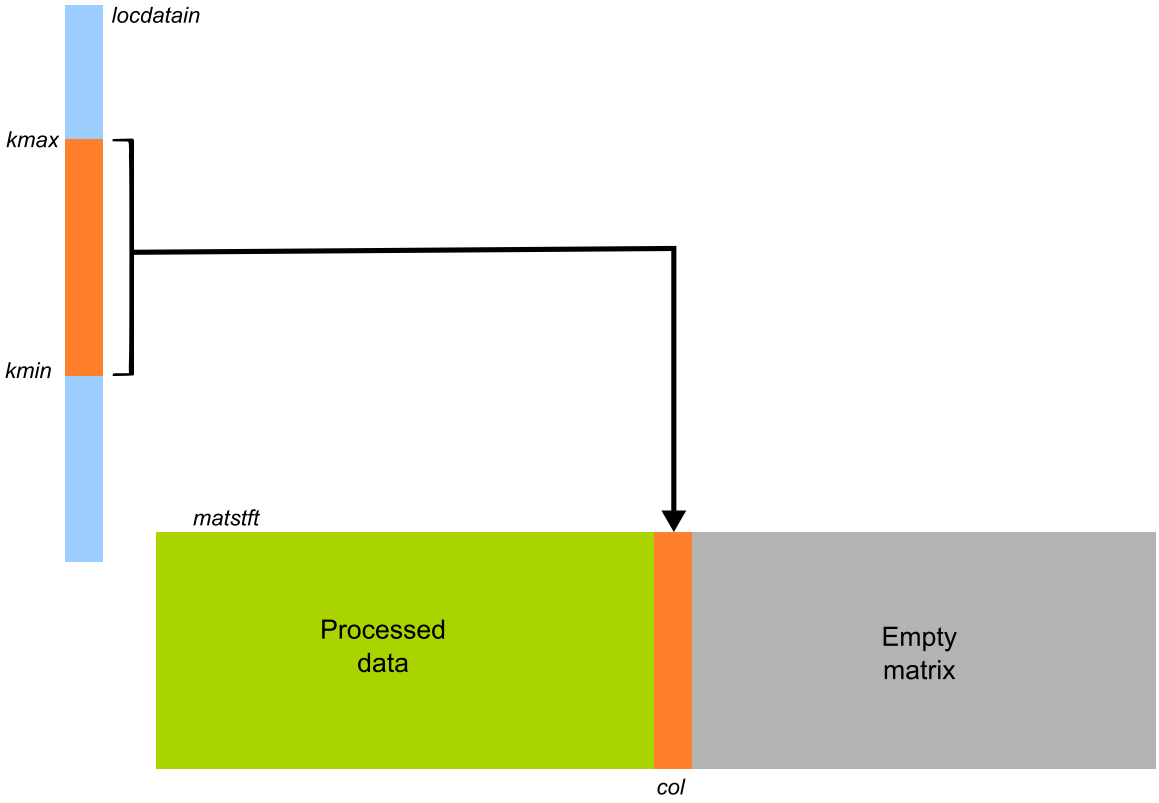


Figure 7–8: Process to copy a segment *[kmin,kmax]* of *locdatain* to the column *col* in the image *matstft* .

Convert to Colormap

After *matstft* is completed, is necessary to convert the double-precision values to an equivalent grayscale image. It is necessary to make a conversion of the double to an RGB value. This function does the operation based in the color indexation of the *colormap* array defined in the function 7.4.2. The parameters of this function are: the *imagin* that is the double precision image matrix, the *imagout* is the RGB image matrix, and the *maximo* that is the maximum value of *imagin*. The *maximo*

is used to normalize *imagein* and be able to perform the indexation according to the colormap of 256 values (*colormap[256][3]*).

```
void conv2colormap( IplImage*     imagin,
                    IplImage*     imagout,
                    double        maximo,
                    int           colormap[256][3]);
```

7.4.3 Aritmetlib

These group of functions are related with basic arithmetic operations performed over the data arrays of SIRLAB. The functions included in this library are the Hadamard product operation, signal normalization, and get the maximum of an array. The purpose of this library is contain a set of very general mathematical functions for digital signal processing operations.

Hadamard Product

For two matrices of the same dimensions, we have the Hadamard product also known as the entry-wise product and the Schur product. Formally, for two matrices of the same dimensions:

$$A, B \in \mathbb{R}^{m \times n} \tag{7.2}$$

the Hadamard product $A \odot B$ is a matrix of the same dimensions

$$A \odot B \in \mathbb{R}^{m \times n} \tag{7.3}$$

with elements given by

$$(A \odot B)_{i,j} = (A)_{i,j} \odot (B)_{i,j} \tag{7.4}$$

In the case of this function the order of the matrices A and B are $\mathbb{R}^{1 \times N}$. The matrix A corresponds to the *data1* array and the matrix B corresponds to the *data2* array. N is the number of elements of *data1* and *data2*.

```
void haddamard_sig( fftw_complex     *data1,
                    fftw_complex     *data2,
                    int              N);
```

Signal Normalization

This function takes an input complex data array (*data1*) and scales the norm of all the data elements. The function first finds the maximum norm in *data1*, then adjust the scale to normalize the magnitude of all the elements in *data1* to the parameter *value*. *N* is the number of elements in *data1*.

```
void normalize_sig( fftw_complex    *data1,
                    double             value,
                    int                N);
```

Get Maximum

This function takes an input complex data array and finds the maximum magnitude. The maximum value is returned in the *maxnorma2* parameter. *N* is the size of *data1*.

```
void getmax_sig( fftw_complex     *data1,
                 double           *maxnorma2,
                 int              N);
```

7.4.4 Outputlib

These functions are responsible to create graphical elements in the output frames. The functions include plotting signals, embed text or draw numerical scales in the template frame to compose an output frame. The template frame is an empty frame which is filled with the T-F output and the metadata associated. The figure 7–9 shows the appearance of an empty frame and the final frame with the data processed.

Signal to Frame

This function plot a time signal (data array) in a selected area in the template frame. The figure 7–10 shows the concept.

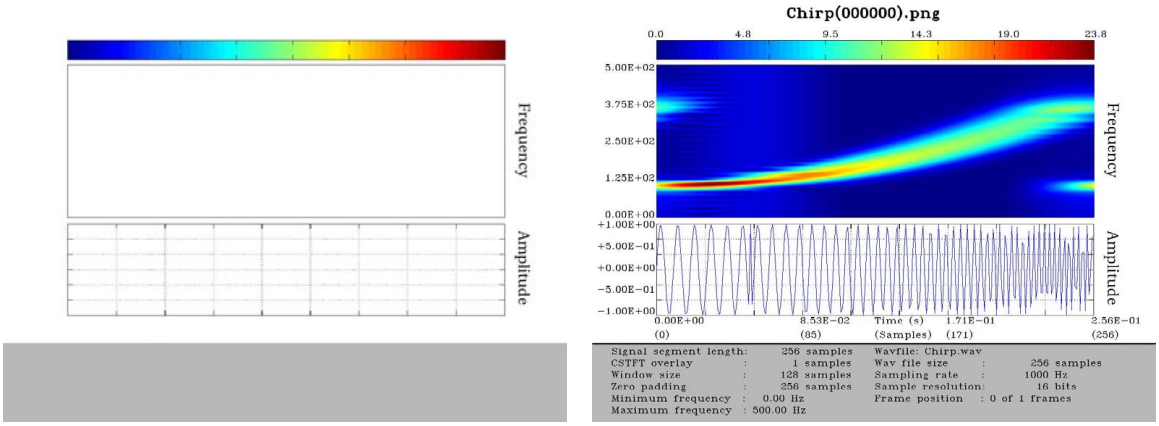


Figure 7–9: Empty Frame Template vs Fulfilled Frame Using the CSTFT Operator With a Chirp Signal.

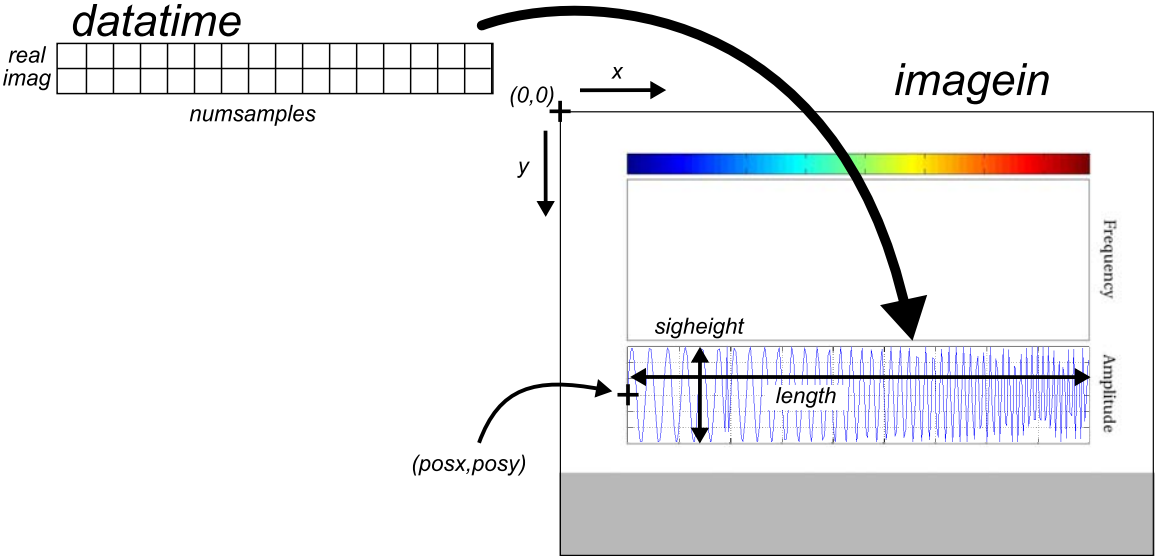


Figure 7–10: *datatime* Array Data is Plot in the *imagein* Array.

```
void signal2frame( fftw_complex       *datatime,
                   int                option,
                   IplImage*          imagin,
                   unsigned int       numsamples,
                   int                posy,
                   int                posx,
                   int                sigheight,
                   int                length,
                   double             *maxval);
```

The figure 7–10 shows the parameter *datatime* that is a complex data array. The function parameter *option* specifies what field of the complex data array will be plotted in the image. For the real part (*option=0*), the imaginary part (*option=1*), or the magnitude of the *datatime* array elements (*option=2*). The parameter *numsamples* is the number of samples of the *datatime* array. *posx, posy* are the parameters to set the origin (in pixels) where the signal will be draw (see figure 7–10). The *sigheight* and *length* parameters indicate the height and the width (in pixels) of the area where the signal will be draw. The *maxval* returns the maximum value of the signal.

Text to Frame

This function receives a message in the *texto* input parameter, and place it in the *imagein* at the *posx, posy* coordinates. The scale of the text is specified by *hscale, vscale* parameters. The *shear* takes values from 0.0 to 1.0 indicating the slant of the text font. The *thickness* is the width line of the font. And finally *line_type* that takes 3 values: *line_type*=8 generates an 8-connected line, *line_type*=4 generates a 4-connected line, and *line_type*=3 produces an anti-aliased line.

```
void text2frame( IplImage*       imagein,
                 char            texto[50],
                 int             posx,
                 int             posy,
                 double          hscale,
                 double          vscale,
                 double          shear,
                 int             thickness,
                 int             line_type);
```

Axis to Frame

```
void axisx2frame( IplImage*      imagein,
                  double         startval,
                  double         endval,
                  int            numoftags,
                  int            dir,
                  int            fmtprt,
                  int            posx,
                  int            posy,
                  int            width);
```

This function receives the *startval*, the *endval*, and the *numoftags* and generates a numerical scale in the *imagein*. Other parameters are necessary to define extra attributes of the numerical scale. The direction of the scale is defined by *dir* which takes values of: *dir*=1 for x orientation, and *dir*=2 for y orientation. The *fmtprt* indicates the format of the numbers in the scale (see table 7–3 and figure 7–11). The *posx* and *posy* define the position of the numerical scale in the *imagein*. Finally, the *width* specifies the width of the numerical scale in pixels.

Value	Format	Example (700)
1	$\%.2E$	$7.00E2$
2	$\%+.2E$	$+7.00E2$
3	$\%.1f$	700.0
4	$(\%.0f)$	(700)

Table 7–3: Values of *fmtprr* and the Format of the Number

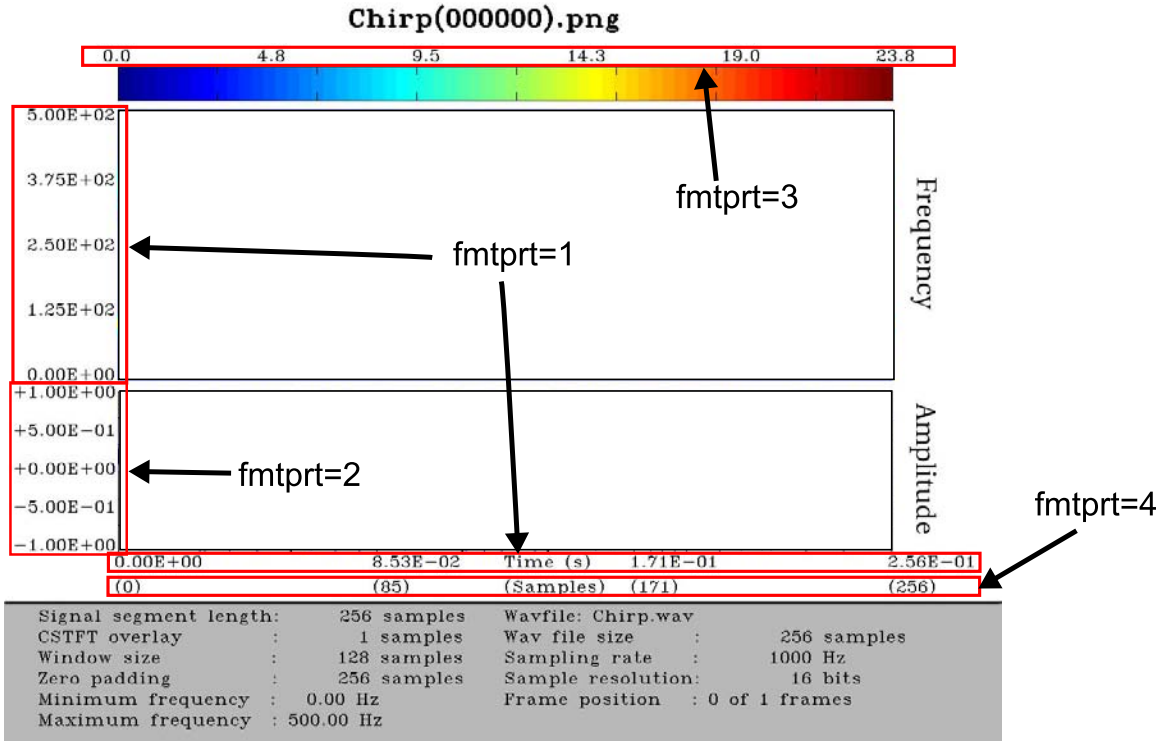


Figure 7–11: Frame with *fmtprt* parameter values

CHAPTER 8
SIRLAB GENERAL GUIDELINES

This chapter contains some general guidelines about the use of SIRLAB. The content is a complement of the chapter 7 and are mostly focused in practical considerations for the users of the CMF.

8.1 Bare Frame of the SIRLAB for CSTFT Implementation

The Fig. 8–1 depicts an empty frame that later will be completed with the CSTFT or signal processing operator output data. In the empty frame there are several regions defined: *Frame Filename* corresponds to the name of the specific png file (output frame) associated with a segment of the wav input file. Below appears a colorbar to show the intensity levels of the output. The *CSTFT spectrum* corresponds to the magnitude of the output array of the CSTFT with colors adjusted to the colormap selected, x-axis and y-axis corresponds to the time and frequency axis respectively. The *Time Signal* area corresponds to the segment of the wav file used to perform the CSTFT. Finally, the *metadata* have the information about the parameters used to process the output frame. An example of a typical frame is shown in Fig. 8–2, the input signal is a chirp with the following equation $x[n] = sin(2\pi * 100 * (n/f_s) + 2\pi * 1500 * (n/f_s)^3)$ with $f_s = 1000$ and $n \in \mathbb{Z}_{256}$.

CSTFT computational method

There exist two manners to compute the CSTFT: the filter method and the transform method. The transform method is more computationally efficient than filter method because it only involves a time shift (m axis) and a Fourier transform (k axis) to

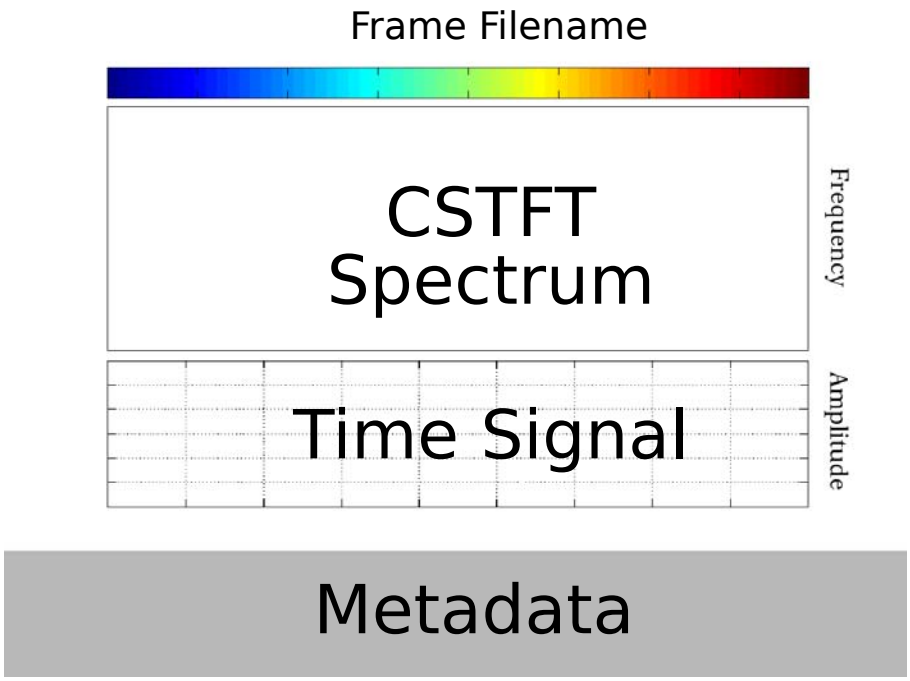


Figure 8–1: Bare Frame with Description of Fields.

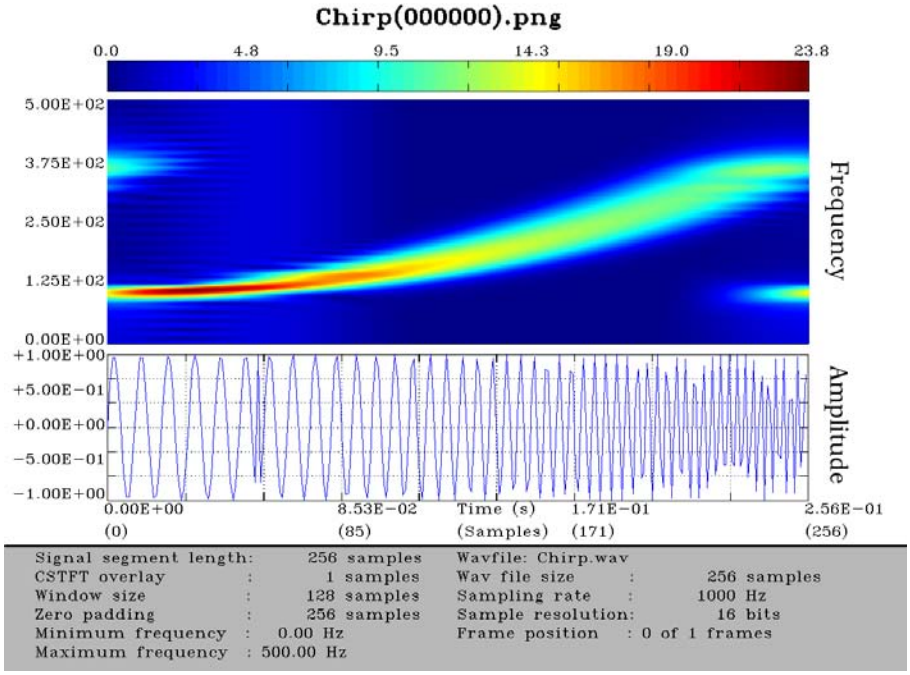


Figure 8–2: Example of a Frame Output with Chirp Signal Input.

compute one column of the time-frequency output matrix (see Fig. 8–3). According to the definition of the CSTFT in section 3.1.4 a complete CSTFT is an array of size $N \times N$. In this case the output has all the components of time and frequency possible. In some applications full CSTFT may not be necessary. In non-full CSTFT case we can compute some time shift resulting in a reduction in the number of computations required to obtain an approximate result to the full CSTFT. Also, not all frequency components may be needed, because we are interested in a range of frequencies. In this case we can extract the range of frequencies adjusting the parameter file of the CSTFT computation.

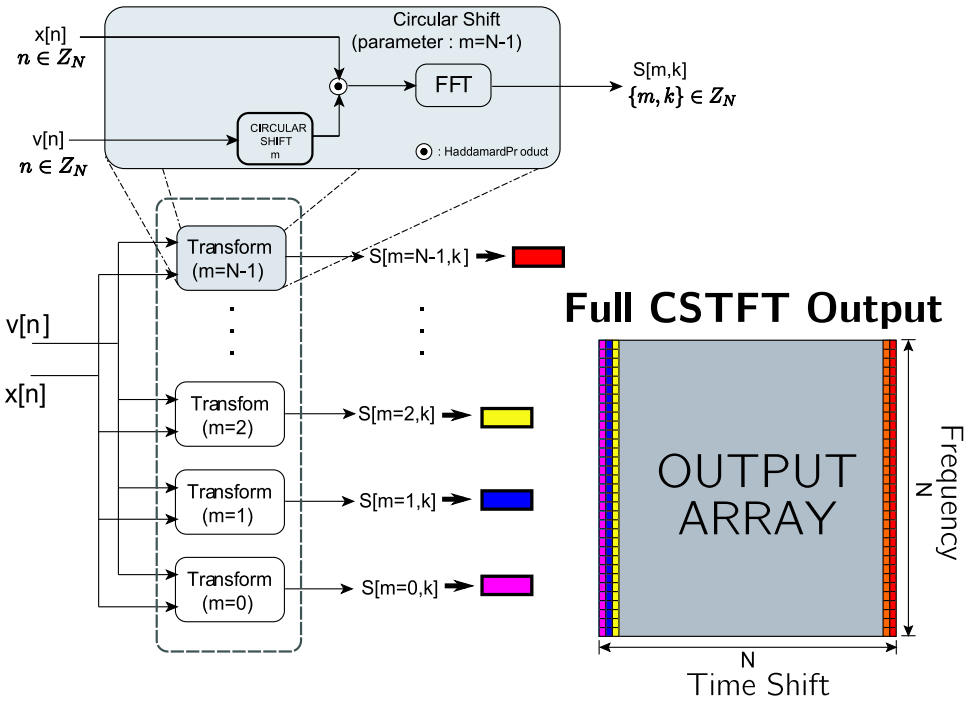


Figure 8–3: Transform method CSTFT computation diagram.

8.2 SIRLAB Ordered Set of Spectrograms

It is important to point out that the spectrograms are delivered as ordered sets which may be conformed into digital streaming videos readily available in a Linux-based environment to an information user for further signal analysis. A conceptual

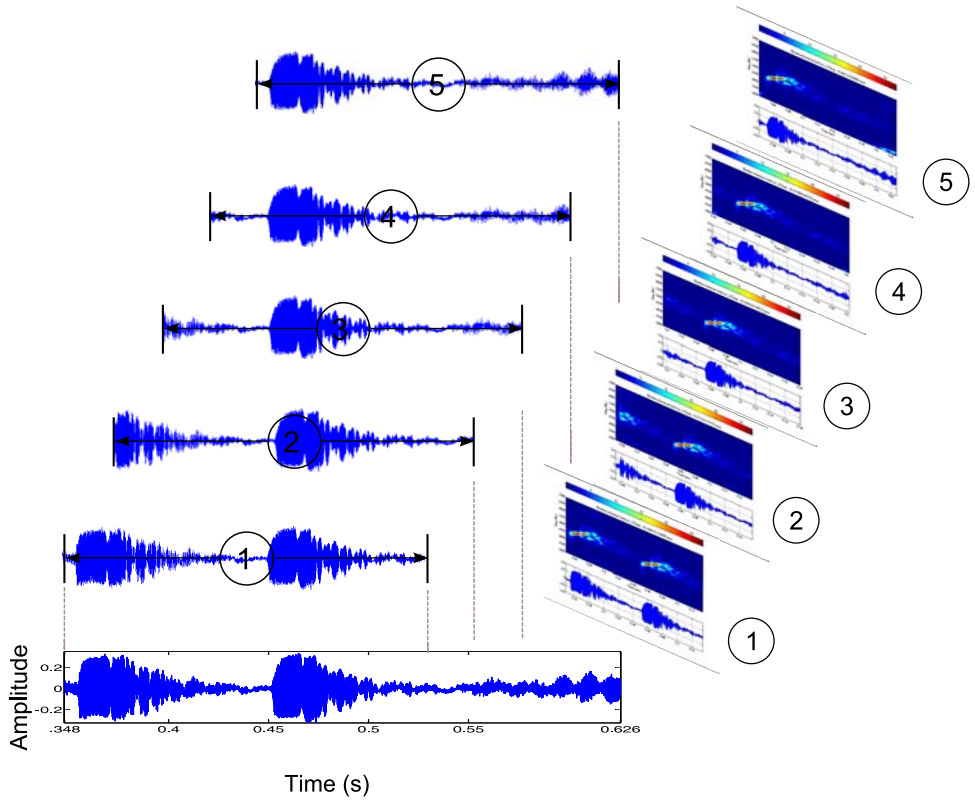


Figure 8–4: Simple depiction of the concept of a set of ordered spectrogram frames (only five frames are presented here) which may be presented as a digital streaming video.

depiction is presented Fig. 8–4 of how an acoustic signal is being segmented, with a desired segment length and window overlap, in order to produce a sequence of time-frequency representations in the conformed set of ordered spectrograms. The SIRLAB CMF produces these ordered sets in automated manner, starting from the recorded signal in waveform audio format. Again, the spectrograms are envisioned to serve as a first stage in a sequence of information processing operations seeking to extract signal intelligence important to an information user.

CHAPTER 9
CONCLUSION AND FUTURE WORKS

9.1 Definition of Standar Frame in SIRLAB

An standard frame (see figure 9–2) is the frame unit selected to conduct performance comparison between computational structures. This frame requires 128 operations that include FFTs of 8192 samples. The frequencies of interest for this frame are from 700 Hz up to 6000 Hz, that is equivalent to 1969 frequency values.

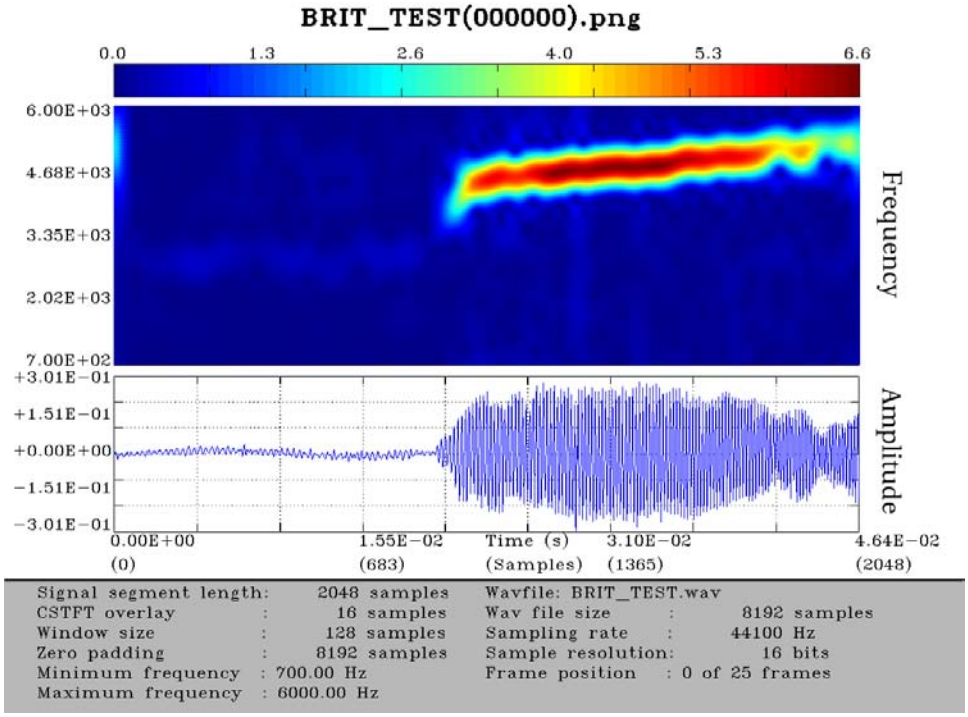


Figure 9–1: Standard Frame for Eleutherodactylus brittoni.

9.2 Speedup and Video Streaming Using SIRLAB

The figure 9–2 shows the spectrogram of an Eleutherodactylus brittoni frog output frame of 46 ms signal length. The parameters resulting of processing the frame are shown in the gray area at the bottom of the figure. Computer speedup of more than 30 times have been reached when compared with MATLAB implementations utilizing the same computational resources and algorithm formulations. The SIRLAB computational framework has shown significant reduction in the processing running time for a standard frame defined above. The comparison of the processing time for the standard frame was conducted using the aipsrv01 computational structure with Matlab and SIRLAB. The Matlab implementation required 1.44 secs/frame and the SIRLAB .038 secs/frame. This is an improvement of 37 times.

The processing time of 0.038 ms, gives us a processing rate of 26 fps, which is near to video streaming capability. The objective of a frame rate of 30 frames per second may be reached, for some parameter specifications. With this frame rate the SIRLAB will support the ATSC digital television standard.

9.3 High Resolution and High Sampling Rate Bioacoustics

Under the WALSAIP-CIMES collaboration, we have access to high resolution and high sample rate sounds recorded at the west area of Puerto Rico. Fig. 9–2 shows a recorded hydrophone sound at Cabo Rojo,PR (192 KHz sampling rate and 24 bits resolution) with the presence of pistol shrimp. SIRLAB is capable to perform a high resolution extraction of an area of the spectrogram as shown in Fig. 9–2. This zoomed area corresponds to the snap of a shrimp.

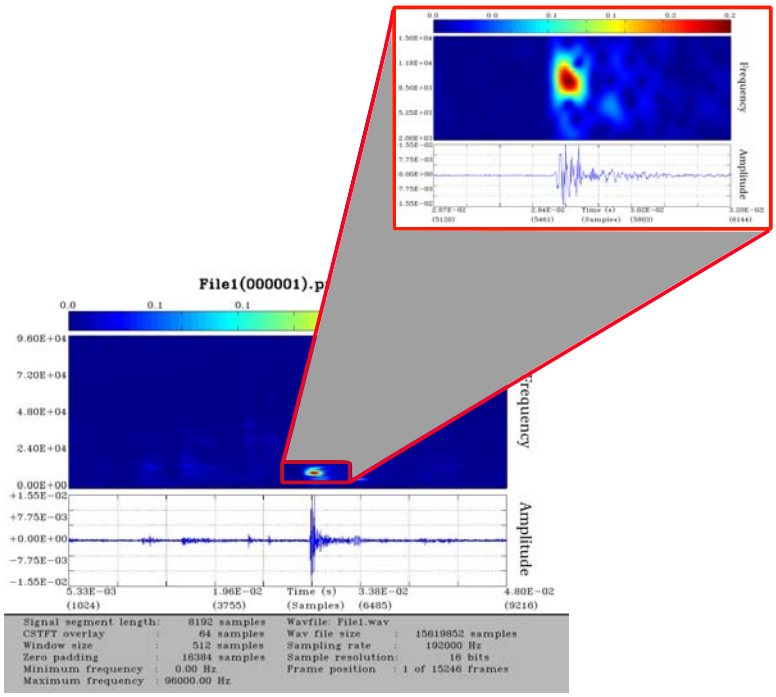


Figure 9–2: Sample Frame of Pistol Shrimp sound acquired at 192 KHz/24 bits with FR-2 recorder.

9.4 SIRLAB Comparison with Other Spectrogram Software

Table 9–1 show a comparison of SIRLAB with other commercial and open source products that perform similar operations.

Table 9–1: Some Spectrogram Software Comparison.

Feature	Software			
	Raven	Song Scope	Sonogram	SIRLAB
OpenSource	No	No	Yes	Yes
GUI Interface	Yes	Yes	Yes	No
High resolution	No	Yes	Yes	Yes
Web Integration	No	No	Yes	Yes
Signal Operator Implementation	STFT	STFT	STFT, Wigner Ville	CSTFT, DCFT, AF
Support of OS Scripting	No	No	No	Yes
Real Time Processing	No	No	No	Yes
Movie Generation	No	No	No	Yes
Stream processing	No	No	No	Yes

9.5 Future Works

- The computational framework is planned to be integrated in a web-application to access the SIRLAB Framework through the NETSIG. An expected feature of the integration is that user can submit a sound recorded in a mobile device to the SIRLAB, process and receive the output of the CSTFT. Fig 9–5 depicts how is expected that mobile devices access the sequence of spectrograms in the NETSIG device.
- The new multicore processors open a window to obtain better results for the speedup of the SIRLAB. Another on going work is the development of a multicore implementation of the SIRLAB Framework.
- In this thesis only three T-F operators (CSTFT, CAF, DCFT) have been implemented. The list of T-F is long and many other operators are suitable to be implemented using SIRLAB.

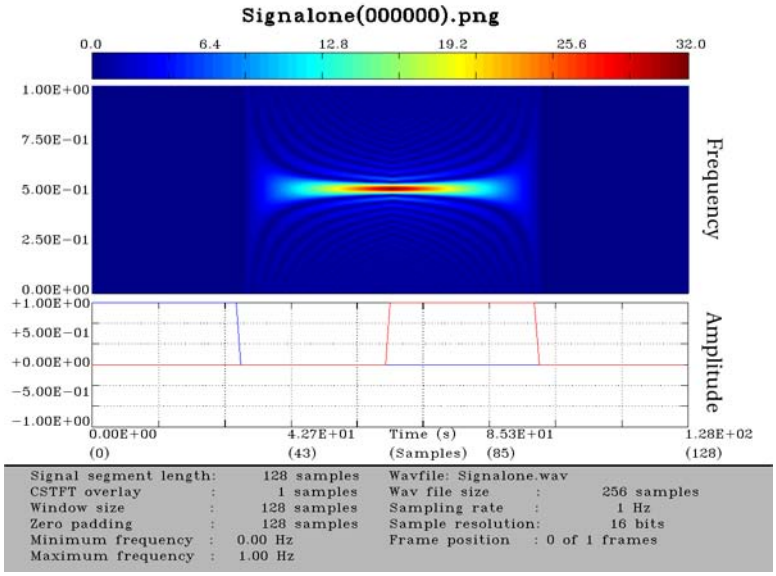


Figure 9–3: Cyclic Ambiguity Function Implementation

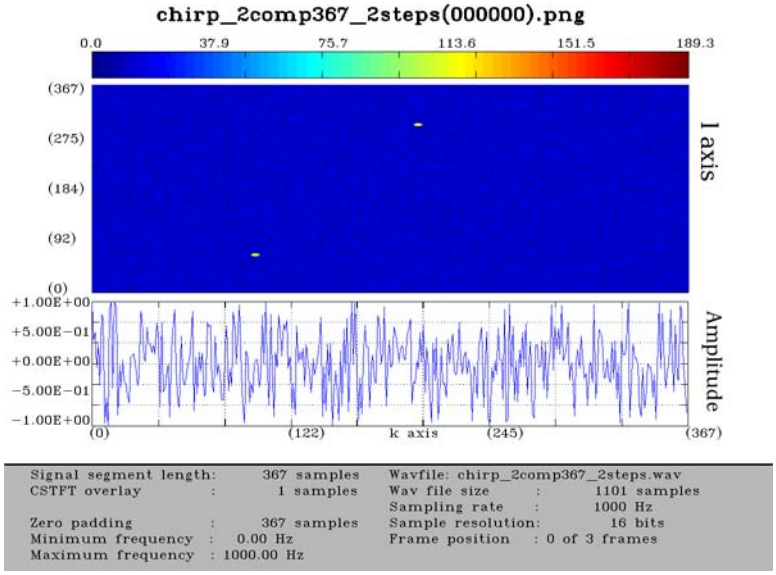


Figure 9–4: DCFT Implementation

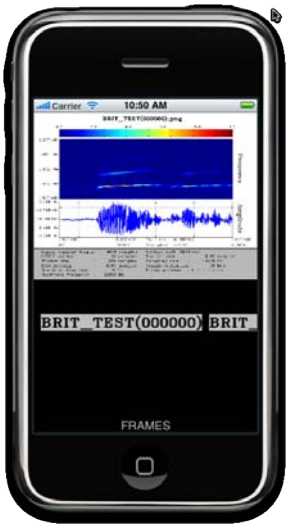


Figure 9–5: iPhone App Presentation.

REFERENCE LIST

[1] Yuji Yunes. Acoustic signal representation for environmental surveillance monitoring (ESM), 2007.

[2] Ervin Sejdic, Igor Djurovic, and Jin Jiang. Time-frequency feature representation using energy concentration: An overview of recent advances. *Digital Signal Processing*, 19(1):153 – 183, 2009.

[3] LJubisa Stankovic. On the time-frequency analysis based filtering. *Annals of Telecommunications*, 55:216–225, 2000. 10.1007/BF02994785.

[4] J Zygierewicz, P J Durka, H Klekowicz, P J Franaszczuk, and N E Crone. Computationally efficient approaches to calculating significant ERD/ERS changes in the time-frequency plane. *Journal of neuroscience methods*, 145(1-2):267–76, June 2005.

[5] Antonio H. Costa and Stephan Hengstler. Adaptive timefrequency analysis based on autoregressive modeling. *Signal Processing*, pages 1–10, August 2010.

[6] Kathleen A Lindlan, Janice Cuny, Allen D Malony, and Sameer Shende. A Tool Framework for Static and Dynamic Analysis of Object-Oriented Software with Templates. *Database*.

[7] Emmanuel J. Candes, Philip R. Charlton, and Hannes Helgason. Detecting highly oscillatory signals by chirplet path pursuit. *Applied and Computational Harmonic Analysis*, 24(1):14 – 40, 2006.

[8] Xeng-Gen Xia. Discrete chirp-fourier transform and its application to chirp rate estimation. *IEEE Trans. Signal Processing*, 48:3122–3133, Nov 2000.

[9] L. Cohen. Time-frequency distributions-a review. *Proceedings of the IEEE*, 77(7):941 –981, jul 1989.

[10] Nicholas N. Bennett and Naoki Saito. Using edge information in time-frequency representations for chirp parameter estimation. *Applied and Computational Harmonic Analysis*, 18(2):186–197, 2005.

[11] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[12] Roberto E Gonzalez and Nelson E Padilla. Automated detection of filaments in the large scale structure of the universe. Technical Report arXiv:0912.0006, Dec 2009. Comments: 14 pages, 12 figures, submitted to MNRAS.

[13] Ery Arias-Castro, David L. Donoho, Xiaoming Huo, and Craig A. Tovey. Connect the dots: How many random points can a regular curve pass through? *Advances in Applied Probability*, 37(3):571–603, 2005.

[14] Ery Arias-castro, David Donoho, and Xiaoming Huo. Near-optimal detection of geometric objects by fast multiscale methods. *IEEE Trans. Inform. Theory*, 51, 2005.

[15] Xiaoming Huo and David Donoho. Multiscale detection of filamentary features. In *in Image Data. SPIE Wavelet-X*, 2003.

[16] Ery Arias-Castro, Boris Efros, and Ofer Levi. Networks of polynomial pieces with application to the analysis of point clouds and images. *J. Approx. Theory*, 162(1):94–130, 2010.

[17] Diane J. Cook and Lawrence B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15:32–41, 2000.

[18] L. Diduch, R. Muller, and G. Rigoll. A framework for modular signal processing systems with high-performance requirements. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1159 –1162, 2-5 2007.

[19] Su Hao, Wing Kam Liu, Brian Moran, Franck Vernerey, and Gregory B. Olson. Multi-scale constitutive model and computational framework for the design of ultra-high strength, high toughness steels. *Computer Methods in Applied Mechanics and Engineering*, 193(17-20):1865 – 1908, 2004. Multiple Scale Methods for Nanoscale Mechanics and Materials.

[20] Kyusoon Lee and Adam W. Bojańczyk. Alps: A software framework for parallel space-time adaptive processing. In *PARA*, pages 423–432, 2004.

[21] James M. Lebak and Adam W. Bojańczyk. Design and performance evaluation of a portable parallel library for space-time adaptive processing. *IEEE Trans. Parallel Distrib. Syst.*, 11(3):287–298, 2000.

[22] Cesar A. Aceros-Moreno. Fast signal transforms for radar information processing. Master's thesis, University of Puerto Rico, 2005.

[23] K. Hoffman and R. Kunze. *Linear Algebra*. Prentice Hall, second edition, 1971.

[24] S. Lang. *Algebra*. Addison-Wesley Publishing Company, third edition, 1993.

[25] N. Hamilton and J. Landin. *Set Theory and the Structure of Arithmetic*. Allyn and Bacon, 1963.

[26] A. Fraenkel. *Set Theory and Logic*. Addison-Wesley Publishing Company, 1966.

[27] H. Krishna. *Digital Signal Processing Algorithms*. CRC Press, 1998.

[28] D. Buchthal and D. Cameron. *Modern Abstract Algebra*. John Wiley and Sons, 1987.

[29] W. Greub. *Graduate Texts in Mathematics. Linear Algebra*. Springer-Verlag, fourth edition, 1981.

[30] Dilia Beatriz Rueda-Serrano. Fast multidimensional convolutions and SAR image formation simulations in a matlab environment, 2000.

[31] Hector M. Lugo-Cordero, Kejie Lu, Domingo Rodriguez, and Sastri Kota. A novel service-oriented routing algorithm for wireless mesh network. In *Proc.*

IEEE MILCOM 2008, San Diego, CA, 2008.

[32] Lu Kejie, Yi Qian, Domingo Rodriguez, Wilson Rivera, and Manuel Rodriguez. Wireless sensor networks for environmental monitoring applications: A design framework. In *IEEE GLOBECOM 2007 proceedings*, 2007.

[33] Gonzalo Vaca-Castano and Domingo Rodriguez. Using syllabic MEL cepstrum features and k-nearest neighbors to identify anurans and birds species. *IEEE Workshop on Signal Processing Systems, SiPS2010 (for publication)*, Oct, 2010.

[34] Ivan J. Rivera-Lebron. Hardware implementation of time frequency tools for power quality applications. Master's thesis, University of Puerto Rico.

[35] Lola X. Bautista-Rozo. Web-base data processing for environmental surveillance monitoring applications. Master's thesis, University of Puerto Rico, 2007.

[36] Ana B. Ramrez-Silva. On implementing tine-frequency representations on hardware/software computational structures for sar applications. Master's thesis, University of Puerto Rico, 2006.

[37] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 1st edition, September 2008.

[38] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".